Artificial Intelligence-Based Fault Prediction Framework for WBAN

Mamoun Awad, Farag Sallabi, Faisal Naeem, and Khaled Shuaib

***Abstract*—Fault-management in Wireless Body Area Network (WBAN) systems is complex and requires careful design and implementation to avoid failure in life-critical supporting systems. Challenges such as unpredicted sensor faults, massive streams of packets, privacy, and the stringent requirements of the WBAN systems need to be addressed. This paper investigates the challenges of the detection and prediction of faults in sensor node health (management) packets in the context of WBAN. We propose a framework that incorporates AI-based prediction models and traditional threshold checking to detect faults. Three Machine learning techniques, namely, Artificial Neural Networks (ANN), Support Vector Machines (SVM), and Decision Trees, are used in the framework to learn and detect faults with high accuracy greater than 96%, yet the framework is flexible and can incorporate other techniques. Besides, the framework is equipped with alarm notifications, prediction model deployment, version control, and sensor profiling. As proof of concept, a fault management prototype is implemented and validated. The prototype utilizes data on the cloud to be shared and managed and accurately classify faults and automation of sensor profiling, training, and validation of new models. The prototype helped uncover new software faults that must be considered in any fault management system.**

***Index Terms*—WBAN; Machine Learning; Fault Management, Fault Prediction, Sensor Health Packets.**

I. Introduction

Sensor networks have grown dramatically, supporting enormous medical and non-medical applications. This is due to the vast advances in wireless communications and semiconductor technologies. Wireless Body Area Network (WBAN) is a special purpose sensor network operating autonomously to connect medical devices, implanted or worn, within a short distance. The adoption of WBAN devices can provide personalized online care and thus bring the cost of healthcare down dramatically**Error! Reference source not found.**.

To make it socially acceptable, WBAN should be reliable and effective. To address reliability, three core characteristics must be appropriately handled, namely, fault tolerance, Quality of Service (QoS), and security**Error! Reference source not found.**. To enable a seamless operation of WBAN, capabilities to handle fault tolerance and recovery must be integrated into the design and architecture of WBAN. For medical applications, strict QoS metrics such as packet delay, error, and loss should be maintained at an acceptable level to avoid any fatalities. The third crucial requirement to achieve reliable WBAN is security. This requirement has been addressed in**Error! Reference source not found.**, and it is out of the scope of this work.

WBAN systems consist of hardware and software components; hence, software and hardware faults need to be addressed. Many researchers have focused on hardware faults in sensors such as irregularity readings, hardware faults, and battery leakage faults. However, faults in software are as critical as hardware faults, which include communications faults, model malformation, software bugs, inadequate validation, and induced human errors.

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 that were used to support such systems**Error! Reference source not found.**[4][5]. However, coping with new learning techniques represents a challenge because of variant trends in data. When the data streams change over time, patterns, and trends change as well; hence, the training of used ML models on new data must be updated continuously. Moreover, new learning models might emerge and provide better capabilities. For example, ANN and SVM were outperformed by Convolutional Neural Network (CNN) and Deep learning techniques lately. Therefore, an AI-based fault management system must be adaptable to use new emerging AI techniques.

Medical sensors that record/monitor a patient's vitals can differ in hardware, software, data/packet format, thresholds, and communication means. Even sensors designed for similar functions, such as the ones used for EEG and EMG signals, may differ in specifications (range, data format, communication means, etc.). Sensors with improved capabilities and safety measures are emerging continuously. Thus the healthcare industry needs to be aware of the latest technologies in this domain and utilize what would be considered the best fit for their applications. Sensor profiling might exhibit a right solution in which sensors specifications from the different manufacturers are maintained and updated regularly. Such profiling will help in the deployment of best practices for WBAN health applications.

In this paper, we focus on the design and implementation of fault-management systems in the context of a WBAN environment where failure cannot be tolerated in life-critical healthcare systems. We attempt to address some of the main aspects in WBAN systems, namely, fault detection, sensor data (both health and biological) management, sensor deployment and profiling, and AI-based models' accuracy, deployment, and

updates. Specifically, we propose an AI-based framework that incorporates different ML techniques to detect faults. The main contributions of this work are as follows.

1. Propose a fault-management framework that incorporates AI/ML models for WBAN applications. The framework considers the distributed nature of WBAN within the context of a healthcare system.
2. Implement a machine learning (training and validation) process to support the fault tolerance framework. The process includes the automation of training, validation, and deploying of prediction models.
3. Generate empirical results for the three considered prediction models, namely, ANN, SVM, and Decision Trees.
4. Develop a prototype for Personal Digital Assistant (PDA) application on the patient site and a prototype of training and data management for the data repository site.

The rest of the paper is organized as follows. In Section II, we present the current state of the art in fault-management systems and applications. Section III provides a background on fault types and classification. In Section IV, we present our AI methodology and framework. Section V presents the empirical results and shows our prototype. Finally, we conclude the paper and state our future research directions in Section VI.

II. Related work

The literature is rich in research work on fault management that focuses on fault diagnosis, detection, and tolerance**Error! Reference source not found.**. However, in this section, we focus on work conducted in the area of IoT and WBAN fault management. Such research is still in its early stages with many open research problems [1].

Zhou *et al.***Error! Reference source not found.** addressed the issue of fault management in an IoT network using sensors with different functions. Their approach was based on the concept of virtual services, which uses data from more than one sensor to cover an actual service in the event of failures. They applied regression analysis to approximate virtual services. They adopted the service-oriented architecture to develop a sensing device adaptation scheme for fault-tolerant IoT. The proposed approach might be appropriate for IoT applications like smart homes, but it does not apply to smart healthcare systems. The paper in**Error! Reference source not found.** presents an IoT-based architecture to support scalability and fault tolerance for healthcare systems. They achieve fault tolerance by deploying a redundant sink node that backups the main sink node in case of communication or node failures. The authors claim that their approach of fault tolerance covers many fault situations, such as malfunctioning of sink node hardware and congestion at the node. This approach handles the problem of faults reactively, i.e., there is no mechanism to proactively determine the situation of the sensor nodes to minimize fault incidents. Furthermore, there is no indication of the performance of this approach in terms of fault detection and recovery.

Yang *et al.* [9] 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. They use a linear regression model based on trapezoidal fuzzy numbers to predict which readings in the outlying data vector are suspected to be faulty. Sensor health packets were not used; only biological data was the source of learning. The authors assumed time and space correlation between physiological readings; hence, the proposed method will not work if the patient is connected with only one sensor. Additionally, the authors did not consider/report false alarm rates. A fault diagnosis model based on Deep Neural Networks (DNN) was proposed in [4]. Temporal coherence of raw time-series sensor data without feature selection and signal processing were used for training DNN. Raw data was segmented and fed to DNN. However, the conducted experiments did not include medical data, and the output of the DNN model was binary, i.e., a prediction is either faulty or normal.

The authors in [10] 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. To save battery life, any faulty detected readings were not forwarded. The authors considered binary outcomes and assumed that sensor nodes could relay a different kind of data besides medical data. Moreover, they employed a threshold to determine faulty versus non-faulty readings. Additionally, the reported prediction rate was relatively low, which made it inapplicable to a WBAN environment.

Sharma *et al.***Error! Reference source not found.** researched 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 were capable of classifying and detecting faults; however, training was shown to be burdensome as estimation methods cannot reliably classify faults.

Opportunities and challenges of healthcare monitoring and management in the IoT paradigm with cloud-based processing are presented in [11]. The paper reviewed the current state and projected future directions for the integration of remote health monitoring technologies into the clinical practice of medicine. They also highlighted several challenges in sensing, analytics, and visualization that need to be addressed.

The paper in [12] proposed a layered fault management scheme for end-to-end transmission that suits the heterogeneous nature of IoT networks. The authors concluded that end-to-end connectivity involves different networks with different performance and connectivity requirements and that sectional monitoring is a better choice to apply as a suitable management technique for each section of the network. 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 [13] to monitor all sensors within a WBAN. VMSFT analyzes data collected from the WBAN, detects any sensor failures, and responds proactively. Also, VMSFT is

based on cloud computing service as IaaS (infrastructure as a service); however, it is restricted to local/limited geographic areas. Another work [14] proposed online fault detection and isolation of erroneous node in a WBAN. The work is based on centralized data fusion detection. A PDA is used to provide data transmission and perform real-time analysis of physiological sensor data to diagnose and identify faulty nodes.

The authors of [15] 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 what was called smart gateways. In [16], the authors presented a deployment of WSN based greenhouse management. The deployment was for the purpose of monitoring to provide certain services. Our work is similar as it provides monitoring and management services, however, we focus on medical context and utilize ML/AI methods to achieve that. The work in [17] surveyed different approaches to designing eHealth monitoring focusing on cost, usability, security and privacy. The authors presented in details different components of the monitoring lifecycle and essential service components. The work did not discuss faults detection and recovery or incorporate data collected in prediction or learning. In [18], the authors proposed and evaluated a wireless relay-enabled task offloading mechanism that consists of a network model and a computation model.

Our work is similar to the work in**Error! Reference source not found.**[13][15][19]. However, we consider biomedical and sensor health packets; we apply machine learning techniques, prediction model management, and update, as well as sensor profiling and threshold checking. Table I presents a comparison between our work and others.

TABLE I
COMPARISON OF OUR WORK VS. OTHER WORKS

Research Reference	Fault Detection	Data Management	Machine Learning Approach	Prediction Model Management	Prototype as proof of concept
Error! Reference source not found.	Yes	No	Yes	No	No
Error! Reference source not found.	Yes	Yes	No	No	Yes
Error! Reference source not found.	Yes	No	Yes	No	No
Error! Reference source not found.	Yes	No	Yes	No	No
[19]	No	Yes	No	No	Yes
Our Work	Yes	Yes	Yes	Yes	Yes

III. FAULTS CLASSIFICATION IN WBAN

A fault is defined as an abnormal condition or defect at the component, equipment, software, or subsystem level, which may lead to a failure**Error! Reference source not found.** or wrong decision being made. To diagnose a fault, three tasks are usually applied: fault detection, fault isolation, and fault identification. Fault detection is to detect a hardware or software malfunction and determine the time of its occurrence. Fault isolation is to locate the faulty component and isolate it. Fault identification is to identify the type, shape, and impact of the fault.

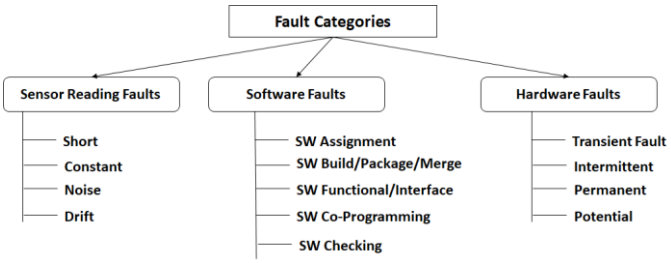

Fig 1. General Fault Categories.

Generally, faults in WBAN can be categorized into sensor reading faults, software faults, and hardware faults [1], as depicted in Fig 1. Sensor reading faults can be further classified into data-centric, system-centric [7], and network-centric, as shown in Table II. In data-centric, a data fault is defined as data reported by a sensor that deviates from its normal behavior. This could be very critical in healthcare and environmental monitoring applications. While in system-centric, several system hardware and software faults have been known to result in sensor faults. Typical hardware faults include damaged sensors, short-circuited connections, low battery, and calibration errors. Network-centric is concerned with managing dependencies between network elements and maintaining end-to-end connectivity. Faults may include end-to-end delay, loss of connectivity, and degraded quality of service, etc.

In this paper, we focus on sensor reading and hardware faults. Table II presents a detailed description of these faults. The reader is referred to [6] for an in-depth fault categorization and details. Channel faults can cause sensor reading faults in WBAN due to the nature of the human body and possible interference. Body movements cause a frequent change in network topology as correlated nodes might move with regard to each other.

For fault diagnosis, a simple threshold-based technique can be considered. However, in WBAN, using such a technique is not adequate because of the hybrid resource capabilities of devices in terms of installed software and data rates. For example, ECG, Temperature, and SpO2 sensors emit different readings; hence, employing threshold checking by just comparing energy levels might deliver inaccurate results. Moreover, setting low threshold values can lead to a high false-positive rate, while high threshold values might lead to a high false-negative rate. Furthermore, medical sensor readings can be interpreted differently based on different situations (normal vs. emergency) and environments (home vs. emergency room) **Error! Reference source not found.**. Another possible technique to deal with faults could be through replication or redundancy. For example, if a sensor fails, the sink node considers receiving data from another redundant deployed

node. However, unlike WSN, WBAN is a sparse communication network as it spans the human body, which is very limited in space. Thus, adding more sensors on the human body might cause interference, which in turns causes other faults**Error! Reference source not found.**.

TABLE II
SELECTED HARDWARE AND SENSOR READING FAULTS DESCRIPTIONS

FAULT TYPE	DESCRIPTION	IMPACT OF FAULT
Data-Centric		
Outlier	Data point that deviates from other observations due to noise, error, events, or malicious attacks.	It may skew the mean, variance, gradient, and other data features.
Spike	The rate of change in the gradient of data over a period of time is higher than expected. Frequent causes include battery failure and other hardware failures or loose wire connections.	Spikes should be discarded and result in a loss of sensor data yield.
Stuck-at	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.
High Noise or Variance	Sensor values experience unexpectedly high variation or noise due to hardware failure, environment out of range, or low battery supply.	If the noisy data tracks other sensors, then the data still offers value and should not be discarded.
System Centric		
Calibration	The sensor reports values that are offset from the ground truth. Calibration error and sensor drift is the primary cause of this fault.	Data should not be discarded. Un-calibrated data can still provide insight.
Hardware	A malfunction in the sensor hardware that causes inaccurate data. This is due to environmental disturbance, short circuit, or loose wire.	Data is meaningless as the sensor is not performing as designed. Should be discarded.
Low 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.
Clipping	The sensor maxes out at the limits of the ADC	It still holds some information content. Indicates data exceeds the upper or lower ADC values.
Calibration	The sensor reports values that are offset from the ground truth. This is might due to environmental, calibration error, and sensor drift.	Data should not be discarded. Un-calibrated data can still provide insight.
Network Centric		
Lost Wireless connection	Connection to the gateway is lost due to interference, noise, low battery, etc.	Loss of data.
Average Packet Delay	A packet arrives late to the final destination due to network congestion.	Delayed or dropped packets
No Internet Connection	Connection to the Internet is lost due to mobility.	No connection to the backend servers.

IV. FAULT MANAGEMENT FRAMEWORK

This section presents our proposed fault management framework based on deep/machine learning. The framework is a continuation of our previous work [22][23] in which a comprehensive telemedicine framework was proposed in different contexts such as the patient's home, outdoor, work, and hospital, as shown in Fig. 2. As was discussed earlier, fault management is concerned with faults that occur in the network, devices, and software. Effective fault management is critical to ensure minimal to no disruption of services provided to healthcare recipients and providers.

Sensors in WBAN transmit three main types of packets, namely, sensor raw data packets, sensor health packets, and route update packets. Sensor raw data packets carry sensed data to the gateway such as temperature, ECG, SpO2, etc. Sensor health packets include data on how well the sensor is performing in the network with regards to radio traffic, battery voltage, Radio Signal Strength Indicator (RSSI), dropped packets, etc. Route update packets include information for routing packets to the gateway.

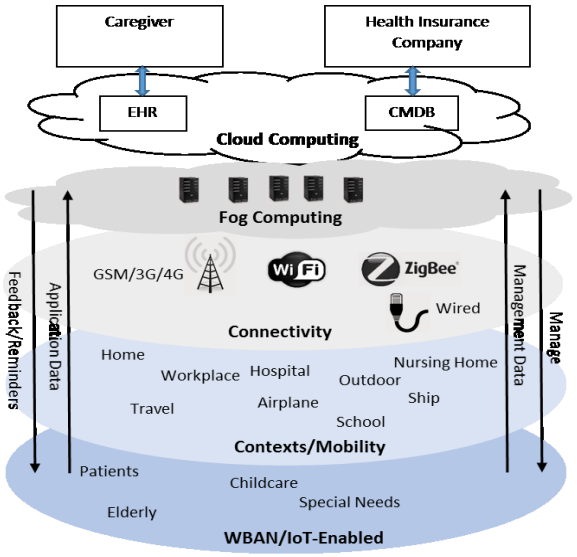

Fig 2. Smart Healthcare Contexts.

A. Use Case Model

In this Use Case, we are concerned with faults that occur in

a WBAN composed of sensors and a gateway. Since medical sensors can deviate from normal behavior due to faults, this could be very critical in healthcare and environmental monitoring applications. In the following, we will elaborate on the use case of detecting faults and alerting stakeholders. We model the fault management subsystem using the UML Use Case Diagrams, as seen in Fig. 3. The use case shows the core functions of the fault management subsystem in collaboration with other external entities and subsystems. This includes analyzing sensor health packets, detecting network emergencies and sensor malfunctioning, and configuring deployed sensors. Actors are external entities that interact directly with the system by providing input and receiving the output. Actors can be software, hardware, or human being. Table III provides descriptors of the various Actors. In Table IV, we present a brief description of the use case along with participating actors.

USE CASE	DETAILS
NAME	Configure Sensors and PDA
ACTORS	Technician
DESCRIPTIONS	After receiving an emergency or a notification, the technician examines the sensors/PDA in the patient site, makes needed fixes, and reports the status
NAME	Collect and Analyze Data (sensor data and sensor health data)
ACTORS	Sensor, Network Manager, caregiver
DESCRIPTIONS	For each implanted sensor, biologically readings are periodically sent to the hospital. The corresponding health data is persisted on the PDA, and a copy is sent to the Network Manager to be persisted and analyzed. In addition, the PDA periodically sends its IP address to the Network Manager for location identification and connectivity checking. In case the IP address was not received within a professionally pre-specified amount of time, the "UC-NM-02 Detect Fault" Use Case is triggered.
NAME	Detect Sensor and PDA Faults
ACTORS	Network Manager, Emergency Unit
DESCRIPTION	Biological data and health packets are examined by an AI algorithm to detect faults and life-threatening emergencies. In case a faulty sensor is detected, a notification/trap is sent to the Network Manager. The notification includes sensor id, type of fault. In case a life-threatening emergency is detected, the PDA sends an emergency notification to the emergency unit.
NAME	Manage Data
ACTORS	PDA
DESCRIPTION	The PDA sends health data to the Network Manager. The network manager persists the health data in the repository (or data center).
NAME	Detect Faults
ACTORS	PDA, Emergency Unit, Technician
DESCRIPTION	Network Manager listens for incoming Traps from PDAs. In case a trap is received, the Network Manager identifies the urgency and type of fault. In case the notification/trap is a Patient Emergency, then the Network Manager contacts the Emergency Unit to dispatch a medical team. In case a faulty sensor (or low-battery) is detected, the Network Manager forwards a request to the technician on duty to examine and fix that sensor. The fault details and status are persisted in the database for future analysis.
NAME	Configure Network (SDN)
ACTORS	PDA
DESCRIPTION	Due to Network changes, new forwarding tables are calculated and sent to the PDAs.
NAME	Analyze Health Data
ACTORS	Timer
DESCRIPTION	Periodically, using an AI Algorithm, accumulated health packets are mined and analyzed to discover traffic patterns on the network. As a result, a reconfiguration of the switches (PDAs) might be initiated.

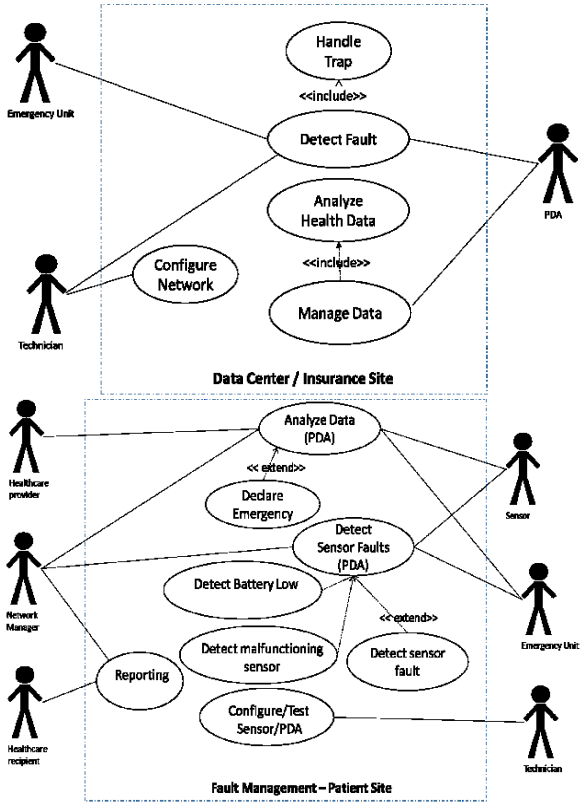

Fig 3. Use case model for data center and patient sites.

TABLE III
USE CASE ACTORS DESCRIPTIONS

ACTOR	DESCRIPTION
NETWORK MANAGER	The software component in the Network Management layer/tier.
PDA	Local Personal Digital Assistant (PDA), which could be a smartphone or a customized device.

SENSOR	The sensor device reads patient biological and environmental data.
HEALTHCARE RECIPIENTS	Elderly and patients who need continuous monitoring.
TECHNICIAN	A trained technical person on PDA configuration and Networking.
EMERGENCY UNIT	24/7 Accident and Emergency department (A&E).
CAREGIVER	A hospital that is assigned to monitor the patient's health condition.

TABLE IV
USE CASE DESCRIPTIONS

USE CASE	DETAILS
NAME	Configure Sensors and PDA
ACTORS	Technician
DESCRIPTIONS	After receiving an emergency or a notification, the technician examines the sensors/PDA in the patient site, makes needed fixes, and reports the status
NAME	Collect and Analyze Data (sensor data and sensor health data)
ACTORS	Sensor, Network Manager, caregiver
DESCRIPTIONS	For each implanted sensor, biologically readings are periodically sent to the hospital. The corresponding health data is persisted on the PDA, and a copy is sent to the Network Manager to be persisted and analyzed. In addition, the PDA periodically sends its IP address to the Network Manager for location identification and connectivity checking. In case the IP address was not received within a professionally pre-specified amount of time, the "UC-NM-02 Detect Fault" Use Case is triggered.
NAME	Detect Sensor and PDA Faults
ACTORS	Network Manager, Emergency Unit
DESCRIPTION	Biological data and health packets are examined by an AI algorithm to detect faults and life-threatening emergencies. In case a faulty sensor is detected, a notification/trap is sent to the Network Manager. The notification includes sensor id, type of fault. In case a life-threatening emergency is detected, the PDA sends an emergency notification to the emergency unit.
NAME	Manage Data
ACTORS	PDA
DESCRIPTION	The PDA sends health data to the Network Manager. The network manager persists the health data in the repository (or data center).
NAME	Detect Faults
ACTORS	PDA, Emergency Unit, Technician
DESCRIPTION	Network Manager listens for incoming Traps from PDAs. In case a trap is received, the Network Manager identifies the urgency and type of fault. In case the notification/trap is a Patient Emergency, then the Network Manager contacts the Emergency Unit to dispatch a medical team. In case a faulty sensor (or low-battery) is detected, the Network Manager forwards a request to the technician on duty to examine and fix that sensor. The fault details and status are persisted in the database for future analysis.
NAME	Configure Network (SDN)
ACTORS	PDA
DESCRIPTION	Due to Network changes, new forwarding tables are calculated and sent to the PDAs.
NAME	Analyze Health Data
ACTORS	Timer
DESCRIPTION	Periodically, using an AI Algorithm, accumulated health packets are mined and analyzed to discover traffic patterns on the network. As a result, a reconfiguration of the switches (PDAs) might be initiated.

B. *Proposed Framework Subsystems and Architecture*

Our proposed fault management framework is composed of hardware and software components distributed on two main sites, namely the patient's site and the data center/repository site. The framework integrates a business model that automates data accumulation, data preprocessing, training process, model deployment process, and fault detection process. Fig. 4**Error! Reference source not found.** presents the distributed subsystems in the framework. The first subsystem is the data center site where data is stored, and AI-based models are managed, while the second subsystem is the PDA application running at the patient's site.

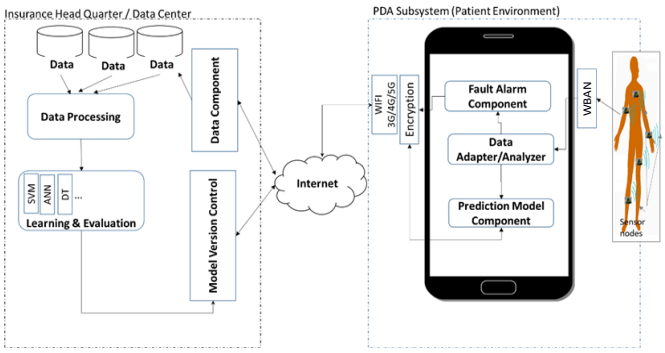

Fig 4. Fault Management Subsystems.

1) *Data Center Subsystem*

At the Data Center, data from patients' sites are accumulated, persisted, and used to generate AI/Deep Learning models to build a fault tolerance system. Fig. 4**Error! Reference source not found.** presents the Data Center subsystem with four components, namely, Data Processing, Learning and Evaluation, Data Component, and Model Version Control (MVC). We focus on two aspects of the fault tolerance system, namely, the training process to predict faults and management and deployment of new updates/releases of prediction models.

a) 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. In training, a data scientist can use different ML techniques to produce a fault tolerance prediction (FTP) model. Fig. 5**Error! Reference source not found.** presents an FTP model captured through UML notation. The model has a simple interface to predict and retrieve metadata and readable labels. The FTP model undergoes an evaluation process to verify its reliability, accuracy, and accomplice to a standard format. Verified FTP models are then deployed to the MVC. In our prototype, we have incorporated three well-known effective techniques in learning, namely, ANN, SVM, and decision trees (DT). Our framework is flexible as more models can be incorporated and deployed later to enhance performance.

b) 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 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.

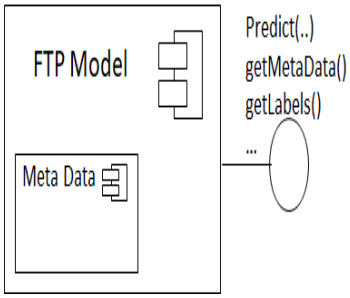

Fig 5. FTP model interface in UML

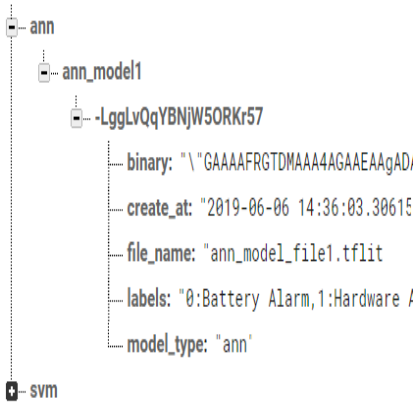

Fig 6. Snapshot of the FTP model format

MVC is a crucial component of the framework. Aside from managing model versions, it notifies PDA components of new updates and releases. Notice that in this framework, we consider different tasks/responses for different faults because learning of one model for all faults leads to a fat model that requires 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 desired high level of reliability of the framework 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 might generate different data formats, and each learning algorithm can have a different model format. Currently, model formats include ProtoBuf (PB) Prototype from google and used in TensorFlow models, Libsvm format, HDF5 format, and WEKA format. The proposed FTP model can hold various model formats because of the attached metadata, as shown in Fig. 6, helps the PDA to interpret, run, and predict faults correctly. In the data center, we can create different models for different faults. The AI/Deep Learning community has used different frameworks and techniques for different applications and contexts. In our prototype, we used two technologies, namely, TensorFlow and Weka, for deep learning (ANN), SVM, and DT. Based on the data complexity, prediction accuracy, and performance, the data scientist decides on the technology and the ML algorithm.

c) Sensor Profiles

Different medical body sensors can be attached to the patient at the same time. Each sensor might use different technology and different means of communication. For this purpose, we introduce the concept of sensor profile which has the specifications and operational thresholds. The sensor profiles are maintained and managed on the data center in the Data Component. For the PDA to work correctly, sensor specifications need to be downloaded, especially when a newly deployed sensor is used or when replacing a sensor with different technology. Fig. 7 shows a snapshot of the sensor profile we use in our prototype. Notice that the profile holds both manufacturing specifications and fault configurations, such as fault thresholds and timing. For example, ft_dead indicates the amount of time in seconds for the PDA to wait until it sends an alert for sensor down the fault. Table V presents the meaning of the proposed sensor profile values. When the configurations in the data center site are changed, the PDA will be notified automatically of the change to download the new profile.

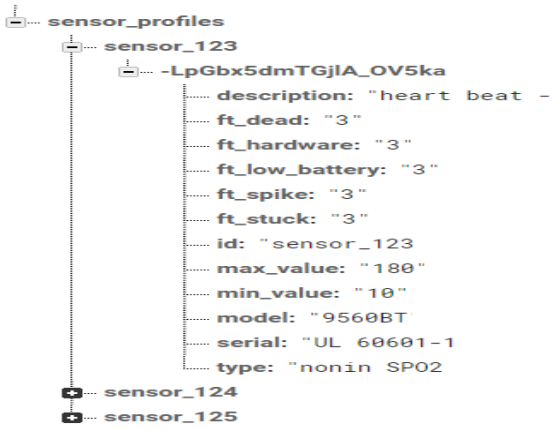

Fig 7. Sensor Profile Snapshot

TABLE V
PROPOSED SENSOR PROFILE ATTRIBUTES DEFINITIONS

CONFIGURATION	DEFINITION
Description	General description of the sensor
Id	A unique Id of the sensor.
Model	Model description of the sensor hardware
Serial	The serial number of the sensor
Type	Type of the sensor
Ft_dead	This is the amount of time in seconds to indicate how long the PDA should wait until it decides there is a sensor down fault.
Ft_hardware	This is the count of consecutive hardware-fault to trigger sensor hardware fault alert, e.g., in Figure 7, the PDA will send an alert of sensor hardware fault if it detects 3 consecutive faults.
Ft_low_battery	This is the count of consecutive low battery faults to trigger sensor low battery fault alert, e.g., in Figure 7, the PDA will send an alert of sensor low battery fault if it detects 3 consecutive faults.
Ft_spike	This is the count of consecutive spiked sensor readings to trigger sensor spike fault alert, e.g., in Figure 7, the PDA will send an alert of sensor spiked fault if it detects 3 consecutive readings higher than a max-values attribute.
Ft_stuck	This is the count of consecutive zero sensor readings to trigger sensor stuck fault alert, e.g., in Figure 7, the PDA will send an alert of sensor stuck fault if it detects 3 consecutive zero readings.
Min-value	This is the minimum value of the sensor reading per the specifications. Values smaller than this value are considered spiked values.
Max-value	This is the maximum value of the sensor reading per the specifications. Values higher than this value is considered spiked values.

The sensor profiles are used in our framework to detect reading faults. It was reported in **Error! Reference source not found.** that using a threshold to isolate and detect faults does not work all the time because of the discrepancies in hardware and other specifications. However, in this framework, we propose to automate this process by building an ML model that automatically generate samples and build a hierarchy of threshold for all sensors. Fig. 8**Error! Reference source not found.** depicts the steps of building such a model. First, we randomly generate sample readings (training and validation) for each sensor based on the sensor's profile. Second, these readings are automatically labeled and passed to an ML technique. Finally, the prediction model is verified using the validation set. If the accuracy of the model is below a threshold (>90%), then we generate more samples and iterate. The prediction model is deployed to be used in the PDA.

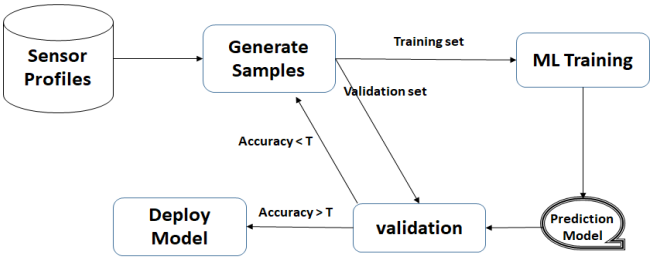

Fig 8. Threshold Based Learning

2) Patient PDA Subsystem

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

a) Data Adapter/Analyzer Component

The Data Adapter/Analyzer (DAA) component is responsible for capturing WPAN 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. For privacy and security concerns, the data must be encrypted during transmission and while residing at the data center. Due to the diversity of WBAN sensors' types of sensed 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. There have been some proposed standards, such as ETSI TS 103 378; however, an adequate, widely used standard is still needed.

Fig. 9 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. Second, Pre-Processor which filters the data such to extract relevant information specific to the task. Third, the Transmitter transmits the data to the data center. Fourth, the Fault Detector which 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 is capable of predicting sensor faults bypassing the pre-processed traffic to a deeply well-trained pool of models managed by the Prediction Model Component.

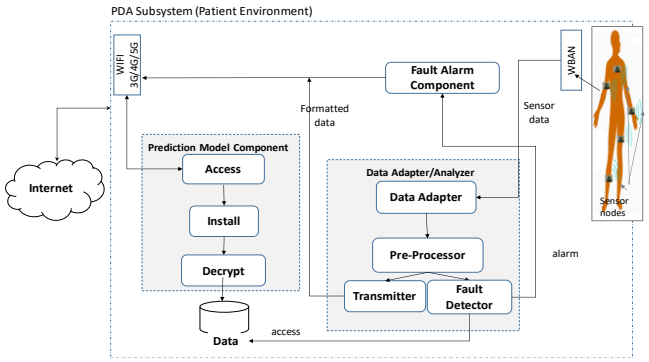

Fig 9. PDA Subsystems in depth

In addition to prediction models, the Fault Detector checks thresholds to detect faults. In our design, we use sensor profiling to make threshold checking very practical and feasible. As explained earlier, on the patient site, e a profile exists that defines each sensor's specifications, which include thresholds. In case 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 keep track of each sensor's last packet information, as seen in Table VI.

TABLE VI
DATA STRUCTURES USED IN THE PROTOTYPE

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

Table VII 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 blocks inline 5 until a new packet arrives—lines 6-7 update data structures. In Table VIII, each packet is passed to the function "predict_fault." The algorithm is designed based on fault sudden causes, though the investigation of progressive degradation might be worth exploring for future work. Lines 8-9 buffer the arriving packets and flush the buffer once full. Lines 10-12 starts the asynchronous call (separate thread of execution) to detect each fault separately. For example, Line 10 starts a thread of execution to detect whether a sensor reading was beyond the min-max range or not. The number of consecutive spiked values received for a given sensor is recorded in the spike_dict in Line 7, so the algorithm in IX will read the latest recorded value and compare it with the profile threshold. In case there was a violation of the threshold, an alert is sent to the data center, see Line 8 in Table IX. The thread after that waits for a given amount of time until the next check, see Line 11 in Table IX. Notice that the designer can start separate threads for each fault, or she can combine them in one thread.

TABLE VII
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
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
11. Start Asynch* check_fault(hb_dict, Sensor-Down) // See Table
12. Start Asynch* check_fault (ft_dict, Low-Battery) // See Table
13. END-DO

TABLE VIII
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]++
6. … 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 IX
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

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 issue an 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 send an 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, for example.

b) Fault Alarm Component

The Fault Alarm Component (FAC) notifies the data center of any sensor faults. It repeats sending the fault notification/trap

until the issue is resolved. FAC alerts the patient or the attendant through either voice, SMS text, or both, as in our porotype, about the fault.

c) Prediction Model Component

The Prediction Model Component (PMC) retrieves the latest releases of prediction models and persists them locally on the PDA local storage. Fig. **Error! Reference source not found.**9 shows the different sub-components of PMC. Based on the patient's profile, the Access sub-component retrieves the prediction models needed to monitor the patient. The Access sub-component checks 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. In our prototype, we utilized built-in asynchronous listeners, which are triggered when new releases of the prediction models are generated on the data center site. The prototype implemented a model validation process to verify proper installation and functionality of the prediction model 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 sensors profiles that contain specifications of each sensor used on the patient's site, as was shown in

Table

V. Experiments and Analysis

This section presents experiments conducted in developing fault tolerance prediction models. First, we elaborate on the data set used in training, then the experimental setup is explained before evaluating the prediction models.

A. Data Sets and Experiments Setup

A WBAN consisting of 15 sensors was deployed in the Lab on a human-like topology. Two kinds of packets, namely, sensor control packets and sensor data packets, were collected for three months. Sensor control packets are regularly transmitted from each sensor node to the base station to provide status information on the sensor node performance and connectivity. On the other hand, sensor data packets contain the actual sensor readings (temperature and light intensity). Collected data was preprocessed by applying aggregations, summarization, and transformations. We examined the sensor control packets manually and identified two types of faults, namely, battery fault and hardware faults. Table X**Error! Reference source not found.** and Table XI**Error! Reference source not found.** present a snapshot of the dataset and the distribution of the faults. Collected data indicated how each sensor might have a different battery lifespan and Mean Time Between Failures (MTBF). This kind of asymmetry happens due to the position, quality of transmission/reception, and role of the sensor if it is relaying (router) or relayed (end device) sensor.

TABLE X
DATASET SNAPSHOT AND SUMMARY (CSV FORMAT)

Pkt	forwarded	dropped	retries	battery	Quality tx	Quality rx	Parent rssi(dBm)	Label
1	0	3	930	2.5	15	15	-67	battery alarm
2	0	3	932	2.5	15	15	-67	battery alarm
3	0	3	936	2.5	15	15	-68	battery alarm
4	7994	7	853	2.8	15	13	-64	normal data
5	7994	7	856	2.8	15	15	-64	normal data
…	1026	241	1516	2.8	7	12	-95	hardware alarm

TABLE XI
DATASET SUMMARY

Fault/Label,type	Percentage/count
Battery faults	61%
Hardware faults	8%
Normal	31%
Total Data	29603

We constructed several models to detect faults using ANN, SVM, and Decision Trees using TensorFlow, LIBSVM, and Weka libraries, respectively. The setup of training the models is presented in Table XII. In all models, the accuracy of the model was verified using a 10-fold-cross-validation technique.

TABLE XII
MODELS' PARAMETERS SETUP

ANN TensorFlow	SVM LIBSVM	DT Weka (J48)
Layers=3	C=1	Binary,Split=true
Nodes/Layer=32,2,3	Kernel=RBF	Unpruned=true
Activation=RELU, Softmax	Type=1-vs-1	Confidence=0.55
Epoch=150	COEF0=0	
Evaluation=10-fold CV	Gamma=0.15	

B. Results and Analysis

This section reports the results and analysis of conducted experiments. Standard cross-validation with 10 folds was used to report the accuracy of the used models. Table **Error! Reference source not found.**XIII presents the prediction accuracy in terms of precision, recall, F1-score, and support of ANN. Table XIV present presents the confusion matrix. The average fold accuracy of fault detection was 96%, with a false-positive rate of 6%.

TABLE XIII
ANN ACCURACY RESULTS
Accuracy Measurement

Fault	Precision	Recall	F1-Score	Support
Battery	0.99	1	1	623
Hardware	0.85	1	0.98	291
Normal	1	0.96	0.98	386

TABLE XIV
ANN CONFUSION MATRIX
Confusion Matrix

Real Faults	Predicted Faults		
Fault	Battery	Hardware	Normal
Battery	1793	9	25
Hardware	5	285	6
Normal	11	44	783

TABLE XV

SVM ACCURACY RESULTS

Accuracy Measurement				
Fault	Precision	Recall	F1-Score	Support
Battery	0.91	0.99	0.95	1829
Hardware	0.96	0.83	0.89	317
9Normal	0.95	0.82	0.88	815

TABLE XVI
SVM CONFUSION MATRIX

Confusion Matrix				
	Predicted Faults			
Real Faults	Fault	Battery	Hardware	Normal
	Battery	1812	1	16
	Hardware	39	262	16
	Normal	134	11	670

Table XV and Table XVI presents the results for SVM and the confusion matrix. The average fold accuracy of fault detection was 92%, with a false-positive rate of 17%. ANN outperforms SVM in terms of accuracy. However, the latter's training time was less.

TABLE XVII
DECISION TREE ACCURACY RESULTS

Accuracy Measurement				
Fault	Precision	Recall	F1-Score	Support
Battery	1	1	1	1
Hardware	1	1	1	1
Normal	1	1	1	1

TABLE XVIII
DECISION TREE CONFUSION MATRIX

Confusion Matrix				
	Predicted Faults			
Real Faults	Fault	Battery	Hardware	Normal
	Battery	1839	0	0
	Hardware	0	307	0
	Normal	0	0	805

Table XVII and Table XVIII show the results of DTs training and the confusion matrix. Surprisingly, DT outperformed both ANN and SVM with 100% accuracy.

The results of our machine learning approach to detecting faults through threshold checking are presented next. As was explained before, sensor readings can be checked against thresholds to determine their validity. However, when dealing with a large number of sensors where each sensor has its own threshold and hardware specifics, using a data mining technique such as DT and SVM, to represent and manage threshold checking becomes needed. To show this, the BIDMC-PPG and respiration data set [24] were used. 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 XIX. The data set was composed of 100,000 readings, where 93.4% of the readings were considered normal, and 6.6% were deemed abnormal. Synthesis data was used to emulate real scenarios assuming 500 sensors are used to measure different biological data. For each sensor, labeled reading was generated for training in DT and SVM. The distribution of the generated data was 46% as normal readings and 54% as faulty. Table XX 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 XXI 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 dataset, sensor reading and labels were done based on sensor profiling. Two learning techniques, namely, DT and SVM, were tested, and the results are discussed below.

TABLE XIX
SNAPSHOT OF BDIMC PPG AND RESPIRATION DATA SETS

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

TABLE XX
SNAPSHOTS OF SYNTHESIS DATA SET- A

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
383	11.59	Normal
215	4.896	Fault
….	….	….

TABLE XXI
SNAPSHOTS OF SYNTHESIS DATA SET WITH CONDITIONS

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
95	-0.4	Normal	Fault
82	3.79	After Surgery	Fault

The results of the BIDMCC-PPG dataset are presented in Table XXII. The 10-fold cross-validation accuracy is 99% for both DT and SVM. As expected, the average training time is longer in the case of SVM.

TABLE XXII
BIDMCC-PPG RESULTS

Prediction	**Decision Tree**	**SVM**
Accuracy	99%	99%
Training Time (seconds)	3.103	311.39

The results of the first synthesis dataset in Table XX is depicted in Fig. 10**Error! Reference source not found.**, and Fig. 11**Error! Reference source not found.**. Fig. 10 **Error! Reference source not found.**presents the change of accuracy of DT and SVM when we change the number of accumulated readings per sensor, N. The focus here is to achieve higher accuracy with a lower number of training data. Both techniques performed with high accuracy; however, SVM was less affected by the size of the dataset, while DT performed well when N was greater than 80. For example, when N=20 (data set size = 20*500), DT achieved 78% accuracy while SVM achieved 90%. Fig. 10 clearly shows that accuracy is proportional to the number of readings per sensor for DT. However, such proportionality is inferior to SVM. 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. In terms of training time, the DT outperforms SVM.

Fig. 11**Error! Reference source not found.** depicts the change of training time with the size of the dataset. As expected, DT is trained faster than SVM, even if the size of the data set is large.

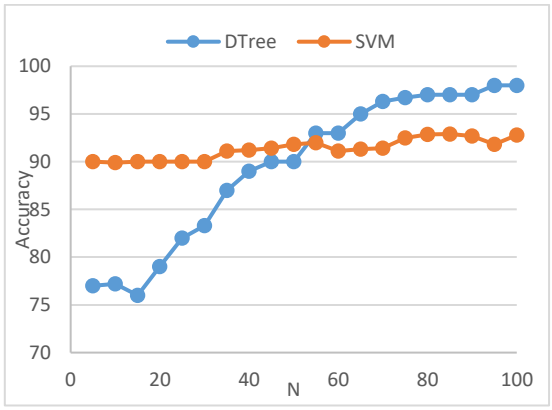

Fig 10. Accuracy of decision tree and SVM (1st synthesis dataset)

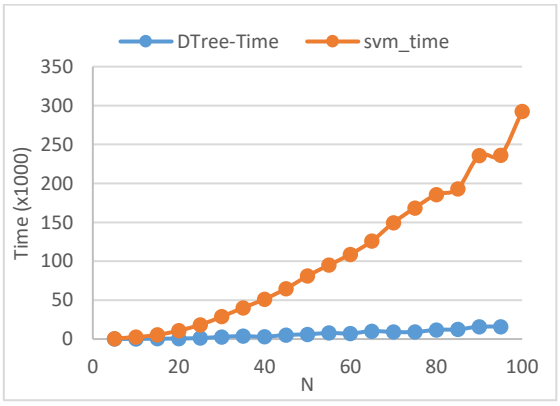

Fig 11. Training Time of Decision Tree and SVM (1st synthesis dataset)

The results of the second synthesis data, as in Table XXI is depicted in Fig. 12, and Fig. 13**Error! Reference source not found.**. In terms of accuracy, Fig. 12 shows high accuracy (> 90%) when the number of samples per sensor is greater than 50. The accuracy of the DT is higher than SVM, especially when the number of samples per sensor increases. In both techniques, the accuracy is lower than the accuracy reported in Fig. 10, which might be related to the complexity of the dataset in the second scenario. Fig. 13 shows the results of increasing the dataset (number of samples per sensor) on training time. As expected, the training time of SVM is larger than the DT due to the model complexity. In both scenarios, the prediction models are reliable to be deployed and used in fault prediction.

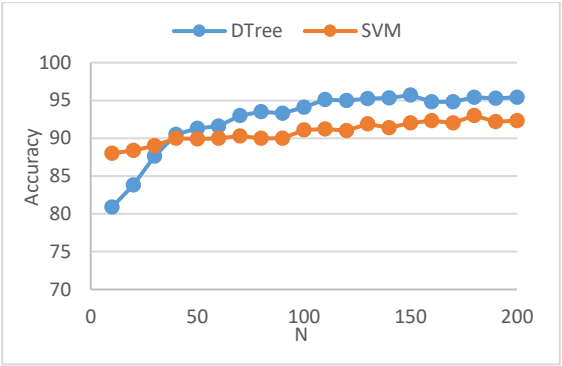

Fig 12. Accuracy of Decision Tree and SVM (2nd synthesis dataset)

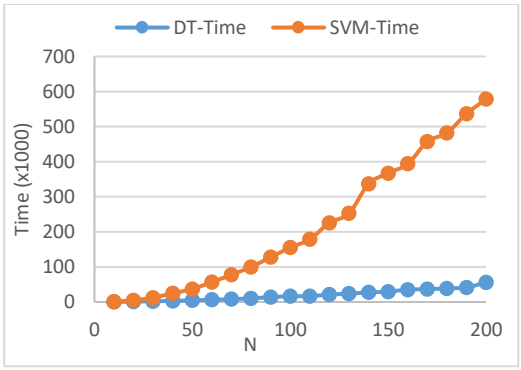

Fig 13. Training Time of Decision Tree and SVM (2nd synthesis dataset)

C. *PDA Prototype*

As proof of concept, a distributed component was developed, as shown in Fig. 4**Error! Reference source not found.**. An android mobile application was developed to manage the data and faults on the patient's side. Prediction Models are assumed to be trained offline and can be 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 FireBase cloud data based on which the mobile application can download and use in fault detection. In Fig. 14**Error! Reference source not found.**, the training process can be conducted using SVM, ANN, or DT. The data engineer configures the training process and launches the training process. Validation is conducted either by applying 10-Cross-validation or by providing a validation dataset. After that, the model can be deployed to the cloud, FireBase DB, see Fig. 15**Error! Reference source not found.**.

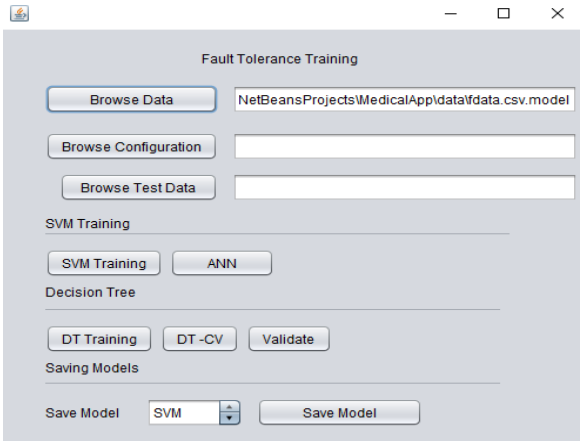

Fig 14. Training Model Gadget

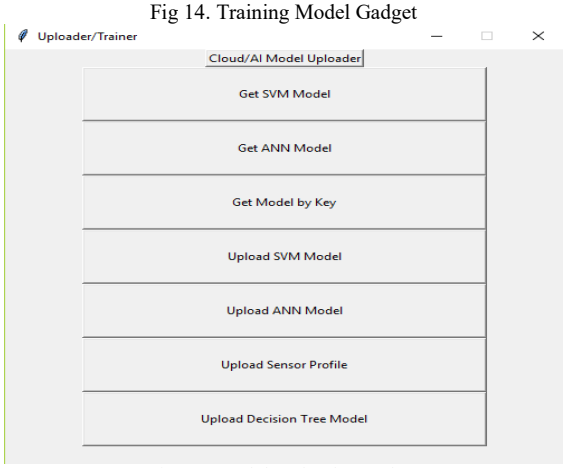

Fig 15. Model Uploader Gadget

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 kinds of sensor faults along with three types of software faults, as shown in Table XXIII. The design is adaptable to include more faults and predictors with small modifications. The application is also capable of automatically update its current prediction models. We have deployed the prediction models presented in the results section on the mobile prototype. Fig. 16 depicts three functional screenshots of the prototype, namely, downloading/retrieving prediction models from the cloud, detecting battery fault, and capability of data management (sensors profiles and prediction models). All the gadgets projects, including the mobile prototype, are available online [25].

TABLE XXIII
LIST OF DETECTED FAULTS IN OUR PROTOTYPE

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

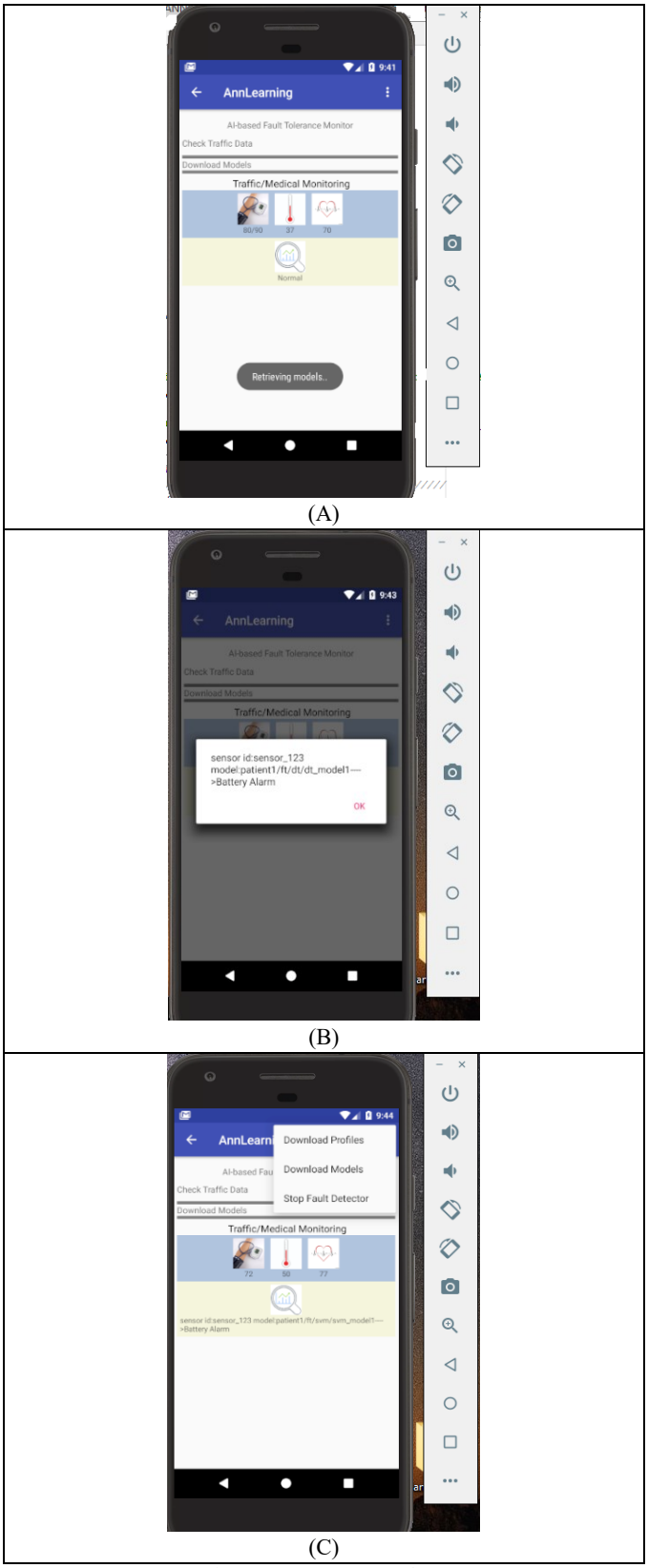

Fig 16. Prototype screenshots

D. *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 some time delay due to data accumulation and training processes.

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 the data scientist can use the framework. The prototype GUI is simple with minimal input 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 **Error! Reference source not found.**, and we assume a customized PDA where unnecessary services and applications can be shut down. The current hardware/storage technologies are sufficient for processing and persistence of traffic on the PDA. Traffic is buffered and forwarded to the edge/cloud site. Although connections are very reliable, in case of communication interruption, the PDA can hold buffered traffic or forward the traffic to a local base station that serves as a backup node. 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, in terms of reliability, we have addressed software and hardware failures for the sensors, PDA, and software components, see Table XXIII. The Simple Network Management Protocol (SNMP) is used to alert the Edge/Cloud side. Table XXIV presents a summary of the strengths and weaknesses of our framework.

TABLE XXIV
FRAMEWORK EVALUATION

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

VI. 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 sensors through profiling.

We investigated the use of three ML algorithms in fault prediction. We trained SVM, DT, and ANN on traffic data to predict battery and hardware faults. We presented the results in terms of accuracy and confusion matrix. Additionally, we proposed a threshold-based detection approach using ML and presented high accuracy results using DT and SVM for both the biomedical dataset and synthesis dataset.

As proof of concept, we developed and presented a prototype for the training process, model deployment, and model prediction. We presented a mobile application that can detect several sensor and software faults.

We plan to extend our future research in the following directions. The first is to explore the software side of the WBAN applications to examine additional software faults and techniques. For example, progressive degradation of fault discovery and PDA resource usage is worth investigating. The second will be to incorporate security and data privacy aspects on the PDA. Finally will be to conduct larger-scale management of prediction models on the data center to uncover additional faults.

Funding: This research was funded by the United Arab Emirates University, grant number 31T078.

REFERENCES

[1] M. Salayma, A. Al-Dubai, I. Romdhani, and Y. Nasser, "Wireless Body Area Network (WBAN) A Survey on Reliability, Fault Tolerance, and Technologies Coexistence," *ACM Comput. Surv.*, vol. 50, no. 1, pp. 1–38, 2017.

[2] C. Academy, *Network Basics Companion Guide, Video Enhanced Edition*. Cisco Press, 2013.

[3] D. M. Barakah and M. Ammad-uddin, "A survey of challenges and applications of wireless body area network (WBAN) and role of a virtual doctor server in existing architecture," in *2012 Third International Conference on Intelligent Systems Modelling and Simulation*, 2012, pp. 214–219.

[4] R. Zhang, Z. Peng, L. Wu, B. Yao, and Y. Guan, "Fault diagnosis from raw sensor data using deep neural networks considering temporal coherence," *Sensors*, vol. 17, no. 3, p. 549, 2017.

[5] A. B. Sharma, L. Golubchik, and R. Govindan, "Sensor faults: Detection methods and prevalence in real-world datasets," *ACM Trans. Sens. Networks*, vol. 6, no. 3, pp. 1–39, 2010.

[6] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—Part I: Fault diagnosis with model-based and signal-based approaches,"

IEEE Trans. Ind. Electron., vol. 62, no. 6, pp. 3757–3767, 2015.

[7] S. Zhou, K.-J. Lin, J. Na, C.-C. Chuang, and C.-S. Shih, "Supporting service adaptation in fault tolerant internet of things," in *2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA)*, 2015, pp. 65–72.

[8] T. N. Gia, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fault tolerant and scalable IoT-based architecture for health monitoring," in *2015 IEEE Sensors Applications Symposium (SAS)*, 2015, pp. 1–6.

[9] Y. Yang, Q. Liu, Z. Gao, X. Qiu, and L. Meng, "Data fault detection in medical sensor networks," *Sensors*, vol. 15, no. 3, pp. 6066–6090, 2015.

[10] T. Zhang, Q. Zhao, and Y. Nakamoto, "Faulty sensor data detection in wireless sensor networks using logistical regression," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2017, pp. 13–18.

[11] M. Hassanalieragh *et al.*, "Health monitoring and management using Internet-of-Things (IoT) sensing with cloud-based processing: Opportunities and challenges," in *2015 IEEE International Conference on Services Computing*, 2015, pp. 285–292.

[12] X. Li, H. Ji, and Y. Li, "Layered fault management scheme for end-to-end transmission in internet of things," *Mob. Networks Appl.*, vol. 18, no. 2, pp. 195–205, 2013.

[13] Y.-S. Jeong, H.-W. Kim, and J. H. Park, "Visual scheme monitoring of sensors for fault tolerance on wireless body area networks with cloud service infrastructure," *Int. J. Distrib. Sens. Networks*, vol. 10, no. 4, p. 154180, 2014.

[14] A. Mahapatro, "Online fault detection and recovery in body sensor networks," in *2011 World Congress on Information and Communication Technologies*, 2011, pp. 407–412.

[15] C. Yi and J. Cai, "Transmission management of delay-sensitive medical packets in beyond wireless body area networks: A queueing game approach," *IEEE Trans. Mob. Comput.*, vol. 17, no. 9, pp. 2209–2222, 2018.

[16] X. Li, Y. Deng, and L. Ding, "Study on precision agriculture monitoring framework based on WSN," in *2008 2nd International Conference on Anti-counterfeiting, Security and Identification*, 2008, pp. 182–185.

[17] A. Sawand, S. Djahel, Z. Zhang, and F. Naït-Abdesselam, "Toward energy-efficient and trustworthy eHealth monitoring system," *China Commun.*, vol. 12, no. 1, pp. 46–65, 2015.

[18] Y. Liao, Q. Yu, Y. Han, and M. S. Leeson, "Relay-Enabled Task Offloading Management for Wireless Body Area Networks," *Appl. Sci.*, vol. 8, no. 8, p. 1409, 2018.

[19] M. Elliott and A. Coventry, "Critical care: The eight vital signs of patient monitoring," *Br. J. Nurs.*, vol. 21, no. 10, pp. 621–625, 2012.

[20] K. M. K. Raghunath and N. Rengarajan, "Investigation of faults, errors and failures in wireless sensor network: A systematical survey," *Int. J. Adv. Comput. Res.*, vol. 3, no. 3, p. 151, 2013.

[21] M. M. Alam and E. Ben Hamida, "Interference mitigation and coexistence strategies in IEEE 802.15. 6 based wearable body-to-body networks," in *International Conference on Cognitive Radio Oriented Wireless Networks*, 2015, pp. 665–677.

[22] F. Sallabi, F. Naeem, M. Awad, and K. Shuaib, "Managing IoT-Based Smart Healthcare Systems Traffic with Software Defined Networks," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, 2018, pp. 1–6.

[23] M. Al Thawadi, F. Sallabi, M. Awad, and K. Shuaib, "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)*, 2019, pp. 397–403.

[24] M. A. F. Pimentel *et al.*, "Toward a robust estimation of respiratory rate from pulse oximeters," *IEEE Trans. Biomed. Eng.*, vol. 64, no. 8, pp. 1914–1923, 2016.

[25] "Github.com Available online." [Online]. Available: https://github.com/mxawad2000/WBAN-Mobile-Prototype.