Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Analyzing the performance improvement of hierarchical binary classifiers using ACO through Monte Carlo simulation and multiclass engine vibration data

Vinodha K. *, E.S. Gopi

Department of Electronics and Communication Engineering, National Institute of Technology Tiruchirappalli, Tamil Nadu 620015, India

ARTICLE INFO

Keywords:
ACO
Gaussian Mixture Model
Adaboost
Monte Carlo simulation

ABSTRACT

Hierarchical binary classifiers are often used for multiclass problems when the number of classes is significantly more. The hierarchical structure suffers from a decrease in detection probability when there is an increase in the number of layers and classes. The order and the choice of classifier used in individual blocks of the hierarchical structure affect the overall probability of detection. In this paper, Ant Colony Optimization (ACO) is proposed to optimize the order of the classifier block at each level. It is also proposed to assign a suitable classifier model for the individual classifier block at each level to maximize the overall probability of detection. The proposed techniques also use an Adaptive boosting (Adaboost) classifier model (weighted average of the constructed classifier) as an alternative approach instead of fixing one classifier model at each classifier block and optimizing the classifier block's order using ACO. The description using the probability of false negative (miss $\rightarrow p_M$), and the probability of false positive (false alarm $\rightarrow p_F$) gives the characteristics of the individual classifier blocks in the hierarchical structure. Monte carlo simulation is performed by randomly assigning values for p_M and p_F. It was found that the overall detection rate is consistently increasing using the proposed method for various attempts made in the Monte carlo simulation. The performance of the optimized binary hierarchical structures obtained using the proposed technique for the Engine vibration data with 42 classes and 53 block classifiers are compared with the default hierarchical structure. It was observed that there has been a significant improvement in the overall detection rate from 87.235% using the default structure (with Random forest ensemble to classify all classes in the hierarchy) to 90.92% using the proposed technique (ACO with Adaboost). The proposed technique can be used for applications that demand hierarchical, multi-level classifications.

1. Introduction

Most researchers focus on hierarchical classification to deal with the multiclass classification problem. Generally, hierarchical structure can be either Tree or a Directed Acyclic Graph. Predicting a single class label in a hierarchical structure is a path from the root node to the indicated leaf node at different hierarchy levels. This class hierarchy can be incorporated for better performance when learning a classification model for a hierarchical classification problem (Carlos et al., 2010; Salabat et al., 2017).

The existing hierarchical classification methods are either local (top-down) and global (big bang). The difference is that in local method, it trains more than one classifier in a hierarchical structure. The global method is used when the leaf node belongs to more than one parent node. If the number of classes is decreased or increased in the hierarchy, the entire model needs to be reconstructed. The local method has three standard ways to build a classifier for multiclass problems. They are local classifier per node(LCN), local classifier per parent node(LCPN) and local classifier per level(LCL) (Carlos et al., 2010; Defiyanti et al., 2019; Fogli et al., 2020; Marwa et al., 2020; Ricardo et al., 2014; Roger et al., 2019; Wei & James, 2014).

In the case of LCN, a binary classifier is built for each hierarchy node in the training phase. In LCPN, the multiclass classifier is constructed at each parent and root node. In LCL, the multiclass classifier is constructed at each hierarchy level. The common weakness in these techniques is they face inconsistent prediction and blocking problems (Fafa et al., 2014; Ricardo et al., 2014; Salabat et al., 2017; Singhal et al., 2020; YaoweiShi et al., 2022; Zulfiqar & Waseem, 2018). This problem may be because each classifier node uses the same classification algorithm in the traditional top-down hierarchical classification method. This method appears less effective because each classifier

* Corresponding author.
E-mail addresses: vinodhakamaraj@gmail.com (Vinodha K.), esgopi@nitt.edu (E.S. Gopi).

https://doi.org/10.1016/j.eswa.2023.121730
Received 14 September 2022; Received in revised form 11 August 2023; Accepted 18 September 2023
Available online 27 September 2023
0957-4174/© 2023 Elsevier Ltd. All rights reserved.

node is an individual classification problem or a distinct training set associated with a different set of classes for prediction.

The authors (Fafa et al., 2014; Jianming et al., 2019; Nicholas & Alex, 2008; Victor & Rodrigues, 2018) suggested improving the classifier tree's prediction accuracy by selecting the best classifier at each node from a predefined list of classifiers in the classification algorithms. Ant colony optimization (ACO) is commonly used to select the classifier in a hierarchy structure. ACO is derived from ants choosing the shortest route to reach their destination. When the ants move, they secrete a chemical component known as a pheromone. The following ants decide to continue moving forward based on the higher concentration of the pheromone (Dorigo et al., 2006; Gopi, 2020). ACO uses this analogy to select the selection rule for each node in a hierarchical structure.

The authors (Victor & Rodrigues, 2018) used a hierarchical Ant Colony (HAC) algorithm for simultaneous classifier selection and hyperparameter optimization. HAC is a metaheuristic algorithm that employs multiple ant colonies to search for optimal classifier and hyperparameter combinations. The algorithm uses a hierarchical structure to balance exploration and exploitation during the search process. They used two objective functions: one of them is mean classification accuracy to primarily guide the search, and the other one is an AUC (area under the receiver operating characteristic) as a tiebreaker (presence of more than one search objective).

The authors (Salabat et al., 2017) focused on the global method of hierarchical classification using ACO, where the leaf node has more than one parent node. In this technique, a set of classifier rules are optimized using ACO to classify the multiclass in the hierarchy. They use the Euclidean distance and variance in the class label space to guide the ants in the right direction.

In this paper (Gunaseelan et al., 2018), the authors use ACO in flat binary classification to choose the order in which the SVM classifier block needs to be cascaded to improve the classification rate in two class problems.

The authors (Nicholas & Alex, 2008) proposed to use a hybrid method of both PSO and ACO algorithms to select the best classifier at each node of the hierarchy, which improves the performance of hierarchical classification. In this algorithm, the population of particles and the pheromone vectors are used to determine the best value of a nominal attribute in each dimension of the problem's search space.

In summary, the existing methods deal with ACO to find the best classifier and the best classifier rule to improve the classification performance in the case of multiclass classification in a hierarchical structure. The fitness function used in previous work for ACO is the mean classification accuracy across all the class levels.

The proposed work involves implementing a binary classifier (from root to leaf node) in a hierarchical structure instead of using multiclass classification in the hierarchy, which deals with the individual classification problem associated with a different set of classes for prediction. Our case study adopts a local method to demonstrate the proposed algorithm, as the leaf node belongs to only one root node in the considered hierarchical structure. Two techniques (based on ACO) are proposed to improve the overall probability of detection and the probability of detection even at the deepest classes of the hierarchy level with multiclass problems. The fitness function uses p_F and p_M, new to the current research work for multiclass problems in a hierarchical classification. ACO is also used to select the best classifier model from the predefined list of models and the order arrangement simultaneously (Proposed technique 1). The proposed technique also uses an adaptive boosting algorithm as a classifier model (instead of selecting one best classifier model, combining all the weak classifier models into a stronger classifier model for each block classifier) with order rearrangement using ACO (Proposed technique 2).

The proposed work involves experiments with Monte carlo simulation for various runs and is demonstrated with an engine vibration dataset. With the benchmark dataset, the imbalance class, which affects the classifier performance, is handled by the autoencoder instead of using a traditional oversampling or undersampling method, which lacks useful information while compressing or expanding the data. With the machinery fault database, a new attempt was made to classify all the faults from the root to the leaf node. The comparison results of proposed techniques 1 and 2 are summarized in Section 5.

The experimental data set used in our paper is the machinery fault database (Ali et al., 2019; Mafaulda, 0000; Pestana Viana et al., 2016) which has a default hierarchical classification structure. In fault classification of vibration data, most researchers have classified only significant faults (primary first-level faults) in the engine but failed to organize the subclass from the root node. The secondary-level engine fault classification is also considered in our proposed method. The description of the data set used is described in Section 2. There is an unequal distribution of sample counts in each class, and the classifiers usually perform poorly in minority classes (Aref et al., 2021; Muhammet et al., 2014). This class imbalance problem is a severe issue that impairs classification performance (Dong & Lili, 2022; Jia et al., 2021; Neema & Gopi, 2021; Xu et al., 2017; YaoweiShi et al., 2022; Zhiqiang et al., 2017). Our proposed technique employs an autoencoder to handle the imbalanced classes in the machinery fault dataset. In the proposed method, a comparative study is given on using the best classifier model with optimized order at each block classifier and combining all the classifier models using Adaboost with optimized classifier order. The proposed technique improves the overall classification performance and makes the defect detection approach more appropriate even in the deepest class level in the hierarchy.

The main contributions of this paper are as follows.

1. ACO-based algorithm to obtain the hierarchical binary structure to improve the overall detection rate.
2. Closed-form expression for computing the overall detection rate of the arbitrary hierarchical binary classifier structure is obtained.
3. The performance of the proposed technique is analyzed using the Monte carlo simulation.
4. Demonstration of the proposed technique using engine vibration data with 53 classifier blocks and 42 classes at each block classifier is minimized.

The rest of the paper is as follows. Section 2 describes the problem formulation of the proposed methodology. The description of the proposed method is in Section 3. Section 4 gives an analysis using Monte carlo simulation. Section 5 summarizes the experiments performed and the proposed methodology's results. The conclusion is discussed in Section 6.

2. Problem formulation

Given: An input data set of multiclass with a default hierarchical structure in the form of a tree. The goal is to rebuild the hierarchical structure by finding the best classifier order and model using p_F and p_M at each block classifier to maximize the overall probability of detection.

Hierarchical structure based on open-source (Mafaulda, 0000) machinery fault dataset is adopted and used as the default hierarchical order to analyze the performance of the proposed methodology using Monte carlo simulation. We use the data set further to demonstrate the real-time usage of the proposed techniques. The details regarding the machinery fault dataset and the default hierarchical structure are summarized below.

The engine under test is subjected to rotation with constant rotation speed (rpm), and the sampled version (sampling frequency 50 kHz) of the signal (frequency ranges from 20 to 20,000 Hz) is obtained for 5 s. In the proposed technique, classifier models with the least probability of false positives and the probability of false negatives are chosen. The selection threshold is 0.2 based on the observation of the performance of all the classifiers considered. The eight sensors used are

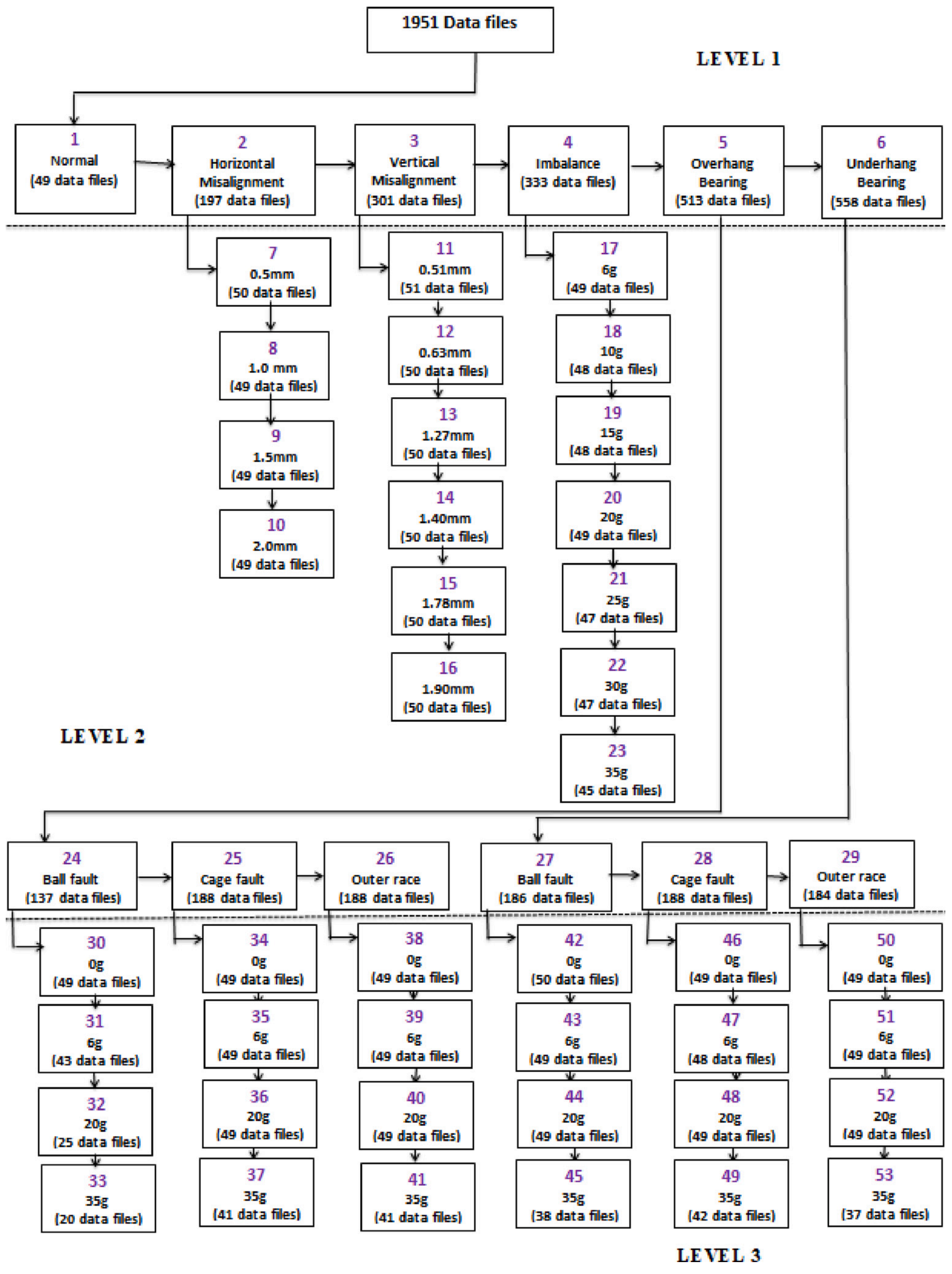


Fig. 1. Default hierarchical classification structure of machinery fault database. The numbers in each block indicate the class label. In the blocks, mm and g indicate millimeter and gram, respectively.

as follows: Three Industrial IMI sensors(Model $601A01$ accelerometers), one IMI sensors triaxial accelerometer(Model $604B31$, which are in the radial, axial, and tangential directions), two accelerometer sensors (National Instruments NI 9234 4 channel analog acquisition modules with a sample rate of 51.2 kHz), one Monarch Instrument MT-190 analog tachometer, and one Shure $SM81$ microphone.

The machinery fault simulator collects several samples at a fixed rotation speed. We formulate the dataset by calculating each sensor outcome's mean values and the corresponding rpm used as the feature vector associated with the corresponding class label. The data collected in the database is 1951, feature vectors of different classes with varying rpm ranging between 737.28 to 3686.4. Since there is an imbalance of data in some classes compared with others, data augmentation using the Autoencoder-based Gaussian Mixture Model to balance the skewed data set is performed. The database has 53 types of fault diagnosis, including root, branch, and leaf nodes subjected to the multiclass classification problem. The default hierarchical order for the multiclass is constructed as given in Fig. 1 by dividing the classes into three levels in the hierarchy structure.

2.1. Objective function

Each block in the hierarchical structure is the binary classifier constructed using the following classifier models. (a) Logistic regression (b) Support vector machine (c) Naive Bayes (d) Random forest ensemble(M_1) (e) Decision tree(M_2) (f) Gradient Boosting(M_3) (g) K-Nearest Neighbor(KNN) (M_4) (h) Multilayer Perceptron(M_5). In our proposed technique, among these eight classifier models, the classifier models that give the least p_M and p_F i.e., d, e, f, g, h are chosen. It is observed that p_M and p_F are lesser than 0.2 for the selected classifier models.

Let the classifier models chosen be represented as M_1, M_2, M_3, M_4, and M_5. The best parameters for all Classifier blocks with the augmented data set are selected using K-fold cross-validation. After training each block with a different model, we calculate p_F and p_M for each block (53 block classifiers) using validation data. The requirement is to rearrange the blocks in the individual classifier blocks under various levels (with the individual blocks constructed with the particular classifier model) to maximize the overall probability of detection (p_d) as given below.

$$p_d = \frac{1}{42} \sum_{i=1, i \in R}^{i=53} p_{k_i}(i/i) \tag{1}$$

where, i ∈ integer, R ← root to leaf node classifier in the hierarchical structure that does not have further branches. $p_{k_i}(i/i)$ is the probability of sample data actually belonging to ith class is declared as ith class (probability of detection) by the constructed hierarchical classifier with ith classifier block constructed using $k_i th$ model. It is noted that the classifier indices are fixed based on the default hierarchical structure (Fig. 1). Let $p_{k_i}(\neq\ i/i)$ be the probability of sample data actually belonging to ith class is not declared as ith class (probability of detection) by the constructed hierarchical classifier with ith classifier block constructed using $k_i th$ model. It is observed that $p_{k_i}(\neq i/i) + p_{k_i}(i/i) = 1$ for all i ∈ to the root-to-leaf node (R). The root-to-leaf nodes are listed below: 1, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, and 53. The closed-form expression for computing the overall probability of detection given the hierarchical structure with the particular model assigned to the individual classifier is given in Appendix. The assignment of the models for the individual blocks and the arrangement of blocks in the individual classifier blocks are optimized using the proposed ACO-based algorithm (refer to Section 3).

3. Proposed methodology using ACO

ACO is a famous meta-heuristic proposed by Dorigo et al. (2006) in the early 90's. Based on its learning style, it is considered a suitable strategy for solving complex combinatorial optimization problems. The motivation for ACO comes from the ant's approach of finding the shortest path to reach the destination. Using this analogy, ACO technique was developed, which determines the order at each hierarchical classification level.

The summary of the algorithm used to obtain the classifier block order and the classifier model assignment that maximizes the overall p_d is given in Algorithm 1. The typical pheromone matrix used for assigning classifier models to the individual classifier blocks is given in Fig. 2(a). The pheromone matrix used to set classifier block orders of the respective levels is given in Fig. 2(b).

The description of constructing the pheromone matrix is as follows: Initialize the default order in which the blocks are arranged. Compute the corresponding cost function by initializing the path of the n ants. The pheromone matrix PM (with size $N \times N$) is initially formulated by filling up with zeros. Let i be the number of class indexes and j be the number of classifier blocks or models used at each level. For the typical ant's path, if jth position of the path has the value i, then the $(i,j)th$ position of the pheromone matrix is said to be active. The functional value of the ith ant's path updates the active positions of the PM. Now the updated pheromone matrix is generated(each ant is described with the vector with N element, with each element filled up with a distinct number between 1 and N). The above procedure is repeated n times to obtain the path of n ants.

Finally, the ant's path corresponding to the maximum value in every iteration is collected, and the best among the final path is declared as the final optimal path. The complete procedure of ACO is given in Algorithms 1 and 3.

3.1. Finding the best fit of classifier block order and classifier model using ACO

Algorithm 1 Ant Colony Optimization for Classifier Selection

```
Require: Number of ants n, evaporation factor $\alpha$, maximum number of iterations max_iter
Ensure: Optimal classifier model M and classifier block order O assignment
 1: Initialize the pheromone matrix PM as a zero matrix.
 2: Initialize the classifier block order O of the first, second, and third levels in a hierarchical structure.
 3: Assign the classifier model M for the individual classifier blocks.
 4: Compute the overall probability of detection $p_d$
 5: Update the corresponding elements of the pheromone matrix $P$ with $+\alpha \times p_d$
 6: while the maximum probability of detection has not been reached do
 7:     Generate new sets of M and O assignments using the updated pheromone matrix PM.
 8:     Compute the overall probability of detection $p_d$
 9:     Update the corresponding elements of the pheromone matrix $P$ with $+\alpha \times p_d$
10: end while
11: return Optimal classifier model and classifier block order assignment
```

3.2. Algorithm of training each block with a combination of five different models

Algorithm 2 Adaboost.

```
 1: Input: Training data $\mathbf{X} = (x_1, t_1), (x_2, t_2), ..., (x_N, t_N)$ where $x_N$ is the $Nth$ sample and $t_i \in 0, 1$ is its label. $y_m(x)$, $\forall m = 1 \cdots 5$
 2: Output: A classifier $y_a(x)$ that predicts the label of a new sample $x$. $y_a(x) = 1$ for correct classification and 0 for mis-classification
 3: Initialize: Set the weights of each training sample to be equal:
        $$w_1^{(1)} = w_2^{(1)} = \dots = w_N^{(1)} = \frac{1}{N}.$$
 4: for $m = 1$ to 5 do
 5:     Compute $y_m(x_n)$ on the training data. $\forall n = 1 \cdots N$.
 6: $y_m(x_n) = 1$ for correct classification and $y_m(x_n = 0$ for mis-classification
 7:     Calculate the error rate of the $mth$ weak classifier :
        $$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(x_n) \neq t_n)}{\sum_{n=1}^{n} w_n^{(m)}}.$$
        $I(y_m(x_n) \neq t_n)$ is the indicator that takes 1 if $y_m(x_n) \neq t_n$
        $$\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right).$$
 8:     Update the weights of the training samples:
        $$w_n^{(m+1)} = w_n^{(m)} e^{-\alpha_m I(y_m(x_n) \neq t_n)}\ \forall n = 1 \cdots N$$
 9:     Normalize the weights: $w_n^{(m+1)} \leftarrow \frac{w_n^{(m+1)}}{\sum_{i=1}^{n} w_i^{(m+1)}}$.
10: end for
11: Output: The final classifier $y_a(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(x)\right)$.
```

In the proposed method 2, ACO is adopted to find the order of arrangements of the classifier block in each level (refer Algorithm 1) with the classifier used in each block is obtained by combining the various classifying models using Adaboost. Adaboost algorithm (refer Algorithm 2) is a machine learning algorithm that combines

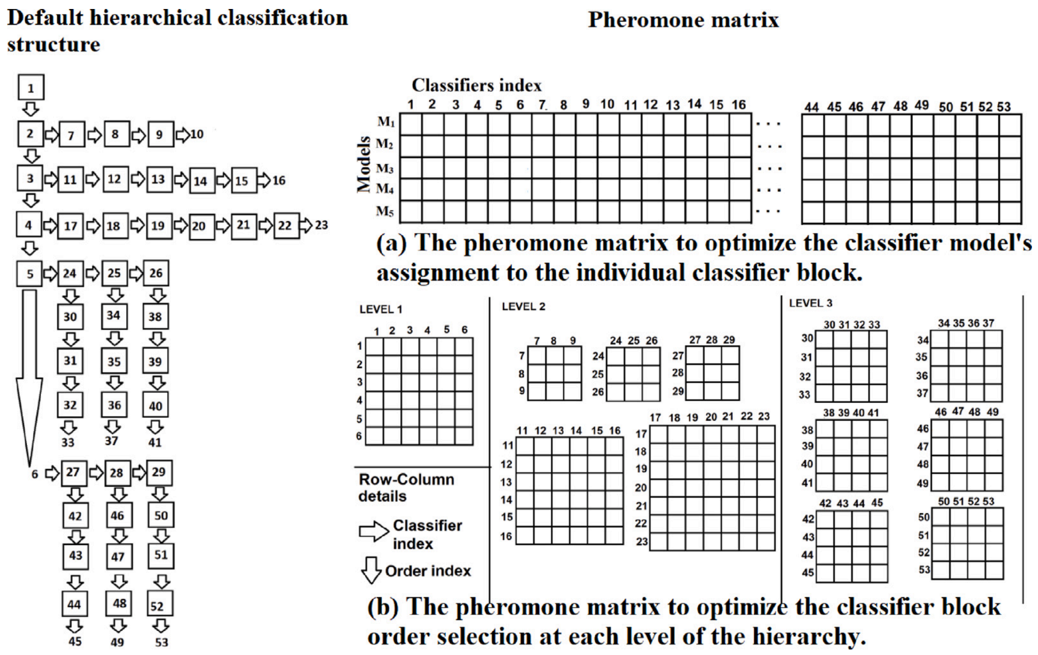

Default hierarchical classification structure
Pheromone matrix
Classifiers index
Models
M1
M2
M3
M4
M5
(a) The pheromone matrix to optimize the classifier model's assignment to the individual classifier block.
LEVEL 1
LEVEL 2
LEVEL 3
Row-Column details
Classifier index
Order index
(b) The pheromone matrix to optimize the classifier block order selection at each level of the hierarchy.

Fig. 2. (a) and (b) (refer Section 2 for details).

weak learners to improve the overall performance of the constructed classifier (Youwei & Lizhou, 2021).

Let x_n and t_n $\forall n = 1 \cdots N$ be the training data and the corresponding label for constructing the binary classifier. N indicates the number of training samples. t_n takes the value either 1 or 0. Let $y_m(x)$ be the mth weak classifiers with $m = 1 \cdots M$ constructed individually using the identical training set. $y_m(x)$ take the value 1 if the vector x belongs to class 1, else 0 if belongs to class 2. Let w_n $\forall n = 1 \cdots N$ be the weights assigned to the individual training data, which is initialized as $\frac{1}{N}$. The values $y_1(x_n)$ obtained for all the training data along with the t_n and w_n are used to compute error rate of the 1st weak classifier as ϵ_1. It is further used to obtain the co-efficient α_1. It is further used to update the weights w_n $\forall n = 1 \cdots N$. Using the latest weights and the corresponding $y_2(x_n)$, t_n are used to compute the error rate ϵ_1 and the co-efficient α_2 for the second weak classifier. It is repeated to obtain the coefficients for all weak classifiers, and hence the combined classifier is obtained as $y_a(x) = sign(\sum_{m=1}^{m=M} \alpha_m y_m(x))$. The estimated weights (α_m in Algorithm 2) depend upon the order of the models chosen in the algorithm. Hence we explored all possible combinations ($5! = 120$ combinations) of 5 classifiers for each block. The best combination in each block was picked and implemented based on accuracy.

3.3. Handling imbalance class using autoencoder

Autoencoder-based data augmentation is adopted to generate augmented engine vibration data as described below.

The original dataset consists of 1951 samples corresponding to 42 classes. In the dataset, 50% of the data is for training, and the remaining is for testing. Using 977 training samples, the augmented data is generated using an autoencoder. The autoencoder with 9 neurons in the input, 2 neurons in the hidden layer, and 9 in the output layer are constructed. Auto-associative training using the constructed model is performed. Two-dimensional vectors corresponding to the hidden layer output are collected and are used to build Gaussian Mixture Model (GMM) with two mixtures. The sample outcomes of the constructed GMM model are treated as the augmented data. Augmented data generation is such that each block in the first level of the hierarchical structure consists of 1200 samples with an equal number of samples in the corresponding subblocks. Thus 7200 samples are used to construct the classifiers. An illustration of original data and the augmented data obtained using a GMM-based autoencoder is given in Fig. 3. Based on the case study (Ali et al., 2019), with the machinery fault database before and after augmentation shows the importance of handling the imbalance class and improvement in classification performance. Since some classes have limited datasets (refer Fig. 1), with this dataset, building the individual block classifier using the classifier model is challenging, it is found that the maximum detection rate for the individual class has increased from 94.2% to 100% with various classification models when data augmentation are used. The training and testing model's performance is shown in Fig. 7, Fig. 8 and Fig. 11 with the augmented dataset.

3.4. Experimental setup

The experimental setup for the proposed work is shown in Fig. 4. It involves the data pre-processing (refer to Section 2) to obtain the standardized data set. After standardization, divide the dataset into two parts 50% is for training and 50% for testing. The training data set is divided into two parts, 80% training, and 20% validation. The dataset consists of a 53 block classifier with 42 root-to-leaf node class indices and three levels in a hierarchical classification structure. In method 1, the hierarchical structure is constructed with binary block classifiers with each classifier block trained using one of the five classification models using training data. In method 2, the hierarchical structure is constructed with binary block classifiers with each classifier block trained using combinations of five classification models using Adaboost for the training data. The individual-trained classifier blocks are validated using validation data. p_M and p_F are calculated using the validation data. Based on the probability of false negative and false positive, ACO is used to find the best order with the optimal model and the best order with the Adaboost model. The final hierarchical structure thus obtained is tested using the test data.

4. Analysis using Monte carlo simulation

Monte carlo simulations are performed to demonstrate the consistent improvement of the detection rate using the proposed technique. The first case uses the randomly generated p_F and the p_M for 53 block classifiers to optimize the order and the model assignment using the proposed ACO that maximizes the overall probability of detection.

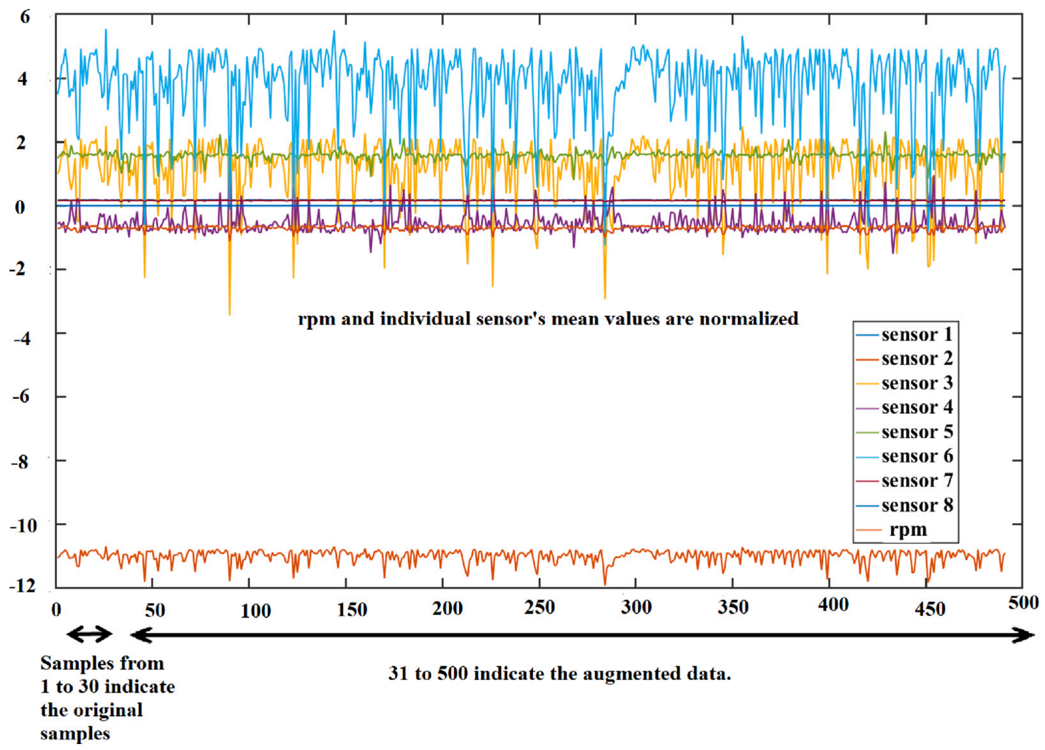


Fig. 3. Illustration of original and the augmented data for the class label (34) using Autoencoder. Each color indicates the distinct elements of the feature vector. Samples from 1 to 30 show the original samples, and 31 to 500 display the augmented data.

It compares the probability of detection attained using the default structure with the identical classification method assigned to all the blocks. In the second case, the Monte carlo simulation uses the randomly generated sequence of the class index (outcome of the discrete random variable that takes the values ∈ root to leaf classifier indices). The Monte carlo simulation mimics the classification of the individual testing data using the constructed classifier. Simulation steps to classify the arbitrary class label i (say) into one among the root classes using the hierarchical classifiers with the specific p_F and the probability of detection assigned to the individual classifiers (Fig. 1) are summarized in Algorithm 3.

For the Monte carlo simulation, the sequence of class indices that are given as the input to the hierarchy classifier is assumed to be the outcomes of the random variable that are uniformly distributed among the root-to-leaf node indices. These are used to mimic the real-time training and testing data. The typical values for p_F and p_M for the individual blocks are obtained as the outcome of the uniformly distributed random variable ranging from 0.0003 to 0.3 (arbitrarily chosen to demonstrate the proposed technique). The sample belonging to the ith class is correctly declared as ith class by the ith binary classifier block (with p_M) if the outcome of the uniformly distributed random variable (between 0 to 1) is greater than p_M. Similarly, samples not belonging to the ith class are declared as ith class by ith binary classifier block (with p_F) if the outcome of the uniformly distributed random variable (between 0 to 1) is lesser than p_F.

Figs. 5 and 6 gives a comparison of the probability of detection attained using a different technique for 100 Runs (Monte carlo simulation). It is found that the ACO with the optimal order and model outperforms in all the runs compared with the default structure with the identical classifier model at all nodes.

Fig. 5 shows the overall probability of detection (computed using p_F and p_M of the individual blocks for the particular hierarchical structures) using various techniques). Fig. 6 shows the overall probability of detection using the randomly generated sequence of the class index (with length 1000) to mimic the classification of the individual testing data using various techniques.

Algorithm 3 Algorithm to classify the arbitrary class label i (say) ∈ root to leaf node using the hierarchical classifiers.

```
 1: Generate $r$: Generate the outcome of the random variable that is
    uniformly distributed between 0 to 1. Let it be $r$.
 2: Identify the first-level and second-level indices for the given class
    label index $i$ (E.g. If the input index is 30, the first-level index is 5,
    and the second-level index is 2. Let it be $f$ and $s$.
 3: for j=1 to 5 do
 4:     if $f \neq j$ then Generate $r$
 5:         if $r > (1 - p_{Fk_j})$ then
 6:             Declare the input classifier index as $j$, where $p_{Fk}$
    represents probability of false positive. Exit loop
 7:         end if
 8:     else if $f == j$ then
 9:         if $r < (1 - p_{Mk_j})$ then
10:             Declare the first level index as $j$, where $p_{Mk}$ represents
    probability of false negative. Exit loop.
11:         end if
12:     end if
13: end for
14: Else the first level index is identified as 6
15: Let the identified first level class index be $\hat{f}$
```

The Monte carlo simulation is repeated for 100 Runs, and the results are summarized in Figs. 5 and 6. Observing that the probability of detection attained using the optimized structure with the best classifier model chosen (obtained using ACO) is significantly more than the default structure with the identical classifier model at all nodes. These experiments reveal the importance of the proposed technique.

5. Demonstrating the importance of the proposed technique using engine vibration data from machinery fault database

The variations of the proposed techniques are summarized below.

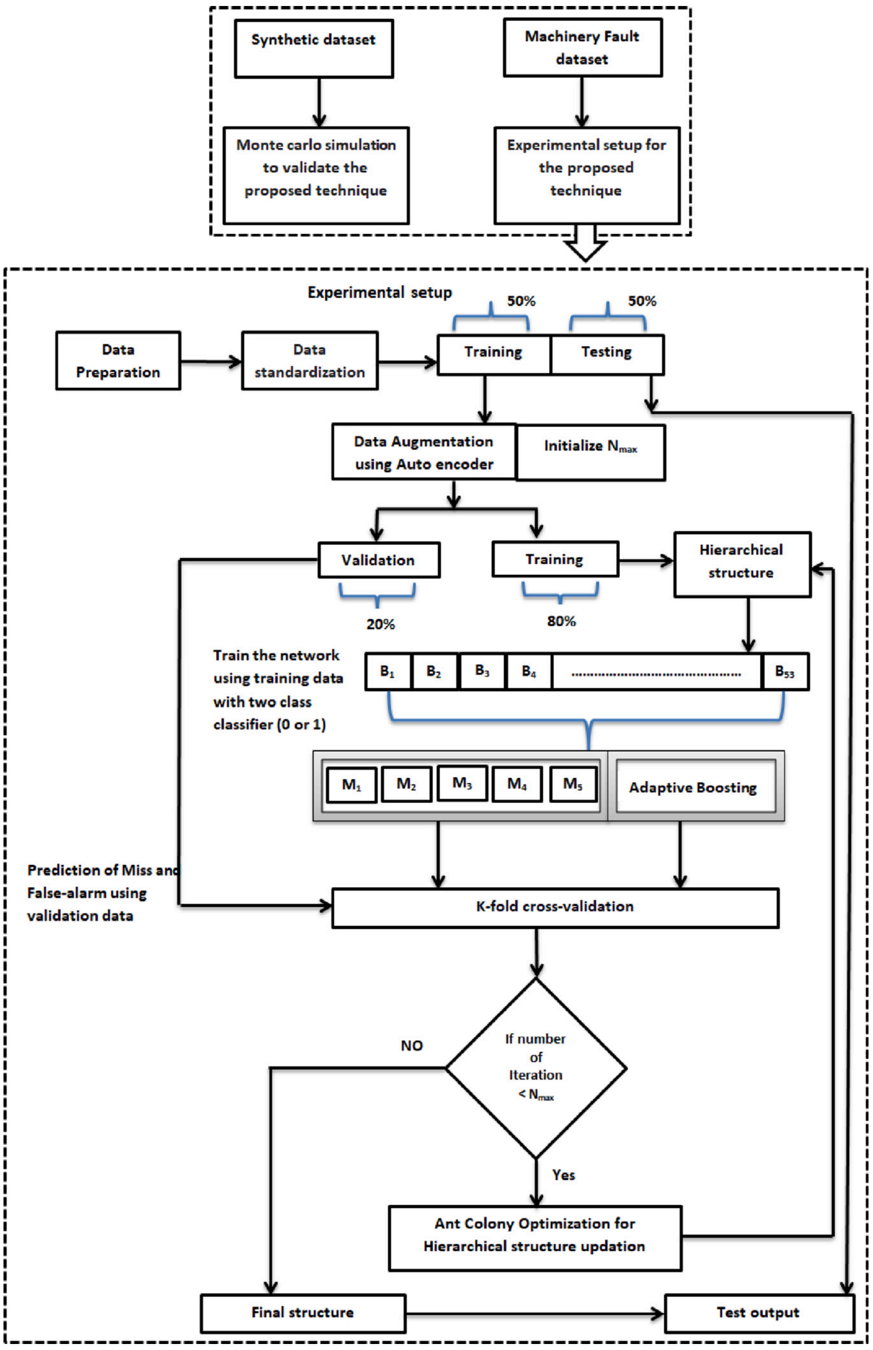

Synthetic dataset
Machinery Fault dataset
Monte carlo simulation to validate the proposed technique
Experimental setup for the proposed technique
Experimental setup
50%
50%
Data Preparation
Data standardization
Training
Testing
Data Augmentation using Auto encoder
Initialize N_{max}
Validation
Training
Hierarchical structure
20%
80%
Train the network using training data with two class classifier (0 or 1)
B1 B2 B3 B4 B53
M1 M2 M3 M4 M5
Adaptive Boosting
Prediction of Miss and False-alarm using validation data
K-fold cross-validation
NO
If number of Iteration < Nmax
Yes
Ant Colony Optimization for Hierarchical structure updation
Final structure
Test output

Fig. 4. Workflow of the proposed technique.

1. Optimize the classifier's order at each level, along with a suitable selection of binary classifiers at each block.
2. Fix the identical binary classifier techniques at each block and optimize the order of the classifiers at each level.
3. Fix the Adaptive boosting binary classifier techniques at each block and optimize the order of the classifiers at each level.

In all the cases described, p_F and p_M for the individual blocks (used in the ACO-based proposed algorithm) are estimated using validation data (refer to Section 4). The estimated values of the p_M and p_F (obtained using the experiments) are summarized in Figs. 7 and 8

Fig. 9(a) shows the best hierarchical structure obtained using ACO with optimal order and model assigned to each node. Fig. 9(b) shows the best hierarchical structure obtained using ACO with optimal order with Adaboost.

Fig. 10 summarizes the comparison of the performances of the proposed method (ACO with the optimal order and model, ACO with the optimal order, and AdaBoost) with the default structure with the identical classifier model at all nodes.

The selection of the optimal order and the optimal models are done during the construction of the optimal hierarchical structure. Once the optimal model is obtained, the time required for classification is instantaneous (see Table 1).

Table 2 compares the overall probability of detection in the default hierarchical structure with identical classifiers at every block and the probability of detection using the proposed technique based on ACO. The overall probability of detection for the machinery fault data set. It is observed that the maximum achievement of 90.92% using ACO with Adaboost, followed by 89.02%, is achieved using ACO with a suitable selection of binary classifiers at each block.

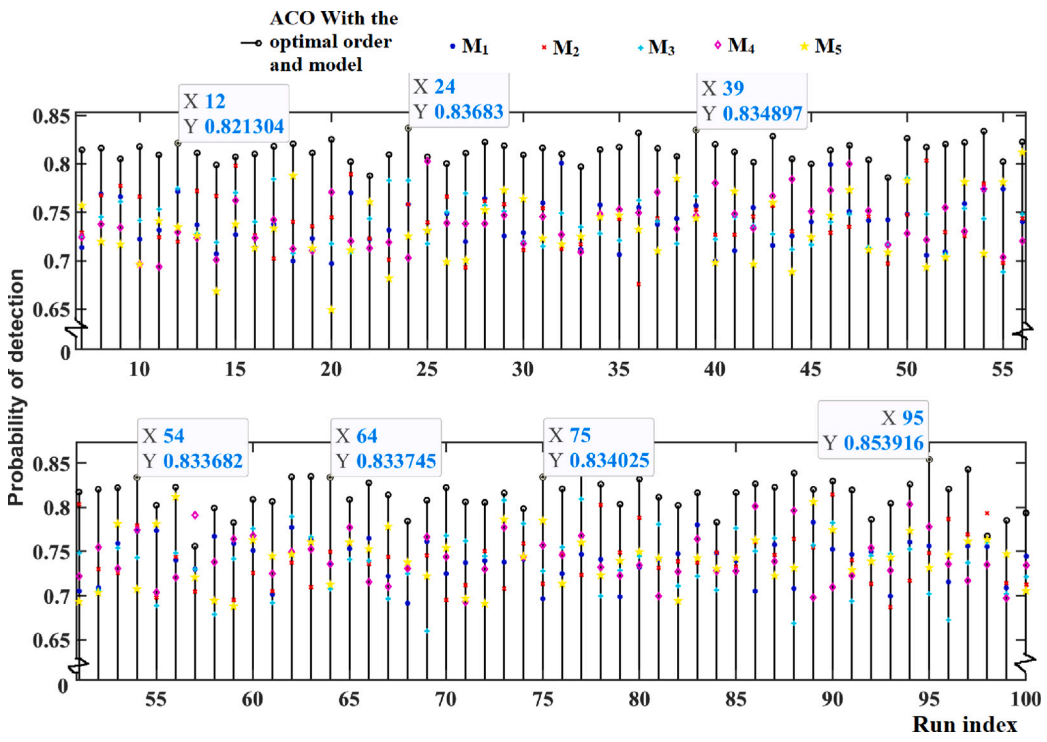


Fig. 5. The Monte carlo simulation with randomly assigned false positive (false alarm) and false negative (miss) for the individual blocks determines the overall probability of detection for different runs using various techniques. It shows that ACO with an optimal model and order performs better than other models in most runs.

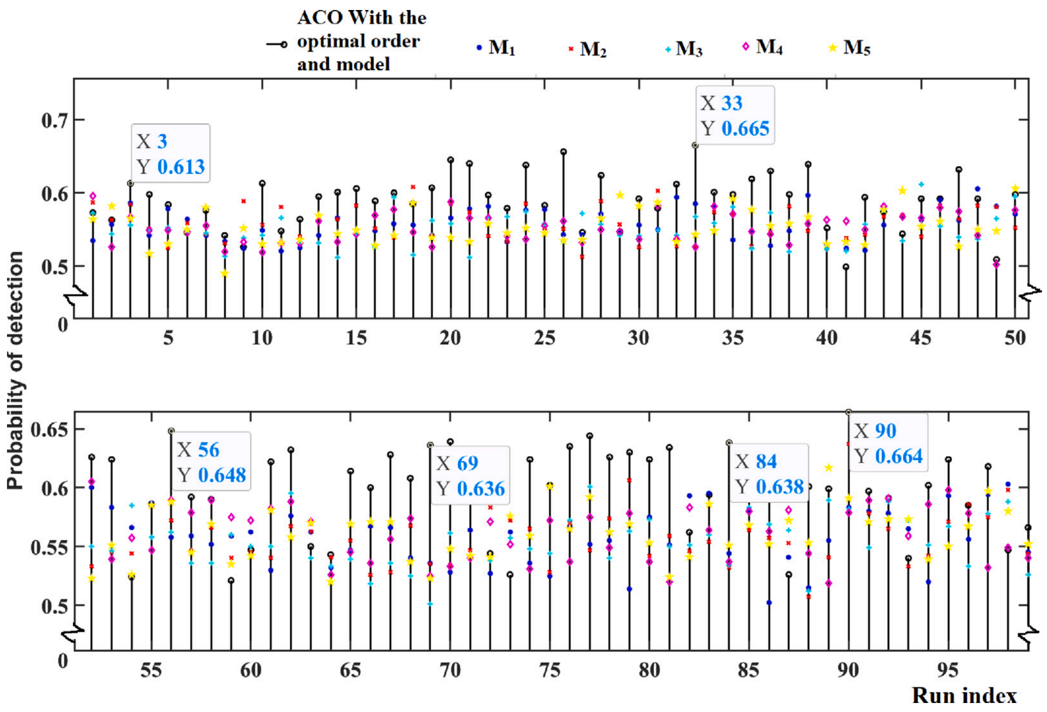


Fig. 6. The Monte carlo simulation with a randomly generated sequence of the class indices belonging to the root node mimics the real-time train and test data. It shows that ACO with an optimal model and order performs better than other models in most runs.

The performance of the proposed technique (Hierarchical structure using ACO based with and without Adaboost) is compared with the default hierarchical structure with identical classifiers at each block using the following metrics: (a) Precision, (b) Recall, (c) F1 score, and (d) Accuracy. The metrics shown in Fig. 11 are for the individual classes. It was found that as the layer of the hierarchical structure increases, there is a significant improvement in all the metrics by using the proposed technique ACO-based with and without Adaboost.

The proposed method aims to increase the hierarchical classifier's overall detection rate (accuracy). Though the performance in terms of metrics like F1 score, precision, and recall are lesser, it is noted that the overall detection rate (hierarchical structure) increases due to the

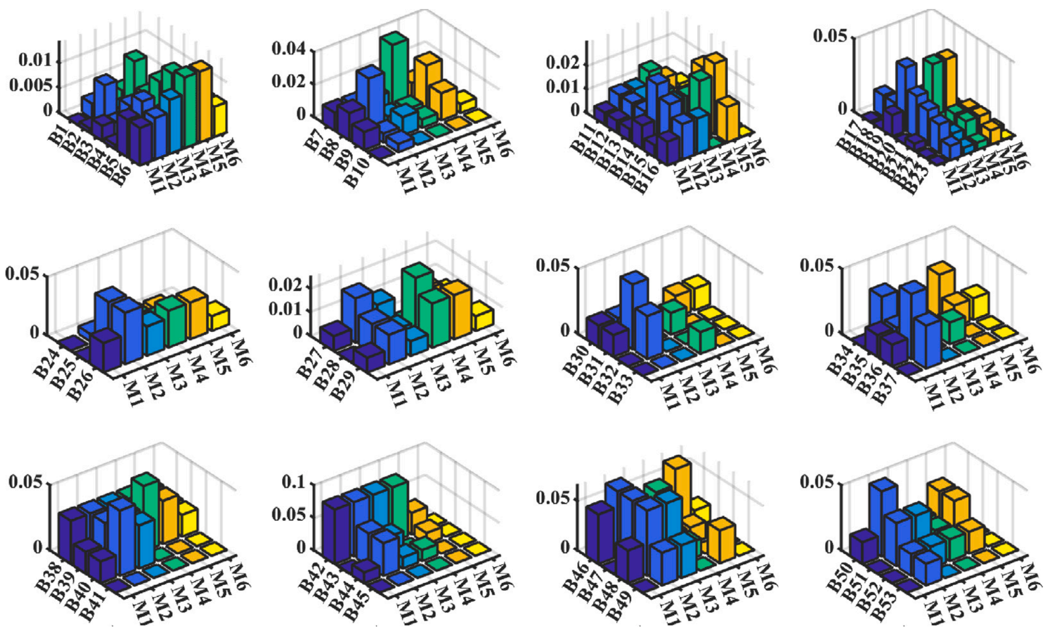


Fig. 7. The probability of a false positive for the model used in each block classifier in the hierarchical structure does not exceed 0.2. It estimates the error probability of the validation dataset. B1 to B53 — Individual block classifier, M1 — Random Forest Ensemble, M2 — Decision Tree, M3 — Gradient Boosting, M4 — K-Nearest Neighbor, M5 — Multilayer Perceptron, M6 — Adaboost.

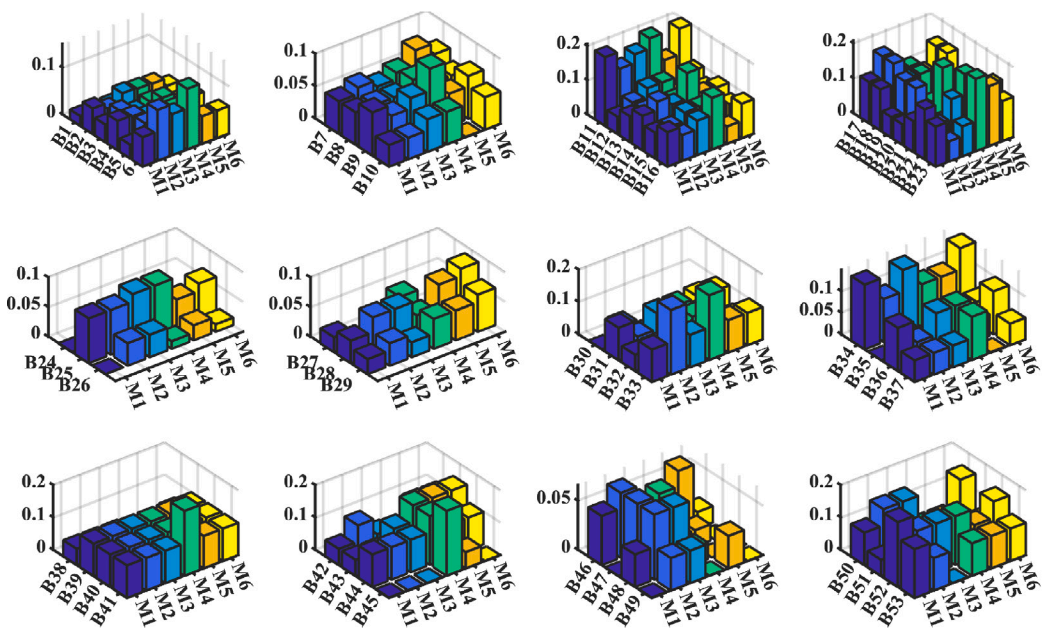


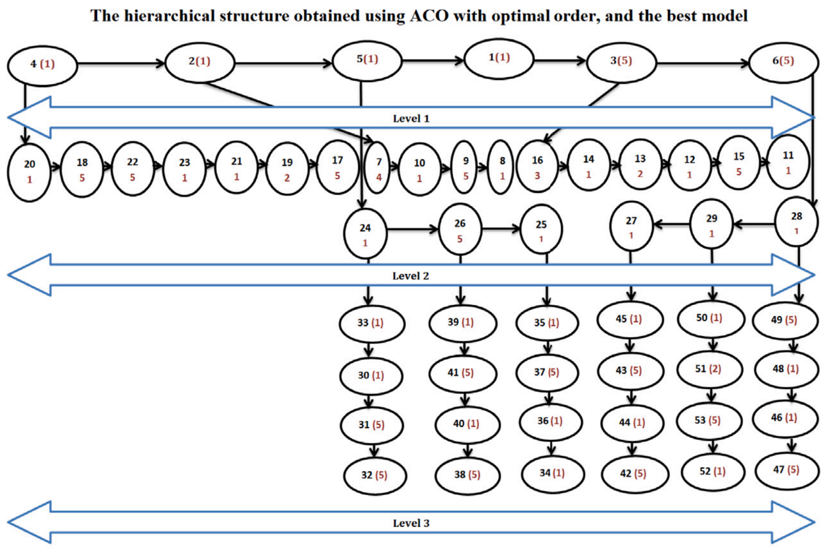
Fig. 8. The probability of a false negative for the model used in each block classifier in the hierarchical structure does not exceed 0.2. It estimates the error probability of the validation dataset. B1 to B53 — Individual block classifier, M1 — Random Forest Ensemble, M2 — Decision Tree, M3 — Gradient Boosting, M4 — K-Nearest Neighbor, M5 — Multilayer Perceptron, M6 — Adaboost.

Table 1
Overall probability of detection for benchmark data set.

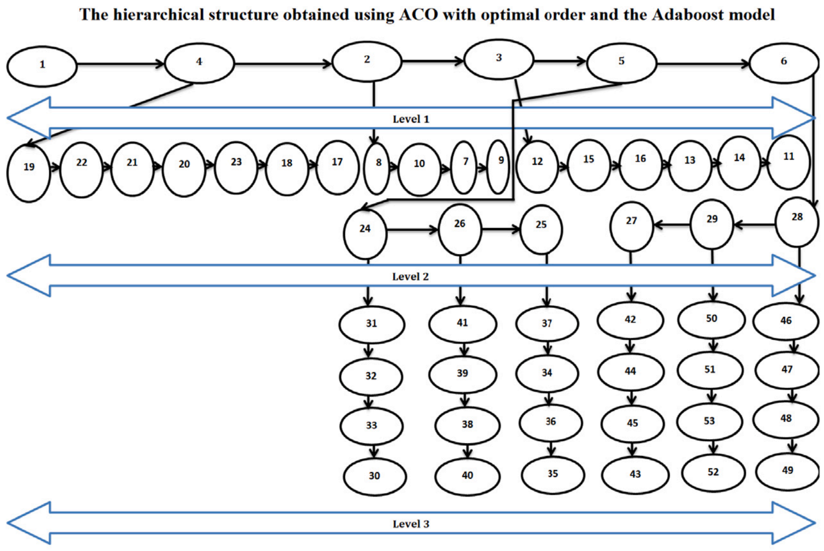
Model	Techniques	Testing
Default order with same model in all blocks		
1	Random Forest Ensemble(M_1)	87.23%
2	Decision Tree(M_2)	84.93%
3	Gradient Boosting(M_3)	86.73%
4	K-Nearest Neighbor(M_4)	82.63%
5	Multilayer Perceptron(M_5)	86.73%
6	Proposed technique using ACO	**89.02%**
7	Proposed technique using ACO with Adaboost	**90.92%**

optimal order (along with the selected models assigned to the classifiers or Adaboost combiner) obtained using the proposed ACO algorithm. The objective function can be reformulated depending on the metric (say, F1 score) that needs to be maximized.

Contrasting previous work for the application engine vibration data: Several studies have proposed a framework to classify mechanical faults in rotating machines. In the method proposed by Pestana Viana et al. (2016), multi-layer perceptron with one hidden layer of 35 neurons and a 31 feature element were proposed to classify three mechanical faults: normal, imbalance, and misalignment. In the method proposed by Ali et al. (2019), Multi-Layer Perceptron with one hidden layer (ten neurons), and 25 feature vectors are used to obtain 96.2% accuracy



(a)



(b)

Fig. 9. The number in red color indicates the best model in each block ($1 \rightarrow M_1$, $2 \rightarrow M_2$, $3 \rightarrow M_3$, $4 \rightarrow M_4$ and $5 \rightarrow M_5$). The numbers from 1 to 53 represent the block classifier at each level. (a) Final hierarchical structure after applying Ant colony optimization with validation data (b) Final hierarchical structure after using Ant colony optimization with Adaboost in the validation data.

Table 2
Comparison of level 1 probability of detection attained using the proposed technique and the method proposed in Ali et al. (2019).

Major class	Type of classes	Existing work	Default hierarchical structure					Proposed method	
		Ali et al. (2019)	M1	M2	M3	M4	M5	ACO	ACO and AdaBoost
1	Normal	167/167	165/167	164/167	164/167	162/167	165/167	164/167	165/167
2	Horiz. Misal.	161/167	164/167	164/167	165/167	165/167	165/167	164/167	164/167
3	Vert. Misal.	158/167	136/167	147/167	149/167	146/167	149/167	162/167	164/167
4	Imbalance	164/167	146/167	145/167	151/167	149/167	151/167	159/167	157/167
5	Overhange	153/167	130/167	117/167	122/167	123/167	121/167	161/167	160/167
6	Underhang	161/167	115/167	90/167	111/167	104/167	111/167	161/167	163/167
	Over all percentage	96.2%	85.43%	82.53%	86.03%	84.73%	86.03%	**96.91%**	**97.11%**

for classifying into six basic classes of the vibration database. SMOTE techniques were used to balance the data, which may lead to the trained model overfitting.

The proposed technique in our paper is analyzed using the Monte carlo simulation and is implemented in engine vibration applications to check the classification performance. The adaptive boosting strategy

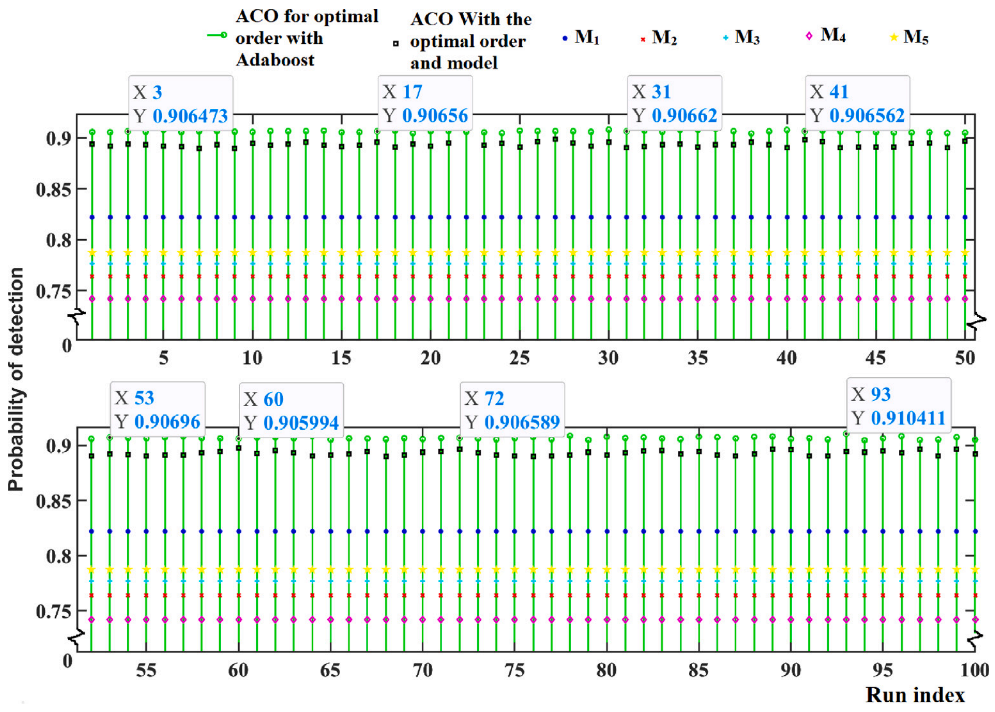


Fig. 10. The experimental result with the machinery fault database compares the overall probability of detection attained using the default hierarchical structure with identical classifiers assigned to each classifier block and the hierarchical optimal blocks with the specific classifier model achieved using ACO. Also, The hierarchical optimal order was attained using ACO with the Adaboost technique for 100 Runs (calculated p_F and p_M for the individual blocks using validation data).

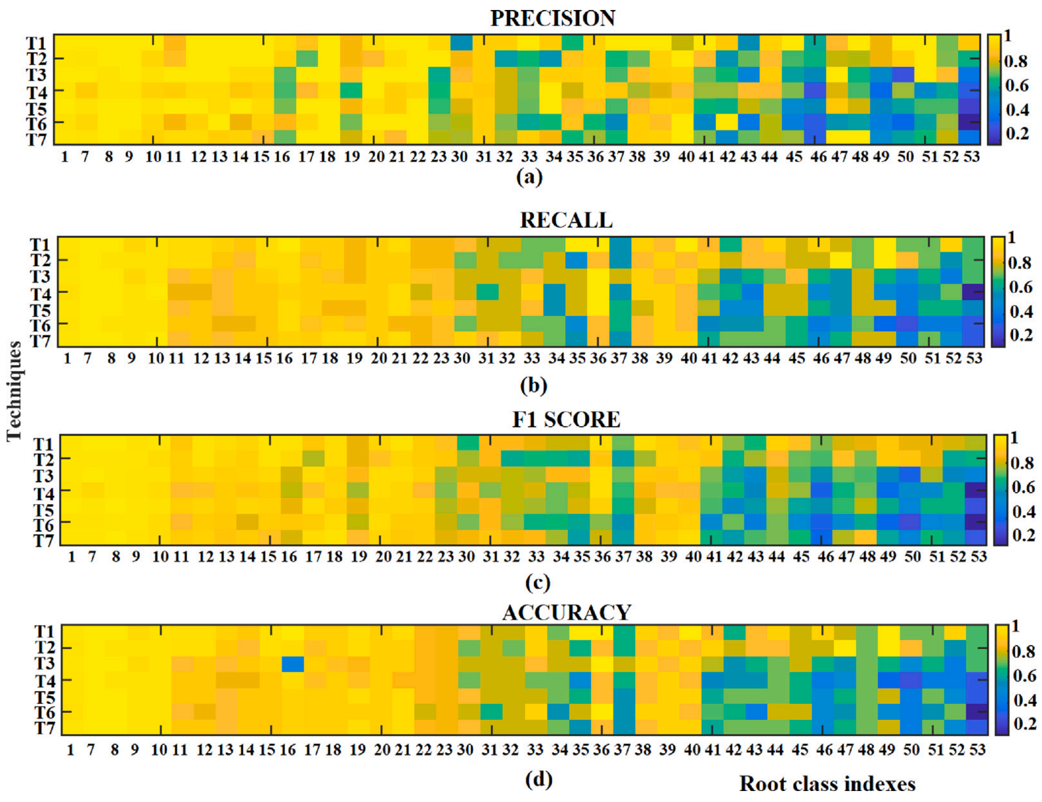


Fig. 11. Test data performance evaluation using precision, recall, F1 score, and accuracy metrics. T1 — ACO with Adaboost, T2 – Optimal Model with ACO, T3 — Random Forest Ensemble (M1), T4 — Decision Tree (M2), T5 — Gradient Boosting(M3), T6 — K-Nearest Neighbor (M4), T7 — Multilayer Perceptron (M5).

with ACO in hierarchical classification achieved a score of 97.11% on our developed technique (Table 2). Adaboost with ant colony optimization performs well with an increase in the number of classes and levels in the hierarchy. The machinery fault data set balances the classes by employing autoencoder to generate augmented data. The proposed network for data augmentation uses 9 attributes as input. The overall probability of detection of 90.92% is achieved for classifying all the root to leaf node classifiers of the hierarchical structure.

Algorithm 3 Algorithm to classify the arbitrary class label i (say) into one of the root-to-leaf classes using the hierarchical classifiers (continuation).

```
 1: for j ∈ set of second-level indices corresponding to the first level identified index $\hat{f}$ do. If the corresponding second-level indices are empty, declare $\hat{f}$ as the identified class index.
 2:     if $s \neq j$ then
 3:         Generate $r$
 4:         if $r > (1 - p_{jFk_j})$ then
 5:             Declare the input classifier index as $j$. Exit loop
 6:         end if
 7:     else if $s == j$ then
 8:         Generate $r$.
 9:         if $r < (1 - p_{jMk_j})$ then
10:             Declare the second level class index as $j$. Exit loop.
11:         end if
12:     end if
13: end for
14: The corresponding last index is identified as the second-level index.
15: if the identified second level index ∈ root classifier then,declare the second level index as the class index.
16:     Similarly, the corresponding third-level index (if available) is identified and declared as the class index.
17: end if
```

Thus, in the existing work, 25 traits are used as input, and classification is performed for six root classes at level 1. The proposed method detects the faulty machinery database with reasonable accuracy. Using ACO, the deepest class in the hierarchy level has a high detection rate compared with the default hierarchical classification structure, evident in the metrics shown in Fig. 11.

6. Conclusion and future direction

This paper demonstrates the importance of optimizing the order in which the blocks are arranged in the hierarchical classifier structure using the engine vibration application (Machinery fault database) with three levels. It is observed that the ACO-based Adaboost technique performs well, with an overall accuracy of 1.9% improvement compared to ACO without Adaboost and an improvement in the overall accuracy of 3.6% compared with the best among other methods without ACO and Adaboost. The proposed algorithm is also validated using Monte carlo simulation. Exhaustive experiments using benchmark datasets with more levels reveal the robustness of the proposed algorithm. The proposed work uses the statistical parameter mean calculated from eight individual sensor outcomes and the corresponding rpm as one of the feature elements as input to train the classifier. The detection rate can be improved if other statistical parameters, along with the mean, are collected from the individual sensors. Developing the algorithm to optimize the order with an imbalanced data set without augmented data is also suggested.

CRediT authorship contribution statement

Vinodha K.: Methodology/Study design, Software, Investigation, Resources, Data curation, Writing – original draft. **E.S. Gopi:** Conceptualization, Validation, Writing – review, Visualization, Supervision.

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed.

We further confirm that the order of authors listed in the manuscript has been approved by all of us. We understand that the Corresponding Author is the sole contact for the Editorial process (including Editorial Manager and direct communications with the office). She is responsible for communicating with the other authors about progress, submissions of revisions and final approval of proofs.

Data availability

Data will be made available on request

Appendix. Closed-form expression for the computation of overall probability of detection for the given hierarchical binary structure

The following is a typical computation of the overall probability of detection for hierarchical blocks stacked, as shown in Fig. 1 Let p_{Fk_i} represent the probability of false positive of the ith classifier using the $k_i th$ technique. Similarly, p_{Mk_i} represents the probability of false negative of the ith classifier using the $k_i th$ technique. Assuming that; the prior equal probabilities, the overall probability of detection is computed as follows.

$$p_{k_i}(i/i) = (1 - p_{Fk_{i-1}})(1 - p_{Fk_{i-2}}) \dots (1 - p_{Fk_{i-i}})(1 - p_{Mk_i}) \quad \forall i = 1, 2, 3, \dots, 6 \tag{A.1}$$

$$p_{k_i}(i/i) = \prod_{j=1}^{2}(1 - p_{Fk_j})(1 - p_{Fk_{i-1}})(1 - p_{Fk_{i-2}}) \dots (1 - p_{Fk_{i-n}})(1 - p_{Mk_i}) \forall i = 7, 8, 9, 10, \forall n = i - 7 \tag{A.2}$$

$$p_{k_i}(i/i) = \prod_{j=1}^{3}(1 - p_{Fk_j})(1 - p_{Fk_{i-1}})(1 - p_{Fk_{i-2}}) \dots (1 - p_{Fk_{i-n}})(1 - p_{Mk_i}) \ \forall i = 11, 12, 13, \dots, 16, \forall n = i - 11 \tag{A.3}$$

$$p_{k_i}(i/i) = \prod_{j=1}^{4}(1 - p_{Fk_j})(1 - p_{Fk_{i-1}})(1 - p_{Fk_{i-2}}) \dots (1 - p_{Fk_{i-n}})(1 - p_{Mk_i}) \forall i = 17, 18, 19, \dots, 23, \forall n = i - 17 \tag{A.4}$$

$$p_{k_i}(i/i) = \prod_{j=1}^{4}(1 - p_{Fk_j})(1 - p_{5Mk_5})(1 - p_{Fk_{i-1}})(1 - p_{Fk_{i-2}}) \dots (1 - p_{Fk_{i-n}})(1 - p_{Mk_i}) \forall i = 24, 25, 26, \forall n = i - 24 \tag{A.5}$$

$$p_{k_i}(i/i) = \prod_{j=1}^{4}(1 - p_{Fk_j})(1 - p_{Mk_5})(1 - p_{Fk_{i-1}})(1 - p_{Fk_{i-2}}) \dots (1 - p_{Fk_{i-n}})(1 - p_{Mk_i}) \forall i = 27, 28, 29, \forall n = i - 27 \tag{A.6}$$

$$p_{k_i}(i/i) = \prod_{j=1}^{4}(1 - p_{Fk_j})(1 - p_{Mk_5})(1 - p_{Mk_{24}})(1 - p_{Fk_{i-1}})(1 - p_{Fk_{i-2}}) \dots (1 - p_{Fk_{i-n}})(1 - p_{Mk_i}) \forall i = 30, \dots, 33, \forall n = i - 30 \tag{A.7}$$

$$p_{k_i}(i/i) = \prod_{j=1}^{4}(1 - p_{Fk_j})(1 - p_{Mk_5})(1 - p_{Fk_{24}})(1 - p_{Mk_{25}})(1 - p_{Fk_{i-1}}) (1 - p_{Fk_{i-2}}) \dots (1 - p_{Fk_{i-n}})(1 - p_{Mk_i}) \forall i = 34, \dots, 37, \forall n = i - 34 \tag{A.8}$$

$$p_{k_i}(i/i) = \prod_{j=1}^{4}(1 - p_{Fk_j})(1 - p_{Mk_5}) \prod_{l=24}^{25}(1 - p_{Fk_l})(1 - p_{Mk_{26}})(1 - p_{Fk_{i-1}}) (1 - p_{Fk_{i-2}}) \dots (1 - p_{Fk_{i-n}})(1 - p_{Mk_i}) \forall i = 38, \dots, 41, \forall n = i - 38 \tag{A.9}$$

$$p_{k_i}(i/i) = \prod_{j=1}^{4}(1-p_{Fk_j})(1-p_{Mk_5})(1-p_{Mk_{27}})(1-p_{Fk_{i-1}})(1-p_{Fk_{i-2}}) \ldots (1-p_{Fk_{i-n}})(1-p_{Mk_i}) \forall \mathrm{i} = 42,\ldots,45, \forall \mathrm{n} = i-42 \tag{A.10}$$

$$p_{k_i}(i/i) = \prod_{j=1}^{4}(1-p_{Fk_j})(1-p_{Mk_5})(1-p_{Fk_{27}})(1-p_{Mk_{28}})(1-p_{Fk_{i-1}}) (1-p_{Fk_{i-2}}) \ldots (1-p_{Fk_{i-n}})(1-p_{Mk_i}) \forall \mathrm{i} = 46,\ldots,49, \forall \mathrm{n} = i-46 \tag{A.11}$$

$$p_{k_i}(i/i) = \prod_{j=1}^{4}(1-p_{Fk_j})(1-p_{Mk_5}) \prod_{l=27}^{28}(1-p_{Fk_l})(1-p_{Mk_{29}})(1-p_{Fk_{i-1}}) (1-p_{Fk_{i-2}}) \ldots (1-p_{Fk_{i-n}})(1-p_{Mk_i}) \forall \mathrm{i} = 50,\ldots,53, \forall \mathrm{n} = i-50 \tag{A.12}$$

References

Ali, M. A., Bingamil, A. A., & Jarndal, A. (2019). The influence of handling imbalance classes on the classification of mechanical faults using neural networks. In *8th int. conf. on modeling simulation and applied optimization (ICMSAO)*. http://dx.doi.org/10.1109/ICMSAO.2019.8880437.

Aref, E., Jafar, M., & Mohammadreza, A. (2021). Fault detection and classification for photovoltaic systems based on hierarchical classification and machine learning technique. *IEEE Transactions on Industrial Electronics*, *68*, 12750–12759. http://dx.doi.org/10.1109/TIE.2020.3047066.

Carlos, N., Silla, J., & Alex, A. F. (2010). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, *22*, 31–72. http://dx.doi.org/10.1007/s10618-010-0175-9.

Defiyanti, S., Winarko, E., & Priyanta, S. (2019). A survey of hierarchical classification algorithms with big-bang approach. In *5th international conference on science and technology (ICST)*. http://dx.doi.org/10.1109/ICST47872.2019.9166313.

Dong, Z., & Lili, Z. (2022). A multi-feature fusion-based domain adversarial neural network for fault diagnosis of rotating machinery. *Measurement*, *200*, http://dx.doi.org/10.1016/j.measurement.2022.111576.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. In *IEEE computational intelligence magazine, Vol. 1* (pp. 28–39). http://dx.doi.org/10.1109/MCI.2006.329691.

Fafa, C., Baoping, T., Tao, S., & LiLi (2014). Multi-fault diagnosis study on roller bearing based on multi-kernel support vector machine with chaotic particle swarm optimization. *Measurement*, *47*, 576–590. http://dx.doi.org/10.1016/j.measurement.2013.08.021.

Fogli, D., Guida, G., Redolfi, M., & Tonoli, R. (2020). A knowledge-based approach to hierarchical classification: A voting metaphor. *Expert Systems with Applications*, *161*, http://dx.doi.org/10.1016/j.eswa.2020.113737.

Gopi, E. S. (2020). *Pattern recognition and computational intelligence techniques using Matlab* (pp. 167–186). Georgia, USA: Springer.

Gunaseelan, J., Gopi, E. S., & Subbu (2018). Ant colony technique for optimizing the order of cascaded svm classifier for sunflower seed classification. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*, 78–88. http://dx.doi.org/10.1109/TETCI.2017.2772918.

Jia, G., Zhaoqi, X., & Chenyang, Y. (2021). Learning fairly with class-imbalanced data for interference coordination. *IEEE Transactions on Vehicular Technology*, *7*, 7176–7181. http://dx.doi.org/10.1109/TVT.2021.3080678.

Jianming, L., Jiajie, Z., Jintao, L., & Zhenguo, Y. (2019). ACE: Ant colony based multi-level network embedding for hierarchical graph representation learning. *IEEE Access*, *7*, 73970–73982. http://dx.doi.org/10.1109/ACCESS.2019.2920671.

Mafaulda Machinery fault database. online. available. http://www02.smt.ufrj.br/~offshore/mfs/page_01.html.

Marwa, C., Bouhamed, A. C. S. A., Solaiman, I. K. K. B., & Derbel, H. (2020). Binary hierarchical multiclass classifier for uncertain numerical features. In *5th international conference on advanced technologies for signal and image processing (ATSIP)*. http://dx.doi.org/10.1109/ICMLA52953.2021.00125.

Muhammet, U., Mustafa, O., Mustafa, D., & Haluk, K. (2014). Fault diagnosis of rolling bearings using a genetic algorithm optimized neural network. *Measurement*, *58*, 187–196. http://dx.doi.org/10.1016/j.measurement.2014.08.041.

Neema, M., & Gopi, E. S. (2021). Data driven approach for mmwave channel characteristics prediction using deep neural network. *Wireless Personal Communications*, *120*, 2161–2177. http://dx.doi.org/10.1007/s11277-021-08768-7, springer.

Nicholas, H., & Alex, A. F. (2008). *Lecture notes in computer science*: *vol. 4973*, *Improving the performance of hierarchical classification with swarm intelligence*. Springer, http://dx.doi.org/10.1007/978-3-540-78757-0_5.

Pestana Viana, D., Zambrano-Lopez, R., Lima, A. A. d., de, T., Prego, M., Netto, S. L., & Silva, E. A. B. d. (2016). The influence of feature vector on the classification of mechanical faults using neural networks. In *IEEE 7th latin american symposium on circuits and systems (LASCAS)*. http://dx.doi.org/10.1109/LASCAS.2016.7451023.

Ricardo, C., Rodrigo, C., Barros, & Carvalho, A. C. (2014). Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, *80*, 39–56. http://dx.doi.org/10.1016/j.jcss.2013.03.007.

Roger, A. S., Jaques, P. A., & aJoão Francisco Valiati 2019. An analysis of hierarchical text classification using word embeddings, 471, 216–232. http://dx.doi.org/10.1016/j.ins.2018.09.001.

Salabat, K., Abdul, & Rauf, B. (2017). Ant colony optimization based hierarchical multi-label classification algorithm. *Applied Soft Computing*, *55*, 462–479. http://dx.doi.org/10.1016/j.asoc.2017.02.021.

Singhal, M., Hegde, S. V., Makam, R., & George, K. (2020). A hierarchical approach for multi-class galaxy classification. In *IEEE 17th India council international conference (INDICON)*. http://dx.doi.org/10.1109/INDICON49873.2020.9342441.

Victor, O. C., & Rodrigues, C. R. (2018). Hierarchical ant colony for simultaneous classifier selection and hyperparameter optimization. http://dx.doi.org/10.1109/CEC.2018.8477834.

Wei, B., & James, T. K. (2014). Mandatory leaf node prediction in hierarchical multilabel classification. *IEEE Transactions on Neural Networks and Learning Systems*, *12*, 2275–2287. http://dx.doi.org/10.1109/TNNLS.2014.2309437.

Xu, Q., Wu, Z., Yang, Y., & Zhang, L. (2017). The difference learning of hidden layer between autoencoder and variational autoencoder. In *29th Chinese control and decision conference (CCDC)*. http://dx.doi.org/10.1109/CCDC.2017.7979344.

YaoweiShi, Deng, A., Deng, M., Xu, M., Liu, Y., & XueDing (2022). A novel multiscale feature adversarial fusion network for unsupervised cross-domain fault diagnosis. *Measurement*, *200*, http://dx.doi.org/10.1016/j.measurement.2022.111616.

Youwei, W., & Lizhou, F. (2021). An adaptive boosting algorithm based on weighted feature selection and category classification confidence. *Applied Intelligence*, *51*, 6837–6858. http://dx.doi.org/10.1007/s10489-020-02184-3.

Zhiqiang, W., Yazhou, Z., & He, H. (2017). Variational autoencoder based synthetic data generation for imbalanced learning. In *IEEE symposium series on computational intelligence (SSCI)*. http://dx.doi.org/10.1109/SSCI.2017.8285168.

Zulfiqar, A., & Waseem, S. (2018). EPACO: a novel ant colony optimization for emerging patterns based classification. *Cluster Computing*, *21*, 453–467. http://dx.doi.org/10.1007/s10586-017-0894-4.

Vinodha K. received a bachelor's degree in electronics and communication engineering from Saranathan College of Engineering, Tiruchirappalli, affiliated with Anna University-Chennai, India, and a master's degree in medical electronics from the College of Engineering, Guindy, Anna University-Chennai, India. She is pursuing PhD at the Department of Electronics and Communication Engineering, National Institute of Technology, Tiruchirappalli, India. Her research interests include data acquisition, machine learning, deep learning, computational intelligence, pattern recognition, and sensors.

Dr. E.S. Gopi has solely authored 8 books and edited 1 book published by Springer in the area of signal processing and pattern recognition. He has got several research papers published in the reputed journals, reviewed book chapters and conference proceedings. He has 25 years of teaching and research experience. He is the coordinator for pattern recognition and the computational intelligence laboratory. He is currently Associate Professor, Department of Electronics and Communication Engineering, National Institute of Technology, Trichy (Government of India). His books are widely used all over the world. His book on "pattern recognition and Computational intelligence using Matlab" , Springer, was recognized as one of the best eBook under the "pattern recognition" and "Matlab" categories by the Book authority, Authorities' leading site for book recommendations by thought leaders. He is the series editor for the series "Signals and Communication Technology", a Springer publication. He has completed the project offered by GTRE (DRDO) as the principal investigator on "Hunting representative sensors and constructing regression model for between sensor outcomes using ML". His video course on "Pattern recognition", "Statistical theory of Communication" and "Linear algebra and stochastic process" are well appreciated by his fellow students. He is also serving as one of the Workshops, Tutorials & Symposia officers for Machine Learning for Communication Emerging Technologies Initiative (IEEE ComSoc). His research interests include Machine intelligence, pattern recognition, and statistical signal processing and Computational intelligence.