

Article

# Hybrid Continuous Density Hmm-Based Ensemble Neural Networks for Sensor Fault Detection and Classification in Wireless Sensor Network

Malathy Emperuman <sup>1</sup> and Srimathi Chandrasekaran <sup>2,\*</sup>

<sup>1</sup> School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu 632014, India; e.malathy@vit.ac.in

<sup>2</sup> School of Computing Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu 632014, India

\* Correspondence: csrimathi@vit.ac.in

Received: 2 December 2019; Accepted: 23 January 2020; Published: 29 January 2020



**Abstract:** Sensor devices in wireless sensor networks are vulnerable to faults during their operation in unmonitored and hazardous environments. Though various methods have been proposed by researchers to detect sensor faults, only very few research studies have reported on capturing the dynamics of the inherent states in sensor data during fault occurrence. The continuous density hidden Markov model (CDHMM) is proposed in this research to determine the dynamics of the state transitions due to fault occurrence, while neural networks are utilized to classify the faults based on the state transition probability density generated by the CDHMM. Therefore, this paper focuses on the fault detection and classification using the hybridization of CDHMM and various neural networks (NNs), namely the learning vector quantization, probabilistic neural network, adaptive probabilistic neural network, and radial basis function. The hybrid models of each NN are used for the classification of sensor faults, namely bias, drift, random, and spike. The proposed methods are evaluated using four performance metrics which includes detection accuracy, false positive rate, F1-score, and the Matthews correlation coefficient. The simulation results show that the learning vector quantization NN classifier outperforms the detection accuracy rate when compared to the other classifiers. In addition, an ensemble NN framework based on the hybrid CDHMM classifier is built with majority voting scheme for decision making and classification. The results of the hybrid CDHMM ensemble classifiers clearly indicates the efficacy of the proposed scheme in capturing the dynamics of change of states which is the vital aspect in determining rapidly-evolving instant faults that occur in wireless sensor networks.

**Keywords:** wireless sensor network; sensor faults; neural networks; classification of faults; hidden Markov model

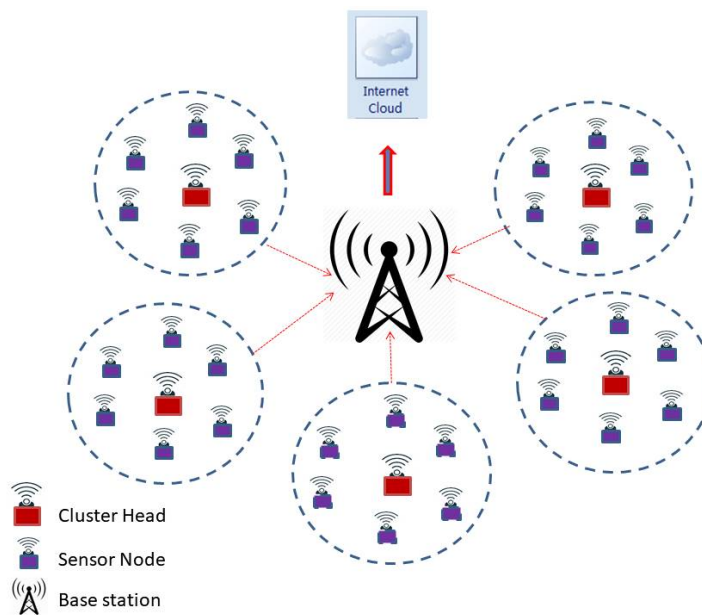
## 1. Introduction

Wireless sensor networks (WSN) are widely deployed in various applications such as environmental monitoring, transportation, health care and industrial automation [1]. These applications utilize large numbers of wireless sensor nodes to measure the parameters of interest with energy and computational constraints [2]. Recent sensor networks have the capability to handle huge data like image or visual sensing, which is in turn integrated with the cloud to enhance the capability of the internet of things (IOT) [3]. Furthermore, the WSN has been implemented in mission critical applications like body sensor networks, security, space, underwater communication, and military [4,5]. Often, critical sensor networks are installed in harsh, inaccessible, and uncontrolled environments, which may affect the sensor performance and cannot be attended immediately. Normally, sensors get affected due to the

surrounding environment and various malfunctions within the sensor nodes [6,7]. Hence, faults in sensor nodes can create substantial loss of network failures and may even affect the economy of the concerned application [8]. Furthermore, fault data also affects the bandwidth and energy consumption of the network. Therefore, sensor fault detection is an important process, and it is essential for all wireless sensor networks [9]. Fault detection and diagnosis can improve the reliability of sensor networks and enhance its bandwidth of operation. There are various techniques as reported in recent research surveys related to fault detection and diagnosis in sensor networks [10–13]. Incidentally, it is worth noting that while choosing fault detection methods, care should be taken to reduce the energy consumption and ease of integration with the existing networks. On the other hand, the quality of service of the fault detection should not be compromised to ensure the effective detection of faults. Upon successful detection or diagnosis of faults, the node can be removed or inactivated. Hence, the data from other nodes are re-routed with alternative nodes in order to sustain the network life.

Depending on the appropriate application and domain of sensor network system, each sensor node can contain single or multiple sensors with supporting processor and wireless communication systems [14]. Faults may occur either in the hardware or software of a node, and they are classified as hard and soft faults. The hard faults represent the complete damage of hardware of the sensor whereby the component/device is unable to communicate with other devices. In soft faults, on the other hand, the node is in active mode but communicates erroneous readings to the central or neighboring nodes. Often, the node get damaged at various levels and stages such as during sensing, processing, storing, or actuation. The general sources of failure may be attributed to calibration error, hardware failure, hostile environment, very low battery, and link failure [8]. Based on the timing, faults are divided into three major categories, namely permanent, intermittent, and transient. Among these, transient faults are difficult to detect since the occurrence is only for a brief period i.e., on the millisecond to nanosecond scale. Transient faults are further subdivided as random faults and spike faults. These faults are particularly important in mission critical sensor networks. As an example, in the body sensor network, the data from the sensors that are installed internally and externally to the human body are very critical in deciding the health condition of the concerned human being [15]. Therefore, it is evident that the detection of faults in these types of the sensor networks with high accuracy is currently very important.

A typical WSN that uses clustering-based data communication with the base station is shown in Figure 1 and the same structure of the network is taken up for research in this study and analysis. Here, we propose the continuous density hidden Markov model (CDHMM), which is well studied in several mechanical and electrical applications. However, the research work based on CDHMM in WSN has not yet been studied in detail for fault detection since there was no requirement. Recently, in the critical wireless sensor network and more specifically in health monitoring systems, such implementations have started receiving specific focus and attention from the research community under the WSN category. For these networks, very low level faults and more so transient responses are required to be detected at the nascent stages itself to ensure the reliability of network. Hence, this proposed works aims at assessing the dynamics of the sensor fault based on the optimized hidden state transitions, which is in fact reflective of the inherent changing dynamics of the faults in the sensors. In addition, hence, a comprehensive diagnostic system has been taken up in this study which utilizes a hybrid CDHMM-Neural Network implementation wherein the dynamics of the fault is obtained from HMM while the classification of the faults is carried out by ensemble NN systems.



**Figure 1.** A typical wireless sensor networks (WSN) uses clustering technique for data communication from nodes to cloud.

### 1.1. Objective and Problem Statement

The main objective of the paper is to detect the permanent, intermittent, and transient sensor faults with better accuracy, as compared to existing techniques which suffer from the limitation of poor recognition due to the dynamic nature of faults. Therefore, a hybrid CDHMM with an ensemble of NNs, namely learning vector quantization (LVQ2), probabilistic neural network (PNN), adaptive probabilistic neural network (APNN) and radial basis function neural network (RBF) to detect and classify various faults in sensor node has been implemented. Further, to arrive at a credible and unbiased decision on the category of faults, a majority voting scheme is proposed.

### 1.2. Contributions

Unique contributions in this research study are summarized as follows:

1. Hybridization of continuous density HMM with NNs is the first of its kind proposed for the detection and classification of sensor faults in a WSN.
2. CDHMM is specifically used to determine the state transition of sensor data during faults. Based on the detection of such transitions, classification of faults and percentage of fault data, even the prediction of a sensor lifetime can be carried out more precisely.
3. When CDHMM is combined with NNs, the classification efficiency can be increased; even a low level of fault that occurs in the sensor data can exhibit a considerable change in the state transition. Besides, this can also detect the mixture of faults for different percentages of fault rates.

The second section focuses on the related works on sensor faults in WSN, with a detail discussion on the limitations in each technique. The third section introduces the continuous density HMM followed by various neural networks. The fourth section proposes the framework for fault detection and classification of different sensor faults. The fifth section discusses the results and analysis of various algorithms for different types of faults with varying fault rate percentage.

## 2. Related Works

Fault detection techniques have been first proposed for identifying machine and control system faults. The various faults that are studied in the machines are bias, drift, gain, precision degradation, complete failure, and constant output with noise. These faults are entirely different from those of

faults that occur in the wireless sensor network [16]. Over the past few decades, artificial neural networks, support vector machine and k-nearest neighbor have been used for fault detection in machine applications [17]. These techniques have been successfully implemented for rolling fault detection and other machine control applications. However, it is evident that sensor fault detection necessitates complex techniques which are capable of including the dynamics of the network. Thus, the fault detection techniques which have been used to detect faults in machines are modified according to the requirement in sensor networks and implemented for the diagnosis of faults in the WSN.

The two important categories of faults in WSN are data centric and system faults. Data centric faults include spike, outlier, stuck-at, high noise, or variance. On the other hand, system view faults are namely calibration, connection or hardware, low battery, environment out of range, and clipping. These faults are elaborately discussed along with fault duration, pattern, and their impact on the sensor data [8]. Fault detection in WSN is basically classified as centralized, distributed, and hybrid approaches [18]. In the centralized approach, a computationally powerful node with large memory capacity with uninterrupted power supply is used for the fault detection and classification in the network. The implementation of this approach is complex but the accuracy of the detection is high when compared to distributed and hybrid approaches. In the centralized approach, [19] a proposed machine learning technique like the naive Bayes framework is used to find out the end to end transmission delay to determine the hardware faults in sensor nodes. Likewise, gradient based forwarding protocol is proposed to determine the node failures due to high data rates [20]. Similarly, Support Vector Machine (SVM) learning is reported in distributed learning in WSN and it has demonstrated the capability to exhibit better performance while scaling the network [21]. The SVM technique has also been used in the distributed technique for the classification and detection of sensor faults [22]. The SVM is also observed to adapt based on various kernel functions for nonlinear classification. SVM has been used to define a decision function and implement in the cluster head to detect the fault sensor. This approach has been used only to classify the fault or non-fault sensor, not the types or dynamics of the fault.

In [23], authors proposed a simple technique to detect the fault sensor. Here, a threshold value was used to distinguish between the false and true data. This approach cannot be utilized for mission critical network, since this simple method may result in misclassification for low level of faults. In [24], used a belief function-based decision fusion method for detecting faults in the WSN. Four classification techniques are proposed to enhance the performance of the belief function fusion approach. This method cannot capture the dynamics of the individual sensor faults. In [25], a distributed algorithm was proposed that considered response time and detection accuracy in fault detection in addition to the required quality of service (QoS). The Bayesian model for classification and Pareto optimization to attain the required QoS was studied. The reduced data accuracy with this method restricted the use in critical sensor networks. Authors in [26], proposed an automated fault detection method using feed forward NNs which were trained by the hybrid Meta heuristic algorithm. This model was not suitable for instant or transient fault diagnosis since the fault state transition could not be captured with the NNs alone. In [27], used a semi-supervised method for the fault detection in WSN using pattern classification. As an estimator, posterior probability was used to find the normal or faulty nodes. Both online and offline fault detection could be performed with the proposed method. However, data accuracy with this approach may not meet the requirements of the critical sensor network.

In continuation to the distributed approach, [28] proposed the Neyman–Pearson test to detect the faults such as random faults, stuck at non-zero, stuck at one and stuck at zero in the sensor nodes. The faults of a sensor can be detected based on the neighboring sensor node data. Thus, the accuracy of the fault detection depends on the characteristics of neighbor nodes. Therefore, this technique results in error if the neighbor nodes get affected by faults. In [29], very simple statistical methods such as the Kuiper test and the autoregressive model are used for detection of sensor health. These statistical methods require a larger number of neighboring nodes data to determine the faulty

sensor node. Likewise, [30] proposed the directional diagnosis method to detect the faults in the network or node. The detection was done with high accuracy by repeatable information probing and reasoning of fault. The authors in [31–33], proposed probability-based fault detection approach which used the transmission time of the packet from the sensor node, the incremental probe scheme, and the probabilistic inference model. Data aggregation used the Markov chain controller model, and pattern classification used a neighbor hidden conditional random field method to detect the fault sensor node. These methods, however, could not find the number of faults and types of faults in the network.

In [34], proposed a novel method for fault detection using a sequence-based detection method. They used the Fletcher checksum algorithm and path planning to decrease the distance taken by the data to reach its destination. The network failure could be determined using the continuous path changes and also by using control messages to find the health of the concerned node. Further, comparison-based approaches have been proposed by [35,36], which pertain to the distributed fault detection method. This is implemented by neighbor sensor correlation and reactive fault detection using temporal and spatial correlation. These approaches do not function when the node starts to move. They are primarily based on the neighbor node data, and if the neighbor sensor moves away from the range, then the detection gives an error reading.

In [37], fault sensor data detection is done in the cloud using clustering of WSN which increases the fast detection of error in sensor data and improves the detection accuracy. This approach is scalable to any topology of networks, and the detection is focussed within the spatial and temporal data sets. Therefore, the time it takes for detection of any amount of error can be drastically reduced. In [38], the detection of a sensor fault is done by the distributed approach with clustering of nodes. Determination of sensor fault is done by verifying the table associated with the neighbor sensor nodes. Also, the false alarms are mitigated by using time redundancy and therefore the instant false does not create a false alarm. The cluster-based method depends more on the efficiency of the cluster formation.

The soft computing method proposed by [39] reported the recurrent neural network to detect the faults using the neighbor's data. [40] used principal component analysis with the neural network to determine the soft fault. [41] reported the fault detection using fuzzy inference system with its neighbor to detect the transient and permanent faults. These methods have drawbacks when the nodes are dynamic, resulting in inaccurate low level fault detection, and increased scalability of the network.

The centralized approaches have good detection accuracy compared to the distributed technique, but the implementation of the centralized system needs more computation power. In order to utilize both merits, a hybrid approach is proposed where the algorithm is implemented both in the cluster node and in each sensor node. In [42], the authors reported a hybrid distributed fault detection method using a mobile sink. It detects the fault in both the hardware and software of the sensor network. The detection of fault is accomplished by the cluster head, and it updates the status of each node in the network. This information is further used by the administrator to find out the fault in the sensor node. The main drawback of this method is the delay in detection due to the error in the path planning that determines the performance of the algorithm. The above-mentioned limitations can be overcome by proposing a hybrid method where the state transition of the fault data is captured by the hidden Markov model and the classification is carried out by neural networks.

### 3. Proposed Approach

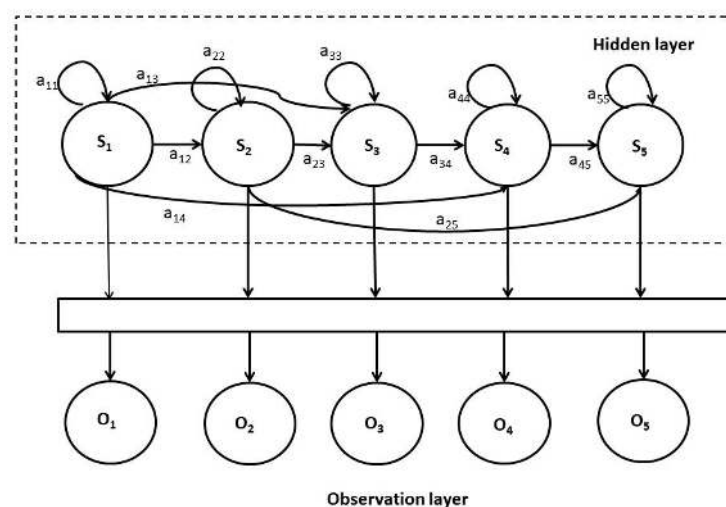
In this section, we propose a centralized approach, where the HMM is hybridized with NNs, and is implemented in each cluster head of the networked sensors as shown in Figure 1. Although the energy and computation requirement for the proposed method is high, the requirement of low level fault detection in the critical sensor network is indispensable. Further, the recent sensor nodes are computationally powerful with less energy consumption. In this work, we detect only the individual sensor faults rather than the network faults. However, the same proposed method can be used for network fault detection using time correlation (inbuilt in the HMM) and space correlation which can be easily implemented using any one of the methods listed in [43]. Once the fault is detected,

the data can be loaded onto the cloud or server for further processing and intimation. Thus, the proposed hybridized HMM with NNs is practically viable, and can be implemented in the online detection of faults in the networks which is the ultimate requirement for implementation in a real time environment. The rationale behind the formulation and implementation of hybridized NNs with CDHMM is attributed to its ability to correlate parameters related to the dynamics of state changes, its capability to associate with the effects of memory propagation during state transitions, its capacity to associate sequence similarity relationships, and its inherent capability to predict time-series sequences. These features are obviously necessitated in order to capture the transient and intermittent fault occurrences in wireless sensor nodes. In addition, the dynamics of the fault characteristics exactly matches with the proposed hybrid models. Thus, the preceding merits succinctly justify the selection of CDHMM and augur well for its suitability in implementing with specifically chosen NNs for the detection of faults in a sensor node.

### 3.1. Hidden Markov Model

In this research, a five-state HMM is adopted and the same is shown in Figure 2, where the top layer represents a Markov process which is the hidden process, while the bottom layer shows the observable states that depends on the hidden states. In this case, the hidden five states  $S = S_1, S_2, S_3, S_4, S_5$  are connected with the output observable states as  $O = O_1, O_2, O_3, O_4, O_5$  using transition probability matrix 'A'. HMM is best described based on four parameters, namely the hidden states, initial, transition, and emission probability. If the observation is continuous in nature, then the continuous density HMM is formulated as  $\lambda = (A, B, \Pi)$ , where  $\Pi$  and 'B' are the initial and the emission probability distribution respectively. Incidentally, it is worth mentioning that the appropriate choice of the type of HMM at the nascent stage and during the design phase of HMM formulation depends on the complexity of the problem, signal processing requirements and applications [44].

The procedure adopted during the implementation of continuous density HMM is indicated in Algorithm 2. In step 3, a minimum distance algorithm is used to cluster the data for training the dataset. The algorithm is primarily a minimum distance metric strategy (Euclidean), utilized to determine the adjacent datasets relative to various junction of intra-cluster and inter-cluster dataset. This is used to realize an optimum common crossing vector that act as the centers denoting the dataset. The various steps followed in the implementation of the minimum distance algorithm is shown in Algorithm 1.



**Figure 2.** A five-State hidden Markov model (HMM) process.

CDHMM is a three-layered stochastic process. The first level, which is related to the choice of the next state, is the same as in the structure with discrete HMM. Although the second and the third levels are identical to the choice of emission symbol with discrete HMM, the second level of CDHMM includes the choice of the mixture density by mixture coefficient. The selection of the output vector by the Gaussian density is the third and last level. Therefore, the classification and training algorithms have to be altered accordingly, although there are few changes in the classification algorithm i.e., the modified probability densities have to be replaced. The Baum–Welch/Viterbi training algorithms are invoked to obtain the optimized state transition labels and the corresponding estimate of the optimized B matrix value. However, the disadvantage of this model may be attributed to its high computational effort. This aspect is attributed to the evaluation of multiple Gaussian distribution mixture densities which may result into a number of parameters, resulting in instabilities.

---

**Algorithm 1** Procedure to cluster the data using minimum distance algorithm for training the data set.

---

- 1: Consider 'n' as number of classes and 'k' as feature vectors
  - 2: For the initial sample, prefer the first 'n' of 'k'
  - 3: In order to populate 'n' classes, assign feature vectors to the nearest sample
  - 4: Give index  $i | k | = n$  to nominate  $x(k)$  associate to class 'n'.
  - 5: Estimate the cluster size with  $s[n]$  and increase ( $s[n] = s[n] + 1$ ) and go to Step 3
  - 6: Retrieve the new centers by averaging the feature vectors in each of the class
  - 7: Assign  $a[m] [n] = 0$ , here 'm' is the component of class 'n' and  $x[m][k]$  is mth component of  $x(k)$
  - 8: For each class 'n', assign the averages
  - 9: Get the sum of the component values for every 'm' and realize the mean cluster 'n'.
  - 10: If there is no change in class or iteration not completed, then goto Step 7 or Step 3 respectively
- 

**Algorithm 2** Procedure to implement continuous density HMM.

---

- 1: Input hidden states(S), Clusters(TN), Windows(tuples)
  - 2: Read the sensor data
  - 3: Clustering using minimum distance algorithm (see Algorithm 1)
  - 4: Obtain  $\Pi, A, B$
  - 5: **for**  $k = 1, \dots, N$  **do**
  - 6:     Determine  $\Pi_k$ , When  $S_1 = 1$
  - 7:      $\Pi_k = \frac{\text{Number of Occurance}}{\text{Number of training Observation}}$
  - 8:     **for**  $m = 1, \dots, N$  **do**
  - 9:         Determine  $a_{km}$ , When  $S_t = k$  and  $S_{t+1} = m$
  - 10:          $a_{km} = \frac{\text{Number of Occurance for } t(S_t = k \text{ and } S_{t+1} = m)}{\text{Number of Occurance for } t(S_t = k)}$
  - 11:     **endfor**
  - 12: **endfor**
  - 13:     Compute Mean Vector  $\mu_k$  and Covariance Matrix  $V_k$
  - 14:      $\mu_k = \frac{1}{N_k} \sum_{k=1}^n O_t, O_t \in k$
  - 15:      $V_k = \frac{1}{N_k} \sum_{k=1}^n (O_t - \mu_k)^T (O_t - \mu_k), O_t \in k$
  - 16: **for**  $m = 1, \dots, N$  **do**
  - 17:     **for**  $k = 1, \dots, N$  **do**
  - 18:         Compute A-posterior observation probability density  $b_k(o_t)$
  - 19:          $b_k(o_t) = \frac{1}{(2\pi)^{u/2} (v_m)^{1/2}} \exp\left[-\frac{1}{2} (O_t - \mu_k) V_m^{-1} (O_t - \mu_k)^T\right]$
  - 20:     **endfor**
  - 21: **endfor**
  - 22: Compute state optimized likelihood using Viterbi algorithm
-

### 3.2. Learning Vector Quantization Neural Network

Learning vector quantization (LVQ) is a supervised version of vector quantization developed by Kohonen which fundamentally utilizes labelled input data and is based on a competitive (winner-takes-all) learning strategy. The basic LVQ approach is quite intuitive. It is based on a standard trained self organizing map (SOM) with input vectors  $x$  and weights/Voronoi vectors  $w_j$ . The main objective of the LVQ1 network is to find the appropriate output unit that is closest to the input vector. If  $x$  is the training vector  $(x_1, \dots, x_i, \dots, x_n)$  and  $w_c$  is the weight vector for the  $j$ th output unit  $(w_{1j}, \dots, w_{ij}, \dots, w_{nj})$  belonging to different classes, then the weights move away from the input vector [45]. For each training input vector  $x$ , the  $j$ th output is obtained so that the Euclidean distance  $\|x - w_j\|$  between the input vector and weight vector for the  $j$ th output unit is minimized. Simultaneously, the weight vector  $w_j$  for the  $j$ th output unit is updated.

$$\text{if} \quad T = c_j, \text{ then} \quad (1)$$

$$w_j(\text{new}) = w_j(\text{old}) + \alpha[x - w_j(\text{old})] \quad (2)$$

$$\text{if} \quad T \neq c_j, \text{ then} \quad (3)$$

$$w_j(\text{new}) = w_j(\text{old}) - \alpha[x - w_j(\text{old})] \quad (4)$$

$T$  = class for the training vector

$x$  = training vector

$w_j$  = weight vector

$c_j$  = class represented by  $j$ th output unit

However in LVQ2, learning weights are updated by using two codebook vectors,  $m_i$  and  $m_j$ , which are taken as the two nearest neighbors of the training vector  $x$ . This concept does not form a part of the strategy in LVQ1. In this case,  $m_i$  belongs to correct class and  $m_j$  belongs to the wrong class. Furthermore, in LVQ2 the vectors  $m_i$  and  $m_j$  are updated only when  $x$  should be located within a zone called a “window” of relative width  $w$  if  $\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s$ , where  $s = \frac{1+w}{1-w}$ . The complete procedure to implement LVQ2 is given in Algorithm 3.

---

**Algorithm 3** Procedure to implement learning vector quantization (LVQ-2).

---

- 1: Prepare reference vectors, weight vectors  $w_j$  and learning rate ( $\alpha$ )
  - 2: **while** (stopping condition false) **do**
  - 3:     **for** every training input vector  $x$  **do** steps 4 to 7
  - 4: Discover the  $j$ th output for each training input vector where Euclidean distance  $\|x - w_j\|$  should be minimum, renew weight vector  $w_j$
  - 5: If the class defined for training vector is equal to the class represented by  $j$ th output unit, then  $w_j(\text{new}) = w_j(\text{old}) + \alpha[x - w_j(\text{old})]$
  - 6: else  $w_j(\text{new}) = w_j(\text{old}) - \alpha[x - w_j(\text{old})]$
  - 7:     **endfor**
  - 8:     **endWhile**
  - 9: Repeat the above procedure for each instance in training dataset by decreasing the learning rate.
- 

### 3.3. Probabilistic Neural Network

A probabilistic neural network (PNN) is predominantly a classifier which can map any input pattern to carry out multiple classifications of categories. Basically, PNN is an implementation of a statistical algorithm called the kernel discriminant analysis in which the operations are organized using four different layers, namely input layer, pattern layer, summation layer, and output layer [46]. The architecture of the probabilistic neural network is shown in Figure 3. According to the classification



theory, if the probability density function (PDF) of each of the populations is known, then an unknown,  $X$ , belongs to class “ $i$ ” if:

$$f_i(X) > f_j(X), \forall j \neq i \tag{5}$$

PNN incorporates with Bayesian method for decision-making along with a non-parametric estimator to achieve the probability density function (PDF). Algorithm 4 shows the procedure to implement the PNN.

In a population, the PDF for a single sample is given by

$$\frac{1}{\sigma} w \left( \frac{x - x_k}{\sigma} \right) \tag{6}$$

where,  $x$  = unknown input,  
 $x_k$  =  $k$ th sample,  
 $w$  = weighting function,  
 $\sigma$  = smoothing parameter.

Probability density function for a single population is given by,

$$\frac{1}{n\sigma} \sum_{k=1}^n w \left( \frac{x - x_k}{\sigma} \right) \tag{7}$$

When Gaussian function is used as an activation function then the PDF becomes

$$g(x) = \frac{1}{n\sigma} \sum_{k=1}^n e^{-\frac{(x-x_k)^2}{\sigma^2}} \tag{8}$$

PNN is one among a few significant powerful statistical classification techniques which when compared to the multilayer feed forward NN, outperforms the latter due to its rapid training speed and operational behavior of the hidden layer with functionally effective kernel functions. Even though PNN has some minor weaknesses related to the requirement of large memory capability in fast classification, the advantages of the PNN is that it encompasses the capability to train many orders of magnitude faster than the multi-layer feed forward NN, its ability to provide mathematically acceptable confidence levels during discrimination, its built-in strength to handle the effect of outliers etc.

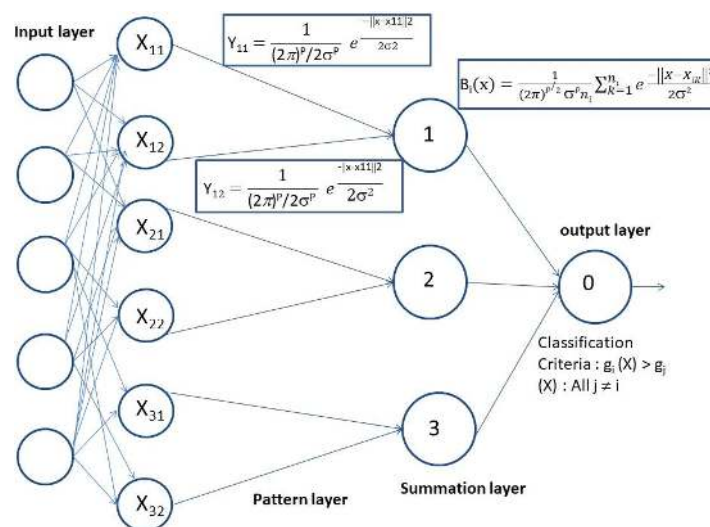


Figure 3. Architecture of probabilistic neural network.

**Algorithm 4** Procedure to implement Probabilistic Neural Network (PNN).

- 1: For every individual training input vector  $X(i)$ , where  $i=1,2,3,\dots,n$ , repeat steps 2 to 5
- 2: Generate pattern unit  $P_i$
- 3: Assign weight vector to pattern unit  $P_i$  as  

$$W_i = X(i)$$
- 4: Connect pattern unit to summation unit
- 5: If  $X(i)$  matches with class 1, then associate pattern unit  $P_i$  to summation unit  $U_1$  else associate to  $U_2$
- 6: At the pattern layer, enumerate Gaussian function values for every class by using the following equation  

$$g_i(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\|x-x_j\|^2}{2\sigma^2}}$$
- 7: The input vectors that belong to same class is summed in the summation unit  

$$f_i(x) = \sum_{i=1}^n g_i(x)$$
- 8: With the help of 'f' values the output layer makes the decision to assign classes for the input vectors.
- 9: If  $f_i(x) \geq f_j(x)$ , then x belong to class 1 else x belong to class 2.

**3.4. Adaptive Probabilistic Neural Network**

It is evident from previous sections that the construction of PNNs for dynamic events is challenging when compared to stationary events. For designing PNNs for the Bayes-optimal decision surfaces (dynamical in nature), it is required to select a smoothing parameter and a learning sequence. This sequence should satisfy conditions that are exemplary for the stochastic approximation scheme [47].

The adaptive probabilistic neural network (APNN) is a basic variant of the probabilistic neural network (PNN). It provides a feasible mechanism to vary the smoothing parameter  $\sigma$  or the variance parameter within a particular class node, while the PNN uses the common variance parameter for all the classes. Based on computing an average distance,  $\sigma = g.d_{ave}$  is determined by the Euclidean distance, whereby PNN employs different values of  $\sigma$  for each class among feature vectors. It is important to note that "g" now serves as the tuning (free) parameter and can be adjusted to suit appropriate density of datasets. Moreover, APNN uses a simplified formula of PDF which obviates the necessity for normalization, so that a considerable amount of computation is reduced.

**3.5. Radial Basis Function Neural Network**

Radial basis function (RBF) neural network implements a supervised learning strategy and can be realized as single or multilayer networks either in linear or non-linear fashion. If the bias function is used as a non-linear model in RBF network then it can expand its hidden layer by one or more. Usually RBF consists of three layers, namely the input, hidden, and output layer. The input layer consists of  $k_0$  source nodes, whereas  $k_0$  is the dimensionality of the input vector  $X$ . The hidden layer units are also known as radial centers, which provide a set of functions that constitute an arbitrary basis for the input vectors. Radial centers are represented by the vectors, namely  $c_1, c_2, \dots, c_h$ . Inside the hidden layer unit, the input vectors are transformed into nonlinear functions whereas the hidden unit vectors are transformed as linear functions to the output layer. In such a case the dimension of each radial center for a 'n' input network is  $n*1$ . If the input vector  $k_i$  lies in the respective field then the center would activate  $c_j$ , and by using proper choice of weights its target output is given by

$$y = \sum \phi_j w_j, \phi = \phi(\|x - c_j\|) \quad (9)$$

When the Gaussian function is used as the radial bias function then the computational unit present in the hidden layer of the network is given by

$$\phi(z) = e^{-\frac{z^2}{2\sigma^2}} \quad (10)$$

$\sigma$  is a measure of width of the  $j$ th Gaussian function with center  $c_j$ . The activation function of the hidden unit layer computes the Euclidean distance between the input vector and the center of that unit.

#### 4. Fault Detection and Classification Framework

The process of the detection and classification of the sensor faults using CDHMM with NNs is summarized as follows:

1. Raw data, from a sensor node in the wireless sensor network, is pre-processed to remove any anonymous data.
2. The fault injection is carried out on the raw data using the established models for various faults.
3. The fault data is fed as the input for CDHMM with the number of hidden states, the number of clusters and the windows defined initially.
4. The transition matrix from the CDHMM is presented as the input to all the classifiers one at a time.
5. The output of the classifiers are compared and the best overall outcome of the classifier is determined based on an appropriately chosen voting scheme (Byzantine or majority voting strategy).

The proposed scheme of hybrid CDHMM with NNs is shown in Figure 4.

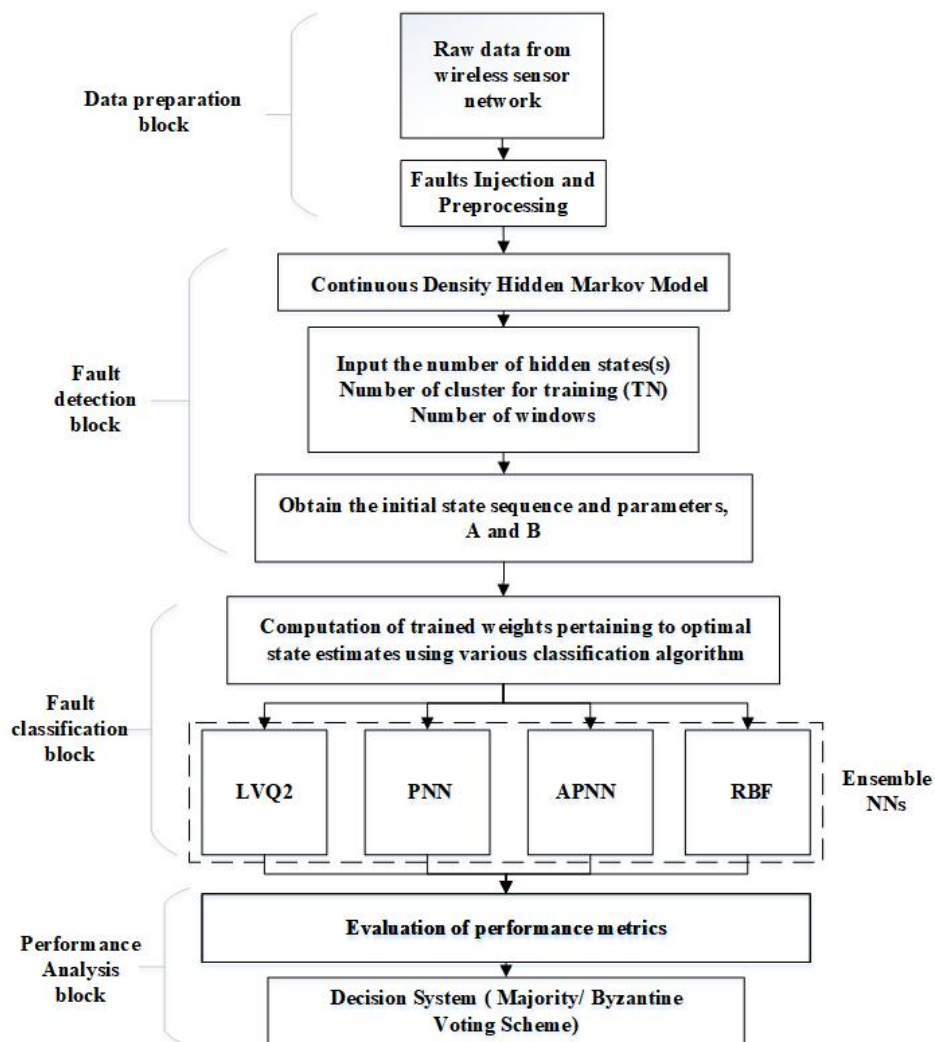
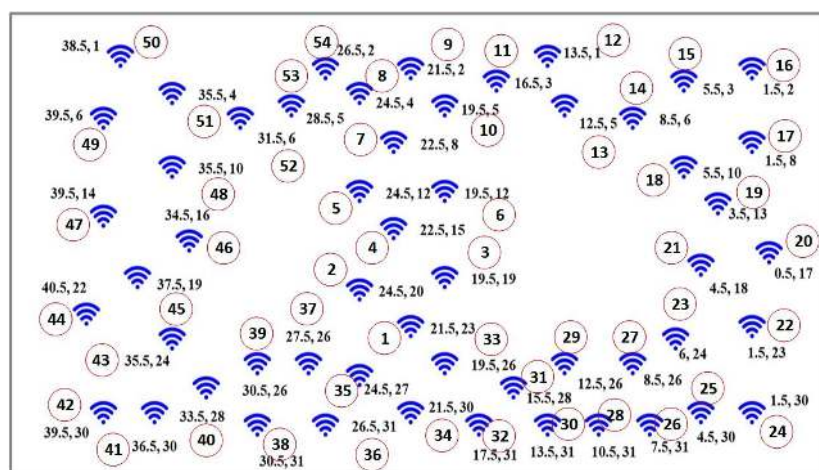


Figure 4. Proposed framework for the detection and classification of sensor faults.

## 5. Experimental Results and Analysis

In this section, the evaluation of the proposed hybrid classifiers is carried out on the real time data set provided by Intel Berkeley research Lab [48]. There are 54 nodes installed in the laboratory in total, recorded from 28th February to 5th April 2004. The acquisition of data had been previously carried out for every 31 seconds from the Mica2Dot sensors along with the topology information. This data had been collected using the TinyDB query processing system which was built on the TinyOS platform. The location of various nodes is shown in Figure 5. The data set contains light, temperature, humidity, and voltage values which had been recorded over the period of a month. In this research, only temperature and light data of one of the nodes (located at (12.5, 26), see Figure 5) was utilized for fault analysis. The reason for choosing, among a large set of parameters, only these two parameters is attributed to the strong correlation between them at a particular instant of time.

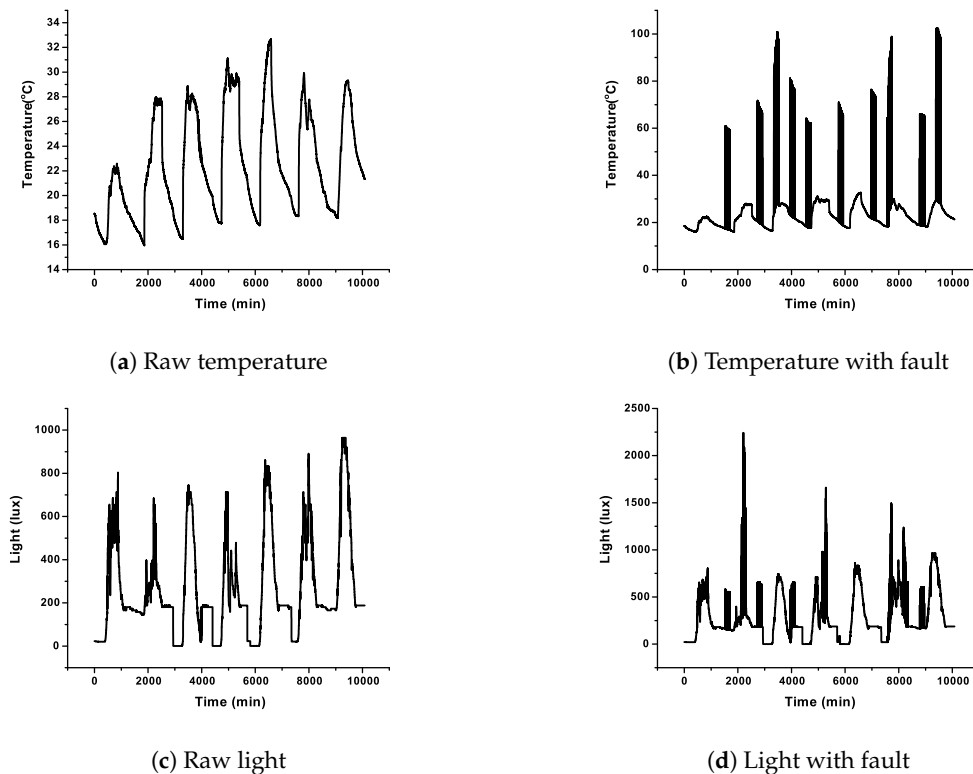
The implementation of hybrid CDHMM with NNs adopts a three-level model. The details of the steps that were followed for the evaluation are shown in Figure 4. The primary aim of this research is to detect and classify the time dependent faults that are due to both discrete or continuous types. The bias and drift faults are considered as continuous faults, while the spike and random faults are characterized as discrete faults. The first step of the detection is to prepare the dataset according to the requirements of the algorithm. All the proposed algorithms are realized in MATLAB 9.6. The proposed NNs have been both verified and validated thoroughly before obtaining the appropriate choice of optimal hyper-parameters. In each case, detailed analyses based on raw datasets were taken up and two major validation techniques, the k-fold method and the one-hold-one out method, were performed.



**Figure 5.** Typical wireless sensor nodes arrangement [48], where wireless sensor nodes are depicted with symbol in 'Blue' and corresponding node number.

In this research, five case studies were taken up for detailed analysis wherein the data sets were organized into two categories, the normal and varying percentage of sensor faults. The dataset was composed of raw light and temperature feature data with an array size of 40 rows by 60 columns. Therefore, a dataset of 2400 data without any fault was prepared. Subsequently, bias, drift, random, and spike fault were introduced on the raw data using the algorithm as discussed in [49]. The methods followed for modeling the fault data were well proven and adopted by various researchers around the globe [50,51]. Figure 6a,c, shows the raw signal of temperature and light. Figure 6b,d, depicts injected random faults in temperature and light. The rate of fault injection started from 10% to 50% with a step size of 10 percent for all the four types of faults. A total of 2400 sets of data were prepared for the performance analysis of the algorithm for each fault type. In each dataset,  $\frac{2}{3}$  of the datasets were utilized for training while the rest of the data were taken up for testing. The first level of the model involved obtaining the fault detection state which comprised of CDHMM, and the outputs were state transition matrix "A" and probability density matrix "B". In the second stage, the optimized values of

the B matrix obtained from the Viterbi algorithm were presented as the input to the next stage which comprised of various neural networks to classify the different faults. In each classifier, the kernel was a Gaussian function, resulting in a group of NNs called ensemble classifiers [52]. The performance of the classifiers could be determined using the four evaluation parameters, namely detection accuracy (DA), false positive rate (FPR), F1-Score, and Matthews correlation coefficient (MCC). Finally, based on an appropriately-chosen voting scheme, the best classifier was determined. To ensure credible validation and reliability of the proposed hybrid CDHMM-NN system, the fault detection techniques were executed repeatedly for obtaining consistent results, and average values were considered for analysis.



**Figure 6.** Temperature and light data with and without fault.

The detection accuracy (DA) [18] is defined as

$$DA = \frac{\text{Number of faulty observation detected}}{\text{Total number of faulty observation}} \quad (11)$$

The DA should be higher for an excellent classifier and it denotes the accuracy of the detection of the algorithm. The DA for all the NNs with various faults was determined. The false positive rate (FPR) [18] measures the number of non-faulty data detected as faulty with the total number of non-faulty data. It is defined as follows:

$$FPR = \frac{\text{Number of Nonfaulty data detected as faulty}}{\text{Total number of Nonfaulty data}} \quad (12)$$

Matthews correlation coefficient (MCC) [18] determines the accuracy of the algorithm and it is defined as below.

$$MCC = \frac{T_p \times T_n - F_p \times F_n}{\sqrt{(T_p + F_p)(T_p + F_n)(T_n + F_p)(T_n + F_n)}} \quad (13)$$

where True Positive ( $T_p$ ) shows the correctness of detection of number of data that are faulty. True Negative ( $T_n$ ) is defined as the detection of number of non-faulty data as non-faulty. False Positive ( $F_p$ ) is defined as detection of non-faulty as faulty. On the other hand, False negative ( $F_n$ ) is defined as the detection of faulty data as non-faulty. When the MCC is near to +1, it means the detection is accurate. MCC can be categorized as excellent, moderate and weak based on the values. If MCC is of +0.7 or higher, it can be considered as excellent, from +0.4 to +0.69, it can be categorized as good and from +0.30 to +0.39 it is moderate. When it reaches +0.20 to +0.29, it is considered as weak.

The F1-Score [13] is used to measure the performance of the classifier based on false positive and false negative values. It is defined as follows.

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (14)$$

where, precision =  $TP / (TP + FP)$  deals with the accuracy of the detection and recall =  $TP / (TP + FN)$  is defined as the measure of determining the particular node as faulty or non-faulty.

### 5.1. Analysis on Capturing the Dynamics of Sensor Faults with CDHMM

Since the major focus of this research is on devising a unique mechanism to ascertain and capture the dynamics of sensor fault by implementing CDHMM, the findings of variation in the state transitions unique to various types of sensor faults are depicted in Figure 7. Here, the transition in the sensor fault is estimated, described and optimized by the probability density function (PDF) of hidden state labels of the CDHMM. Figure 7a shows the PDF for raw data of temperature and light whereas Figure 7b depicts the PDF for random fault with 10% fault rate.

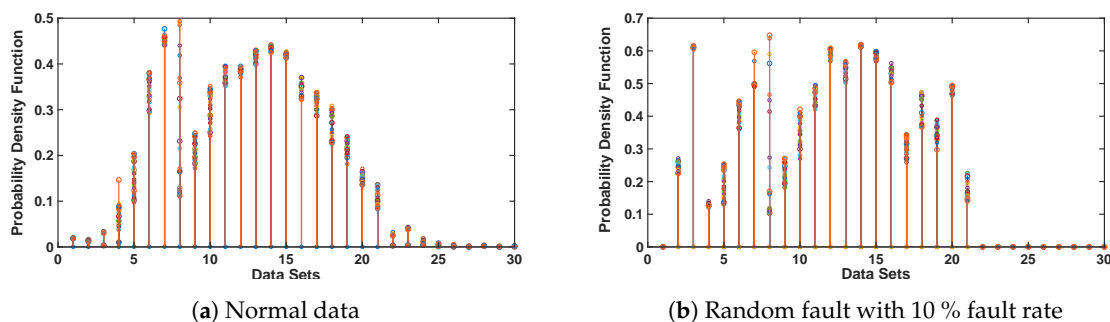
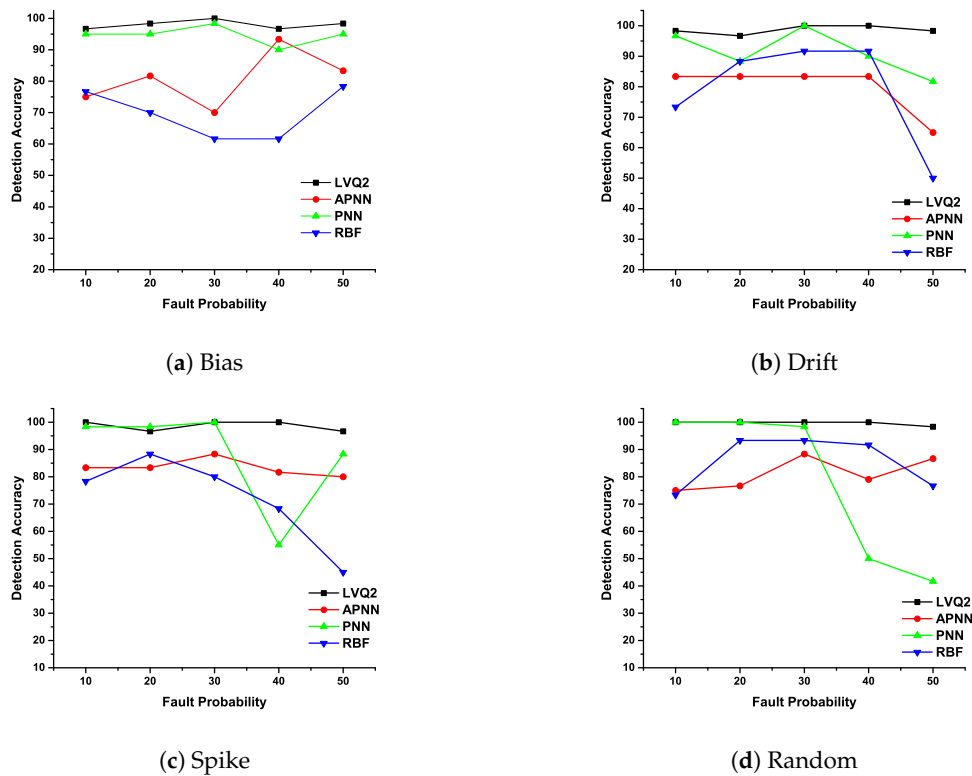


Figure 7. Dynamics of the sensor faults.

It can be noted clearly that the signature patterns of the optimized B-Matrix for the normal and fault data are distinct. Likewise, for various types or percentage of faults, significant and unique variations of patterns can occur. It is also evident from Figure 7a,b that there are significant changes in the PDF values in the data bin between 0 to 10 and from 15 to 25 bins while the values of B-matrix are very similar in the region of data bins between 10 to 15. This clearly demonstrates the strength of CDHMM as a tool in capturing the dynamics of the time sequence data in wireless sensors. These PDFs are then provided to ensemble classifiers to classify the faulty and non-faulty data.

### 5.2. Detection Accuracy and False Positive Rate Analysis

In CDHMM, based on repeated experimentation, iteration value is fixed as 1000. The detection accuracy and false positive rate of various faults are plotted in Figures 8 and 9.



**Figure 8.** Detection accuracy of various faults with different neural networks (NNs).

The DA and FPR are determined for different NN classifiers and for different percentage of faults. In Figure 8a, DA is plotted for bias fault and it can be noted that DA for LVQ2 is constant for all percentages of faults whereas APNN showed lower accuracy at smaller fault rates, but performance was better at higher percentages of faults. On the other hand, RBF and PNN did not perform well for higher percentages of faults. In Figure 8b, the LVQ2 and APNN were found to achieve better accuracy compared to RBF and PNN. In Figure 8c, except RBF, all other classifiers demonstrated strong performance. In Figure 8d, for random faults, the LVQ2 and PNN display a high accuracy compared to APNN and RBF. From the aforementioned discussions, it is clear that the DA for LVQ2 was almost constant for all faults compared to other NNs. It is also evident from the analysis that for the lower extent of faults, either LVQ2 or PNN could be preferred. The reasons for the erroneous response of APNN and RBFNN may be attributed to the aspects summarized.

1. Though the APNN comprises a separate variance parameter for each category (class) of defect pattern, it is evinced from this study that the variance parameter displayed a tendency of peaked nature of the Gaussian kernel so as to ensure meaningful separation of hyper-boundaries for classes that are very similar in its patterns.
2. The values of the spread parameter (variance parameter) pertaining to each class resulted in very small values, indicating the nature of the peaked distribution of datasets for each class. This aspect invariably led to overlapped nature of classes and hence arguably higher misclassification rates.
3. On the other hand, the rudimentary version of the RBFNN tried to circumvent the limitations of the backpropagation NN by ensuring learning of weights which then proceeds to obtain a linear separable model to reach the target output. However, the performance of the NN did not indicate considerable improvement in its classification accuracy. This may be attributed to the need for the appropriate choice of centers during the training phase.

The Figure 9a depict the characteristics of LVQ2 for different fault probability and it is considerably strong compared with the other three NNs. The FPR for both PNN and RBF at lower and higher fault

rates did not demonstrate satisfactory results. Further, APNN exhibited a poor performance for lower percentages of fault. In Figure 9b, except RBF, all other NNs such as LVQ2, PNN, and APNN detection performed well for all percentage of faults. In Figure 9c, FPR is plotted for spike fault where LVQ2 obtained zero misclassifications for all percentage of faults. This shows the exceptional characteristic of LVQ2 in the precise classification of faults. Incidentally, PNN and APNN exhibit the same variation trend for all probabilities. However, the RBFNN shows a lower accuracy for lesser percentage of faults, and at 30% the performance is almost same as other NNs. However, for higher fault rates, the FPR is within an acceptable limit. In Figure 9d, for random faults, the LVQ2 and PNN are perfectly constant for any percentage of faults, while other NNs eschewed themselves differently for the increase of fault percentage. From these observations, it is clear that the FPR for LVQ2 performance is the best in comparison with other NNs.

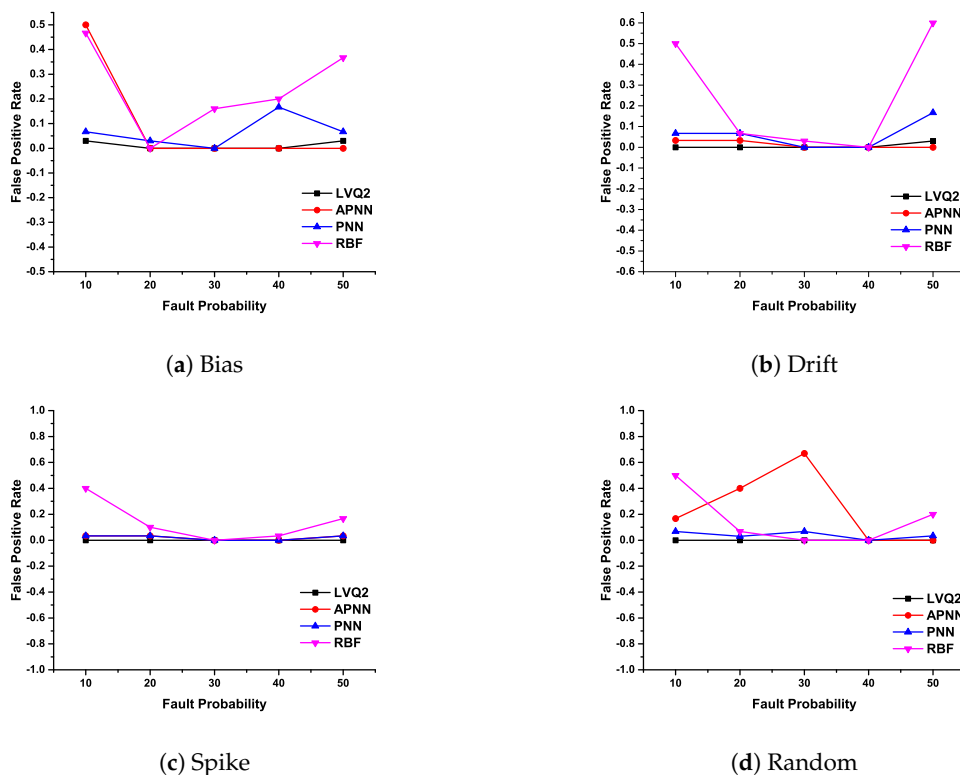


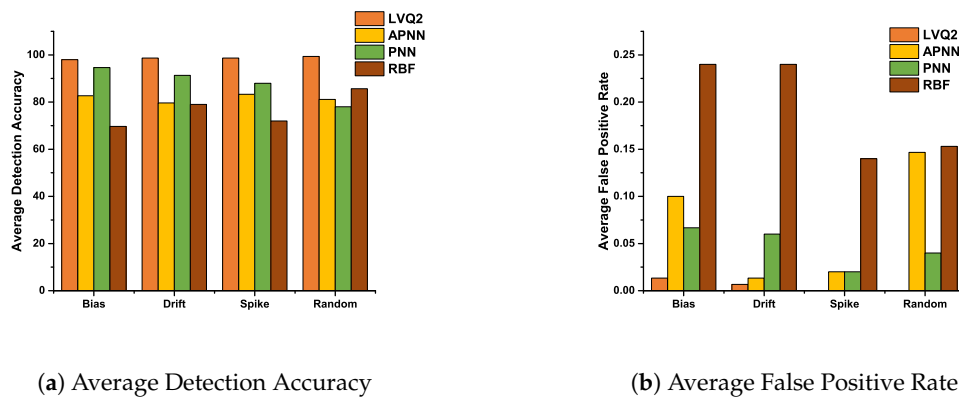
Figure 9. False Positive Rate of various faults with different NNs.

The erroneous performance of the RBF may be attributed to the similarity in structure with that of the PNN which also considers a Gaussian-like kernel for estimation of densities and utilizes the variance for describing the distribution of datasets pertaining to the appropriate class. Though the NN is a supervised learning-based architecture, since functionally a similar structure of mathematical formulation and computation is utilized, it is evident that the results of classification accuracy was quite similar to that of the PNN variants considered in this research study.

The average detection accuracy and false positive rate for various NNs of all faults are plotted in Figure 10. These can be determined by averaging the detection accuracy and the false positive rate for all fault probability for each fault. The performance of various NNs can be easily comprehended from the bar graphs in Figure 10a,b. In Figure 10a, LVQ2 and PNN performed better than RBF and APNN in detecting accurately the faults. On the other hand, detection by APNN is almost constant for all the faults. Similarly, RBF is better for higher fault rates compared to lower cases. From Figure 10b, it is evident that LVQ2 did not detect any non-faulty data as a faulty type i.e., the false detection had almost zero misclassifications for LVQ2. It is also evident that for all transient or instant faults



the PNN could be used as an alternate to LVQ2. For constant fault, APNN may be selected in place of LVQ2. Similarly, the performance of RBFNN was observed to be only just average for any type of faults. Table 1 shows the consolidated detection performance of various classification techniques. In terms of ranking, it is summarized that LVQ2 is the most preferred, followed by the PNN, APNN, and RBFNN classifiers.



**Figure 10.** Average detection accuracy and average false positive rate of various NNs for all faults.

**Table 1.** Detection performance of various classification techniques.

Techniques	Average DA	Average FPR
LVQ2	98.66 %	0.005
APNN	81.69 %	0.07
PNN	87.99 %	0.047
RBF	76.58 %	0.193

It is evident that for large values of the smoothing parameters there are more misclassifications, which is obvious due to the nature of the density of the overlapped data because of various types of sensor faults taken up for study. However, with reduced values of smoothing parameters, substantial improvement in classification accuracy is evinced. These aspects are observed in the detailed analysis of both PNN as well as APNN. In the case of APNN with variation in “g”, it is observed that unique smoothing values resulted in better class discrimination. It is also observed that for the best performance of APNN, having more training sets which could provide appropriate class representative vectors would yield enhanced classification rates.

### 5.3. F1-Score and Matthews Correlation Coefficient Analysis

In this subsection, we analyze the F1-Score and Matthews correlation coefficient for various faults at 10% and 50% fault rates. These two fault rates are chosen to analyze the performance of various hybrid CDHMM-NNs during low and high fault occurrence in a sensor node. Figures 11a and 12a shows F1-Score for all NNs at minimum and maximum percentage of faults. The comparison of these two characteristics reveals a similar trend follows for both LVQ2 and PNN. The characteristics are almost constant for all faults and values; also, it lies within the acceptable limit in both percentages of faults. Even the APNN and RBF give better performances and the least percentages of faults. On the contrary, while the percentage of fault increases, APNN performs well for bias and random fault, whereas negative trend in drift fault. On the contrary, RBF performs to a moderate level for bias and spike faults, while for drift and random, the performance does not meet the requisite expectations. Figures 11b and 12b shows MMC values for all NNs at a minimum and maximum percentage of faults. From the comparison, the LVQ2 performance is constant for all faults, and PNN is next to LVQ2 and its trend also lies within the acceptable limit for all percentages of fault. However, APNN performs well for bias and random fault at 50 % fault, while for lower level faults it shows moderate performance.

At the same time, RBF performs to an acceptable level for bias and spike faults whereas in drift and random types, the performance did not meet the expectations at both lower and higher percentage of faults. Table 2 shows the performance of all the NNs for all faults at 10% and 50% fault rate. These coefficients are determined by repeated simulation of fault data for various algorithms. In this research, the LVQ2 shows an excellent performance compared to all other NNs. Furthermore, the performance of PNN is par with LVQ2 at lower percentage of faults.

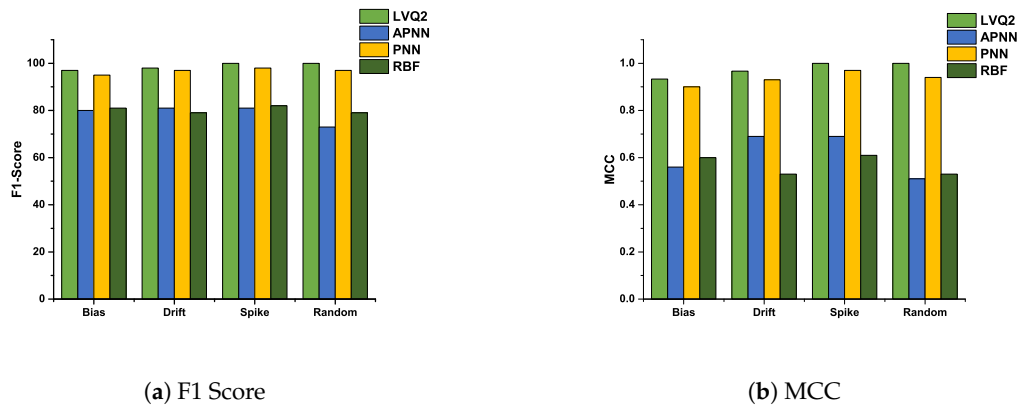


Figure 11. F1-Score and MCC value of various NNs for all faults at 10 % fault rate.

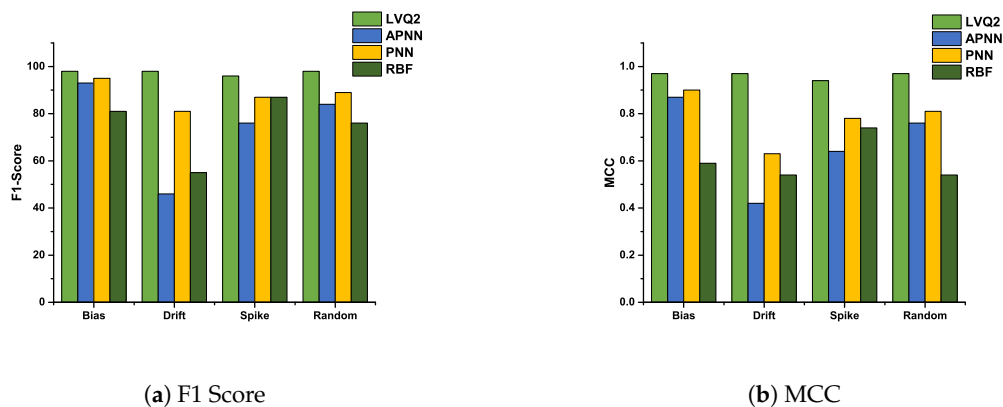


Figure 12. F1-Score and MCC value of various NNs for all faults at 50 % fault rate.

Table 2. MCC of various classification techniques at 10 % and 50% fault rate.

Techniques	MCC(10%)	MCC(50%)	Performance
LVQ2	0.95	0.96	Excellent
APNN	0.61	0.71	Good
PNN	0.93	0.78	Good
RBF	0.57	0.52	Moderate

Though the LVQ2 does not directly utilize the Gaussian kernel, the inherent structure of obtaining the distance vector through the operator  $\epsilon$  in LVQ2 provides a window (range) for the association of similar classes to particular classes during training. Hence, structurally similar types of NNs have been taken up in this research for carrying out meaningful and cogent analyses. For lower values of  $\epsilon$  the density of data with a peaked variance is observed. The tendency of data displayed a peaked distribution for  $\epsilon$ . When there is substantial overlap in the signal (fully or partially overlapped), this may be attributed to the dynamic nature of normal or abnormal fault analyses.

Normally, in order to improve the classification accuracy, the number of iterations in the algorithm is increased. However, this would not be a convincing strategy for the improvement of detection. This is attributed to the fact that the traditional mean square error-based algorithms may not result in

global minima during the training phase. Therefore, in our research, the iteration values have been optimized to be 1000 throughout the analysis of all fault classifications.

#### 5.4. Ensemble Classifier-Based Majority Voting Decision

In this subsection, the research analysis has been consolidated to ascertain the classification accuracy of various CDHMM-NN, and to determine the classification capability of an ensemble classifier for different percentage of faults using the majority and Byzantine voting systems which are appropriate for the hybrid CDHMM-NNs taken up for detailed studies. The total training and testing datasets considered for each percentage of fault is 60. For HMM-RBF, the tuning parameters  $\alpha$  and  $\eta$  are fixed as 0.06 and 0.06 respectively. In HMM-PNN,  $\sigma$  is taken as 0.1, while in HMM-APNN,  $g$  is taken as 0.9 whereas in HMM-PNN,  $\sigma$  is fixed as 0.6. It is worth mentioning that these values are optimized after multiple iterations, and they are fixed as constant for all percentage of faults. Based on detailed analyses carried out on gamut of faults encountered in sensor input datasets, few major findings are summarized:

1. It is evident from Table 3, that for all versions of CDHMM-NN classifier systems that with increasing percentage of faults in the dataset, better classification accuracy of the hybrid classifiers have been observed. This aspect is obvious since unique features that describe the nature of faults are exhibited in the case of higher density estimates of such datasets. Though in this study and implementation a common value of the tunable parts of the NN (variance parameter,  $\epsilon$ ,  $\eta$ ) has been utilized for assessing the role played by the Gaussian kernel in creating hyper-boundaries during the training phase, appropriate tweaking of the free parameter led to much more enhanced results than the outcome indicated in Table 3.
2. Though for the dataset taken up during the training phase in this research work resulted into superior classification capability of LVQ2 as compared to other NNs, it is to be noted that for large real-time datasets (with substantially varying background noise) the decision based on only LVQ2 will not be the best strategy as more training, input validation studies etc. would be essential before arriving at a meaningful decision. It clearly indicates that the decision based on LVQ2 as a single NN alone would be insufficient for obtaining a comprehensive decision.
3. Another interesting aspect is on the role played by the Gaussian kernel utilized as the common underlying activation function in the ensemble classifier system. It is observed that similar types of misclassifications have been reported during detailed studies based on various faults, and the nature of data distribution and the density function resulted in the sharp nature of Gaussian distribution functions for each class. This aspect is made evident from the few significant changes in the misclassification of a few of the input datasets.
4. The ensemble classifier performed considerably well in the case of the hybrid CDHMM-NN classifier system which utilizes the majority voting scheme, while the Byzantine voting scheme which utilizes a more conservative procedure for decision-making displayed inferior results.

**Table 3.** Classification accuracy of hybrid continuous density hidden Markov model (CDHMM)-NN ensemble classifier system with different voting schemes.

Type of Dataset	Classification Accuracy of CDHMM-NN Classifier System (%)				Accuracy of Ensemble Classifier	
	HMM-LVQ2	HMM-APNN	HMM-PNN	HMM-RBF	M.Voting(%)	B.Voting(%)
<b>Bias Fault</b>						
10%	96.67	75	95	76.67	95	76.67
20%	98.33	81.67	95	70	95	81.67
30%	100	70	98.33	61.67	98.33	70
40%	96.66	93.33	90	61.67	93.33	90
50%	98.33	83.33	95	78.33	95	83.33
<b>Drift Fault</b>						
10%	98.33	83	96.67	73.33	96.67	83
20%	96.67	83.33	88.33	88.33	88.33	83.33
30%	100	83.33	100	91.67	100	91.67
40%	100	83.33	90	91.67	91.67	90
50%	98.33	65	81.67	50	81.67	65
<b>Spike Fault</b>						
10%	100	83	98.33	78.33	98.33	83
20%	96.67	83.33	98.33	88.33	98.33	88.33
30%	100	88.33	100	80	100	88.33
40%	100	81.67	55	68.33	81.67	68.33
50%	96.67	80	88.33	45	88.33	80
<b>Random Fault</b>						
10%	100	75	100	73.33	100	75
20%	100	76.67	100	93.33	100	93.33
30%	100	88.33	98.33	80	98.33	88.33
40%	100	79	50	91.67	91.67	79
50%	98.33	86.67	41.67	76.67	86.67	76.67

## 6. Conclusions

A hybrid CDHMM with NN was implemented to detect and classify the faults such as bias, random, drift, and spike. The dataset consisted of temperature and light, while pre-processing and injection of fault from 10% to 50% was carried out before it was presented to the CDHMM. The probability density function from CDHMM was given as input for training and testing data to carefully chosen classifiers, namely LVQ2, PNN, APNN, and RBFNN. The performance of various classifiers were compared in terms of detection accuracy, false positive rate, F1-score and Matthews Correlation Coefficient. The results showed that the hybridized CDHMM with LVQ2 turned out to be the best machine learning algorithm for determining the instant changes in the sensor data due to the effect of various faults. Although LVQ2 performed well for most of the faults, the other NNs showed better performance for unique and specific faults. Therefore, the results showcased in this work were most appropriate for the selection of NN for hybridization with CDHMM for a particular fault. Hence, the proposed method is best suited for a time-critical wireless sensor network, where the detection of faults is essential instantly, yet with substantial reliability. Further, this research work reports detailed studies based on only one major kernel, i.e., Gaussian kernel. Although the Gaussian kernel is the most common and popularly utilized kernel function when the nature of distribution of data is random, other distribution functions with kernels such as elliptical, Laplacian, Cauchy etc. could provide better representation of the data. These hybridized CDHMM and NNs can be extended to a few possible aspects in WSN sensor fault detection:

1. Prognosis and prediction of remaining useful life and ageing characterization of the sensors could be possibly attempted using HMM and other methods such as bath-tub distribution functions, Weibull distribution, and other higher order statistical measures.
2. Online detection of faults can be implemented with this hybridized algorithm in a real-time environment. However, challenges related to training speed in utilizing the HMM may

- be attempted by taking up alternative algorithms (for maximum likelihood estimation and optimization of transition states).
- From the perspective of the training and implementation of more parsimonious sets of centers for NN training, advanced algorithms that optimize and provide reasonably good representative centers that ensure compact training sets may aid one in reducing the time during the training phase, specifically in on-line fault detection.

**Author Contributions:** M.E. conceived the core idea of the proposed work and S.C. helped in achieving the end results. All authors have read and agreed to the published version of the manuscript.

**Funding:** There was no external funding for this work.

**Acknowledgments:** Both the authors are willing to acknowledge the support rendered by S. Venkatesh, School of Electrical Engineering, Vellore Institute of Technology, in the technical discussion on HMM and various NNs.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

- Yick, J.; Mukherjee, B.; Ghosal, D. Wireless sensor network survey. *Comput. Netw.* **2008**, *52*, 142–149. [[CrossRef](#)]
- Anisi, M.H.; Abdullah, A.H.; Razak, S.A. Energy-efficient and reliable data delivery in wireless sensor networks. *Comput. Netw.* **2013**, *19*, 495–505. [[CrossRef](#)]
- Merezeanu, D.; Vasilescu, G.; Dobrescu, R. Context-aware control platform for sensor network integration in IoT and cloud. *Stud. Inform. Control* **2016**, *25*, 489–498. [[CrossRef](#)]
- Xu, F.; Ye, H.; Yang, F.; Zhao, C. Software Defined Mission-Critical Wireless Sensor Network: Architecture and Edge Offloading Strategy. *IEEE Access* **2019**, *7*, 10383–10391. [[CrossRef](#)]
- Munir, A.; Antoon, J.; Gordon-Ross, A. Modeling and analysis of fault detection and fault tolerance in wireless sensor networks. *ACA Trans. Embed. Comput. Syst.* **2015**, *14*, 1–43. [[CrossRef](#)]
- Chudzikiewicz, A.; Bogacz, R.; Kostrzewski, M.; Konowrocki, R. Condition monitoring of railway track systems by using acceleration signals on wheelset axle-boxes. *Transport* **2018**, *33*, 555–566. [[CrossRef](#)]
- Chudzikiewicz, A.; Bogacz, R.; Kostrzewski, M. Using acceleration signals recorded on a railway vehicle wheelsets for rail track condition monitoring. In Proceedings of the 7th European Workshop on Structural Health Monitoring, EWSHM 2014—2nd European Conference of the Prognostics and Health Management (PHM) Society Nantes, France, 8–11 July 2014; pp. 167–174.
- Ni, K.; Ramanathan, N.; Chehade, M.N.H.; Balzano, L.; Nair, S.; Zahedi, S.; Kohler, E.; Pottie, G.; Hansen, M.; Srivastava, M. Sensor network data fault types. *ACM Trans. Sens. Netw.* **2009**, *5*, 1–29. [[CrossRef](#)]
- Nagaraju, S.; Gudino, L.J.; Tripathi, N.; Sreejith, V.; Ramesha, C.K. Mobility assisted localization for mission critical Wireless Sensor Network applications using hybrid area exploration approach. *J. King Saud Univ. Comput. Inf. Sci.* **2018**. [[CrossRef](#)]
- Raposo, D.; Rodrigues, A.; Silva, J.S.; Boavida, F. A Taxonomy of Faults for Wireless Sensor Networks. *J. Netw. Syst. Manag.* **2017**, *25*, 591–611. [[CrossRef](#)]
- Mehmood, A.; Alrajeh, N.; Mukherjee, M.; Abdullah, S.; Song, H. A survey on proactive, active and passive fault diagnosis protocols for WSNs: Network operation perspective. *Sensors* **2018**, *18*, 1787. [[CrossRef](#)] [[PubMed](#)]
- Warriach, E.U.; Tei, K.; A comparative analysis of machine learning algorithms for faults detection in wireless sensor networks. *Int. J. Sens. Netw.* **2017**, *22*, 1–13. [[CrossRef](#)]
- Noshad, Z.; Javaid, N.; Saba, T.; Wadud, Z.; Saleem, M.Q.; Alzahrani, M.E.; Sheta, O.E. Fault detection in wireless sensor networks through the random forest classifier. *Sensors* **2019**, *7*, 1568. [[CrossRef](#)] [[PubMed](#)]
- Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E., Wireless sensor networks: A survey. *Comput. Netw.* **2002**, *38*, 393–422. [[CrossRef](#)]
- Salem, O.; Liu, Y.; Mehaoua, A.; Boutaba, R. Online Anomaly Detection in Wireless Body Area Networks for Reliable Healthcare Monitoring. *IEEE J. Biomed. Health Inf.* **2014**, *18*, 1541–1551. [[CrossRef](#)] [[PubMed](#)]
- Gao, M.; Jiang, S.; Pan, F. Fault detection for time-delayed networked control systems with sensor saturation and randomly occurring faults. *Int. J. Innov. Comput. Inf. Control* **2018**, *14*, 1511–1525.

17. GLu, S.; Zhou, P.; Wang, X.; Liu, Y.; Liu, F.; Zhao, J. Condition monitoring and fault diagnosis of motor bearings using undersampled vibration signals from a wireless sensor network. *J. Sound Vib.* **2018**, *414*, 81–96.
18. Muhammed, T.; Shaikh, R.A. An analysis of fault detection strategies in wireless sensor networks. *Wirel. Networks* **2017**, *78*, 267–287. [[CrossRef](#)]
19. Lau, B.C.P.; Ma, E.W.M.; Chow, T.W.S. Probabilistic fault detector for Wireless Sensor Network. *Expert Syst. Appl.* **2014**, *41*, 3703–3711. [[CrossRef](#)]
20. Huang, P.; Chen, H.; Xing, G.; Tan, Y. SGF: A state-free gradient-based forwarding protocol for wireless sensor networks. *ACM T. Sens. Netw.* **2009**, *5*, 14. [[CrossRef](#)]
21. Kim, W.; Stankovic, M.S.; Johansson, K.H.; Kim, H.J. A distributed support vector machine learning over wireless sensor networks. *IEEE Trans. Cybernetics* **2015**, *11*, 2599–2611. [[CrossRef](#)]
22. Zidi, S.; Moulahi, T.; Alaya, B. Fault detection in wireless sensor networks through SVM classifier. *IEEE Sens. J.* **2018**, 340–347. [[CrossRef](#)]
23. Haque, S.A.; Rahman, M.; Aziz, S.M. Sensor anomaly detection in wireless sensor networks for healthcare. *Sensors* **2015**, *17*, 8764–8786. [[CrossRef](#)]
24. Javaid, A.; Javaid, N.; Wadud, Z.; Saba, T.; Sheta, O.E.; Saleem, M.Q.; Alzahrani, M.E. Machine learning algorithms and fault detection for improved belief function based decision fusion in wireless sensor networks. *Sensors* **2019**, *19*, 1334. [[CrossRef](#)] [[PubMed](#)]
25. De Paola, A. and Lo Re, G. and Milazzo, F. and Ortolani, M. QoS-aware fault detection in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2013**, *9*, 165732. [[CrossRef](#)]
26. Swain, R.R.; Khilar, P.M.; Dash, T. Multifault diagnosis in WSN using a hybrid metaheuristic trained neural network. *Digit. Commun. Netw.* **2018**. [[CrossRef](#)]
27. Zhao, M.; Tian, Z.; Chow, T.W.S. Fault diagnosis on wireless sensor network using the neighborhood kernel density estimation. *Neural Comput. Appl.* **2019**, *31*, 4019–4030. [[CrossRef](#)]
28. Panda, M.; Khilar, P.M. Distributed Byzantine fault detection technique in wireless sensor networks based on hypothesis testing. *Comput. Electr. Eng.* **2015**, *48*, 270–285. [[CrossRef](#)]
29. Jin, X. ; Chow, T.W.S. ; Sun, Y. ; Shan, J. ; Lau, B.C.P. Kuiper test and autoregressive model-based approach for wireless sensor network fault diagnosis. *Wirel. Netw.* **2015**, *21*, 829–839. [[CrossRef](#)]
30. Gong, W.; Liu, K.; Liu, Y. Directional Diagnosis for Wireless Sensor Networks. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 1290–1300. [[CrossRef](#)]
31. Artail, H.; Ajami, A.; Saouma, T.; Charaf, M. A faulty node detection scheme for wireless sensor networks that use data aggregation for transport. *Wirel. Commun. Mob. Comput.* **2016**, *95*, 1956–1971. [[CrossRef](#)]
32. Tang, P.; Chow, T.W.S. Wireless Sensor-Networks Conditions Monitoring and Fault Diagnosis Using Neighborhood Hidden Conditional Random Field. *IEEE Trans. Ind. Inf.* **2016**, *12*, 933–940. [[CrossRef](#)]
33. Zhao, J.; Qu, S. The fuzzy nonlinear enhancement algorithm of infrared image based on curvelet transform. *Procedia Eng.* **2011**, *15*, 3754–3758. [[CrossRef](#)]
34. Kamal, A.R.M.; Bleakley, C.J. ; Dobson, S. Failure detection in wireless sensor networks: A sequence-based dynamic approach. *ACM Trans. Sens. Netw.* **2014**, *10*, 1–29. [[CrossRef](#)]
35. Mahapatro, A. ; Khilar, P.M. Online distributed fault diagnosis in wireless sensor networks. *Adv. Intell. Syst. Comput.* **2013**, *71*, 1931–1960. [[CrossRef](#)]
36. Sharma, K.P.; Sharma, T.P. rDFD: Reactive distributed fault detection in wireless sensor networks. *Wirel. Netw.* **2017**, *23*, 1145–1160. [[CrossRef](#)]
37. Yang, C.; Liu, C.; Zhang, X.; Nepal, S.; Chen, J. A time efficient approach for detecting errors in big sensor data on cloud. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 329–339. [[CrossRef](#)]
38. Nitesh, K.; Jana, P.K. Distributed fault detection and recovery algorithms in two-tier wireless sensor networks. *Int. J. Commun. Netw. Distrib. Syst.* **2016**, *16*, 281–296. [[CrossRef](#)]
39. Moustapha, A.I.; Selmic, R.R. Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection. *IEEE Trans. Instrum. Meas.* **2008**, *57*, 981–988. [[CrossRef](#)]
40. Zhu, D.; Bai, J.; Yang, S.X. A multi-fault diagnosis method for sensor systems based on principle component analysis. *Sensors* **2010**, *10*, 241–253. [[CrossRef](#)]
41. Khan, S.A.; Daachi, B.; Djouani, K. Application of fuzzy inference systems to detection of faults in wireless sensor networks. *Neurocomputing* **2012**, *94*, 111–120. [[CrossRef](#)]
42. Chanak, P.; Banerjee, I.; Sherratt, R.S. Mobile sink based fault diagnosis scheme for wireless sensor network. *J. Syst. Softw.* **2016**, *119*, 45–57. [[CrossRef](#)]

43. Abbasi, A.A.; Younis, M. A survey on clustering algorithms for wireless sensor networks. *Comput. Commun.* **2007**, *30*, 2826–2841. [[CrossRef](#)]
44. Saihi, M.; Zouinkhi, A.; Boussaid, B.; Abdelkarim, M.N.; Andrieux, G. Hidden Gaussian Markov model for distributed fault detection in wireless sensor networks. *Trans. Inst. Meas. Control* **2018**, *40*, 1788–1798. [[CrossRef](#)]
45. Fausett, L. *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*, 1st ed.; Pearson Education: Tamil Nadu, India, 2004; pp. 187–192.
46. Timothy, M. *Practical Neural Network Recipes in C++*; Morgan Kaufmann: San Francisco, CA, USA, 1993.
47. Bose, N.K.; Liang, P. *Neural Network Fundamentals with Graphs, Algorithms and Applications*; McGraw-Hill Inc.: New York, NY, USA, 1995.
48. Intel Lab Data. Available online: <http://db.csail.mit.edu/labdata/labdata.html> (accessed on 13 June 2019).
49. De Bruijn, B.; Nguyen, T.A.; Bucur, D.; Tei, K. Benchmark datasets for fault detection and classification in sensor data. In Proceedings of the 5th International Conference on Sensor Networks, Orlando, FL, USA, 30 October–2 November 2016; pp. 185–195.
50. Wang, Z.M.; Song, G.H.; Gao, C. An isolation-based distributed outlier detection framework using nearest neighbor ensembles for wireless sensor networks. *IEEE Access* **2019**, *7*, 96319–96333. [[CrossRef](#)]
51. Titouna, C.; Naït-Abdesselam, F.; Khokhar, A. DODS: A Distributed Outlier Detection Scheme for Wireless Sensor Networks. *Comput. Netw.* **2019**, *161*, 93–101. [[CrossRef](#)]
52. Paolanti, M.; Romeo, L.; Liciotti, D.; Pietrin, R.; Cenci, A.; Frontoni, E.; Zingaretti, P. Person re-identification with RGB-D camera in top-view configuration through multiple nearest neighbor classifiers and neighborhood component features selection. *Sensors* **2018**, *18*, 10. [[CrossRef](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).