Deep heterogeneous ensemble.

NGUYEN, T.T., DANG, M.T., PHAM, T.D., DAO, L.P., LUONG, A.V., MCCALL, J. and LIEW, A.W.C.

2019

Deep Heterogeneous Ensemble

Tien Thanh Nguyen[1], Manh Truong Dang[1], Tien Dung Pham[2], Lan Phuong Dao[3], Anh Vu Luong[4], John McCall[1], and Alan Wee Chung Liew[4]

[1] School of Computing Science and Digital Media, Robert Gordon University, Aberdeen, UK
[2] Zalo Company, Hochiminh city, Vietnam
[3] AN Company, Hanoi, Vietnam
[4] School of Information and Communication Technology, Griffith University, Australia
Email: t.nguyen11@rgu.ac.uk

Abstract. In recent years, deep neural networks (DNNs) have emerged as a powerful technique in many areas of machine learning. Although DNNs have achieved great breakthrough in processing images, video, audio, and text, it also has some limitations such as needing a large number of labeled data for training and having a large number of parameters. Ensemble learning, meanwhile, provides a learning model by combining many different classifiers such that ensemble of classifier is better than using single classifier. In this study, we propose a deep ensemble framework called Deep Heterogeneous Ensemble (DHE) for the supervised learning tasks. In each layer of our algorithm, the input data is passed through a feature selection method to remove irrelevant features and prevent overfitting. The cross-validation with K learning algorithms is applied to the selected data to obtain the meta-data and the K base classifiers for the next layer. By this way, one layer will output the meta-data as the input data for the next layer, the base classifiers, and the indices of the selected meta-data. A combining algorithm is then applied on the meta-data of the last layer to obtain the final class prediction. Experiments on 30 datasets confirm that the proposed DHE is better than a number of well-known benchmark algorithms.

Keywords: Ensemble method · Deep learning · Multiple classifiers · Ensemble of classifiers · Ensemble system.

1 Introduction

It is well known that there is no one learning algorithm that perform well on all datasets. Ensemble learning, where multiple classifiers are trained and combined to get better performance than using a single classifier, offers an effective solution [10]. Nowadays, ensemble learning have been applied in many diverse fields such as computer vision, bioinformatics, and software engineering. The success of ensemble learning was summarized in [3] in its ability to reduce the risk of choosing a wrong learner and the ability to obtain a better approximation for the unknown feature-class relationship. Among the many ensemble systems introduced, Random Forest [1] and XgBoost [2] have been reported as the superior methods for supervised learning [4, 19].

In recent years, deep neural networks (DNNs) have emerged as a powerful supervised learning technique applied to images, video, audio and text processing tasks [6]. Despite its successes, there are some limitations of DNNs. First, these deep models have a large amount of parameters, for example, 144 million parameters in the DNN model in [14]. Such complex models can only be trained on specially-designed hardware. Second, DNNs require a huge amount of labeled data. When the cost of labeled data is too prohibitive, deep models might not bring about the expected gains in performance. Finally, there is still little insight into the behaviors of DNNs, or how they were able to achieve such a good performance. Even though there have been some works on understanding and visualizing deep convolutional neural networks, such as in [18], the design of deep learning models remains without a coherent theoretical basis.

In this study, we aim to design a novel deep ensemble learning model for the classification problem based on multiple layers of ensemble of different learning algorithms (called DHE) which can get the advantages of both ensemble system and DNNs as well as overcome the limitations of DNNs. We propose a cascade ensemble model of many layers in which one layer produces new data in the form of the predictions of base classifiers for the next layer. On each layer, the input training data will pass through a feature selection process to select a subset of features as input. We then train the set of learning algorithms on the selected data to obtain the base classifiers for the layer and the predictions. These predictions will form new training data for the next layer. The DHE starts from the original training data and then goes deeply layer-by-layer until a performance score does not improve with the addition of further layers. We then train a combining algorithm on the output of the final layer for the final class prediction. The classification process works in a straightforward manner where an instance passes through the layers and the prediction from the base classifiers in last layer is combined to obtain final predicted label.

In section 2, we briefly review ensemble learning for classification and recent developments in ensemble learning-based deep model. In section 3, we give a detailed description of the general architecture for DHE. Experimental studies on a number of datasets are provided in Section 4, followed by conclusion in Section 5.

2 Background and Related Work

2.1 Ensemble Learning

In ensemble learning, classifiers can be generated: either heterogeneously or homogeneously [9, 10]. In homogeneous ensemble, many classifiers are generated by training one learning algorithm on many different training sets obtained from the original one. These classifiers are then combined for the collaborated decision. There are many homogeneous ensemble frameworks and among them Bagging, Boosting, Random Subspace, and Random Forest are the most well-known methods. These methods generate the diversity in the ensemble by using different approaches for new training data generation. Bagging, for example, uses the bootstrap method, i.e. sampling without replacement to generate the training datasets. Random Subspace method, meanwhile, puts attention on the feature space by randomly selecting many subsets of feature set to get the new training data associated with these subsets. Despite many successes, homogeneous ensemble methods have some limitations such as they only work with unstable learning algorithms like in Bagging or they are only applicable to high-dimension data for the method of Random Subspace.

On the other hand, heterogeneous ensemble uses different learning algorithms to generate system diversity. A popular framework in heterogeneous ensemble learning is stacking [15] in which the predictions of the set of base classifiers on the training data is used to train a combining algorithm for the final prediction. The research on heterogeneous ensemble focuses on designing combining algorithms that effectively combine the predictions of the base classifiers.

The combiners for heterogeneous ensemble system are mainly categorized into two groups: fixed combiner and trainable combiner. Fixed combiner does not require training using the meta-data of the training set. Although the training cost is reduced in the fixed combiner, ignoring the label information in the meta-data of the training set can degrade the performance of this approach. The trainable combiner on the other hand is trained on meta-data of training set for better classification accuracy. Some examples of trainable combining algorithms are Bayesian-based method with Gaussian [11] and Gaussian Mixture Model [7], MRL [15], and Information granularity-based method [10, 12].

2.2 Ensemble learning-based deep models

Recently, some deep learning models using homogeneous ensemble methods have been introduced using the representation learning through layer-by-layer processing. The first ensemble learning-based deep model was proposed by Zhou and Feng [19] where a deep learning system called gcForest was constructed with two random forests and two completely-random trees working in each layer. In detail, the class distribution vector from each tree in a forest for an instance is obtained by counting the percentage of different classes of training instances at the leaf node where that instance falls into. The prediction vector of a forest is computed as the mean of averaging class distribution vectors of all the trees inside. This vector is then concatenated with the original data as the input data for the next layer. Utkin et al. [16] extended gcForest by considering the weights of the trees in the same forest when averaging their predictions. These weights are found by minimizing a loss function on the training data based on the Euclidean distance between the weighted average vector and the crisp distribution vector based on class labels of the training instances. Utkin et al. [17] also introduced a Siamese Deep Forest (SDF) which concatenates the pairs of weighted averaged prediction vectors with original data as the input data for the next layer. Although these deep learning models achieve superior performance on diverse datasets, the homogeneous ensemble requires a large number of base classifiers, resulting in expensive computation. For heterogeneous ensemble, Nguyen et al. [8] introduced a two-stage model including K base classifiers in the first layer and one classifier in the second layer. The meta-data from the output of the first layer is separated into K data pieces associated with the K base classifiers. The learning algorithm on the second layer trains on these data pieces to obtain the meta-data which is then combined by a combining algorithm for the final classification. Our idea here is to design a deep heterogeneous ensemble system where a multi-layer architecture with a small number of diverse classifiers in each layer offers superior performance than existing homogeneous ensemble based deep models.

3 Deep Heterogeneous Ensemble System

3.1 General description

Let $\mathcal{D} = \{(\mathbf{x}_n, \hat{y}_n)\},\ |\mathcal{D}| = N$ be the training data, where $\mathbf{x}_n \in \mathcal{R}^d$ is the feature vector of the n^{th} training observation and $\hat{y}_n \in \mathcal{Y},\ |\mathcal{Y}| = M$ is its corresponding label. We denote $\boldsymbol{\mathcal{K}} = \{\mathcal{K}_k\}$ as the set of learning algorithms in which $|\boldsymbol{\mathcal{K}}| = K$. Our task is to approximate the relationship $f : \mathbf{x}_n \to \hat{y}_n$ by a hypothesis and then use it to assign a label for each test instance. In the context of ensemble system, an ensemble of classifiers is trained on the given training set to obtain different approximations for the relationship f. These classifiers are then combined for the final approximation.

In DHE, the data is passed through several layers. The combining algorithm then trains the combiner on the output of the last layer. Each layer receives the meta-data of training data obtained from the previous layer and outputs the triple $\{\mathcal{F}_i, \mathcal{L}_i, \mathcal{H}_i\}$:

- $\mathcal{F}_i$: The indices of features selected by a feature selection method.
- $\mathcal{L}_i$: The meta-data generated at layer i which is used as input for layer i.
- $\mathcal{H}_i$: The set of base classifiers which predicts on an instance at layer i.

In the design of each layer, before generating the meta-data and the set of base classifiers, we use a feature selection method to select a subset of features to prevent overfitting as well as to improve ensemble performance. Here we are inspired by the idea of drop out in DNNs in which units (both hidden and visible) in a neural network are dropped to reduce overfitting. Feature selection is an important research topic in pattern recognition, data analysis, and data mining. In fact, real-world data usually contain redundant attributes which increases storage and computation costs. Besides, some of the attributes may be irrelevant to the class that the data belongs to. Hence, eliminating redundant or irrelevant attributes will improve classification performance and efficiency. Feature selection is not only beneficial in avoiding overfitting but it also provide a deep insight into the data generation process [13].

As commented in [19], the stacking model uses a layer framework in the ensemble since re-training on multiple layers of meta-data can lead to overfitting. In this work, we apply the filter feature selection strategy [13] on the input data of a layer to prevent overfitting. As the filter approach focuses only on the intrinsic properties of the data while ignoring the classification process, it can reduce overfitting when multiple layers of meta-data are generated. Each layer outputs the indices of the selected features in a set $\mathcal{F}_i$ which will be used for feature selection during classification.

To generate the meta-data $\mathcal{L}_i$ from the labeled data at the i^{th} layer, we use T_i-fold cross validation on the selected meta-data $\hat{\mathcal{L}}_{i-1}$ obtained the meta-data $\mathcal{L}_{i-1}$ of the $(i-1)^{th}$ layer. In detail, $\hat{\mathcal{L}}_{i-1}$ is divided into T_i disjoin parts in which the cardinality of each part is nearly similar. One part will be used as the test set and the other will be used as the training set. By doing this, an observation in $\hat{\mathcal{L}}_{i-1}$ will be used as the training data for $T_i - 1$ times and as test data for 1 time. At the end, we obtain $p^i_{k,m}(\mathbf{x}_n)$, which is the prediction of the k^{th} classifier at the i^{th} layer that $\mathbf{x}_n$ belongs to the class label y_m. We assume that the classifiers output the prediction in the form of soft label [10, 12]:

$$\sum_{m=1}^{M} p^i_{k,m}(\mathbf{x}_n) = 1;\ \forall k = 1, ..., K; n = 1, ..., N \tag{1}$$

From these predictions, we can now create the meta-data $\mathcal{L}_i$ as the input for the next layer. This is done by concatenating the prediction vector of each classifier for each observation. The meta-data of the training set at the i^{th} layer is as an $N \times MK$ matrix. The row of this matrix is the predictions of all classifiers for a specific instance and the column is the predictions of all instances arranged in the order:

$$\mathcal{L}_i = [\mathcal{L}_i(\mathbf{x}_1)...\mathcal{L}_i(\mathbf{x}_N)]^T \tag{2}$$

$$\mathcal{L}_i(\mathbf{x}_n) = \left[p^i_{1,1}(\mathbf{x}_n), p^i_{1,2}(\mathbf{x}_n), ..., p^i_{K,M-1}(\mathbf{x}_n), p^i_{K,M}(\mathbf{x}_n)\right] \tag{3}$$

The set of base classifiers of the i^{th} layer $\mathcal{H}_i$ is also learned by training K given learning algorithms on the selected meta-data $\hat{\mathcal{L}}_{i-1}$ i.e. $\mathcal{H}_i = \{h_{i,k}\}\ h_{i,k} : \hat{\mathcal{L}}_{i-1}(\mathbf{x}_n) \to \hat{y}_i$. At the end of the last layer, a combining algorithm is used to train the combiner on the meta-data for the collaborated decision.

Given a test instance $\mathbf{x}$, it will pass through the layers to finally obtain the meta-data when reaching the last layer. Referring to the set of selected features, we select associated features $\hat{\mathcal{L}}_{i-1}(\mathbf{x})$ in the meta-data of $\mathbf{x}$ from the $(i-1)^{th}$ layer. The base classifiers $\mathcal{H}_i$ then work on this selected meta-data to obtain the meta-data of $\mathbf{x}$ at the i^{th} layer. Once again, the meta-data of $\mathbf{x}$ i.e. $\mathcal{L}_i(\mathbf{x})$ is given in a shape of a MK-vector. The meta-data output of the last layer is fed into the combiner to generate the predicted label.

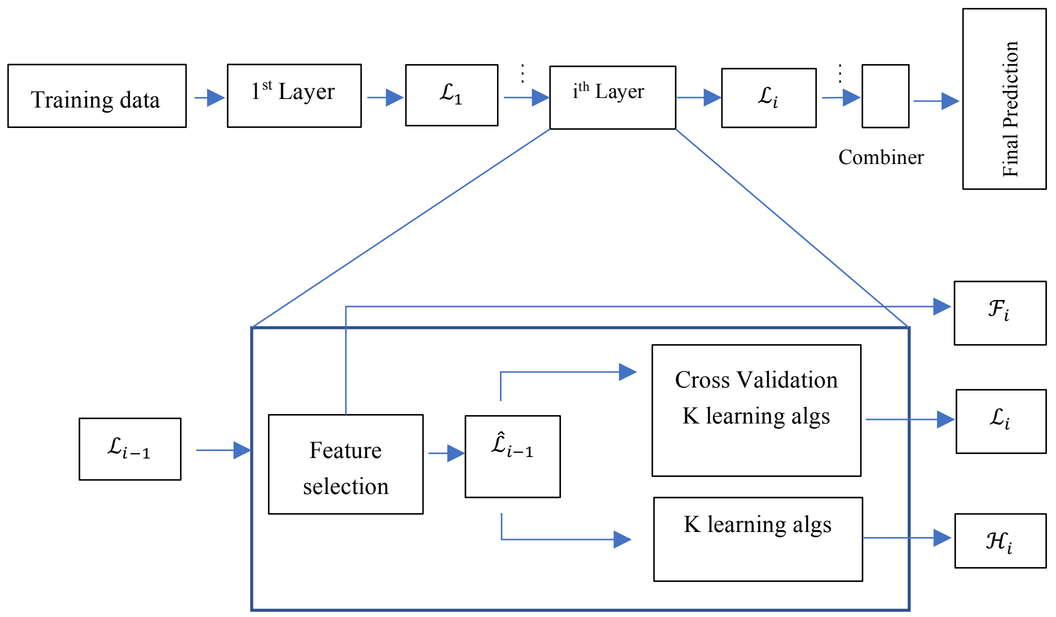

Fig. 1. The deep heterogeneous ensemble system

3.2 Combining algorithms

To examine the influence of the combining algorithm on the proposed framework, we experimented on two popular combining algorithms. The first method is the Sum Rule, a fixed combiner for heterogeneous ensemble systems [5]. Sum Rule simply sums the meta-data of each test instance for each class and assigns label to the class with the maximum value. In the proposed framework, the Sum Rule for a test instance $\mathbf{x}$ at the i^{th} layer is given by:

$$\mathbf{x} \in y_t \text{ if } t = \text{argmax}_{m=1,\ldots,M} \sum_{k=1}^{K} p_{k,m}^{i}(\mathbf{x}) \tag{4}$$

We also used MLR [15], a trainable combiner, to obtain the final collaborated decision. MLR is a weight-based technique in which the weigh that each classifier contributes to the combination is found by solving:

$$\min \left\{ \sum_{i=1}^{N} \left(\mathbb{I}[y_m = \hat{y}_i] - \sum_{k=1}^{K} \omega_{km} p_{k,m}^{i}(\mathbf{x}_i) \right)^2 \right\} \tag{5}$$

where $\mathbf{x}_i$ is a training observation and $\mathbb{I}[\cdot]$ is the indicator function which returns 1 if the condition is true and 0 otherwise. Once the weights are obtained, prediction is given by:

$$\mathbf{x} \in y_t \text{ if } t = \text{argmax}_{m=1,\ldots,M} \left\{ \sum_{k=1}^{K} \omega_{km} p_{k,m}^{i}(\mathbf{x}) \right\} \tag{6}$$

4 Experimental Studies

4.1 Configurations

We used the same configuration for DHE in all experiments: in each layer, we used three learning algorithms namely K Nearest Neighbor (K was set to 5), Logistic Regression, and Naïve Bayes to generate the meta-data and the base classifiers. We used the 10-fold Cross Validation in all layers to generate the meta-data. To automatically determine the number of layers, we split the training data into the training part and the validation part. We then used the training part to grow the model and trained the combiner on the meta-data of each layer. The performance of

Table 1. The information of experimental datasets

Data	**# of training**	**# of validation**	**# of test**	**# of class labels**	**# of dimensions**
Appendicitis	59	15	32	2	7
Artificial	392	98	210	2	10
Australian	386	97	207	2	14
Balance	349	88	188	3	4
Banana	2968	742	1590	2	2
Biodeg	590	148	317	2	41
Blood	418	105	225	2	4
Breast-Cancer	382	96	205	2	9
Breast-Tissue	59	15	32	6	9
Cleveland	165	42	90	5	13
Contraceptive	824	207	442	3	9
GM4	560	140	300	3	1000
Haberman	171	43	92	2	3
Heart	151	38	81	2	13
Hill_Valley	1356	340	728	2	100
Isolet	4365	1092	2340	26	617
Led7digit	280	70	150	10	7
Madelon	1120	280	600	2	500
Mammographic	464	117	249	2	5
Musk1	266	67	143	2	166
Newthyroid	120	30	65	3	5
Penbased	6155	1539	3298	10	16
Ring	4144	1036	2220	2	20
Sonar	116	29	63	2	60
Titanic	1232	308	661	2	3
Twonorm	4144	1036	2220	2	20
Vehicle	473	119	254	4	18
Waveform_w_noise	2800	700	1500	3	40
Waveform_wo_noise	2800	700	1500	3	21
Wine	99	25	54	3	13

DHE at each layer was evaluated by the combiner on the validation part. We followed the experiments in [19] in which 80% of training data is used for the training part and the remainder is used for the validation part. The layer growing process is stopped if the classification error rate on the validation part does not improve after 5 layers.

We evaluated DHE using 3 different feature selection methods. On the input meta-data of each layer, we computed (i) chi-squared statistic between each meta-data feature and class (ii) the ANOVA F-value, and (iii) the mutual information. These methods were implemented from the scikit-learn library. We then ranked the meta-data features based on the results and selected the top 50% of the meta-data as the final input features for the layer.

We compared DHE to some well-known benchmark algorithms in deep ensemble learning and standard ensemble learning: A standard Random Forest with 2000 trees, gcForest (with 500 trees in each of 8 forests), XgBoost with 2000 trees, and Multi-Layer Perceptron (MLP). As the performance of MLP significantly depends on the network structure, we examined a range of parameters including input-30-20-output, input-50-30-output, and input-70-50-output; and then reported the best performance.

To assess the statistical significance in the comparisons, we used the Friedman test to test the null hypothesis that all methods perform equally on all datasets. If the p-value of this test is smaller than a significant threshold e.g. 0.05, we reject the null hypothesis and conduct the Nemenyi post-hoc test for pairwise comparison on all datasets. The experiments were conducted on 30 datasets selected from various sources such as the UCI Machine Learning Repository and [9] (for the GM4 dataset). The detailed information of these datasets are presented in Table 1.

4.2 Experimental Results

Different feature selection methods: Fig 2 and 3 present the classification error rates of DHE with MLR and Sum Rule using the three different feature selection methods (denoted by DHE_MLR and DHE_Sum). Clearly, different feature selection methods have a significant effect on a number of datasets such as Cleveland, Blood, Artificial with DHE_MLR and Breast-Tissue, Cleveland, Balance, and Penbased with DHE_Sum. In the next section, we use the results with mutual information-based feature selection method to compare with those of the benchmark algorithms.

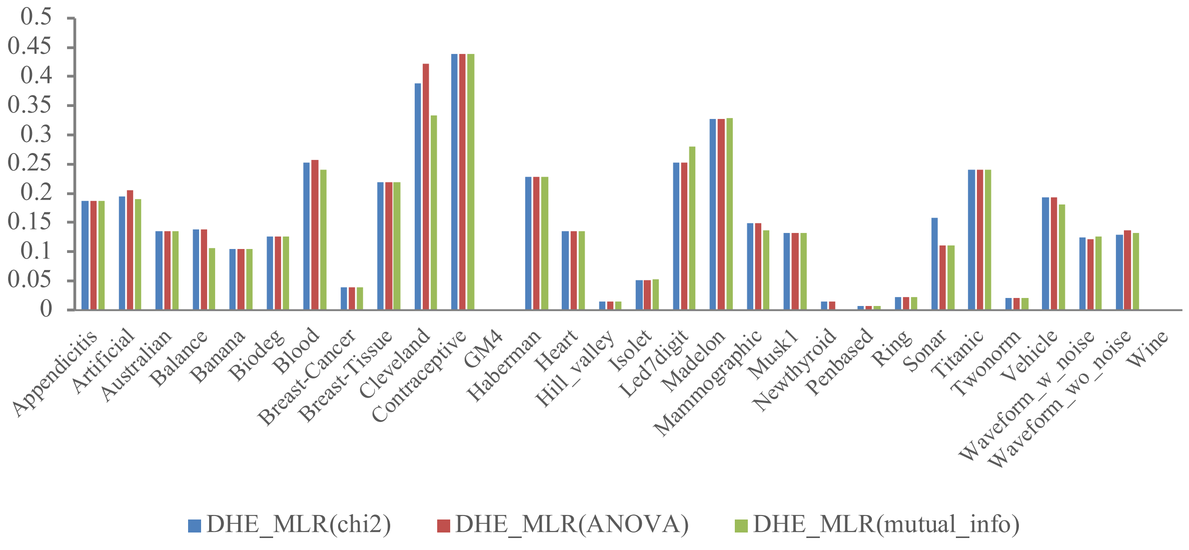

Fig. 2. The classification error rate of DHE_MLR using three feature selection methods

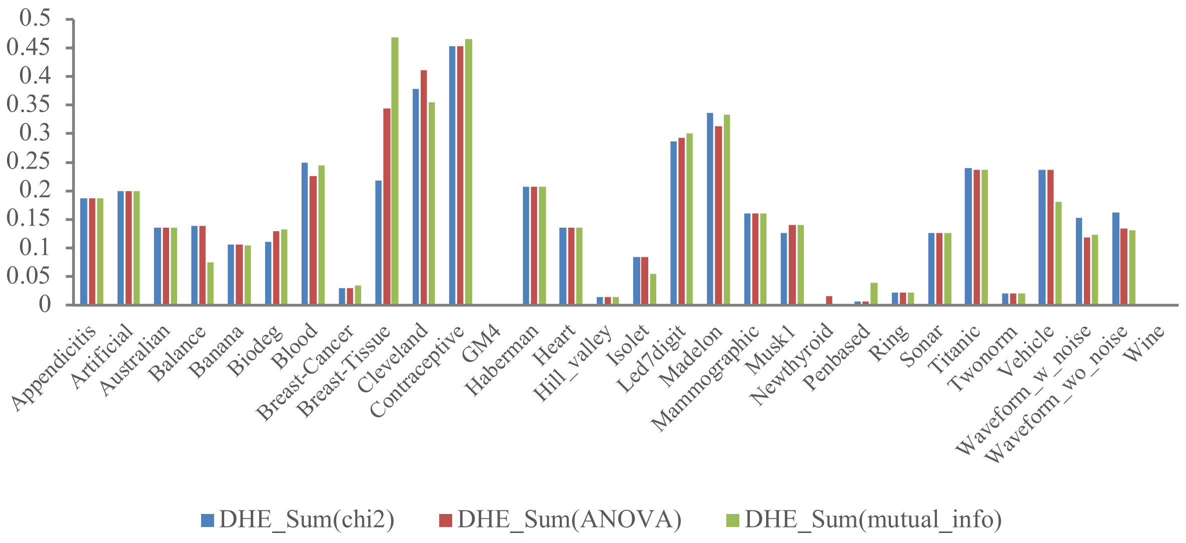

Fig. 3. The classification error rate of DHE_Sum using three feature selection methods

Comparison to baselines: Table 2 presents the classification error rates of DHE_MLR and DHE_Sum using mutual information and 4 well-known benchmark algorithms. Based on the experimental results and the statistical test results, some observations can be made:

- The P-value of the Friedman test is 1.261E-08 thus we rejected the null hypothesis. The Nemenyi test result in Fig 4 shows that DHE_MLR is better than all four benchmark algorithms and DHE_Sum is better than MLP and XgBoost. Meanwhile, there are no differences between DHE_MLR and DHE_Sum and between DHE_Sum and gcForest/Random Forest.
- DHE_MLR achieves the lowest average rank among all methods (rank value 2.03). On the 30 datasets, DHE_MLR ranks first in 12 cases (40%), ranks second in 12 cases (40%), and does not rank below third on any datasets. DHE_Sum ranks second with a rank value 2.6. DHE_Sum only performs poorly on 3 datasets: Penbased, Contraceptive, and Breast-Tissue. XgBoost and MLP meanwhile are the two poorest methods where MLP ranks fifth (rank value 4.40) and XgBoost ranks sixth (rank value 4.68).
- The proposed DHE is significantly superior than the benchmark algorithms on some datasets such as Breast Tissue (0.2188 of DHE_MLR vs. 0.25 of gcForest), Cleveland (0.3333 of DHE_MLR vs. 0.4 of gcForest), Haber-

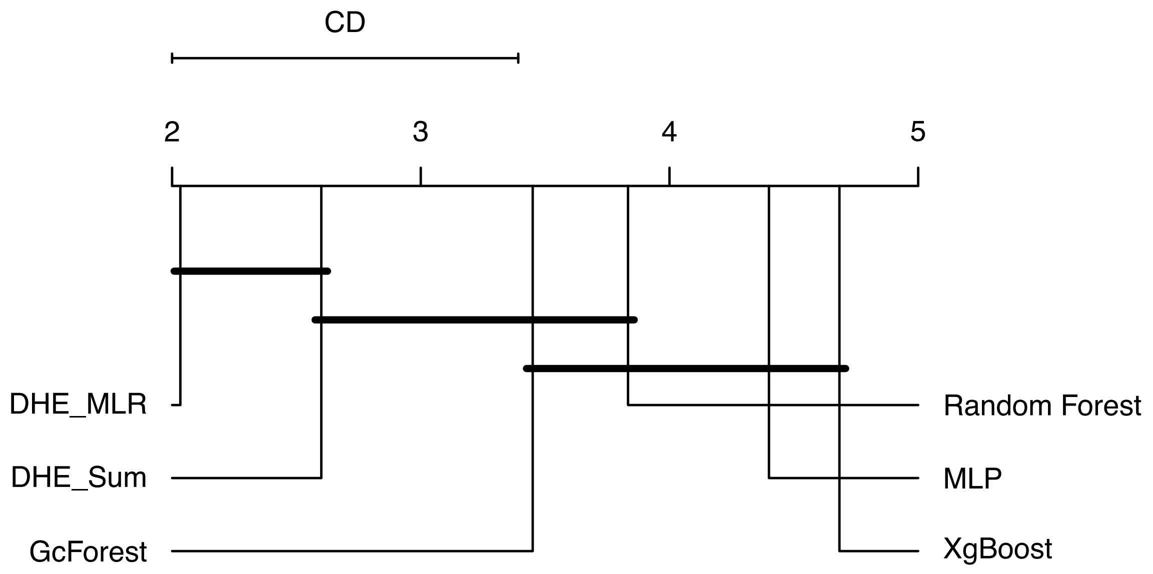

Fig. 4. The Nemenyi test result

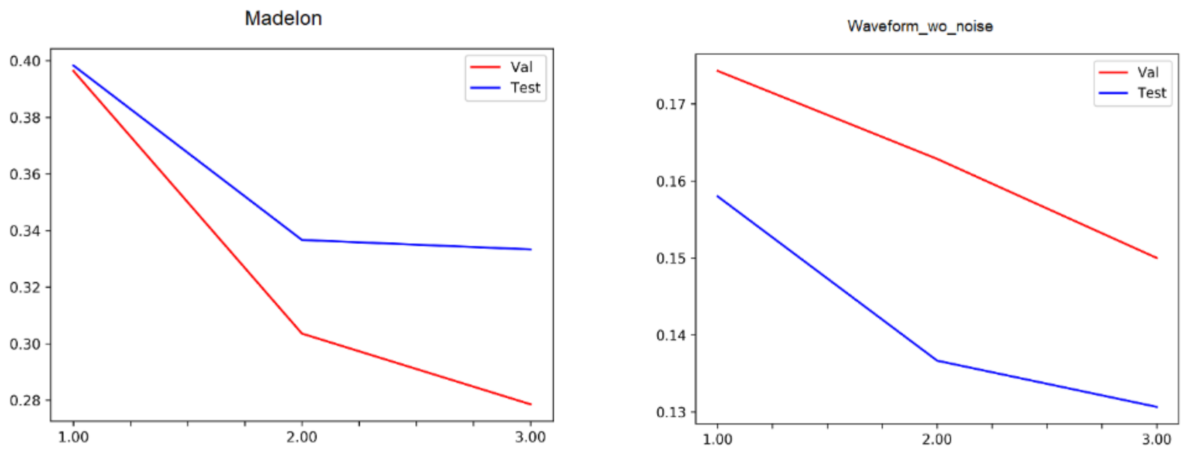

Fig. 5. The classification error rate of DHE_Sum on the validation set and test set after each layer on two datasets

man (0.2065 of DHE_Sum vs. 0.25 of MLP), Heart (0.1358 of DHE_MLR and DHE_Sum vs. 0.1728 of gcForest), and Hill_valley (0.0137 of DHE_Sum vs. 0.3214 of XgBoost).

- DHE_MLR is better than DHE_Sum on some datasets such as Penbased, Breast-Tissue, Contraceptive, and Mammographic, and Led7digit. It is because MLR trains the combiner on the meta-data for class label prediction. However, there are some exceptions in which DHE_Sum outperforms DHE_MLR, for example, on Balance and Haberman dataset.
- In comparison to gcForest, DHE is simpler and faster with only three different learning algorithms in each layer. In contrast, gcForest uses a large number of trees in each forest (500 trees in each of 8 forests in the original paper and in our experiment), resulting in very expensive training.

Benefit of layer-by-layer processing: Fig 5 shows the classification error rate of DHE_Sum with mutual information feature selection on the validation set and test set for the Madelon and Waveform_wo_noise dataset. It can be seen that the classification error rate of DHE_Sum on the validation set and test set both reduces through each layer. This is evidence of overfitting avoidance. This also indicates that the layer-by-layer processing works

Table 2. The classification error rate of DHE_MLR, DHE_Sum, and the benchmark algorithms

	MLP	**Random Forest**	**gcForest**	**XgBoost**	**DHE_MLR**	**DHE_Sum**
Appendicitis	0.1875 (3)	0.2188 (6)	0.1875 (3)	0.1875 (3)	0.1875 (3)	0.1875 (3)
Artificial	0.3238 (6)	0.2381 (4)	0.2048 (3)	0.2429 (5)	0.1905 (1)	0.2000 (2)
Australian	0.2802 (6)	0.1159 (1)	0.1353 (3)	0.1401 (5)	0.1353 (3)	0.1353 (3)
Balance	0.0266 (1)	0.1862 (6)	0.1543 (5)	0.1117 (4)	0.1064 (3)	0.0745 (2)
Banana	0.1019 (1)	0.1044 (2)	0.1075 (5)	0.1107 (6)	0.1050 (3.5)	0.1050 (3.5)
Biodeg	0.1199 (2)	0.1104 (1)	0.1388 (5)	0.1546 (6)	0.1262 (3)	0.1325 (4)
Blood	0.2667 (3)	0.2711 (4)	0.2800 (5.5)	0.2800 (5.5)	0.2400 (1)	0.2444 (2)
Breast-Cancer	0.0390 (3.5)	0.0439 (5)	0.0293 (1)	0.0488 (6)	0.0390 (3.5)	0.0341 (2)
Breast-Tissue	0.8438 (6)	0.2813 (3)	0.2500 (2)	0.3125 (4)	0.2188 (1)	0.4688 (5)
Cleveland	0.6222 (6)	0.4222 (4)	0.4000 (3)	0.4444 (5)	0.3333 (1)	0.3556 (2)
Contraceptive	0.4299 (1)	0.4570 (3)	0.4615 (4.5)	0.4615 (4.5)	0.4389 (2)	0.4661 (6)
GM4	0.0200 (5)	0.0000 (2.5)	0.0000 (2.5)	0.1033 (6)	0.0000 (2.5)	0.0000 (2.5)
Haberman	0.2500 (3)	0.2717 (4)	0.2826 (5)	0.3587 (6)	0.2283 (2)	0.2065 (1)
Heart	0.3086 (6)	0.1852 (4)	0.1728 (3)	0.2222 (5)	0.1358 (1.5)	0.1358 (1.5)
Hill_valley	0.4382 (6)	0.3365 (4)	0.3832 (5)	0.3214 (3)	0.0151 (2)	0.0137 (1)
Isolet	0.0500 (1)	0.0611 (5)	0.0615 (6)	0.0543 (3)	0.0526 (2)	0.0556 (4)
Led7digit	0.2800 (1.5)	0.3133 (5)	0.3000 (3.5)	0.3400(6)	0.2800 (1.5)	0.3000 (3.5)
Madelon	0.4867 (6)	0.3217 (2)	0.3933 (5)	0.3117 (1)	0.3283 (3)	0.3333 (4)
Mammographic	0.2570 (6)	0.2169 (5)	0.1526 (2)	0.2048 (4)	0.1365 (1)	0.1606 (3)
Musk1	0.1818 (6)	0.1538 (4)	0.1259 (1)	0.1678 (5)	0.1329 (2)	0.1399 (3)
Newthyroid	0.0154 (3.5)	0.0154 (3.5)	0.0308 (5)	0.0462 (6)	0.0000 (1.5)	0.0000 (1.5)
Penbased	0.0121 (5)	0.0088 (4)	0.0076 (1.5)	0.0082 (3)	0.0076 (1.5)	0.0391 (6)
Ring	0.1482 (6)	0.0428 (5)	0.0221 (1.5)	0.0284 (4)	0.0225 (3)	0.0221 (1.5)
Sonar	0.1587 (5)	0.1587 (5)	0.1270 (2.5)	0.1587 (5)	0.1111 (1)	0.1270 (2.5)
Titanic	0.2496 (4.5)	0.2496 (4.5)	0.2496 (4.5)	0.2496 (4.5)	0.2405 (2)	0.2375 (1)
Twonorm	0.0293 (6)	0.0248 (5)	0.0221 (3)	0.0239 (4)	0.0212 (1.5)	0.0212 (1.5)
Vehicle	0.4882 (6)	0.2165 (3)	0.2205 (4)	0.2283 (5)	0.1811 (1.5)	0.1811 (1.5)
Waveform_w_noise	0.1633 (6)	0.1367 (4)	0.1273 (3)	0.1380 (5)	0.1260 (2)	0.1240 (1)
Waveform_wo_noise	0.1447 (5)	0.1440 (4)	0.1373 (3)	0.1713 (6)	0.1320 (2)	0.1307 (1)
Wine	0.6111 (6)	0.0000 (2.5)	0.0000 (2.5)	0.0556 (5)	0.0000 (2.5)	0.0000 (2.5)
Average rank	4.4	3.83	3.45	4.68	2.03	2.6

effectively with heterogeneous ensemble system in which using the meta-data as the input data for the next layer in a multi-layer deep framework reduces the overall classification error rate.

5 Conclusions

In summary, we introduced a novel DHE algorithm inspired by the layer-by-layer processing of DNNs. In each layer of our DHE, we first applied a feature selection method to the input data (i.e. the meta-data from the previous layer) which remove the irrelevant features and prevent overfitting for the model. A cross-validation procedure and K different learning algorithms is then applied to the selected data to obtain the meta-data and the base classifiers. The meta-data then is then used as the input data for the next layer. We used a combining algorithm on the meta-data of the last layer to train a combiner for the class prediction. The layer growing process is controlled by using a validation set where we stop creating new layer if the classification error rate does not improve after a specific number of layers. Experiments on 30 datasets confirm that DHE is better than four well-known benchmark algorithms.

The proposed DHE resolves some limitations of DNNs mentioned before. First, like the gcForest, DHE only has two parameters: the number of layers for early stopping control and the number of folds in cross-validation which can simply set with default values. Second, DHE only uses a small number of learning algorithms on each layer (3 in our experiments), resulting in a very fast training process. Finally, by using only traditional learning algorithms in each layer, this model thus does not require a large amount of labeled data or any specialized hardware for training like in many DNNs.

References

1. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
2. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. pp. 785–794. ACM (2016)

3. Duin, R.P.: The combining classifier: to train or not to train? In: Object recognition supported by user interaction for service robots. vol. 2, pp. 765–770. IEEE (2002)
4. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? The Journal of Machine Learning Research **15**(1), 3133–3181 (2014)
5. Kittler, J., Hater, M., Duin, R.P., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(3), 226–239 (1998)
6. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436 (2015)
7. Nguyen, T.T., Liew, A.W.C., Tran, M.T., Nguyen, M.P.: Combining multi classifiers based on a genetic algorithm–a gaussian mixture model framework. In: International Conference on Intelligent Computing. pp. 56–67. Springer (2014)
8. Nguyen, T.T., Liew, A.W.C., Tran, M.T., Nguyen, T.T.T., Nguyen, M.P.: Fusion of classifiers based on a novel 2-stage model. In: International Conference on Machine Learning and Cybernetics. pp. 60–68. Springer (2014)
9. Nguyen, T.T., Nguyen, M.P., Pham, X.C., Liew, A.W.C.: Heterogeneous classifier ensemble with fuzzy rule-based meta learner. Information Sciences **422**, 144–160 (2018)
10. Nguyen, T.T., Nguyen, M.P., Pham, X.C., Liew, A.W.C., Pedrycz, W.: Combining heterogeneous classifiers via granular prototypes. Applied Soft Computing **73**, 795–815 (2018)
11. Nguyen, T.T., Nguyen, T.T.T., Pham, X.C., Liew, A.W.C.: A novel combining classifier method based on variational inference. Pattern Recognition **49**, 198–212 (2016)
12. Nguyen, T.T., Pham, X.C., Liew, A.W.C., Pedrycz, W.: Aggregation of classifiers: A justifiable information granularity approach. IEEE transactions on cybernetics **49**(6), 2168–2177 (2018)
13. Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. bioinformatics **23**(19), 2507–2517 (2007)
14. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229 (2013)
15. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of artificial intelligence research **10**, 271–289 (1999)
16. Utkin, L.V., Kovalev, M.S., Meldo, A.A.: A deep forest classifier with weights of class probability distribution subsets. Knowledge-Based Systems (2019)
17. Utkin, L.V., Ryabinin, M.A.: A siamese deep forest. Knowledge-Based Systems **139**, 13–22 (2018)
18. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. pp. 818–833. Springer (2014)
19. Zhou, Z.H., Feng, J.: Deep forest: Towards an alternative to deep neural networks. arXiv preprint arXiv:1702.08835 (2017)