# Which classifiers are connected to others? An optimal connection framework for multi-layer ensemble systems.

DANG, T., NGUYEN, T.T., LIEW, A.W.-C., ELYAN, E. and MCCALL, J.

2024

Knowledge-Based Systems 304 (2024) 112522

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

# Which classifiers are connected to others? An optimal connection framework for multi-layer ensemble systems

Truong Dang [a], Tien Thanh Nguyen [a,*], Alan Wee-Chung Liew [b], Eyad Elyan [c], John McCall [a]

[a] National Subsea Centre, Robert Gordon University, Aberdeen, UK
[b] School of Information and Communication Technology, Griffith University, Gold Coast, Australia
[c] School of Computing, Robert Gordon University, Aberdeen, UK

ARTICLE INFO

*Keywords:*
Ensemble learning
Ensemble selection
Classifier selection
Ensemble of classifiers
Differential evolution

ABSTRACT

Ensemble learning is a powerful machine learning strategy that combines multiple models e.g. classifiers to improve predictions beyond what any single model can achieve. Until recently, traditional ensemble methods typically use only one layer of models which limits the exploration of different aspects in the classifiers' predictions. On the other hand, the rise of deep learning has introduced multi-layer architectures that can learn complex functions by transforming data into multiple levels of representation. This characteristic of deep learning suggests that multi-layer ensembles may potentially provide better performance compared to single-layer ensembles. However, a problem which might arise is that in the subsequent layers, not all the inputs to a classifier are desirable, leading to lower performance. In this paper, we introduce a novel multi-layer ensemble of classifiers named COME in which each classifier at a specific layer is connected to multiple classifiers in the previous layer. These connections signify the use of the previous-layer-classifiers' outputs as inputs for training the current layer's classifier. Each classifier can be connected to different classifiers in the previous layer, which allows inputs in each layer to be optimally selected. We propose a binary encoding scheme to encode the topology of the proposed multi-layer ensemble with defined connections between layers. Differential Evolution, a popular evolutionary computation method, is used as the optimisation algorithm to search for the optimal set of connections. Experimental results on 30 datasets from the UCI Machine Learning Repository and OpenML demonstrate that our proposed ensemble outperforms many state-of-the-art ensemble learning algorithms.

## 1. Introduction

Machine learning is a field of computer science focusing on creating data-driven algorithms that can perform tasks at a level comparable to humans. It is important to note that each machine learning model has its own strengths and limitations, and the choice of the most suitable model depends on the characteristics of the data and the specific task at hand. One approach to enhance predictions is to combine diverse machine learning models into an Ensemble of Classifiers (EoC). Many studies have shown that an EoC can achieve better results compared to using a single classifier. The past decades have witnessed many successes of ensemble learning on international machine learning competitions, such as Kaggle and KDD-Cups [1]. Ensemble learning leverages the diversity and collective intelligence of multiple models to improve accuracy and robustness, making it an effective strategy for solving complex problems.

There are three stages in designing an EoC, namely ensemble generation, ensemble combination, and ensemble selection. In ensemble generation, the classifiers are generated by either training different algorithms on the same training set (heterogeneous ensemble method) or training a single algorithm on many different training sets (homogeneous ensemble method) [2]. Afterward, an ensemble combination method is used to combine the output of the different models to obtain the final result [3]. Ensemble selection, an intermediate stage, is performed to choose a subset of diverse classifiers from a larger pool to form the final ensemble. The goal is to improve performance by selecting models that complement each other, allowing for the construction of an ensemble that optimally combines the strengths of different models. The selection process can be performed using a heuristic criterion or using an optimisation algorithm.

Traditionally, the majority of ensemble learning methods have been limited to utilising just a single layer of ensembles. In recent years, deep

* Corresponding author.
*E-mail address:* t.nguyen11@rgu.ac.uk (T.T. Nguyen).

https://doi.org/10.1016/j.knosys.2024.112522
Received 14 June 2024; Received in revised form 2 September 2024; Accepted 13 September 2024
Available online 28 September 2024

learning has achieved great success in multiple areas. Deep learning is based on layer-by-layer processing, in which the original data is transformed into multiple levels of representation. This allows deep learning models to learn very complex relationships within the data. This consequently means that using multiple layers of ensembles can potentially achieve better results compared to traditional single-layer ensembles. Multi-layer ensembles involve stacking multiple layers of EoCs, with each layer contributing to the final prediction. This hierarchical structure allows for capturing complex relationships in the data by combining models with different levels of abstraction [4]. Recognising the need to enhance the performance of multi-layer ensembles, it is acknowledged that ensemble selection is necessary for identifying the optimal structure tailored to a specific dataset. It is also necessary to design a multi-layer ensemble that would facilitate diversity amongst the base classifiers in each layer since it is known that diversity is very important in creating an effective ensemble system.

In this paper, we propose a novel optimal connection framework for multi-layer heterogeneous ensemble systems named COME. The main contributions of our work are as follows:

- In existing multi-layer EoC approaches, classifiers in a layer are typically trained using predictions from all classifiers in the preceding layer [2]. However, this method may not always be effective, as different learning algorithms should ideally be trained on different data. In this study, we propose a novel multi-layer EoC, where each classifier in a specific layer is connected to a subset of classifiers from the previous layer. The connection between a classifier in one layer and those in the previous layer implies using the output from the preceding-layer classifier as input for training the current-layer classifier. At each layer, each classifier is trained solely on the inputs from its connected classifiers. The process continues through all layers of the multi-layer ensemble until reaching the last layer. The predictions of the EoC in the final layer are combined using the Sum Rule, a simple but popular combining method, to generate the final prediction. To our best knowledge, our work is the first approach to investigating the concept of connections in multi-layer ensemble systems.
- We propose a binary encoding to represent the topology of the multi-layer ensemble with the defined connections. As the number of connections to each classifier at each layer may vary, the length of the encoding at each layer also varies accordingly. The encoding for the entire multi-layer ensemble is generated by concatenating the encodings for each layer. The search process iterates through multiple layers until a predefined threshold is reached. The optimal encoding obtained from the last layer is used to train the multi-layer ensemble.
- We conducted experiments on 30 datasets from the UCI Machine Learning repository. The proposed ensemble is compared to several popular benchmark algorithms including multi-layer ensemble and state-of-the-art ensemble methods. We used 3 popular nature-based optimisation methods namely Differential Evolution (DE), Genetic Algorithm (GA) and Particle Swarm Optimisation (PSO) to search for optimal ensemble topology. The results show that DE is a better choice to find optimal topology for the proposed ensemble compared to GA and PSO. We then show the effectiveness of our proposed method compared to several popular benchmark algorithms.

The paper is organised as follows. In Section 2, we briefly review the existing approaches in ensemble learning and topology optimisation for deep neural networks. In Section 3, we give a detailed description of the proposed method. Experimental studies on 30 datasets are provided in Section 4, followed by conclusions in Section 5.

## 2. Background and related work

### 2.1. Ensemble learning

Ensemble learning refers to a sub-field of supervised machine learning algorithms in which the decisions of a number of classifiers are combined to improve the final prediction. In recent decades, the use of ensemble learning has significantly increased across multiple fields such as engineering, physics, earth sciences, and planetary sciences. It is known that ensemble learning-based algorithms usually achieve better classification results compared to single machine learning algorithms [1], and ensemble learning was used in many best solutions in data science and machine learning competitions [5,6].

With the successes of deep learning models since 2013, there has been a growing number of works on building an ensemble of deep learning models. Calisto et al. proposed AdaEn-Net [7], which is an ensemble of deep 2D-3D fully convolutional networks for medical image segmentation. In [8], the authors proposed a novel Monte-Carlo-based ensemble of 2D CNNs to learn the 3D relevance of the features across multiple slices of brain images and showed that the proposed method is competitive on several brain image datasets. He et al. [6] proposed ResNet, a deep residual learning-based neural network that can extend deeply into hundreds of layers. The ensemble of these networks achieved an error rate of 3.57% on the ImageNet dataset, ranking first place in the ILSVRC 2015 competition. Xie et al. [9] proposed a stacked ensemble of 18 1D and 2D CNN models based on the classic LeNet5 framework for daily runoff predictions. Dvornik et al. [10] explored the application of ensemble methods in few-shot classification, proposing an ensemble of deep networks that assesses classifier variance. They also introduced new strategies to promote a cooperative learning scheme among networks while preserving diversity in predictions.

On the other hand, one of the most popular approaches in ensemble learning is Stacking, in which classifiers first predict on the training data, and then a meta-classifier i.e. combiner would be used to learn from these predictions to output the final predictions [2]. It is noted that research on Stacking only used one layer of ensemble, and this has a restriction on the ability to harness the predictions of classifiers on the training data. In recent years, deep neural networks have achieved great successes in multiple areas. Deep neural networks rely on composing multiple layers of functions to create increasingly complex representations of the data. Based on this observation, there has been significant interest in developing multi-layer ensembles, which is a generalisation of stacking, drawing inspiration from the mechanisms of DNNs. By exploring predictions through several layers (creating several representations of predictions), it is expected to capture more information from the predictions to train the combiner compared to the traditional Stacking algorithm.

One of the first multiple layer ensembles is gcForest [4], a deep cascade of completely random trees and random forests in which the output of each layer is used as the input to the next layer, and new layers are automatically added until the result does not improve compared to the last layer. A weight average approach for gcForest was proposed by Utkin et al. [11] to combine class distribution vectors. The weight vectors of the trees in a forest in one layer are found by minimising the distance between the class label vector in a binary encoding scheme and the weighted prediction vector of this forest. To reduce the number of weight vectors, the authors grouped the class distribution vectors of the trees and set a weight vector for each group. Nguyen et al. [2] proposed a novel multi-layer heterogeneous ensemble that automatically selects the best classifiers and features in each layer. The selection process at each layer is modelled as a bi-optimisation problem, with the objectives being the classification accuracy and the diversity of the ensemble at the current layer. A two-layer ensemble of deep learning models for medical image segmentation was proposed by Dang et al. [12], in which the predictions of each pixel for each training image produced by the first layer's models are used as augmented data for the models in the second layer. The predictions of the second layer are then combined using a weighted approach. Luong et al. [13] proposed a multi-layer ensemble for the data stream setting and used a dynamic Genetic Algorithm (GA) to develop an online ensemble selection method to find the optimal subset of classifiers at each layer.

### 2.2. Ensemble selection and optimisation approaches

Ensemble selection refers to an intermediate stage in the ensemble design process in which classifiers that contribute the most to the overall result are selected, leading to better performance. There are several approaches in ensemble selection, namely ordering-based, optimisation-based, and dynamic techniques. Ordering-based ensemble selection methods try to order the classifiers based on ranking criteria, such as validation error. In [14], the authors proposed the Complementariness measure which chooses new classifiers that have the highest prediction accuracy over the set of instances misclassified by the chosen classifiers so far. Cao et al. [15] proposed Discriminant classifier pruning (DISC), which maximises the relevancy between correct decisions made by the candidate classifier and the chosen classifiers so far and maximises the relevancy between correct decisions by the candidate classifier and target classifier in terms of misclassified instance made by currently selected classifier subset simultaneously. In [16], the authors proposed Margin and Diversity-based Ensemble Pruning (MDEP), which is based on a heuristic measure for evaluating the importance of each classifier in the ensemble that explicitly considers both sample margins and ensemble diversity.

Optimisation-based ensemble selection methods formulate the ensemble selection process as an optimisation problem, which is then solved by heuristic optimisation or mathematical programming. Nguyen et al. proposed MULES [2], which is a multi-layer ensemble system that performs both classifier and feature selection at each layer. The optimal configuration of each layer is found by formulating a bi-objective optimisation problem, with the objectives being classification accuracy and ensemble diversity. Dang et al. [17] proposed a weighted ensemble of deep learning models, in which the optimal weights are found by using Comprehensive Learning Particle Swarm Optimisation (CLPSO), and the Dice coefficient, a popular performance metric for image segmentation, is used as the fitness criteria. In [18], the authors proposed an ensemble selection algorithm that simultaneously selects the optimal set of meta-data and features to be used for each classifier using Ant Colony Optimisation (ACO). In [19], the authors introduced the multi-objective semi-supervised classifier ensemble (MOSSCE) approach for classification of high-dimensional data with limited labels. Firstly, the optimal combination of feature subspaces is generated using a multiobjective subspace selection process based on three objective criteria. Secondly, an auxiliary training set is generated based on the sample confidence. Finally, the training set and the auxiliary training set are used to select the optimal subset of the ensemble. In [20], the authors proposed a Multiview optimization (MVO) based ensemble system for imbalanced data. Multiple subviews are generated from the data, combined with a new evaluation criterion ensemble selection based on subviews, and an oversampling approach to obtain a new class rebalanced subset for the classifier.

In dynamic ensemble selection, an EoC is selected differently for each test sample based on the competence level of the classifiers, which are calculated according to some criteria on a local region [21]. In [22], the authors proposed a novel dynamic ensemble selection strategy based on Error Correcting Output Code by matching each column in the coding matrix with a set of feature subsets generated by various feature selection methods. In the decoding process, a criterion based on data complexity theory is used to select the optimal feature subset of the ensemble. Garcia et al. [23] proposed DES-MI, a novel dynamic ensemble selection method which can handle multi-class imbalanced datasets. DES-MI performs preprocessing to rebalance the dataset using random balance, and uses a weighting mechanism to highlight the competence of classifiers that are more powerful in classifying samples in the region of underrepresented competence. In [24], the authors proposed to incorporate dynamic ensemble selection, an adaptive technique for managing imbalanced multiclass data streams, and a K-nearest neighbor (KNN) algorithm-based concept drift detector, to improve the classification of imbalanced multiclass drifted data streams. While the adaptive oversampling method ensures that the data is not imbalanced, KNN is used to make sure that the generated samples do not overlap, and the classifiers are selected dynamically based on incoming data by using the concept drift detector. Luong et al. [13] proposed a streaming multi-layer dynamic ensemble selection algorithm based on a dynamic GA and dropout and demonstrated the effectiveness of the proposed ensemble on an insect stream classification dataset. Zhu et al. [25] proposed MLDE, a novel dynamic ensemble selection algorithm for multi-label classification. Initially, classifiers are generated by partitioning the original multi-label dataset into single-label datasets, with individual classifiers constructed for each label. Then, ranking loss, single-label, and multi-label accuracy are used to improve the model's label correlation exploiting abilities. Finally, a novel integration mechanism is proposed that fuses the outputs of the base classifiers.

### 2.3. Topology optimisation for deep neural networks

There have been many approaches in topology optimisation for DNNs which aim to search for optimal configuration of DNNs for a particular dataset. In reinforcement learning-based approaches, an agent executes an action at a specific timestep to sample a new candidate DNN, and the network's performance on a validation set is used as a reward to update the agent [26]. An example is [27] in which the authors used a RNN (Recurrent Neural Network) policy to sample a string that sequentially encodes the DNN. Another approach is Bayesian optimisation, which constructs a probability model between DNN architecture and its performance by using a training dataset of sampled architectures. At each step, the next promising candidate is selected using a probability model, then the model is updated based on the new samples until a termination criterion is satisfied [28]. The gradient-based approach transforms the discrete hyperparameter search space into a continuous one, employing gradient-based methods to find the optimal architecture. A notable example is DARTS [29] in which the output of each layer is considered as a convex combination of a set of operations. A bi-level optimisation method is then applied to find the optimal architecture. Another approach is Evolutionary Computation (EC), a generic population-based metaheuristic optimisation algorithm that takes inspiration from biological processes [28]. In this approach, each candidate DNN is encoded, and a number of biologically inspired operations, such as crossover and mutations, are applied at each generation to evolve the optimal DNN architecture. Among these approaches, EC has been shown to have several advantages, such as faster convergence, and diverse representational abilities, and its multi-objective extension has been shown to have been successfully applied to solve problems with multiple objective functions [7].

Evolutionary DL approaches typically encode the network hyperparameters before applying an EC method to find the optimal network. There are two types of encoding strategies: Fixed-length encoding and variable-length encoding [28]. In fixed-length encoding, the individuals have the same length during the evolutionary process while in variable-length encoding, the individuals can have different lengths during evolution. An example of fixed-length encoding as described by [30], involves proposing a fixed-length encoding for CNN where node connections are represented as a binary string. The authors utilised Genetic Algorithms to discover the optimal architecture. An example of variable-length encoding is CoDeepNEAT [31] which was based on NEAT [32], a popular evolutionary algorithm. NEAT begins with a minimal encoding and during evolution, new connections and nodes are added, and each gene in an encoding has a global innovation number for crossover. Within CoDeepNEAT, two populations evolve: blueprints, representing the overall structure of the deep neural network (DNN), and modules, representing each component. For every candidate, each blueprint node is substituted with a randomly selected chromosome module. The fitness of each blueprint and module is calculated as the average of the fitness scores of all DNN chromosomes associated with that blueprint or module. In [33], the authors proposed to evolve DNNs based on an Inception-based module called cells, with each being connected to the previous two cells. Each cell is either a normal cell, which preserves the image size, or reduction size, which has a pooling operator of stride 2. The cells are constructed by applying five different pairwise combinations of the hidden states, and afterward, the remaining hidden states are concatenated to give the final results.

It should be noted that evolving DNNs usually require a lot of computation time. For example, the Genetic CNN method, proposed in [30] required 17 GPU days to optimise a 3-stage LeNet network on the CIFAR-10 dataset. Another example is the evolved model in [33] which use 450 GPUs to train the model in 3150 GPU days [28]. In recent years, there has been many works in overcoming this limitation. In [34], the authors used a weight-sharing mechanism in which the weights in each component of a trained SuperNet will be used by the candidates in the next generations, thereby reducing computation time. Sun et al. [35] proposed E2EPP (End-to-End Performance Predictor), in which a number of candidate CNNs are trained, and then their discrete encodings are fed into a regressor. During evolution, given a new architecture encoding, the average prediction of a number of surrogate trees in the regressor is used as the fitness value instead of training the network, and the proposed method could save 2/3 of computation time while retaining the same accuracy. Domhan et al. [36] used parametric learning models to model the partially observed learning curve during the training process to extrapolate the performance and terminate the training of models that are expected to perform poorly. Dang et al. [37] proposed an ensemble framework of DNNs for medical image segmentation, in which the outputs of the deep segmentation models are combined using a weighted mechanism obtained by using a swarm-based EC method. Radial Basis Function (RBF) is used as the surrogate model to reduce the computational time by half.

## 3. Proposed method

### 3.1. General architecture

We denote $\mathscr{D}$ to be the training data with $N$ observations $\{(\mathbf{x}_n, \widehat{y}_n)\}$ $n = 1, \dots, N$, where $\mathbf{x}_n = (x_{n1}, x_{n2}, \dots, x_{nD})$ is the $D-$ feature vector of the $n^{th}$ training instance and $\widehat{y}_n$ is its true label. We also denote

$$\mathscr{X} = \begin{bmatrix} x_{11}, x_{12}, \dots, x_{1D} \\ x_{21}, x_{22}, \dots, x_{2D} \\ \dots \\ x_{N1}, x_{N2}, \dots, x_{ND} \end{bmatrix}, \quad \mathscr{Y} = \begin{bmatrix} \widehat{y}_1 \\ \widehat{y}_2 \\ \dots \\ \widehat{y}_N \end{bmatrix} \tag{1}$$

as the set of feature vectors and ground truths of all training instances. For supervised learning (i.e. classification), we learn a hypothesis $\boldsymbol{h}$ (i.e., classifier) to approximate the unknown relationship between the feature vector and its corresponding label. This hypothesis will be used to assign a label for each unlabelled instance. In single-layer ensemble learning, we learn $K$ hypotheses $\{\boldsymbol{h}_k\}$ by training $K$ learning algorithms $\{\mathscr{K}_k\}$ on the training data $\mathscr{D}$ and then use a combining algorithm $C$ on $\{\boldsymbol{h}_k\}$: $\widetilde{\boldsymbol{h}} = C\{\{\boldsymbol{h}_k\}, k = 1, \dots, K\}$ to combine $K$ hypothesis to reach the final decision. Recently, multi-layer ensemble learning has captured the attention of the machine learning community because this approach can further improve the performance of the conventional one-layer ensemble [2]. A multiple layer ensemble system consists of $S$ layers, each of which has $K$ classifiers $\left\{\boldsymbol{h}_k^{(i)},\ k = 1, \dots, K\right\}$ for $i = 1, \dots, S$. The classifiers in one layer are obtained by training $\{\mathscr{K}_k\}$ on training data generated by the subsequent layer. A combining algorithm $C$ works on the outputs of the classifiers in the last layer $\left\{\boldsymbol{h}_k^{(S)}\right\}$: $\widetilde{\boldsymbol{h}} = C\left\{\boldsymbol{h}_k^{(S)}, k = 1, \dots, K\right\}$ to generate a combined hypothesis.

We first describe the mechanism to populate the training data through layers of the ensemble. In the first layer, we obtain EoC $\left\{\boldsymbol{h}_k^{(1)}, k = 1, \dots, K\right\}$ by training $K$ learning algorithms on the original training data $\mathscr{D}$. The first layer also generates input data for the second layer by using the Stacking algorithm with the set of learning algorithms $\mathscr{K}$ [2,4]. Specifically, $\mathscr{D}$ is divided into $T_1$ disjoint parts in which the cardinality of each part is nearly similar. For each part, we train classifiers on its complementary and use these classifiers to predict for

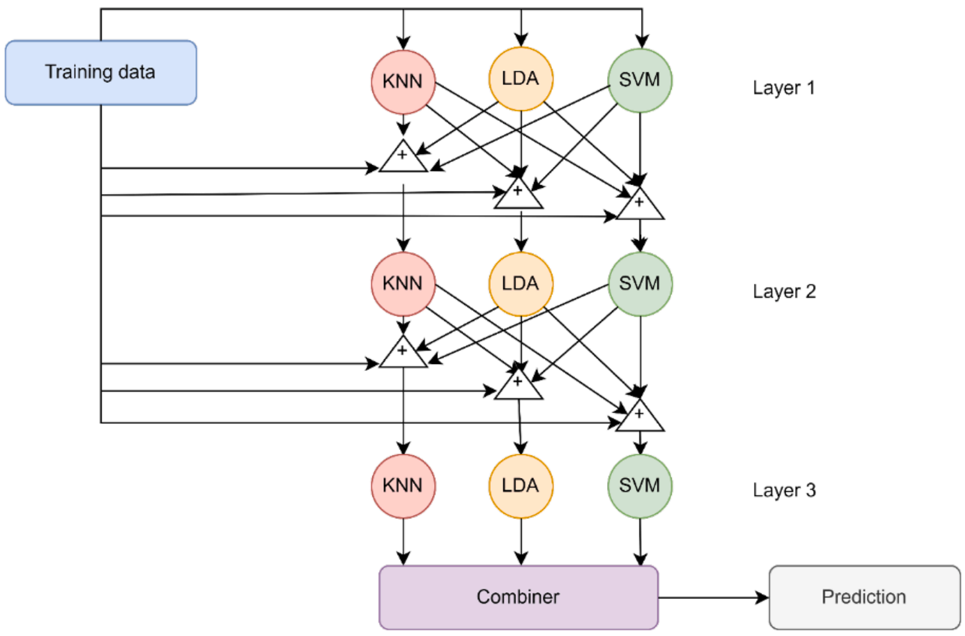

**Fig. 1.** An illustration of the original multi-layer ensemble system. $\triangle$ means the concatenation between the training data and predictions of classifiers.

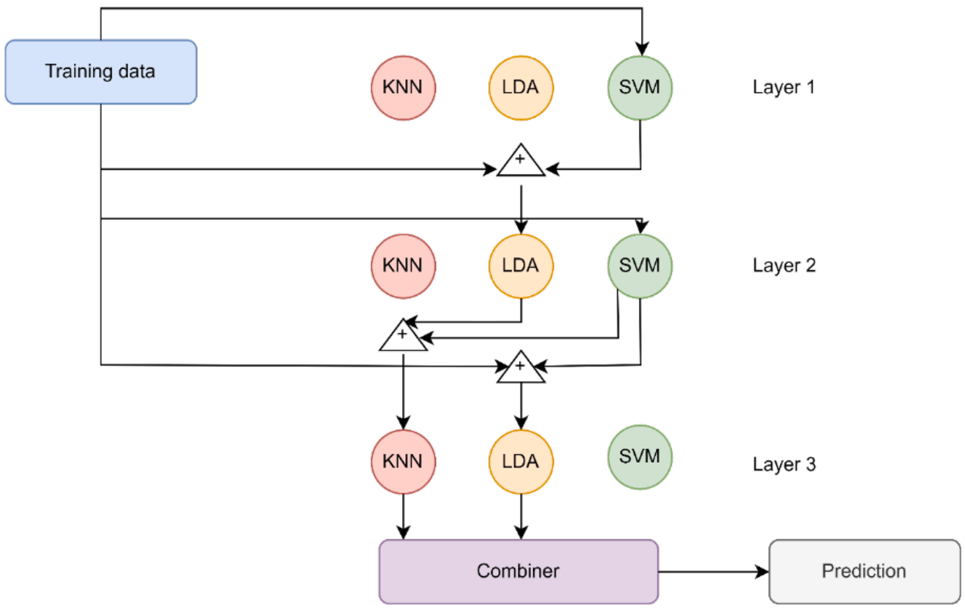

**Fig. 2.** An illustration of the proposed multiple layers ensemble system. $\triangle$ *means the concatenation between the training data and predictions of classifiers.*

observations of this part. If we solve a $M$-class classification problem, for observation $\mathbf{x}_n$, we obtain $p^{(1)}_{k,m}(\mathbf{x}_n)$ as the prediction of the $k^{th}$ classifier in the first layer that observation belongs to the class label $y_m$ $(m = 1\ldots M)$.

The predictions of the EoC in the first layer for $\mathbf{x}_n$ in terms of $M$ class labels are given in the form of $(M\times K)$ probability vector $P^{(1)}(\mathbf{x}_n) = \left[p^{(1)}_{1,1}(\mathbf{x}_n),\ p^{(1)}_{1,2}(\mathbf{x}_n),\ldots,p^{(1)}_{K,M}(\mathbf{x}_n)\right]$. The prediction vectors for all observations in $\mathscr{D}$ are given in the form of a $N\times(MK)$ matrix.

$$\mathscr{P}_1 = \left[\,P^{(1)}(\mathbf{x}_1) \quad P^{(1)}(\mathbf{x}_2) \quad \ldots \quad P^{(1)}(\mathbf{x}_N)\,\right]^T \tag{2}$$

The study in [4] claimed that by concatenating the original training data to the predictions, the discriminative characteristic of the input training data is likely to improve when growing to the next layer. A similar scheme will be conducted on the 2nd layer in terms of generating EoC and populating input training data for the 3rd layer.

We let $\mathscr{L}_1$ be the new data generated by the 1st layer which serves as the input for the 2nd layer. Normally, $\mathscr{L}_1$ is created by concatenating the original feature vectors of training instances and the predictions of EoC of the 1st layer as below:

$$\mathcal{L}_1 = (\mathcal{X} \triangle \mathcal{P}_1, \mathcal{Y}) \tag{3}$$

in which $\triangle$ denotes the concatenation operator between two matrices $\mathscr{X}$ of size $N\times D$ and $\mathscr{P}_1$ of size $N\times(MK)$. Thus $\mathscr{L}_1$ is obtained in the form of a $N\times(D+MK+1)$ matrix including $D$ features of original data, $M\times K$ prediction values, and ground truth of training instances. A similar process is conducted on the next layers until reaching the last layer in which at the $i^{th}$ layer, we train the EoC of $K$ classifiers $\left\{\boldsymbol{h}^{(i)}_k, k = 1,\ldots,K\right\}$ on the input data $\mathscr{L}_{i-1}$ generated by $(i-1)^{th}$ layer and generate input data $\mathscr{L}_i$ for the $(i+1)^{th}$ layer

$$\mathcal{L}_i = (\mathcal{X} \triangle \mathcal{P}_i, \mathcal{Y}) \tag{4}$$

The predictions of EoC of the last layer i.e. $S^{th}$ layer are combined for the collaborated decision. In this study, the Sum rule is used for the combination [2]. For an instance $\mathbf{x}$, the Sum rule summarises the predictions of EoC of the last layer concerning each class label. The label associated with the maximum value is assigned to this instance as follows:

$$\widetilde{\boldsymbol{h}} : \mathbf{x} \in y_t \text{ if } t = \operatorname{argmax}_{m=1,\ldots,M}\left\{\sum_{k=1}^{K} p^{(S)}_{k,m}(\mathbf{x})\right\} \tag{5}$$

in which $p^{(S)}_{k,m}(\mathbf{x})$ denotes the probability assigned by the $k^{th}$ classifier in layer $S$ of the ensemble for the $m^{th}$ class. In the classification process, each unseen instance is fed forward through the layers until reaching the last layer. The predictions of $K$ classifier at the last layer i.e. $P^{(S)}(.) = \left[p^{(S)}_{1,1}(.),\ p^{(S)}_{1,2}(.),\ldots,p^{(S)}_{K,M}(.)\right]$ are combined by the Sum Rule in (5) to obtain the predicted label.

An example of a multiple layer ensemble system with 3 layers is presented in Fig. 1. In each layer, we choose three machine learning algorithms namely SVM, KNN, and LDA to train the classifiers. In the first layer, the KNN, SVM, and LDA algorithms train on the original training data to obtain the EoC of the first layer. The $T$-fold cross-validation procedure is also applied to the original training data to split the training data into $T$ separated parts. SVM, LDA, and KNN train on the $(T-1)$ parts and predict for observations on the remained part. The outputs of the classifiers are concatenated with the original training data to generate the input training data for the second layer. A similar scheme is used for the second layer and the third layer. A combiner finally is applied to the output of the third layer to obtain the combining result of the ensemble.

### 3.2. Optimal connections

In multiple layer ensemble systems, the original training data is concatenated to the predictions of classifiers to generate the input for the next layer. Thus, unique input data will be used for all learning algorithms in a layer to train EoC. This approach has two issues: (1) there is no evidence to show that the concatenation of original training data and the predictions can improve the discriminative characteristic of data. Some research has even shown the effectiveness of using the predictions of only the new training data on some datasets [2]. (2) Learning algorithms use different approaches in approximating the relationship between the feature vectors of training instances and their class labels. In fact, different learning algorithms should use different sets of training data when training classifiers. Based on these observations, we propose an optimal connection approach to optimise the topology of multiple layer ensemble systems. We aim to choose suitable input data among the

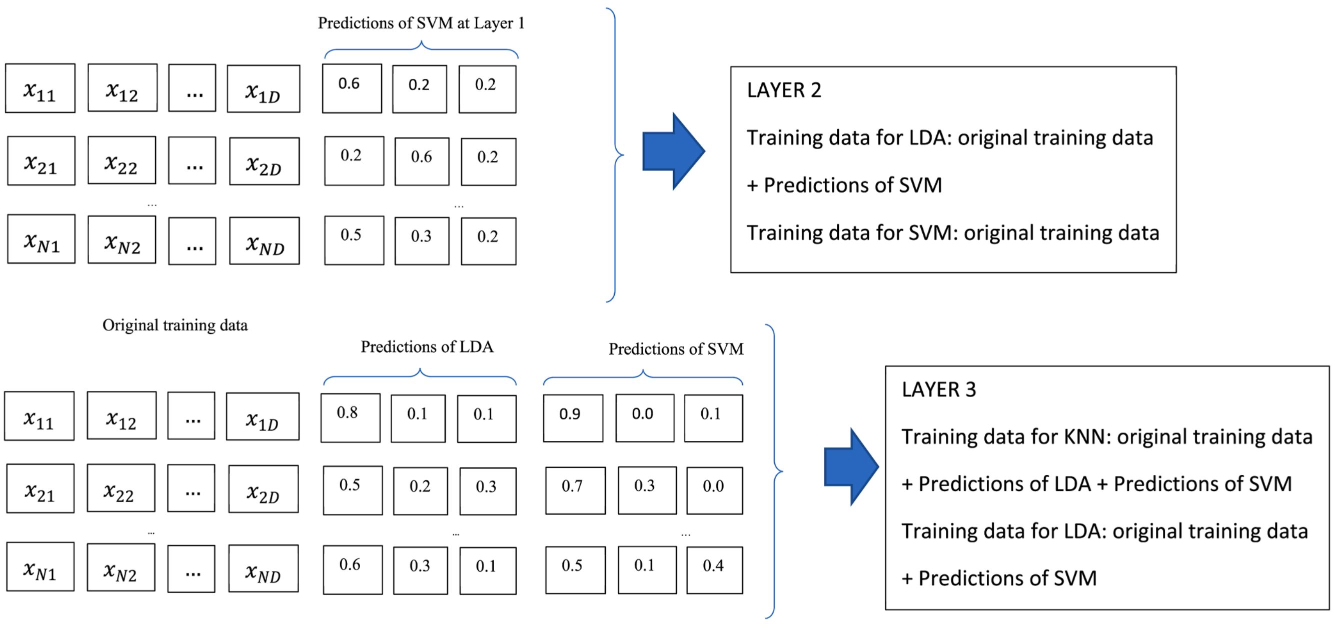
Predictions of SVM at Layer 1
$x_{11}$ $x_{12}$ ... $x_{1D}$ 0.6 0.2 0.2
$x_{21}$ $x_{22}$ ... $x_{2D}$ 0.2 0.6 0.2
$x_{N1}$ $x_{N2}$ ... $x_{ND}$ 0.5 0.3 0.2
LAYER 2
Training data for LDA: original training data
+ Predictions of SVM
Training data for SVM: original training data
Original training data
Predictions of LDA
Predictions of SVM
$x_{11}$ $x_{12}$ ... $x_{1D}$ 0.8 0.1 0.1 0.9 0.0 0.1
$x_{21}$ $x_{22}$ ... $x_{2D}$ 0.5 0.2 0.3 0.7 0.3 0.0
$x_{N1}$ $x_{N2}$ ... $x_{ND}$ 0.6 0.3 0.1 0.5 0.1 0.4
LAYER 3
Training data for KNN: original training data
+ Predictions of LDA + Predictions of SVM
Training data for LDA: original training data
+ Predictions of SVM

**Fig. 3.** An example of the training data flow in Fig. 2.
*Fig. 3 shows an example of the new input training data generated by the first and second layers in Fig. 2 for a three class-classification problem. In the first layer, only the connection between the training data and SVM is kept meaning that only SVM classifier is generated. The prediction of the SVM classifier is given in the form of 3-dimension vectors of probabilities showing an observation belongs to a class. In layer 2, since the connections between the original training data and LDA and SVM, and the connection between the SVM classifier in the 1st layer to LDA are kept, there are two classifiers generated in the 2nd layer: LDA classifier by training LDA on (D + 3) dimensional new training data, SVM classifier by training SVM on original training data. The LDA and SVM classifiers in the 2nd layer will output 3-dimension prediction vectors for each observation. At the 3rd layer, KNN classifier is generated by training KNN on the (D + 6) dimensional new training data (original training data + predictions of LDA and SVM classifier in the 2nd layer) while LDA classifier is generated by training LDA on the (D + 3) dimensional new training data (original training data + predictions of SVM classifier in the 2nd layer).*

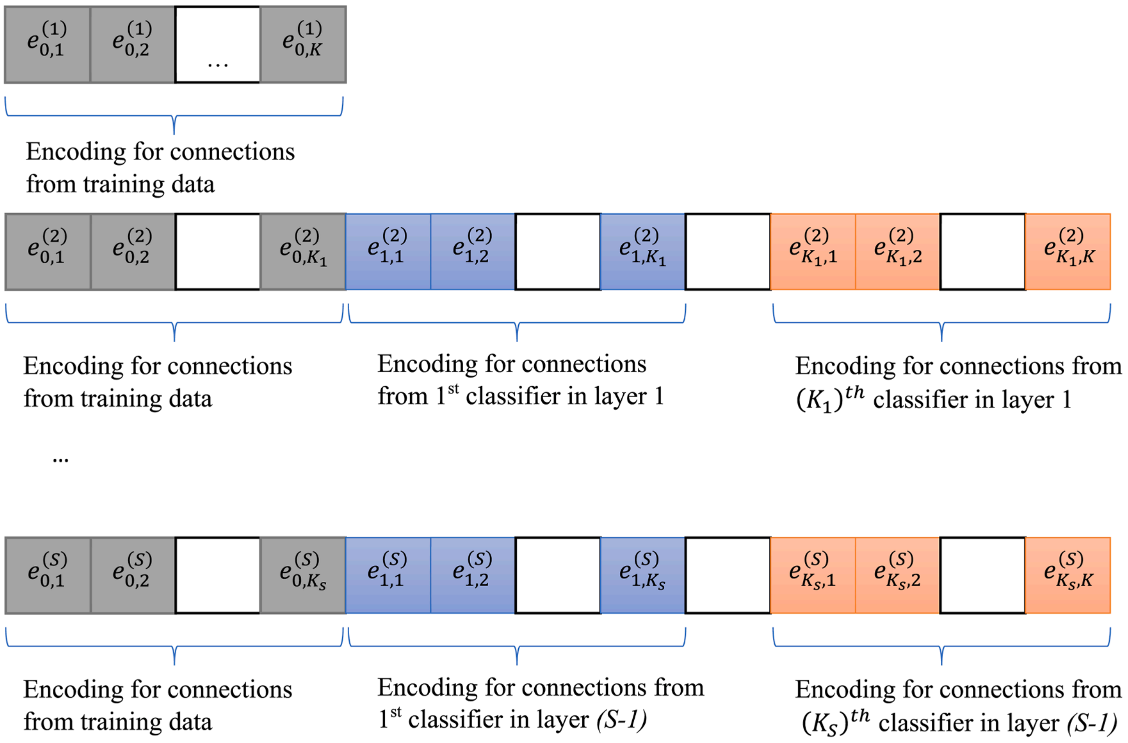
$e^{(1)}_{0,1}$ $e^{(1)}_{0,2}$ … $e^{(1)}_{0,K}$
Encoding for connections from training data
$e^{(2)}_{0,1}$ $e^{(2)}_{0,2}$ $e^{(2)}_{0,K_1}$ $e^{(2)}_{1,1}$ $e^{(2)}_{1,2}$ $e^{(2)}_{1,K_1}$ $e^{(2)}_{K_1,1}$ $e^{(2)}_{K_1,2}$ $e^{(2)}_{K_1,K}$
Encoding for connections from training data
Encoding for connections from 1st classifier in layer 1
Encoding for connections from $(K_1)^{th}$ classifier in layer 1
...
$e^{(S)}_{0,1}$ $e^{(S)}_{0,2}$ $e^{(S)}_{0,K_S}$ $e^{(S)}_{1,1}$ $e^{(S)}_{1,2}$ $e^{(S)}_{1,K_S}$ $e^{(S)}_{K_S,1}$ $e^{(S)}_{K_S,2}$ $e^{(S)}_{K_S,K}$
Encoding for connections from training data
Encoding for connections from 1st classifier in layer *(S-1)*
Encoding for connections from $(K_S)^{th}$ classifier in layer *(S-1)*

**Fig. 4.** Proposed encoding for selection of classifiers in the multi-layer ensemble model.

original training data and the predictions of classifiers in the preceding layer to a machine learning algorithm in the next layer to work on. First, we propose the following definition of a connection in a multiple layer ensemble system:

**Definition 1.** The $i^{th}$ classifier of the $s^{th}$ layer and the $j^{th}$ classifier of the $(s+1)^{th}$ layer are said to be connected if the output of the $i^{th}$ classifier of the $s^{th}$ layer is used in the training data for the $j^{th}$ classifier of the $(s+1)^{th}$ layer.

**Algorithm 1**
Stacking.

```
Input: $T$ training set folds $\{(\mathscr{X}_{train,i}, \mathscr{Y}_{train,i}), i = 1,\ldots,T\}$, validation set $\mathscr{X}_{val}$, learning algorithm $\mathscr{K}$
Output: The predicted values for all observations in the training set and validation set, and the trained classifier
  1. $\mathscr{P}^{train} = \phi, \mathscr{P}^{val} = \phi$
  2. For $t$ from 1 to $T$ do
  3.    Train $\mathscr{K}$ on the $\{(\mathscr{X}_{train,i}, \mathscr{Y}_{train,i}), i = 1,\ldots,T; i \neq t\}$ to obtain a classifier, predict on $\mathscr{X}_{train,t}$
  4.    Add the results to $\mathscr{P}^{train}$
  5. End for
  6. Train $\mathscr{K}$ on $(\mathscr{X}_{train}, \mathscr{Y}_{train})$ to obtain classifier $\mathscr{H}$
  7. $\mathscr{H}$ predicts on $\mathscr{X}_{val}$, add the results to $\mathscr{P}^{val}$
8. Return $\mathscr{P}^{train}$, $\mathscr{P}^{val}$, and $\mathscr{H}$
```

Fig. 2 shows an example of the optimal connection topology for the system in Fig. 1. In the 1st layer, since only the connection between SVM and the training data is formed, one classifier is generated by training SVM on the training data. In the 2nd layer, there are 2 connections from the training data to SVM and LDA and one connection from the SVM classifier in the 1st layer to LDA. Therefore, in the 2nd layer, there are two classifiers: one classifier is generated by training LDA on the concatenation of the training data and the prediction of the SVM classifier in the 1st layer, and the other classifier is generated by training SVM on the training data. Similarly, in the 3rd layer, there are two classifiers: one classifier is generated by training KNN on the concatenation of the predictions of SVM and LDA classifier in the 2nd layer while the other classifier is generated by training LDA on the concatenation of the training data and the prediction of SVM classifier in the 2nd layer. Fig. 3 shows which predictions and training data are used by each machine learning algorithm in each layer. The predictions of KNN and LDA are combined to obtain the combined result.

Next, we propose a binary encoding representation to encode the connections in the multiple layer ensemble systems. Fig. 4 shows the encoding representation for the connection from the original training data and the 1st, to the $S^{th}$ layer. Firstly, since there are $K$ learning algorithms to train classifiers for the 1st layer, there are $K$ genes encode for the connections from the original training data to these learning algorithms. The encoding is given in the binary format $E_1 = \left[e_{0,k}^{(1)}, k = 1\ldots K\right]$ showing which connection is absent or present. The presence of a connection means the associated learning algorithm will train on the original training data to generate a classifier for the 1st layer. Assume that $K_1$ connections are present in the final solution i.e. $K_1 \leq K$ classifiers are generated in the 1st layer. In the 2nd layer, there are $K$ genes encode for the connections from the original training data and $K$ genes encode for the connections from one classifier in the 1st layer to $K$ learning algorithms, therefore, there are $K + K \times K_1 = (K_1 + 1) \times K$ genes encode for the connections $E_2 = \left[e_{j,k}^{(2)}, k = 1\ldots K, j = 0\ldots K_1\right]$. It is recognised that the length of encoding of one layer depends on the number of remaining connections (i.e., the number of classifiers) in the preceding layer. Similar encoding representations will be used for the 2nd, 3rd,…, and $S^{th}$ layer with $(K_2 + 1) \times K$, $(K_3 + 1) \times K$, …,$(K_S + 1) \times K$ genes respectively.

Each element is defined as follows:

$$e_{j,k}^{(i)} = \begin{cases} 1, & \textit{the connection from } j^{th} \textit{ classifier at layer } (i-1) \textit{ to } k^{th} \textit{ classifier at layer } i \textit{ is made}* \\ 0, & \textit{otherwise} \end{cases} \quad (6)$$

*j = 0 shows the connections from the original training data.*

The search process is conducted in each layer to obtain the optimal set of connections. First, we search among $K$ connections from the training set to $K$ learning algorithms to obtain the optimal set. Based on the configuration of the optimal set, we train classifiers for the 1st layer. The search process continues with the 2nd layer in which the length of encoding representation depends on the number of classifiers in the 1st layer. For a $S$ layers-ensemble system, we need to conduct the search process $S$ times. In this study, the search for the optimal set of connections in each layer is done by using Binary Differential Evolution (Binary DE) [38,39], since it is generally recognised that DE is a reliable and versatile algorithm [39]. Here the objective function is the accuracy of the classification task on the validation set. In this way, we solve $S$ optimisation problems for the $S$ layers - ensemble systems:

$$\begin{aligned} &\max_{E_1}\left\{\frac{1}{|\mathscr{V}_1|}\sum_{n=1}^{|\mathscr{V}_1|}[\![C(\mathbf{x}_n) = \widehat{y}_n]\!]\right\} \\ &\max_{E_2}\left\{\frac{1}{|\mathscr{V}_2|}\sum_{n=1}^{|\mathscr{V}_2|}[\![C(\mathbf{x}_n) = \widehat{y}_n]\!]\right\} \\ &\ldots \\ &\max_{E_S}\left\{\frac{1}{|\mathscr{V}_S|}\sum_{n=1}^{|\mathscr{V}_S|}[\![C(\mathbf{x}_n) = \widehat{y}_n]\!]\right\} \end{aligned} \quad (7)$$

where $C(.)$ is the combiner working on the predictions of the classifiers at the $i^{th}$ layer, $\mathscr{V}_i$ is the validation set at the $i^{th}$ layer $(i = 1,\ldots,S)$, $|\cdot|$ denotes the cardinality of a set, and $[\![.]\!]$ is equal to 1 if the condition is true, otherwise equal to 0. In this study, the Sum Rule method was used to summarise the predictions of each instance concerning each class label and assign the instance to the class label associated with the maximum value. The combined prediction on an instance $\mathbf{x}$ at the $i^{th}$ layer is given by:

$$\mathbf{x} \in y_t \text{ if } t = \text{argmax}_{m=1,\ldots,M}\left\{\sum_{k=1}^{K} p_{k,m}^{(i)}(\mathbf{x})\right\} \quad (8)$$

## 3.3. Algorithms

We describe Differential Evolution (DE) algorithm which is used to solve the optimisation problem in (7). Let *popSize* be the number of candidates in the population, *nGen* be the number of generations, $f_n$ be the fitness value of the n-th candidate in the population. The DE algorithm [38,39] maintains a number of candidates $u_1,\ldots,u_{popSize} \in R^D$ where $D$ is the number of dimensions. Let $u_{i,d}$ denotes the $d^{th}$ dimension of the $i^{th}$ candidate (with $1 \leq i \leq popSize$, $1 \leq d \leq D$). At each

**Algorithm 2**
Sum_Rule.

```
Input: Predictions $\mathscr{P}$ of size $(N,\ MK_1)$, the number of instances $N$, the number of classes $M$, the number of classifiers used in the current layer $K_1$.
Output: The prediction by the ensemble
 1. $pred = \varnothing$
 2. For $n$ from 1 to $N$ do
 3.   $sum = \varnothing$
 4.   For $m$ from 1 to $M$ do
 5.     $sum_m = 0$
 6.     For $k$ from 1 to $K_1$ do
 7.       $sum_m = sum_m + p_{n,m+(k-1)*M}$
 8.     End for
 9.   End for
10.   $m_{pred} = argmax(sum)$
11.   Add $m_{pred}$ to $pred$
12. End for
13. Return $pred$
```

**Algorithm 3**
Create_Predictions_First_Layer.

```
Require: Training set $(\mathscr{X}_{\text{train}},\ \mathscr{Y}_{train})$, validation set $\mathscr{X}_{val}$, $K$ classifiers $\{\mathscr{K}_k\}_{k=1}^{K}$, number of folds $T$, encoding of the first layer $E_1 = \left\{e_{0,\ k}^{(1)}\right\}_{k=1}^{K}$.
Output: The predictions on the training and validation set.
 1. Divide $(\mathscr{X}_{\text{train}},\ \mathscr{Y}_{train})$ into $\{(\mathscr{X}_{train,i},\ \mathscr{Y}_{train,i}), i = 1,\dots,T\}$
 2. $\mathscr{P}_1^{train} = \varnothing,\ \mathscr{P}_1^{val} = \varnothing,\ \mathscr{H}_1 = \varnothing$
 3. For $k$ from 1 to $K$ do
 4.   If $e_{0,\ k}^{(1)} == 1$ then
 5.     $\left(\mathscr{P}_{1,k}^{train},\ \mathscr{P}_{1,k}^{val}, \mathscr{H}_{1,k}\right) = \text{Stacking}(\{(\mathscr{X}_{train,i},\ \mathscr{Y}_{train,i}), i = 1,\dots,T\}, \mathscr{K}_k)$
 6.     Add $\mathscr{P}_{1,k}^{train}$ and $\mathscr{P}_{1,k}^{val}$ to $\mathscr{P}_1^{train}$ and $\mathscr{P}_1^{val}$ respectively
 7.     Add $\mathscr{H}_{1,k}$ to $\mathscr{H}_1$
 8.   End if
 9. End for
10. Return $(\mathscr{P}_1^{train},\ \mathscr{P}_1^{val},\ \mathscr{H}_1)$
```

generation, and for each candidate $u_r(t)$, three random candidates $u_{r_1}(t)$, $u_{r_2}(t), u_{r_3}(t)$ are chosen, where $r_1, r_2, r_3 \in \{1, 2, \dots, popSize\}/\{r\}$, and a new offspring is created by the following mutation process:

$$u_{off}(t) = u_{r_1}(t) + F * \left(u_{r_2}(t) - u_{r_3}(t)\right) \tag{9}$$

where $F \in [0, 2]$ is a scaling factor. Afterwards, to increase the diversity, the following crossover is performed on the offspring for each dimension to create the trial vector:

$$u_{tr,d}(t) = \begin{cases} u_{r,d}(t)\ if\ rand() \le CR \\ u_{off,d}(t)\ if\ rand() > CR \end{cases} \tag{10}$$

**Algorithm 4**
Create_Predictions_and_Classifiers_Subsequent_Layers.

```
Input: Training set $(\mathscr{X}_{\text{train}},\ \mathscr{Y}_{train})$, validation set $\mathscr{X}_{val}$, $K$ learning algorithm $\{\mathscr{K}_i\}_{i=1}^{K}$, number of folds $T$, encoding of the current layer $E_i$, outputs from the previous layer $(\mathscr{P}_{i-1}^{train}, \mathscr{P}_{i-1}^{val})$
Output: The predictions of the current layer on the training and validation set and trained classifiers
 1. $K_{prev}$ = Number of classifiers used in $E_{i-1}$
 2. prevList = The list of classifiers which were used in $E_{i-1}$
 3. $\mathscr{P}_i^{train}=\varnothing, \mathscr{P}_i^{val}=\varnothing, \mathscr{H}_i=\varnothing$
 4. For $k$ from 1 to $K$ do
 5. $\mathscr{L}_k^{train} = \varnothing, \mathscr{L}_k^{val} = \varnothing$
 6.   //Add inputs based on connections from the previous layer
 7.   For $j$ from 0 to $K_{prev}$ do
 8.     If $q_{j,k}^{(i)} == 1$ then
 9.     If $j == 0$ then
10.         Add $\mathscr{X}_{\text{train}}$ and $\mathscr{X}_{val}$ to $\mathscr{L}_k^{train}$ and $\mathscr{L}_k^{val}$ respectively.
11.       Else
12.         $\mathscr{H}$ = prevList$_j$
13.         Add the predictions of $\mathscr{H}$ in $\mathscr{P}_{i-1}^{train}$ and $\mathscr{P}_{i-1}^{val}$ to $\mathscr{L}_k^{train}$ and $\mathscr{L}_k^{val}$ respectively.
14.       End if
15.       End if
16.     End for
17.     //Train on the current layer
18.     Divide $(\mathscr{L}_k^{train}, \mathscr{Y}_{train})$ into $\left\{\left(\mathscr{L}_{k,i}^{train},\ \mathscr{Y}_{train,i}\right), i = 1,\dots,T\right\}$
19.     $(\mathscr{P}_{i,k}^{train},\ \mathscr{P}_{i,k}^{val}, \mathscr{H}_{i,k}) = Stacking\left(\left\{\left(\mathscr{L}_{k,i}^{train},\ \mathscr{Y}_{train,i}\right), i = 1,\dots,T\right\}\right)$
20.     Add $\mathscr{P}_{i,k}^{train}$ and $\mathscr{P}_{i,k}^{val}$ to $\mathscr{P}_i^{train}$ and $\mathscr{P}_i^{val}$ respectively
21.     Add $\mathscr{H}_{i,k}$ to $\mathscr{H}_i$
22.   End for
23. Return $(\mathscr{P}_i^{train},\ \mathscr{P}_i^{val}, \mathscr{H}_i)$
```

**Algorithm 5**
Training multi-layer ensemble.

```
Input: Training set $(\mathscr{X}_{\text{train}}, \mathscr{Y}_{train})$, validation set $(\mathscr{X}_{val}, \mathscr{Y}_{val})$, $K$ learning algorithms $\{\mathscr{K}_i\}_{i=1}^{K}$, number of folds $T$, number of maximum layers $T_{stop}$, number of early stopping epochs $T_{earlyStopping}$. Binary DE parameters: number of generations maxGen, number of individuals popSize, the scaling factor $F$, the crossover rate $CR$.
Output: Optimal encoding of connections $E_1, E_2, \ldots, E_s$
1. $i = 1$
  2. While True do
  3.   If $i == 1$ then
  4.     $K_{prev} = 0$
  5.   Else
  6.     $K_{prev} =$ Number of classifiers used in the previous layer
  7.   End if
  8.   Randomly initialize popSize binary arrays of size $(K_{prev} + 1,\ K)$
  9.   For gen from 1 to maxGen do
  10.    For pop from 1 to popSize do
  11.      Let $q^{(i,pop)}$ be the pop -th candidate for the $i$-th layer
  12.      If $i == 1$ then
  13.      $(\mathscr{P}_i^{train}, \mathscr{P}_i^{val}, \mathscr{K}_i) =$ Create_Predictions_First_Layer($\mathscr{X}_{\text{train}}, \mathscr{Y}_{train}, \mathscr{X}_{val}, \{\mathscr{K}_k\}_{k=1}^{K}, T, q^{(i,pop)}$)
  14.    Else
  15.      $(\mathscr{P}_i^{train}, \mathscr{P}_i^{val}, \mathscr{K}_i) =$ Create_Predictions_Subsequent_Layer($\mathscr{X}_{train}, \mathscr{Y}_{train}, \mathscr{X}_{val}, \{\mathscr{K}_k\}_{k=1}^{K}, T, q^{(i,pop)}, (\mathscr{P}_{i-1}^{train}, \mathscr{P}_{i-1}^{val})$)
  16.    End if
  17.      $pred_{pop} = Sum\_Rule(\mathscr{P}_i^{val})$
  18.      $fitness_{pop} = Accuracy(pred_{pop}, \mathscr{Y}_{val})$
  19.    End for
  20.    Perform crossover and mutation procedures according to Eq. (12), (13) and (14)
  21.  End for
  22.  Let $E_i$ be the candidate with the highest fitness value
  23.  If $i == T_{stop}$ or the result has not improved after $T_{earlyStopping}$ then
  24.    $S = i$
  25.    Break.
  26.  End if
  27.  $i = i + 1$
  28. End while
29. Return $(E_1, E_2, \ldots, E_s)$
```

Where $CR$ is the crossover rate and $rand()$ denotes the random function (within the $[0,\ 1]$ range). This is known as the DE/rand/1/bin strategy, which is one of the most popular strategies in DE. The objective value of $u_{tr}(t)$ is then compared with that of $u_r(t)$, and the one with the better objective value is chosen for the next generation.

$$u_r(t+1) = \begin{cases} u_r(t) \text{if } obj(u_r(t)) \geq obj(u_{tr}(t)) \\ u_{tr}(t) \text{ if } obj(u_r(t)) < obj(u_{tr}(t)) \end{cases} \quad (11)$$

Since the proposed encoding is of binary form, during the optimisation process, the candidates are constrained to be within the $[0,\ 1]$ range, and each value is rounded off to either 0 or 1 based on its value.

The pseudo-code of the proposed method is presented in Algorithms 1-5. Algorithm 1 describes the Stacking procedure. Given $T$ training set folds $\{(\mathscr{X}_{train,i}, \mathscr{Y}_{train,i}),\ i = 1, \ldots, T\}$, validation set $\mathscr{X}_{val}$, number of cross-validation folds $T$, learning algorithm $\mathscr{K}$, the algorithm first sets the predictions of the training and validation sets to the empty set (line 1). Then, for each fold, the algorithm trains a classifier by using $\mathscr{K}$ on the remainder of that fold and then uses this classifier to predict observations on that fold. The predictions are added to $\mathscr{P}^{train}$ (lines 2–5). In lines 6–7, a classifier $\mathscr{K}$ is trained on the entire training set and predict on the validation set, then the result is added to $\mathscr{P}^{val}$. Finally, the predictions for the training and validation set and the classifier $\mathscr{K}$ are returned in line 8.

Algorithm 2 describes the sum rule procedure to evaluate the predictions at each layer. The algorithm receives the predictions $\mathscr{P}$ of size $(N,\ MK_1)$ where $N$ is the number of observations, $M$ is the number of classes, and $K_1$ is the number of classifiers used in the current layer. Initially, the output predictions *pred* is set to the empty set (line 1). For each observation, the probabilities of each classifier for each class are summed (line 7) where $p_{n,m+(k-1)*M}$ is the prediction of $k^{th}$ classifier for the $m^{th}$ class on the $n^{th}$ observation. The output is the class with the highest sum value (line 10). Finally, in line 13, the final prediction is returned.

Algorithm 3 describes the procedure to create the predictions for the first layer given an encoding. The algorithm receives as inputs the training set $(\mathscr{X}_{train}, \mathscr{Y}_{train})$, validation set $\mathscr{X}_{val}$, $K$ learning algorithms $\{\mathscr{K}_k\}_{k=1}^{K}$, number of cross-validation folds $T$, and encoding of the first layer $E_1 = \left\{e_{0,\ k}^{(1)}\right\}_{k=1}^{K}$. In lines 1–2, the training set is first divided into $T$ disjoint folds, and the predictions of the training and validation sets are set to the empty set. Then, in lines 3–8, for each learning algorithm, if the corresponding value in the encoding is 1, Algorithm 1 is called. The outputs of Algorithm 1 are added to the predictions of the training and validation sets as well as the ensemble of classifiers for layer 1.

Algorithm 4 describes the procedure to create the predictions for an $i^{th}$ layer with an encoding. The inputs of the algorithm are the training set $(\mathscr{X}_{train}, \mathscr{Y}_{train})$, validation set $\mathscr{X}_{val}$, $K$ learning algorithm $\{\mathscr{K}_i\}_{i=1}^{K}$, number of cross-validation folds $T$, encoding of the current layer $E_i$ and outputs from the previous layer $(\mathscr{P}_{i-1}^{train}, \mathscr{P}_{i-1}^{val})$. In lines 1–3, the number and list of classifiers used in the previous layers are retrieved while the predictions of the training and validation set by the current layer are set to the empty set. In lines 4–16, the inputs to each classifier $\mathscr{L}_k^{train}$ and $\mathscr{L}_k^{val}$ are added based on connections from the previous layer, as well as connections to the original data (lines 8–15). Afterwards, in lines 18–20, $\mathscr{L}_k^{train}$ is divided into $T$ folds, then the Stacking procedure (Algorithm 1) is performed, and the results are added to $\mathscr{P}_i^{train}$ and $\mathscr{P}_i^{val}$ as well as the ensemble of classifiers of $i^{th}$ layer $\mathscr{K}_i$. Finally, in line 22, $\mathscr{P}_i^{train}$, $\mathscr{P}_i^{val}$, and $\mathscr{K}_i$ are returned.

Algorithm 5 describes the training procedure. The algorithm receives as inputs the training set $(\mathscr{X}_{\text{train}}, \mathscr{Y}_{train})$, validation set $(\mathscr{X}_{val}, \mathscr{Y}_{val})$, $K$ learning algorithm $\{\mathscr{K}_i\}_{i=1}^{K}$, number of cross-validation folds $T$, number of maximum layers $T_{stop}$, number of early stopping epochs $T_{earlyStopping}$, and the binary DE parameters including the number of generations *maxGen*, number of individuals *popSize*, the scaling factor $F$, and the crossover rate $CR$. In lines 1–8, $K_{prev}$ is set to 0 for the first layer, or to the number of classifiers used in the previous layer, and *popSize* binary arrays of size $(K_{prev} + 1,\ K)$ are randomly initialized. From lines 9–15, for

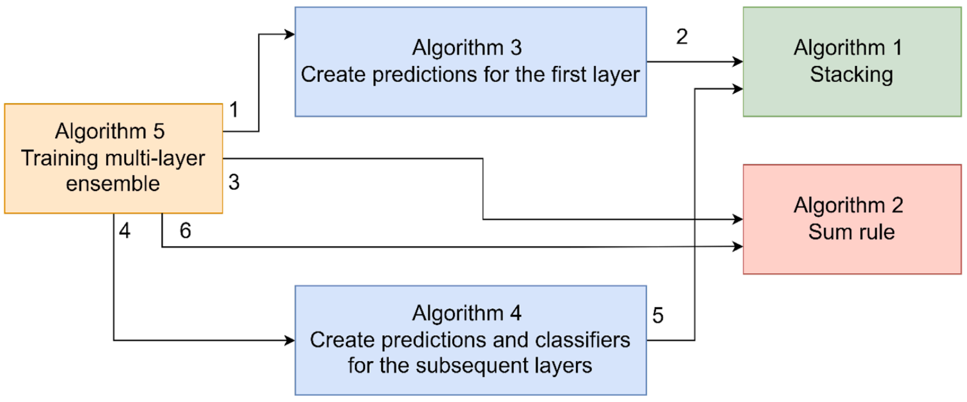
Algorithm 5
Training multi-layer ensemble
Algorithm 3
Create predictions for the first layer
Algorithm 1
Stacking
Algorithm 2
Sum rule
Algorithm 4
Create predictions and classifiers for the subsequent layers
1
2
3
4
5
6

**Fig. 5.** The flows between algorithms of the proposed multi-layer ensemble model.

**Table 1**
Information of experimental datasets.

| Dataset | Number of instances | Number of dimensions | Number of classes |
|---|---|---|---|
| Balance | 625 | 4 | 3 |
| Banana | 5300 | 2 | 2 |
| Breast-Tissue | 106 | 9 | 6 |
| Cleveland | 297 | 13 | 5 |
| Colon | 62 | 2000 | 2 |
| Conn-Bench-Vowel | 528 | 10 | 11 |
| Contraceptive | 1473 | 9 | 3 |
| Electricity-Normalised | 45,312 | 8 | 2 |
| Embryonal | 60 | 7129 | 2 |
| Fertility | 100 | 9 | 2 |
| GM4 | 1000 | 1000 | 3 |
| Heart | 270 | 13 | 2 |
| Isolet | 7797 | 617 | 26 |
| Leukemia | 72 | 7129 | 2 |
| Madelon | 2000 | 500 | 2 |
| Mammographic | 830 | 5 | 2 |
| Multiple-Features | 2000 | 649 | 10 |
| Musk1 | 476 | 166 | 2 |
| Musk2 | 6598 | 166 | 2 |
| Newthyroid | 215 | 5 | 3 |
| Penbased | 10,992 | 16 | 10 |
| Phoneme | 5404 | 5 | 2 |
| Plant-Margin | 1600 | 64 | 100 |
| Ringnorm | 7400 | 20 | 2 |
| Satimage | 6435 | 36 | 6 |
| Sonar | 208 | 60 | 2 |
| Tic-Tac-Toe | 958 | 9 | 2 |
| Titanic | 2201 | 3 | 2 |
| Vertebral-3C | 310 | 6 | 3 |
| Wine-Red | 1599 | 11 | 6 |

**Table 2**
Classification algorithms and their parameters in the experiments.

| Classification algorithms | Parameters | Number of classifiers |
|---|---|---|
| Decision Tree | Split criterion: “gini”, “entropy”. Split strategy: “best”, “random”. | 4 |
| KNN | Number of neighbours: 1, 3, 5, 7, 9. | 5 |
| Random Forest | Number of estimators: 200. | 1 |
| Bagging | Number of estimators: 200. | 1 |
| XgBoost | Number of estimators: 200. | 1 |
| MLP | Size of hidden layer: 20, 40, 60, 80, 100. Learning rate: 0.3, 0.6. | 10 |
| Gaussian Naive Bayes | Default. | 1 |
| AdaBoost | Number of estimators: 200. | 1 |
| Logistic regression | Inverse of regularisation strength: 0.001, 0.01, 0.1, 1, 10, 100. | 6 |

each candidate encoding, Algorithm 3 is called for the first layer, or Algorithm 4 is called for the subsequent layers, and the output of the current layer is created. Then, in lines 17–18, Algorithm 2 is called to output the final predictions so as to calculate the fitness of the candidate encoding.

In line 20, crossover and mutation are performed using the following equations:

$$e_{j,k}^{off} = e_{j,k}^{i,r_1} + F * \left(e_{j,k}^{i,r_2} - e_{j,k}^{i,r_3}\right) \tag{12}$$

$$e_{j,k}^{tr} = \begin{cases} e_{j,k}^{i,r} \text{ if } rand() \le CR \\ e_{j,k}^{off} \text{ if } rand() > CR \end{cases} \tag{13}$$

$$e_{j,k}^{i,r} = \begin{cases} e_{j,k}^{i,r} \text{ if } obj\left(e_{j,k}^{i,r}\right) \ge obj\left(e_{j,k}^{tr}\right) \\ e_{j,k}^{tr} \text{ if } obj\left(e_{j,k}^{i,r}\right) < obj\left(e_{j,k}^{tr}\right) \end{cases} \tag{14}$$

The algorithm retrieves the candidate $E_i$ with the highest fitness value. In lines 23–28, if the maximum number of layers has been achieved or the result has not improved after $T_{earlyStopping}$ layers, and then the algorithm terminates. Finally, in line 29, the optimal encodings for each layer are returned.

Each test sample will pass through layers in the ensemble and be predicted by the ensemble of classifiers in each layer. The test data is populated based on the connections between two layers until the last layer. Afterward, the Sum rule is applied to the predictions of the ensemble of classifiers in the last layer to get the final prediction.

Fig. 5 shows the block diagram of the algorithms. The numbers denote the calling sequence during the training and testing process. Firstly, Algorithm 5 is called to begin the training process to create the multi-layer ensemble. Then, Algorithm 3 is called to create the predictions for the first layer (step 1), and within this algorithm, for each candidate, Algorithm 1 is also called as well (step 2). Then, Algorithm 2 is called to return the fitness result (step 3). Afterwards, the predictions for the first layer are returned, and Algorithm 4 is called to create the predictions and classifiers for the subsequent layers (step 4). For each candidate, Algorithm 1 is called (step 5), and afterwards, Algorithm 2 is called to return the fitness result (step 6).

## 4. Experimental studies

### 4.1. Configurations

We chose 30 datasets from the UCI Machine Learning Repository and

**Table 3**
The comparison of the accuracy of COME with GA, PSO, and DE.

| Dataset | COME-GA | COME-PSO | COME-DE |
|---|---|---|---|
| Balance | 0.9202 | 0.9202 | **0.9681** |
| Banana | 0.8981 | 0.8994 | **0.9013** |
| Breast-Tissue | 0.7188 | 0.6875 | **0.7813** |
| Cleveland | **0.6222** | 0.6000 | **0.6222** |
| Colon | 0.6842 | 0.6842 | **0.8421** |
| Conn-Bench-Vowel | 0.9811 | **0.9874** | 0.9811 |
| Contraceptive | 0.5362 | **0.5837** | 0.5814 |
| Electricity-Normalised | 0.9288 | **0.9320** | 0.9311 |
| Embryonal | 0.5000 | 0.6111 | **0.6667** |
| Fertility | **0.9333** | **0.9333** | **0.9333** |
| GM4 | 0.9967 | 0.9900 | **1.0000** |
| Heart | 0.8519 | 0.8272 | **0.8765** |
| Isolet | **0.9564** | 0.9543 | 0.9534 |
| Leukemia | **0.9545** | 0.9091 | **0.9545** |
| Madelon | 0.7650 | 0.7517 | **0.7983** |
| Mammographic | 0.8554 | 0.8434 | **0.8594** |
| Multiple-Features | 0.9767 | 0.9833 | **0.9850** |
| Musk1 | 0.8811 | **0.8881** | 0.8671 |
| Musk2 | 0.9909 | 0.9924 | **0.9934** |
| Newthyroid | **0.9692** | **0.9692** | **0.9692** |
| Penbased | 0.9955 | **0.9958** | 0.9951 |
| Phoneme | 0.902 | **0.9069** | 0.9051 |
| Plant-Margin | **0.85** | 0.8375 | 0.8271 |
| Ring | 0.9802 | 0.9793 | **0.9811** |
| Satimage | 0.9342 | 0.9332 | **0.9358** |
| Sonar | **0.9365** | 0.9048 | **0.9365** |
| Tic-Tac-Toe | 0.9931 | **1.0000** | **1.0000** |
| Titanic | **0.7595** | **0.7595** | **0.7595** |
| Vertebral-3C | 0.8387 | **0.8495** | **0.8495** |
| Wine-Red | 0.6688 | 0.6771 | **0.6854** |

**Table 4**
The comparisons of the F1 score of COME with GA, PSO, and DE.

| Dataset | COME-GA | COME-PSO | COME-DE |
|---|---|---|---|
| Balance | 0.8681 | 0.8783 | **0.9426** |
| Banana | 0.8968 | 0.8981 | **0.9003** |
| Breast-Tissue | 0.6912 | 0.6629 | **0.7650** |
| Cleveland | 0.2891 | 0.2838 | **0.3013** |
| Colon | 0.5250 | 0.5250 | **0.8081** |
| Conn-Bench-Vowel | 0.9833 | **0.9891** | 0.9814 |
| Contraceptive | 0.5183 | 0.5450 | **0.5514** |
| Electricity-Normalised | 0.9270 | **0.9303** | 0.9292 |
| Embryonal | 0.4582 | 0.5786 | **0.6250** |
| Fertility | **0.7321** | **0.7321** | **0.7321** |
| GM4 | 0.9969 | 0.9907 | **1.0000** |
| Heart | 0.8363 | 0.7975 | **0.8635** |
| Isolet | **0.9567** | 0.9544 | 0.9538 |
| Leukemia | **0.9454** | 0.8854 | **0.9454** |
| Madelon | 0.7650 | 0.7513 | **0.7983** |
| Mammographic | 0.8554 | 0.8433 | **0.8594** |
| Multiple-Features | 0.9761 | 0.9827 | **0.9843** |
| Musk1 | 0.8796 | **0.8874** | 0.8650 |
| Musk2 | 0.9823 | 0.9854 | **0.9874** |
| Newthyroid | 0.9453 | 0.9453 | **0.9548** |
| Penbased | 0.9955 | **0.9958** | 0.9952 |
| Phoneme | 0.8825 | **0.8867** | 0.8851 |
| Plant-Margin | **0.8475** | 0.8307 | 0.8177 |
| Ringnorm | 0.9802 | 0.9793 | **0.9811** |
| Satimage | 0.9191 | 0.9176 | **0.9214** |
| Sonar | 0.9357 | 0.9028 | **0.9361** |
| Tic-Tac-Toe | 0.9921 | **1.0000** | **1.0000** |
| Titanic | **0.7059** | **0.7059** | **0.7059** |
| Vertebral-3C | 0.7573 | **0.7870** | 0.7772 |
| Wine-Red | 0.3245 | **0.3270** | 0.3252 |

OpenML for the experiment. The information of these datasets including the number of observations, the number of classes, and the number of dimensions is given in Table 1. To generate the set of classifiers, we trained 30 machine learning algorithms on the layer's training data. These classifiers are based on the following algorithms: Decision Tree, kNN, Random Forest, Bagging, XgBoost, MLP, Gaussian Naïve Bayes, AdaBoost, and Logistic regression. The classifiers were created using different sets of hyperparameters for these algorithms. The detailed information of these hyperparameters is presented in Table 2. The training data for one layer was generated by using the 5-fold cross-validation on the output of the subsequent layer. The validation set was formed by choosing 20% training observations.

We first compared COME using three different optimisation algorithms namely Genetic Algorithm, Particle Swarm Optimisation, and Differential Evolution, which are denoted COME-GA, COME-PSO, and COME-DE respectively. For all three methods, we set the maximum number of generations as 50, and the population size as 100. For COME-GA, the crossover and mutation probability were set as 0.8 and 0.2, respectively. For COME-PSO, the cognitive and social parameters $c_1$ and $c_2$ were set to 0.5 and the inertia parameter $w$ was set to 0.9. For COME-DE, the scaling factor $F$ was set to 2.0, and the crossover rate $CR$ was set to 0.7 [40,41]. The proposed method was also compared with some benchmark algorithms in terms of classification accuracy and F1 score. We chose 2 well-known ensemble methods namely Random Forest and XgBoost in which each method included 200 classifiers. These algorithms were shown to be top-performing methods based on the experiments in [2]. We also chose two multiple layer ensemble models as the benchmark algorithms: gcForest [4] and MULES [2]. For gcForest, 4 forests with 200 trees in each forest like in the original papers were implemented while the parameters of MULES were set similarly to the original papers. We also choose Multiple Linear Regression (MLR) [42], a popular weighted combining ensemble learning method, as a benchmark algorithm.

We used the Friedman test to test the null hypothesis "the performance of the benchmark algorithms and proposed methods is equal". In this study, the significant threshold was set to 0.05. If the P-value of this test is smaller than a significant threshold, we reject the null hypothesis, which means there is a difference in the performance results. In this case, we compared each pair among 6 methods by using the Nemenyi post-hoc test [43].

## 4.2. Comparisons between different optimisation algorithms

Table 3 shows the accuracy of COME-GA, COME-PSO, and COME-DE. It can be seen that COME-DE achieves the best results on 22 out of 30 datasets while the others also have high performance on several datasets. On the Balance, Breast-Tissue, Colon, Embryonal, Heart, and Madelon datasets, the differences between COME-DE and the other two methods are significant. For example, on Breast-Tissue, COME-DE obtains an accuracy of 0.7813, which is higher than that of the second-best (COME-GA) by 6.25%. Similarly, on the Colon dataset, the accuracy of COME-DE is 0.8421 while both COME-GA and COME-PSO only achieve a score of 0.6842. On many other datasets, COME-DE also obtains the best score, however, the difference is not as large as those datasets. For example, on the Wine-Red dataset, COME-GA obtains an accuracy of 0.6854, which is higher than the second best (COME-PSO) at just around 0.83%. On 8 datasets (Cleveland, Fertility, Leukemia, Newthyroid, Sonar, Tic-Tac-Toe, Titanic, and Vertebral-3C), COME-DE shares the best position with one or two other methods. For example, on the Titanic dataset, all three methods have a score of 0.7595, while on the Vertebral-3C dataset, both COME-DE and COME-PSO have an accuracy of 0.8495, while COME-GA only obtains a score of 0.8387. On several datasets, the performance of COME-DE is lower than that of COME-GA and COME-PSO. For example, the accuracy of COME-GA on the Plant-Margin dataset is 0.85, which is higher than that of COME-DE by around 0.023. However, in most instances, the performance differences between the methods are not substantial. An example is Electricity-Normalised, in which the best-performing method is COME-PSO at 0.932, while COME-DE has a score of 0.9311, which is only slightly worse compared to COME-PSO.

The F1 scores of COME-GA, COME-PSO, and COME-DE are shown in Table 4. Similar to the results concerning accuracy score, COME-DE

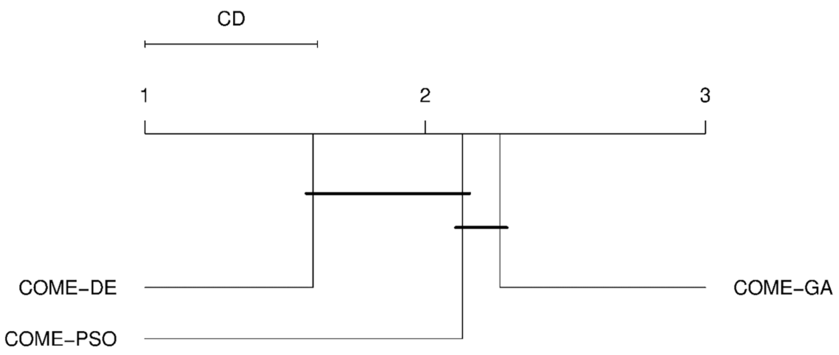

**Fig. 6.** Nemenyi test for the accuracy of COME with GA, PSO, and DE
*COME-DE* > *COME-GA*.

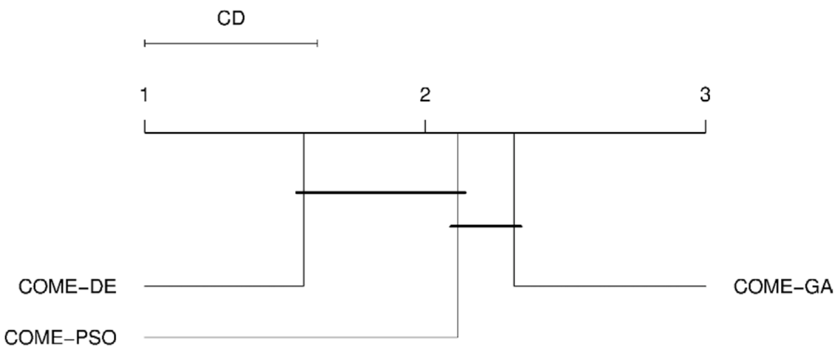

**Fig. 7.** Nemenyi test for the F1 score of COME with GA, PSO, and DE
*COME-DE* > *COME-GA*.

achieves the highest scores on a significant number of datasets. The highest difference in performance between COME-DE and the other two methods is in the Colon dataset, in which COME-DE obtains a score of 0.8081, which is higher than both COME-GA and COME-PSO by 0.2831. On five other datasets, namely Balance, Breast-Tissue, Embryonal, Heart, and Madelon, COME-DE also perform significantly better than the others, with an average difference of 4.934%. Unlike the results related to accuracy score, in this case, only four datasets (Fertility, Leukemia, Tic-Tac-Toe, and Titanic) in which COME-DE shares the first position with other methods. There are five datasets in which COME-GA obtains the best results (Fertility, Isolet, Leukemia, Plant-Margin, and Titanic), however, the average difference between COME-GA and the second best on these datasets is only 0.382%. Similarly, COME-PSO has the highest F1 score on 10 datasets, however, its performance is the same as COME-DE on three datasets (Fertility, Tic-Tac-Toe, and Titanic). For the remaining datasets, COME-PSO only scored higher than COME-DE by around 0.0064 on average.

Figs. 6 and 7 show the Nemenyi test for the accuracy and F1 score of COME-GA, COME-PSO, and COME-DE. In both cases, it can be seen that COME-DE is better than COME-GA and that COME-DE ranks first, followed by COME-PSO and COME-GA, however, there is no statistical difference between COME-DE and COME-PSO.

Fig. 8 shows the number of optimal layers found by COME-GA, COME-PSO, and COME-DE. On average, all three methods found the same number of optimal layers, however on some datasets, some methods found more layers compared to the other methods. There are 9 datasets where all three methods found just one optimal layer and 4 datasets where 2 layers are considered optimal by all three methods. In contrast, on the Colon dataset, COME-DE found 7 layers to be optimal while both COME-GA and COME-DE found only 1 layer. In this case, according to Table 3 and 4, COME-DE obtains a much better score compared to COME-GA and COME-PSO. Another example is Balance, in which COME-DE found 3 layers and obtained an accuracy of 0.9681, as opposed to COME-GA and COME-PSO which only found 2 layers while achieving an accuracy of just slightly over 0.92. However, on the Pen-based dataset, even though COME-DE found more layers compared to COME-GA and COME-PSO, the accuracy is slightly lower than that of these two methods. COME-GA and COME-PSO also found many more layers compared to the other methods on several datasets. For example, on the Contraceptive dataset, the optimised ensemble found by COME-PSO has 8 layers, compared to just 4 layers by COME-GA and 2 layers by COME-DE.

Table 5 shows the optimisation time (in hours) of COME-GA, COME-PSO, and COME-DE. COME-DE requires the most time for optimisation on most datasets, however, the differences in time on many datasets are negligible. On the Isolet dataset, COME-DE took 100 h compared to just around 63 and 36 h for COME-GA and COME-PSO, while on the Plant-margin dataset, COME-DE took 119 h, which is significantly higher as that of COME-PSO. However, for other datasets, the difference in optimisation time between COME-DE and the other two methods are not very significant. For example, on the Balance dataset, COME-DE took 1.5 h, which is higher than COME-GA and COME-PSO by just 0.5 and 1.0 h respectively. There are five datasets in which COME-GA requires more time than COME-DE (Breast-Tissue, Cleveland, Mammographic, Musk2, Wine-Red) and two datasets for COME-PSO (Contraceptive and Vertebral). It should be noted that although COME-DE might require more time compared to the other two methods, it provides better results on many datasets.

### 4.3. *Comparisons with the benchmark algorithms*

Next, we compared COME-DE with the following benchmark algorithms: gcForest, Random Forest, XgBoost, MULES, and MLR. For the following experiments, COME-DE is now denoted as COME.

Figs. 9 and 10 show the Nemenyi test for the accuracy and F1 score of COME and the benchmark algorithms. In both cases, COME is better

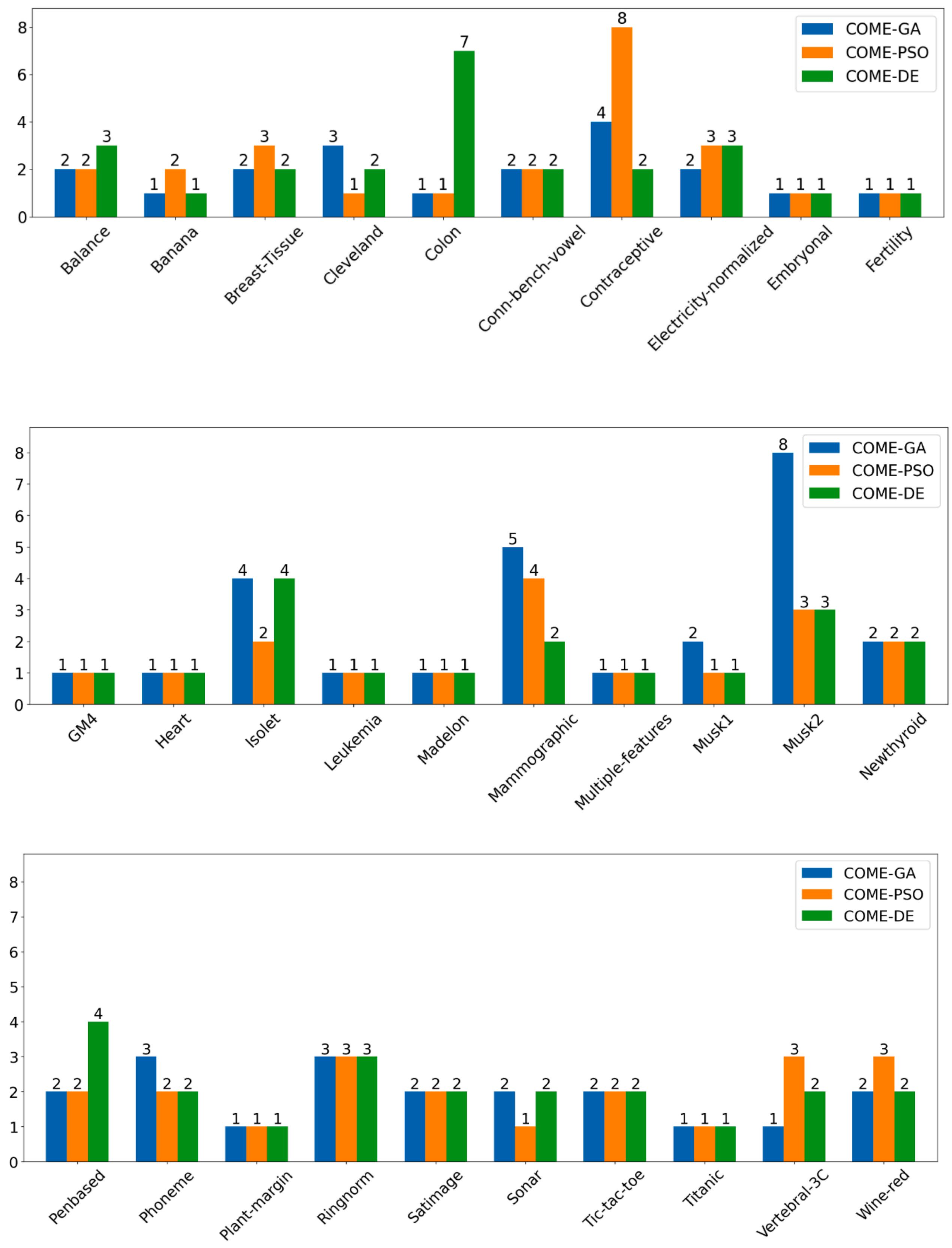

**Fig. 8.** Number of optimal layers found by COME-GA, COME-PSO, and COME-DE on experimental datasets.

**Table 5**
Optimisation time of COME with GA, PSO, and DE (in hours).

| Dataset | COME-GA | COME-PSO | COME-DE |
|---|---|---|---|
| Balance | 1.0975 | 0.6010 | 1.5023 |
| Banana | 3.0058 | 3.1760 | 4.8259 |
| Breast-Tissue | 0.7645 | 0.6840 | 0.6196 |
| Cleveland | 0.8904 | 0.5556 | 0.8308 |
| Colon | 0.3836 | 0.6191 | 2.0441 |
| Conn-Bench-Vowel | 1.4069 | 1.1531 | 1.4452 |
| Contraceptive | 2.5370 | 3.7200 | 1.8606 |
| Electricity-Normalised | 24.556 | 23.1680 | 35.5188 |
| Embryonal | 0.9977 | 1.7581 | 1.8373 |
| Fertility | 0.1703 | 0.1812 | 0.2069 |
| GM4 | 2.4926 | 3.0685 | 3.5406 |
| Heart | 0.2462 | 0.2670 | 0.3061 |
| Isolet | 63.3973 | 36.4517 | 100.1772 |
| Leukemia | 0.6926 | 1.4652 | 1.4659 |
| Madelon | 5.8683 | 5.8151 | 12.7560 |
| Mammographic | 1.2879 | 0.8617 | 0.8208 |
| Multiple-Features | 6.1392 | 6.8573 | 8.7926 |
| Musk1 | 1.0331 | 0.7663 | 0.8015 |
| Musk2 | 25.2961 | 11.0347 | 17.4254 |
| Newthyroid | 0.2970 | 0.2786 | 0.3264 |
| Penbased | 18.7128 | 26.6560 | 29.8070 |
| Phoneme | 5.2769 | 3.3581 | 5.3429 |
| Plant-Margin | 100.0346 | 79.9919 | 119.3066 |
| Ring1 | 6.6760 | 5.6050 | 9.6177 |
| Satimage | 7.7718 | 6.0654 | 10.1016 |
| Sonar | 0.37635 | 0.3245 | 0.5215 |
| Tic-Tac-Toe | 0.701825 | 0.5334 | 0.8900 |
| Titanic | 0.5451 | 0.4233 | 0.6711 |
| Vertebral-3C | 0.3206 | 0.4777 | 0.4530 |
| Wine-Red | 4.42855 | 3.2037 | 3.1914 |

than MULES, XgBoost, gcForest, and Random Forest. Additionally, concerning accuracy, MLR is better than XgBoost, gcForest, and Random Forest, while for the F1 score, MLR is better than MULES, gcForest, and Random Forest. On the other hand, there is no statistical difference between COME and MLR. For both the accuracy and F1 score, COME ranks first followed by MLR. For accuracy, MULES has the third rank and XgBoost has the fourth rank, while for the F1 score, the ranks are reversed. gcForest and Random Forest are consistently the lowest-ranking methods in both cases.

Table 6 shows the accuracy of the proposed method and the benchmark algorithms. It can be seen that COME has the highest scores on most datasets, while amongst the benchmark algorithms, MLR has the highest number of first places. COME achieves the highest difference compared to the benchmark algorithms on the Balance dataset, in which the accuracy of COME is 0.9681, which is higher than the second-best method (MLR) by around 10%. Another dataset in which the difference is significant is Breast-Tissue, where COME has an accuracy of 0.7813 while most of the benchmark algorithms only achieve a score of 0.7188. There are 6 datasets (Colon, GM4, Leukemia, Newthyroid, Tic-Tac-Toe, Vertebral-3C) where COME's performance is the same as some other benchmark algorithms. gcForest shares the highest accuracy on GM4 (1.0) and Newthyroid (0.9692), with COME. XgBoost achieves the highest score on just one dataset (Newthyroid) at 0.9692, which is the same score as that of COME, gcForest, and MLR. Although there are 4 datasets where Random Forest obtains the highest accuracy, the scores of Random Forest are the same score as those of COME on 3 datasets (Colon, Leukemia, and Vertebral-3C). Additionally, among the benchmark algorithms, MLR performs the best where it achieves the highest accuracy on 10 datasets, but 5 of the highest scores are shared by COME as well.

Table 7 shows the F1 score of COME and the benchmark algorithms. It can be seen that COME achieves first or equal first place on 20 out of

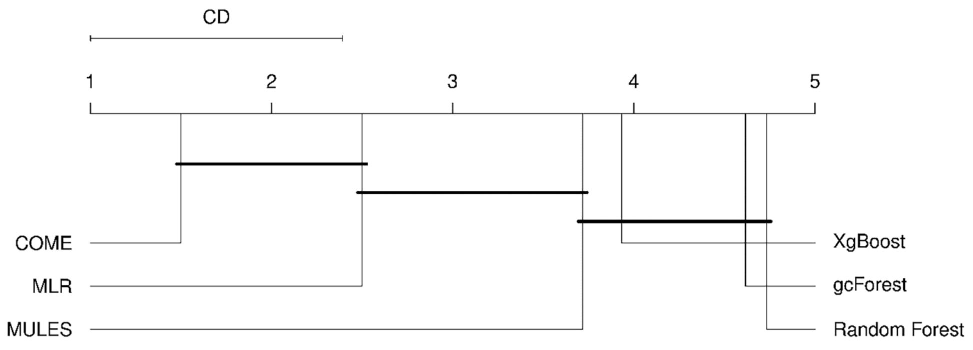

**Fig. 9.** Nemenyi test for the accuracy of COME and the benchmark algorithms
COME > MULES, XgBoost, gcForest, Random Forest
MLR > XgBoost, gcForest, Random Forest.

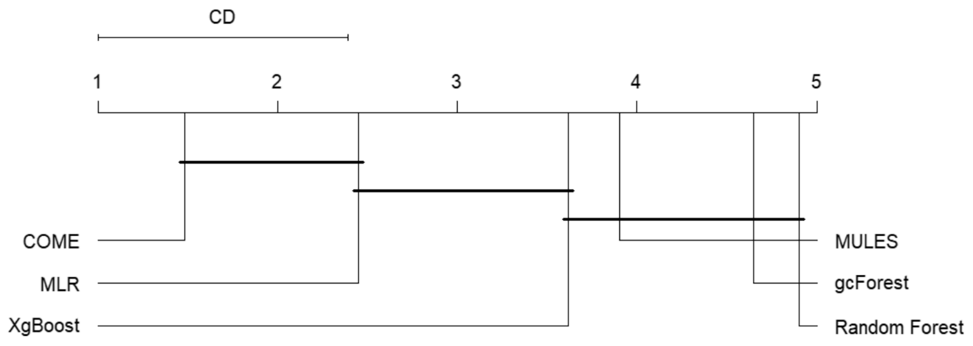

**Fig. 10.** Nemenyi test for the F1 score of COME and the benchmark algorithms
COME > XgBoost, MULES, gcForest, Random Forest
MLR > MULES, gcForest, Random Forest.

**Table 6**
The comparison of COME and benchmark algorithms for the accuracy.

| Dataset | COME | gcForest | Random Forest | XgBoost | MULES | MLR |
|---|---|---|---|---|---|---|
| Balance | **0.9681** | 0.8564 | 0.8085 | 0.8457 | 0.8351 | 0.8617 |
| Banana | **0.9013** | 0.8654 | 0.8365 | 0.8969 | 0.8899 | 0.8943 |
| Breast-Tissue | **0.7813** | 0.6875 | 0.7188 | 0.7188 | 0.7188 | 0.7188 |
| Cleveland | 0.6222 | 0.6222 | **0.6333** | 0.5889 | 0.5889 | 0.5778 |
| Colon | **0.8421** | 0.7368 | **0.8421** | 0.7895 | 0.7368 | 0.7895 |
| Conn-Bench-Vowel | 0.9811 | 0.6415 | 0.6101 | 0.8239 | 0.8050 | **0.9874** |
| Contraceptive | **0.5814** | 0.5566 | 0.5588 | 0.5701 | 0.5724 | 0.5747 |
| Electricity-Normalised | 0.9311 | 0.7998 | 0.7745 | 0.8529 | **0.9325** | 0.9255 |
| Embryonal | **0.6667** | 0.5000 | 0.3889 | 0.5000 | 0.6111 | 0.5000 |
| Fertility | **0.9333** | 0.9000 | 0.9000 | 0.9000 | 0.8667 | 0.9000 |
| GM4 | **1.0000** | **1.0000** | 0.9767 | 0.9033 | 0.9967 | **1.0000** |
| Heart | **0.8765** | 0.8272 | 0.8025 | 0.7531 | 0.7654 | 0.8519 |
| Isolet | 0.9534 | 0.8141 | 0.7534 | 0.9462 | 0.9517 | **0.9543** |
| Leukemia | **0.9545** | 0.9091 | **0.9545** | 0.9091 | **0.9545** | **0.9545** |
| Madelon | 0.7983 | 0.6350 | 0.6200 | 0.7000 | 0.7717 | **0.8000** |
| Mammographic | **0.8594** | 0.8394 | 0.8514 | 0.8233 | 0.2811 | 0.8394 |
| Multiple-Features | **0.9850** | 0.9783 | 0.9733 | 0.9800 | 0.9767 | 0.9833 |
| Musk1 | **0.8671** | 0.8531 | 0.8042 | 0.8322 | **0.8671** | **0.8671** |
| Musk2 | **0.9934** | 0.9510 | 0.8944 | 0.9768 | 0.9727 | 0.9909 |
| Newthyroid | **0.9692** | **0.9692** | 0.9538 | **0.9692** | 0.9385 | **0.9692** |
| Penbased | 0.9951 | 0.8957 | 0.8357 | 0.9897 | 0.9912 | **0.9955** |
| Phoneme | 0.9051 | 0.8181 | 0.8033 | 0.8570 | 0.9014 | **0.9100** |
| Plant-Margin | **0.8271** | 0.5333 | 0.4813 | 0.7167 | 0.7646 | 0.8167 |
| Ringnorm | **0.9811** | 0.9635 | 0.9270 | 0.9707 | 0.9658 | 0.9770 |
| Satimage | **0.9358** | 0.8695 | 0.8576 | 0.9120 | 0.9244 | 0.9337 |
| Sonar | **0.9365** | 0.8254 | 0.8413 | 0.8413 | 0.8254 | 0.8730 |
| Tic-Tac-Toe | **1.0000** | 0.8160 | 0.8056 | 0.9861 | 0.9792 | **1.0000** |
| Titanic | 0.7595 | 0.7474 | 0.7534 | 0.7504 | **0.7716** | 0.7504 |
| Vertebral-3C | **0.8495** | 0.7957 | **0.8495** | 0.8280 | 0.8280 | 0.8280 |
| Wine-Red | **0.6854** | 0.5854 | 0.5938 | 0.6375 | 0.6750 | 0.6688 |

**Table 7**
The comparison of COME and the benchmark algorithms for the F1 score.

| Dataset | COME | gcForest | Random Forest | XgBoost | MULES | MLR |
|---|---|---|---|---|---|---|
| Balance | **0.9426** | 0.6985 | 0.5664 | 0.5970 | 0.5863 | 0.6056 |
| Banana | **0.9003** | 0.8646 | 0.8341 | 0.8956 | 0.8890 | 0.8929 |
| Breast-Tissue | **0.7650** | 0.6389 | 0.7037 | 0.6989 | 0.6962 | 0.7013 |
| Cleveland | 0.3013 | 0.2559 | 0.2571 | **0.3016** | 0.2555 | 0.2642 |
| Colon | **0.8081** | 0.6360 | **0.8081** | 0.7286 | 0.6360 | 0.7286 |
| Conn-Bench-Vowel | 0.9814 | 0.6442 | 0.6134 | 0.8208 | 0.8089 | **0.9891** |
| Contraceptive | **0.5514** | 0.5244 | 0.5204 | 0.5461 | 0.5224 | 0.5335 |
| Electricity-Normalised | 0.9292 | 0.7961 | 0.7571 | 0.8483 | **0.9306** | 0.9235 |
| Embryonal | **0.6250** | 0.4109 | 0.3378 | 0.4582 | 0.5786 | 0.4109 |
| Fertility | **0.7321** | 0.4737 | 0.4737 | 0.6727 | 0.4643 | 0.4737 |
| GM4 | **1.0000** | **1.0000** | 0.9738 | 0.8955 | 0.9963 | **1.0000** |
| Heart | **0.8635** | 0.8056 | 0.7817 | 0.7271 | 0.7502 | 0.8363 |
| Isolet | 0.9538 | 0.8102 | 0.7369 | 0.9462 | 0.9518 | **0.9545** |
| Leukemia | **0.9454** | 0.8854 | **0.9454** | 0.8854 | **0.9454** | **0.9454** |
| Madelon | 0.7983 | 0.6248 | 0.6196 | 0.6996 | 0.7717 | **0.7999** |
| Mammographic | **0.8594** | 0.8390 | 0.8506 | 0.8231 | 0.2795 | 0.8394 |
| Multiple-Features | **0.9843** | 0.9775 | 0.9731 | 0.9803 | 0.9759 | 0.9825 |
| Musk1 | 0.8650 | 0.8513 | 0.7980 | 0.8292 | **0.8669** | 0.8658 |
| Musk2 | **0.9874** | 0.8968 | 0.7163 | 0.9538 | 0.9447 | 0.9824 |
| Newthyroid | **0.9548** | 0.9453 | 0.9260 | 0.9453 | 0.9052 | 0.9453 |
| Penbased | 0.9952 | 0.8961 | 0.8318 | 0.9897 | 0.9912 | **0.9955** |
| Phoneme | 0.8851 | 0.7822 | 0.7566 | 0.8280 | 0.8794 | **0.8909** |
| Plant-Margin | **0.8177** | 0.4914 | 0.4499 | 0.7001 | 0.7520 | 0.8065 |
| Ringnorm | **0.9811** | 0.9635 | 0.9266 | 0.9707 | 0.9658 | 0.9770 |
| Satimage | **0.9214** | 0.8322 | 0.8092 | 0.8932 | 0.9035 | 0.9197 |
| Sonar | **0.9361** | 0.8209 | 0.838 | 0.8393 | 0.8225 | 0.8714 |
| Tic-Tac-Toe | **1.0000** | 0.7715 | 0.7383 | 0.9841 | 0.9763 | **1.0000** |
| Titanic | 0.7059 | 0.6441 | 0.6966 | 0.6662 | **0.7143** | 0.6662 |
| Vertebral-3C | **0.7772** | 0.6809 | 0.7660 | 0.7395 | 0.7436 | 0.7527 |
| Wine-Red | 0.3252 | 0.2620 | 0.2517 | **0.3535** | 0.3226 | 0.3248 |

30 datasets. On several datasets, COME obtains substantially better results than the benchmark algorithms, i.e. on the Balance dataset, COME obtains an F1 score of 0.9426 while the scores of the benchmark algorithms are between 0.5664–0.6985, Breast-Tissue (0.765 vs 0.6389–0.7037), Colon (0.8081 vs 0.6360–0.7286, except Random Forest), Fertility (0.7321 vs 0.4643–0.6727). gcForest, Random Forest, XgBoost, and MULES achieve highest scores on 1, 2, 2, and 4 datasets respectively, but several of these highest scores are also shared by COME, or the difference with the second-best method is not large. Although MLR achieved the highest scores on 8 datasets, 3 highest scores are shared with COME, while the remaining 5 highest scores are not substantially larger than the 2nd highest scores achieved by COME.

The training time of COME is significantly higher than those of the other benchmark algorithms, except MULES. For example, on the Tic-tac-toe dataset, MULES and gcForest took 0.8761 h and 0.0867 h for training, respectively, while COME took 0.89 h. Several factors contribute to this difference. Firstly, both COME and MULES solve $S$ optimization problems for the ensemble of $S$ layers to find optimal connections in each layer. These algorithms use evolutionary-based approaches to search for optimal solutions, involving the evaluation of numerous candidates in multiple generations, which increases training time. Furthermore, COME uses 30 machine learning algorithms in each layer to train classifiers, whereas MULES employs only 5 classifiers per layer as in the original paper [2]. This further extends COME's training time compared to its counterparts. On the other hand, the test time of COME is lower than that of MLR on datasets where the optimal number of layers is just one (e.g., Banana, Embryonal, Fertility, GM4, Heart, Leukemia, Madelon, Multiple Features, Musk1, Plant-Margin, and Titanic). This is due to the retention of a subset of connections between training data and classifiers, resulting in fewer classifiers being used in COME, whereas MLR uses all 30 classifiers in the ensemble. For other datasets, COME's testing time exceeds that of MLR because it generates more layers with 30 classifiers each. Compared to gcForest, COME's testing time is substantially higher. For instance, on the Balance and Madelon datasets, gcForest took 0.1406 and 0.2188 s, respectively, while COME took 0.3032 and 0.682 s.

## 5. Conclusions

In this paper, we introduced COME, a novel multi-layer ensemble of classifiers. In COME, each classifier at a specific layer is connected to some classifiers from the previous layer. The connection between two classifiers in two layers indicates that the output of the classifier in the previous layer serves as the input for the classifier in the current layer. The Sum Rule combines predictions from the last layer to produce final predictions. We search for the optimal set of connections in the multi-layer ensemble to maximise the classification accuracy. We first encoded the connections between layers by using a binary encoding scheme. We then used Differential Evolution, one of the most popular evolutionary computation-based methods, to find the optimal encoding which exactly is the optimal set of connections for multi-layer ensemble systems. For each candidate, the accuracy on a validation set is used as the fitness criteria. In our work, the optimisation algorithm only plays a role to solve the optimisation problem in Eq. (7). Thus, it is note that not only DE but also any optimisation algorithms can be used in our method. We focused on developing a novel multi-layer ensemble algorithm based on the idea of connections between classifiers in two consecutive layers instead of developing a new optimisation algorithm.

To evaluate the performance of COME over benchmark algorithms, we conducted experiments on 30 datasets from the UCI Machine Learning Repository and OpenML. We first evaluated the performance of COME with three different optimisation algorithms, DE, GA, and PSO. The experimental results indicated the outperformance of DE when working in COME compared to the others. More specifically, COME-DE obtains the best accuracy and F1 scores on more than 20 datasets. Notably, on some datasets, COME-DE has higher results compared to COME-Gaand COME-PSO by more than 6%. Additionally, we compared the results of our proposed ensemble with several state-of-the-art ensemble learning algorithms, namely gcForest, Random Forest, XgBoost, MULES and MLR. The experimental results demonstrate the effectiveness of our proposed ensemble.

There are a number of potential future directions for our proposed ensemble. Firstly, by parallelising the candidate evaluation during the DE optimisation procedure on GPUs [44] our proposed ensemble can achieve considerable reductions in computational time. Secondly, our current multi-layer ensemble follows a sequential structure where outputs from one layer feed into the next. Enhancements could be made by exploring new types of connections in the multi-layer ensemble, such as skip-connection, or by finding new ways to enhance the input data for each layer, such as training the classifiers on different subsets of features or data. Thirdly, adapting our ensemble for various scenarios such as incremental learning or data stream-based learning is another avenue. This involves developing methods for the ensemble to adjust its configuration with new data arrivals, balancing accuracy, and time considerations during updates.

## Ethical approval

This article does not contain any studies with human participants performed by any of the authors.

## CRediT authorship contribution statement

**Truong Dang:** Writing – original draft, Methodology, Data curation. **Tien Thanh Nguyen:** Writing – original draft, Supervision, Conceptualization. **Alan Wee-Chung Liew:** Writing – review & editing, Validation, Methodology. **Eyad Elyan:** Writing – review & editing, Validation, Methodology. **John McCall:** Writing – review & editing, Validation, Supervision, Conceptualization.

## Declaration of competing interest

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

## Data availability

The raw data required to reproduce the above findings are available to download from https://archive.ics.uci.edu/datasets and https://www.openml.org/search?type=data&status=active. The processed data required to reproduce the above findings are available to download from https://1drv.ms/u/s!Aor3F4KdvJjrkwG4JAl3a2qObFDc?e=L9h5P9.

## Funding Declaration

There is no funding declaration.

## Appendix Notation table

| Notation | Explanation |
|---|---|
| $\mathbf{x}_n = (x_{n1}, x_{n2}, \ldots, x_{nD})$ | The $D-$ feature vector of the $n^{th}$ training instance |
| $\widehat{y}_n$ | The true label of the $n^{th}$ training instance |
| $\mathscr{D} = \{(\mathbf{x}_n, \widehat{y}_n)\}$ | The training set |
| $\mathscr{X} = \begin{bmatrix} x_{11}, x_{12}, \ldots, x_{1D} \\ x_{21}, x_{22}, \ldots, x_{2D} \\ \ldots \\ x_{N1}, x_{N2}, \ldots, x_{ND} \end{bmatrix}$ | The set of all training instances |
| $\mathscr{Y} = \begin{bmatrix} \widehat{y}_1 \\ \widehat{y}_2 \\ \ldots \\ \widehat{y}_N \end{bmatrix}$ | The ground truth of all training instances |
| $\boldsymbol{h}$ | A hypothesis (i.e. classifier) |
| $\{\boldsymbol{h}_k\}$ | The set of $K$ hypotheses |
| $\mathscr{K} = \{\mathscr{K}_k\}$ | $K$ learning algorithms |
| $C$ | The combining algorithm |
| $\widetilde{\boldsymbol{h}} = C\{\{\boldsymbol{h}_k\}, k = 1, \ldots, K\}$ | The final decision (after the combining procedure) |
| $\left\{\boldsymbol{h}_k^{(i)},\ k = 1, \ldots, K\right\}$ for $i = 1, \ldots, S$ | The $K$ classifiers in the $i^{th}$ layer in a multi-layer ensemble with $S$ layers |
| $p_{k,m}^{(1)}(\mathbf{x}_n)$ | The prediction of the $k^{th}$ classifier in the first layer that observation belongs to the class label $y_m$ |
| $P^{(1)}(\mathbf{x}_n) = \left[p_{1,1}^{(1)}(\mathbf{x}_n),\ p_{1,2}^{(1)}(\mathbf{x}_n), \ldots, p_{K,M}^{(1)}(\mathbf{x}_n)\right]$ | The prediction vector by the classifiers in the first layer for observation $\mathbf{x}_n$ |
| $\mathscr{P}_1 = \left[P^{(1)}(\mathbf{x}_1) \quad P^{(1)}(\mathbf{x}_2) \quad \ldots \quad P^{(1)}(\mathbf{x}_N)\right]^T$ | The prediction vectors by the classifiers in the first layer for all observations in the training set |
| $\triangle\!\!\!\!+$ | Concatenation operator |
| $\mathscr{P}_i$ | The prediction vectors by the classifiers in the $i^{th}$ layer for all observations |
| $\mathscr{P}_i^{train}$ | The prediction vectors by the classifiers in the $i^{th}$ layer for all observations in the training set |
| $\mathscr{P}_i^{val}$ | The prediction vectors by the classifiers in the $i^{th}$ layer for all observations in the validation set |
| $\mathcal{L}_i = (\mathcal{X} \triangle\!\!\!\!+ \mathcal{P}_i, \mathcal{Y})$ | The input data $\mathscr{L}_i$ for the $(i+1)^{th}$ layer |
| $p_{k,m}^{(S)}(\mathbf{x})$ | The probability assigned by the $k^{th}$ classifier in layer $S$ of the ensemble for the $m^{th}$ class. |
| $T$ | The number of cross-validation folds |
| $E_1 = \left[e_{0,k}^{(1)}, k = 1 \ldots K\right]$ | The binary encoding for the 1st layer |
| $E_i = \left[e_{j,k}^{(i)}, k = 1 \ldots K,\ j = 0 \ldots K_1\right] (i = 2, \ldots, S)$ | The binary encoding for the $i^{th}$ layer |
| $e_{j,k}^{(i)}$ | Denotes whether there is a connection between j$^{\text{th}}$ classifier at $(i-1)^{\text{th}}$ layer and k$^{\text{th}}$ classifier at the i$^{\text{th}}$ layer. $(j \neq 0)$ If j is zero: Denotes whether there is connection from training data to the k$^{\text{th}}$ classifier at the i$^{\text{th}}$ layer. |
| $\mathscr{V}_i$ | The validation set at the $i^{th}$ layer |
| $\lvert \cdot \rvert$ | The cardinality of a set |
| $[\![ . ]\!]$ | The indicator function, which is equal to 1 if the condition is true, else returns 0. |
| $popSize$ | The population size |
| $maxGen$ | The maximum number of generations |
| $obj(.)$ | The objective function |
| $u_{i,d}$ | The $d^{th}$ dimension of the $i^{th}$ candidate in the DE algorithm |
| $u_r(t)$ | The $r^{th}$ candidate at time $t$ in the DE algorithm |
| $u_{off}(t)$ | The offspring created at time $t$ in the DE algorithm |
| $u_{tr,d}(t)$ | The $d^{th}$ dimension of the trial vector at time $t$ |
| $F$ | The scaling factor |
| $CR$ | The crossover rate |
| $rand()$ | The random function (within the $[0,\ 1]$ range) |
| $\mathscr{H}_i$ | The ensemble of classifiers of $i^{th}$ layer |
| $T_{stop}$ | Number of maximum layers |
| $T_{earlyStopping}$ | Number of early stopping epochs |
| $K_{prev}$ | Number of classifiers used in the previous layer |
| $e_{j,k}^{i,r_1}$ | The $(j,k)$ entry of the $r_1{}^{\text{th}}$ candidate of the binary encoding at the $i^{\text{th}}$ layer. |
| $e_{j,k}^{off}$ | The $(j,k)$ entry of the offspring of the binary encoding. |
| $e_{j,k}^{tr}$ | The $(j,k)$ entry of the trial vector of the binary encoding. |

## References

[1] X. Dong, Z. Yu, W. Cao, et al., A survey on ensemble learning, Front. Comput. Sci. 14 (2020) 241–258.

[2] T.T. Nguyen, N.V. Pham, T. Dang, A.V. Luong, J. McCall, A.W.-C. Liew, Multi-layer heterogeneous ensemble with classifier and feature selection, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference (GECCO), 2020, pp. 725–733.

[3] T.T. Nguyen, M.P. Nguyen, X.C. Pham, A.W.C. Liew, Heterogeneous classifier ensemble with fuzzy rule-based meta learner, Inf. Sci. (Ny) 422 (2018) 144–160.

[4] Z.-H. Zhou, J. Feng, Deep forest: towards An alternative to deep neural networks, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI), 2017, pp. 3553–3559.

[5] A.V. Luong, T.T. Nguyen, A.W.-C. Liew, S. Wang, Heterogeneous ensemble selection for evolving data streams, Pattern Recognit. 112 (2021) 107743.

[6] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[7] M. Baldeon Calisto, S.K. Lai-Yuen, AdaEn-Net: an ensemble of adaptive 2D-3D Fully Convolutional Networks for medical image segmentation, Neural Netw. 126 (2020) 76–94.

[8] C. Liu, F. Huang, A. Qiu, Monte Carlo ensemble neural network for the diagnosis of Alzheimer's disease, Neural Netw. 159 (2023) 14–24.
[9] Y. Xie, W. Sun, M. Ren, S. Chen, Z. Huang, X. Pan, Stacking ensemble learning models for daily runoff prediction using 1D and 2D CNNs, Expert Syst. Appl. 217 (2023) 119469.
[10] N. Dvornik, J. Mairal, C. Schmid, Diversity with cooperation: ensemble methods for few-shot classification, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 3722–3730.
[11] L.V. Utkin, M.S. Kovalev, A.A. Meldo, A deep forest classifier with weights of class probability distribution subsets, Knowl. Based Syst. 173 (2019) 15–27.
[12] T. Dang, T.T. Nguyen, J. McCall, E. Elyan, C.F. Moreno-García, Two layer ensemble of deep learning models for medical image segmentation, Cognit. Comput. (2024).
[13] A.V. Luong, T.T. Nguyen, A.W.-C. Liew, Streaming multi-layer ensemble selection using dynamic genetic algorithm, in: 2021 Digital Image Computing: Techniques and Applications (DICTA), 2021, pp. 1–8.
[14] G. Martínez-Muñoz and A. Suárez, 'Aggregation ordering in bagging', 2004.
[15] J. Cao, W. Li, C. Ma, Z. Tao, Optimizing multi-sensor deployment via ensemble pruning for wearable activity recognition, Inf. Fusion 41 (C) (2018) 68–79.
[16] H. Guo, H. Liu, R. Li, C. Wu, Y. Guo, M. Xu, Margin & diversity based ordering ensemble pruning, Neurocomputing 275 (2018) 237–246.
[17] T. Dang, T.T. Nguyen, C.F. Moreno-García, E. Elyan, J. McCall, Weighted ensemble of deep learning models based on comprehensive learning particle swarm optimization for medical image segmentation, in: 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 744–751.
[18] T.T. Nguyen, A.V. Luong, T.M. Van Nguyen, T.S. Ha, A.W.-C. Liew, J. McCall, Simultaneous meta-data and meta-classifier selection in multiple classifier system, in: GECCO `19: Genetic and Evolutionary Computation Conference, 2019, pp. 39–46.
[19] Z. Yu, et al., Multiobjective semisupervised classifier ensemble, in: IEEE Transactions on Cybernetics 49, 2019, pp. 2280–2293.
[20] Y. Xu, Z. Yu, C.L.P. Chen, Classifier ensemble based on multiview optimization for high-dimensional imbalanced data classification, IEEE Trans. Neural Netw. Learn. Syst. 35 (1) (2024) 870–883.
[21] D.T. Do, T.T. Nguyen, T.T. Nguyen, A.V. Luong, A.W.-C. Liew, J. McCall, Confidence in prediction: an approach for dynamic weighted ensemble, in: N. T. Nguyen, K. Jearanaitanakij, A. Selamat, B. Trawiński, S. Chittayasothorn (Eds.), Intelligent Information and Database Systems, Springer International Publishing, Cham, 2020, pp. 358–370.
[22] J.-Y. Zou, M.-X. Sun, K.-H. Liu, Q.-Q. Wu, The design of dynamic ensemble selection strategy for the error-correcting output codes family, Inf. Sci. (Ny) 571 (2021) 1–23.
[23] S. García, Z.-L. Zhang, A. Altalhi, S. Alshomrani, F. Herrera, Dynamic ensemble selection for multi-class imbalanced datasets, Inf. Sci. (Ny) 445-446 (2018) 22–37.
[24] A.H. Madkour, H.M. Abdelkader, A.M. Mohammed, Dynamic classification ensembles for handling imbalanced multiclass drifted data streams, Inf. Sci. (Ny) 670 (2024) 120555.
[25] X. Zhu, J. Li, J. Ren, J. Wang, G. Wang, Dynamic ensemble learning for multi-label classification, Inf. Sci. (Ny) 623 (2023) 94–111.
[26] T. Elsken, J.H. Metzen, F. Hutter, Neural architecture search: a survey, J. Mach. Learn. Res. 20 (55) (2019) 1–21.
[27] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 8697–8710.
[28] X. He, K. Zhao, X. Chu, AutoML: a survey of the state-of-the-art, Knowl. Based Syst. 212 (2021) 106622.
[29] X. Chen, L. Xie, J. Wu, Q. Tian, Progressive differentiable architecture search: bridging the depth gap between search and evaluation, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 1294–1303.
[30] L. Xie, A. Yuille, Genetic CNN, in: ICCV, 2017, pp. 1379–1388.
[31] R. Miikkulainen et al., 'Evolving deep neural networks', arXiv:1703.00548, 2017.
[32] K.O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evol. Comput. 10 (2) (2002) 99–127.
[33] E. Real, A. Aggarwal, Y. Huang, Q.V. Le, Regularized evolution for image classifier architecture search, in: Proceedings of the AAAI Conference on Artificial Intelligence 33, 2019, pp. 4780–4789.
[34] Z. Yang, et al., CARS: continuous evolution for efficient neural architecture search, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 1826–1835.
[35] Y. Sun, H. Wang, B. Xue, Y. Jin, G.G. Yen, M. Zhang, Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor, IEEE Trans. Evol. Computat. 24 (2) (2020) 350–364.
[36] T. Domhan, J.T. Springenberg, F. Hutter, Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves, in: Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI), 2015, pp. 3460–3468.
[37] T. Dang, A.V. Luong, A.W.C. Liew, J. McCall, T.T. Nguyen, Ensemble of deep learning models with surrogate-based optimization for medical image segmentation, in: IEEE Congress on Evolutionary Computation (CEC), 2022, pp. 1–8.
[38] R. Storn, K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359, https://doi.org/10.1023/A:1008202821328.
[39] F. Neri, V. Tirronen, V. 'Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif. Intell. Rev. 33 (1–2) (2010) 61–106, https://doi.org/10.1007/s10462-009-9137-2.
[40] T.T. Nguyen, T. Dang, V.A. Baghel, A.V. Luong, J. McCall, A.W.-C. Liew, Evolving interval-based representation for multiple classifier fusion, Knowl. Based Syst. (2020) 106034.
[41] X. Zeng, J. Song, S. Zheng, G. Xu, S. Zeng, Y. Wang, A. Esamdin, Y. Huang, S. Xia, J. Huang, A fast inversion method of parameters for contact binaries based on differential evolution, Astron. Comput. 47 (2024).
[42] K.M. Ting, I.H. Witten, Issues in stacked generalization, J. Artif. Intell. Res. 10 (1999) 271–289.
[43] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[44] D.M. Janssen, W. Pullan, A.W.C. Liew, "GPU based differential evolution: new insights and comparative study", arXiv:2405.16551v1, 2024.