A Dynamic Ensemble Learning Algorithm for Neural Networks

[1]Kazi Md. Rokibul Alam, [2*]Nazmul Siddique, [3]Hojjat Adeli

Abstract This paper presents a novel dynamic ensemble learning (DEL) algorithm for designing ensemble of neural networks (NN). DEL algorithm determines the size of ensemble, the number of individual NNs employing a constructive strategy, the number of hidden nodes of individual NNs employing a constructive-pruning strategy, and different training samples for individual NN's learning. For diversity, negative correlation learning has been introduced and also variation of training samples has been made for individual NN that provide better learning from the whole training samples. The major benefits of the proposed DEL compared to existing ensemble algorithms are 1) automatic design of ensemble; 2) maintaining accuracy and diversity of NNs at the same time; and 3) minimum number of parameters to be defined by user. DEL algorithm is applied to a set of real-world classification problems such as the cancer, diabetes, heart disease, thyroid, credit card, glass, gene, horse, letter recognition, mushroom, and soybean datasets. It has been confirmed by experimental results that DEL produces dynamic NN ensembles of appropriate architecture and diversity that demonstrate good generalization ability.

Keywords Neural network ensemble, back-propagation algorithm, negative correlation learning, constructive algorithms, pruning algorithms.

1 Introduction

Neural network (NN) structures have been used for knowledge representation [1], modeling [2, 3, 4], prediction [5, 6], design automation [7], classification [8, 9], identification [10] and nonlinear control [11] applications in many domains. All these applications mainly used monolithic structure for NN. In a monolithic structure, the NN is represented by a single NN architecture for the whole task to be performed [12, 13, 14]. Scalability is a major impairment for monolithic NN for a wide range of applications. Incremental learning is also not possible as the addition of new elements to NN requires retraining of the NN with old and new data [15, 16]. An inevitable phenomenon in the retraining of NN is the catastrophic forgetting (also known as crosstalk), which was first reported by McCloskey and Cohen [17]. Two types of crosstalk phenomena can get exposed during retraining: temporal and spatial crosstalk. In temporal crosstalk, learned knowledge is lost during retraining of a new task. In spatial crosstalk, NN cannot learn two or more tasks simultaneously [18]. Kemker et al. [19] demonstrated that catastrophic forgetting problem in incremental learning paradigm has not been resolved despite many claims and showed methods of measuring such catastrophic forgetting can be measured. A number of attempts has been made to mitigate the phenomenon such as regularization, rehearsal and pseudorehearsal, life-long learning based dynamic combination, dual-memory models and ensemble methods [16, 20-23]. A collection or committee of individual NNs can also be advantageous for addition of a new NN to store new knowledge mitigating forgetting phenomena where tasks can be subdivided [24]. Instead of employing a large NN for a complex problem, the researchers are impressed by the idea of decomposing the problem into smaller subtasks leading to smaller architecture, shorter training time and increased performance [24, 25]. NN ensemble-based classifier can also improve generalization ability [25, 26]. The structure of an NN ensemble is illustrated in Figure 1. Each NN in the ensemble (1 through *n*) is first trained on the training instances. The output of the ensembles is calculated from the predicted outputs, O_i, $i = 1,2,\cdots,n$, of the individual NNs [26]. The challenge here is to design a learning algorithm for ensemble NN. The initial weights, topology of NNs, training datasets, and training algorithms also play decisive roles in the design of ensembles [23, 25]. The approaches to designing ensembles constitute by varying these parameters.

Many algorithms similar to NN ensembles [25] have been reported in the literature such as mixer of experts [27], boosting [28] and bagging [29]. The main drawbacks of these algorithms are manual design and predefined number of neurons in the hidden layer and the number of NNs in an ensemble.

In general, ensemble and modular approaches are employed for combining NNs. The ensemble approach attempts to

[1]Department of Computer Science and Engineering, Khulna University of Engineering and Technology, Khulna, Bangladesh, Tel: 8801714-087-216, e-mail: rokib@cse.kuet.ac.bd.

[2]School of Computing, Engineering and Intelligent Systems, Ulster University, BT48 7JL, Londonderry, UK, Tel: +44(0)2871675340, e-mail: nh.siddique@ulster.ac.uk.

[3]Department of Civil, Environmental and Geodetic Engineering, the Ohio State University, Columbia, CO 80309 USA, e-mail: adeli.1@osu.edu.

*Corresponding author

generate a reliable and accurate output by combining the outputs of a set of trained NNs rather than selecting the best NN. Whereas the modular approach strives to have each NN as self–contained or autonomous [14, 24]. In modular approach, the problem is divided into a number of tasks. Each task is assigned to an individual NN to be accomplished. It is not possible to know the best size of NN a priori. The size of NN is defined by the number of layers and the number of neurons in each layer. Moreover, the backpropagation (BP) [30, 31] algorithm is not useful for training NN unless the topology is known. Therefore, finding the correct topology is the foremost design issue. In order to define the topology of an NN, a number of parameters such as the number of layers, number of hidden neurons, activation functions, and degree of connectivity have to be determined. A second issue is to determine the training parameters that include the initial weights of the NN, the learning rate, acceleration term, momentum term and weight update rule. The choice of the topological and training parameters has significant impact on the training time and the performance of the NN. Unfortunately, there is no straightforward method of selecting the parameters rather the designer has to depend on the expert knowledge or employ empirical method.

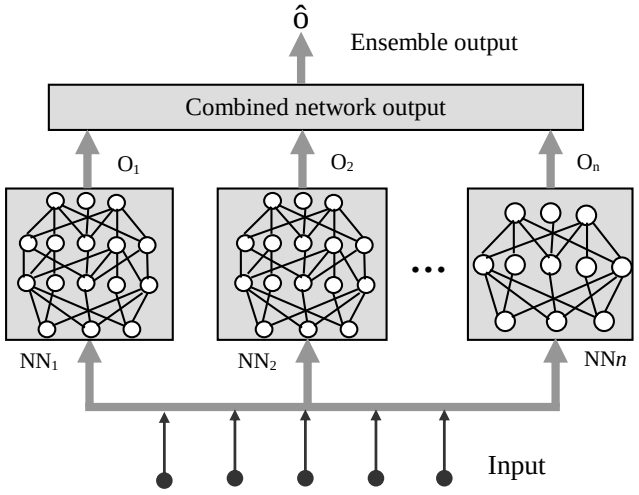


Fig. 1: A neural networks ensemble

The performance of NNs in an ensemble is dependent on a number of factors such as (i) the topology of the NNs and the initial structure; (ii) the training method; (iii) the learning rate; (iv) the input and output representations and (v) the content of the training sample [32]. Eventually, the numbers of NNs and the number of neurons in the hidden layers in NNs determine the performance of an ensemble. In most of the cases, these are predefined by human experts based on available a priori information. Formal learning theory is used to estimate the size of the ensemble system based on the complexity and the examples required learning the particular function. In such cases, the generalization error becomes high if the number of examples is small. Consequently, choosing appropriate NN topology is still something of an art. The data examples play a crucial role in learning where learning is sensitive to initial weights and learning parameters [33, 34, 35].

The purpose of this research is to design an NN ensemble that addresses the following issues: (i) automatic determination of NN ensemble architecture (i.e. the number of NNs in the ensemble), (ii) automatic determination of the size of individual NNs (i.e., the number of hidden neurons in individual NNs) and (iii) variation of training examples for each individual NN's better learning. Real-world classification problems are used to verify the effectiveness and the generalization ability of the ensemble.

The paper is organized as follows: Section 2 presents the related works. Section 3 contains the description of DEL algorithm. Section 4 presents the datasets description, experimental results and comparison. Section 5 presents a discussion. Some conclusions are made in Section 6.

2 Related Works

In ensemble learning, the individual NNs are called base learners. They are single classifiers, which are trained and combined together to ease individual errors and crop generalization independently. Hitherto, efforts have been made to design ensemble by combining NNs based on either the accuracy or the diversity [25, 36, 37]. There are evidences that accurate and diverse NNs can produce a good ensemble that distribute errors over different regions of the input space [38, 39]. Rosen [40] proposed an ensemble-learning algorithm that also trains individual NNs sequentially where the individual NNs minimize training errors as well as de-correlate previous training errors. Sequential training of an NN does not affect the NNs that were previously trained, which is a major disadvantage in ensemble learning. Consequently, there is no correlation between the errors of the individual NNs [41]. The topology of the mixtures-of-experts (ME) [27] can produce biased individual NNs which may be negatively correlated [32]. The disadvantage of ME is that it needs a separate gating NN and also can not provide a balance control over the bias-variance-covariance tradeoff [34].

A two-stage design approach is employed in most of the architectures mentioned above where individual NNs are generated first followed by combining them. As the combination stage does not provide any feedback to design stage, some individual NNs designed independently may not contribute significantly to the ensemble [34]. Therefore, some researchers proposed a one-stage design process and used a penalty term into the error function of each NN. The researchers also proposed simultaneous and interactive training for all NNs in the ensemble instead of independent and sequential training [41]. NNs with negative correlation can be created by reassuring specialization and cooperation among the NNs in an ensemble. This will enable NNs learning the different regions of training data space and ensure the ensemble learns the whole data space.

To ensure interaction between NNs and simultaneous learning in an ensemble, some researchers employed evolutionary computing [32]. Liu et al. [32] applied

evolutionary algorithm for ensemble learning of NNs with negative correlation. This approach can determine the optimal number of NNs and the combinations of NNs in an ensemble using fitness sharing mechanism.

Chen and Yao [33] employed multi-objective genetic algorithm [42] for regularized NCL optimizing errors of the base NNs and their diversity in ensemble. Mousavi and Eftekhari [43] proposed static ensemble selection and deploys the popular multi-objective genetic algorithm NSGA-II [42]. This combination of static ensemble selection and NSGA-II ensures selecting the best classifiers and their optimal combination.

There are two other widely popular approaches to ensemble learning, namely, Constructive NN Ensemble (CNNE) [44] and Pruning NN Ensemble (PNNE) [45]. CNNE determines the number of NNs in the ensemble and the hidden neurons of the individual NNs by employing negative correlation learning (NCL) [34, 41] in an incremental fashion. On the other hand, PNNE employs a competitive decay approach. PNNE uses a neuron cooperation function in each NN for the hidden neurons and a selective deletion of NNs in the ensemble based on the criterion of over-fitting. PNNE employs NCL to ensure diversity of the NNs in the ensemble.

Islam et al. [29] proposed two incremental learning algorithms for NNs in ensemble using NCL: NegBagg and NegBoost. NegBagg fixes the number of hidden neurons of NNs in ensemble by constructive method. NegBoost also uses constructive method to fix the number of hidden neurons of NNs as well as the number of NNs in the ensemble.

Yin et al. [46] proposed a two-stage hierarchical approach to ensemble learning called Dynamic Ensemble of Ensembles (DE^2). DE^2 comprises component classifiers and interim ensembles. The final DE^2 is obtained by weighted averaging. Cruz et al. [47] used a two-phase dynamic ensemble selection (DES) framework. In the first phase, DES extracts meta-features from training data. In the second phase, DES uses a meta-classifier to estimate the competence of the base classifier to be added to the ensemble.

Chen and Yao [48] show that NCL considers the entire ensemble as a single machine with the objective of minimising the mean square error (MSE) and NCL does not employ regularisation while training. They proposed a regularised NCL (RNCL) incorporating a regularisation term for the ensemble which enables the RNCL decomposing the training objectives into sub-objectives each of which is implemented by an individual NN. RNCL shows improved performance over the NCL even when noise level is higher in datasets.

Semi-supervised learning is the mechanism of learning using large amount of unlabelled data and small amount of labelled data. Chen and Wang [49] proposed a Semi-supervised boosting framework taking three assumptions such as smoothness, cluster and manifold into consideration where they used a cost function comprising the margin cost on labelled data and the regularization penalty on unlabelled data. Experiments on benchmarks and real-world classification reveal constant improvement by the algorithm. Semi-supervised learning is a widely popular method due to its higher accuracy at a lower effort.

The generalization of an ensemble is related to the accuracy of the base NNs and the diversity among NNs [37, 38]. Higher accuracy for the base NNs leads to lower diversity among them. To strike a balance of the dilemma between accuracy and diversity in an ensemble, Chen et al. [50] proposed a semi-supervised NCL (semiNCL) where a correlation penalty term on labelled and unlabeled data is incorporated into the cost function of each individual NNs in the ensemble.

Though the semi-supervised learning has been very successful for labelled and unlabelled data, its generalisation ability is sensitive to incorrect labelled data. To mitigate this limitation, Soares et al. [51] proposed a cluster-based boosting (CBoost) with cluster regularization. In CBoost, the base NNs in the ensemble jointly performs a cluster-based semi-supervised optimisation. Extensive experimentation shows that the CBoost has significant generalisation ability over the other ensembles.

Recently, Rafiei and Adeli [52] reported a new neural dynamic classification algorithm. A comprehensive review of multiple classifier systems based on the dynamic selection of classifiers was reported by Britto et al. [53]. Recent developments in ensemble methods are analysed by Ren et al. [54]. Cruz et al. [55] reported a review on the recent advances on dynamic classifier selection techniques. Dynamic mechanism is used in the generalisation phase in those studies, while the dynamic mechanism is employed in the training phase in DEL.

3 Dynamic Ensemble Learning (DEL)

3.1 Main Steps of the Algorithm

Unlike fixed ensemble architecture, DEL automatically determines the number base learner NNs and their architectures in an ensemble during the training phase. The DEL algorithm is presented in 8 steps in the sequel. The flow diagram of the DEL algorithm is shown in Figure 2.

Step 1: Create an ensemble with minimum architecture comprising two NNs. Each NN consists of an input, two hidden and an output layer. The number of neurons in the input and output layers is determined by the system. Next, apply a constructive algorithm [56] based on Ash's [57] dynamic node creation method for the first (later on the odd number NNs in sequence in the ensemble) NN training. Initially, this NN starts with a small architecture containing one node in each hidden layer. For the second (later on even number NNs in sequence in the ensemble) NN training, apply Reed's pruning algorithm [58]. In the pruning phase of NN training, the number of neurons in the hidden layer is larger than necessary (i.e. it starts with a bulky architecture). Initialize the connection weights for each NN randomly within a small interval.

Step 2: Create separate training examples for each NN of the ensemble. In general, subsets of training examples for individual NNs are created by randomly picking from the main set of the training examples. In this work, training sets are

created in such a way that if one NN learns from training examples from the first to the last, other NN learns from the last to the first of the same training examples.

Step 3: Train the NNs in the ensemble partially on the examples for a fixed number of epochs specified by the user using NCL [34, 41] regardless of whether the NNs converge or not [59].

Step 4: Compute the training error E_i for the ith NN in the ensemble according to the following rule:

$$E_i = 100 \frac{O_{\max} - O_{\min}}{N.S} \sum_{n=1}^{N} \sum_{S=1}^{S} [(d(n,s) - F_i(n,s))^2 + \lambda P_i(n,s)] \quad (1)$$

where $O_{\max}$ is the maximum value and $O_{\min}$ is the minimum value of the target outputs respectively, N is the total number of examples, S is the number of output neurons, $d(n, s)$ is the desired output and $F_i(n, s)$ is the actual output of the neuron s in the nth training data. The rule in Eq. (1) is a combination of the rule proposed by Reed [58] and NCL for an NN error. The error E_i is independent of the size of the training examples and the number of output neurons.

Step 5: Compute the ensemble error E where E is the average of E_i of the base learner NNs. If E is small and acceptable, the ensemble architecture is believed to have the highest generalization ability and output the final ensemble. If E is not acceptable, then either the ensemble architecture or the individual base learner NNs undergo change.

Step 6: Check the neuron addition and/or deletion criterion of individual NNs. In this criterion, hidden neurons are added or deleted if the error of individual NNs does not change after a specified number of epochs chosen by the user (see Section 3.2). If the criterion is not met, then the individual NNs are not good enough and the ensemble undergo addition of new learner NN.

Step 7: Add and/or delete hidden neurons to/from the NNs to meet the addition and/or deletion criterion (see Section 3.2) and continue training using NCL.

Step 8: Add a new NN to the ensemble (see Section 3.3) if the previous NN addition improves the performance of the ensemble. Initialize and create different training set for the new NN as in step 2. Go to step 3 for further training of the ensemble.

The above-mentioned procedure (steps 1-8) is implemented in DEL that determines the architecture of ensemble. For example, the networks in Fig. 1 work as follows: network 1 has 2 hidden layers, uses constructive algorithm for node addition, and trains examples from first to last. On the contrary, network 2 has 2 hidden layers, uses pruning algorithm for node deletion, and trains examples from last to first. Then network 3 has a single hidden layer, uses constructive algorithm for node addition, and trains using examples from first to last. Similarly, network 4 has a single hidden layer, uses pruning algorithm for node deletion, and trains using examples from last to first and so on. The idea of varying the training examples is to enable the NNs to learn different regions of the data distribution. Major components of DEL are the addition/deletion of hidden neurons to/from leaners NNs and addition of NN to ensemble described in sections 3.2-3.4.

3.2 Nodes Addition/Deletion to/from Individual NNs

Both constructive and pruning algorithms provide some benefits as well as some drawbacks. At the training period of individual NNs, there may be some portions which may be critical or stable either for constructive or pruning algorithms. If all the NNs in the ensemble learn either only by constructive or only by pruning algorithm, then their learning will be very similar.

Even though NCL forces the NNs to learn from different regions of the data space, the learning will not be perfect if the NNs in the ensemble have the same architecture. Different architectures of the NNs in the ensemble will provide a different weight on the accuracy and diversity, which justifies the deployment of the hybrid 'constructive–pruning strategy' in DEL.

3.3 NN Addition to the Ensemble

In DEL, constructive algorithm is used to add NNs in the ensemble. New NNs are added to the ensemble if the previous addition improves the performance of the ensemble. This addition process continues until the minimum ensemble error criterion has been met.

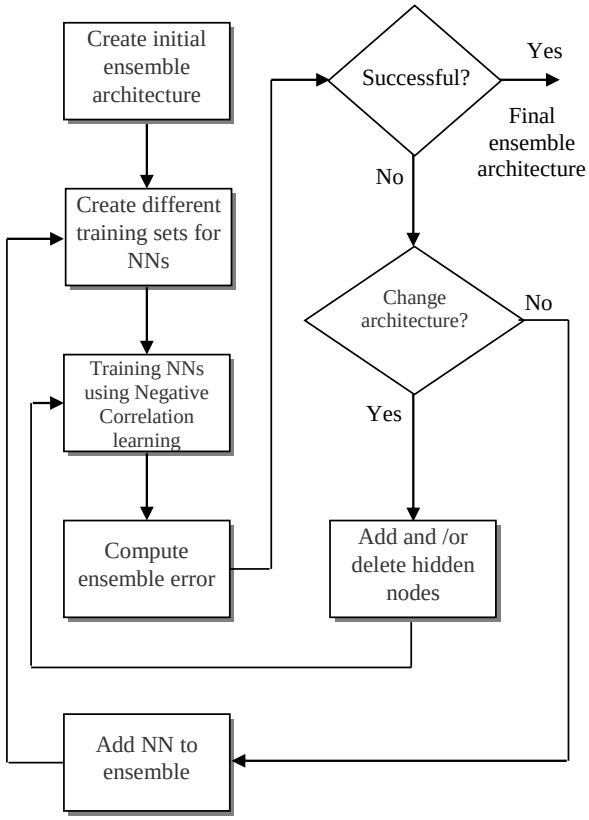


Fig. 2 Flow diagram of the DEL algorithm

3.4 Different Training Sets for Individual NNs

Varying the examples into different training sets enables efficient learning and can help the ensemble learning from the whole training examples. Training sets are varied by maintaining one important criterion i.e., training sets should have appropriate number of examples so that individual NNs obtain the necessary information for learning.

In DEL, if the first NN in the ensemble learns from odd-positioned training examples, the second one learns from even-positioned training examples, and the third one learns from other training examples in a similar fashion. In some cases, subsets of training examples are created just by partitioning or by randomly selecting. The pseudocode of DEL algorithm is shown in **Algorithm 1**.

Algorithm 1: DEL algorithm	
Step 1: Create ensemble with minimum architecture	
1.	Create an ensemble comprising 2 NNs with minimum architecture of 1 input-2 hidden-1 output layers
2.	Number of neurons in input and output layer is determined by the system
3.	Apply Ash's constructive algorithm for dynamic node creation for the first NN training
4.	Apply Reed's pruning algorithm for the second NN training
Step 2: Create training examples	
1	Create separate training examples for each NN
Step 3: Training NNs in ensemble	
1.	Train NNs partially for fixed number of epochs using NCL
Step 4: Compute training error	
1.	Compute the training error E_i for the ith NN using Eq. (1)
Step 5: Compute ensemble error	
1.	Compute the ensemble error E
2.	**If** E < *acceptable*
3.	Output final ensemble
	Endif
Step 6: Check node addition/deletion criterion	
1.	**If** (addition/deletion criterion is not met)
2.	Add NN to ensemble
3.	Go to **Step 2**
4.	**Else**
5.	Add/delete hidden nodes to NN
6.	Go to **Step 3**
7.	**Endif**

4 Experimental Analysis

The effectiveness and performance of DEL are verified on real-world benchmark problems. The datasets of the selected benchmark problems are taken from the UCI machine learning repository [60].

Different tests were carried out on DEL algorithm with varying parameter settings. For setting the correlation strength parameter λ to nonzero, the DEL performs as described in section 3. For the correlation strength parameter λ equal to zero, it is the individual NN's independent training. The independent training is performed using standard backpropagation algorithm [30].

The learning rate and correlation strength parameter λ were chosen between [0.05, 1.0] and [0.1, 1.0] respectively. The initial weights for NNs were randomly generated within the interval of [–0.5, 0.5]. The winner-takes-all method of classification is used. Both the majority voting method and the simple averaging method are used for computing the generalization ability of the DEL. Medical and non-medical datasets described in sections 3.1 and 3.2 are used in the experimentation. Table 1 shows the summary of benchmark datasets.

4.1 Medical Datasets

The medical datasets comprise four data sets from medical domain: the cancer, the diabetes, the heart disease and the thyroid dataset. These data sets have some characteristics in common:

- DEL uses the similar input attributes that an expert uses for diagnosis.
- The datasets pose a classification problem, which the DEL has to classify to a number of classes or predict a set of quantities.
- Acquisition of examples from human subjects is expensive, which results in small datasets for training.
- Very often the datasets have missing values of attributes and contain a small sample of noisy data [59], which make the classification or prediction challenging.

4.1.1 The Breast Cancer Dataset

The breast cancer dataset comprising 699 examples. 458 examples are benign and 241 examples are malignant. There are 9 attributes of a tumor collected from expensive microscopic examinations. The attributes relate to the thickness of clumps, the uniformity of cell size and shape, the amount of marginal adhesion, and the frequency of bare nuclei. The problem is to classify the tumor as either benign or malignant.

4.1.2 The diabetes dataset

The diabetes dataset comprising 768 examples of which 500 belong to class 1 and 268 belong to class 2. Datasets are collected from female patients of 21 years of age or older and of Pima Indian heritage. There are 8 attributes to be classified as either "tested positive for diabetes" or "tested not positive for diabetes".

4.1.3 The Heart Disease Dataset

The heart disease datasets comprise 920 examples. The datasets are collected from expensive medical tests on patients. There are 35 attributes to be classified as presence or absence of heart disease.

4.1.4 The Thyroid Dataset

The thyroid disease dataset comprises 7200 examples collected from patients through clinical tests. There are 21

attributes to be classified in three classes, i.e. normal, hyper-function and subnormal function. 92% of the patients are normal, which insists that the classifier accuracy must be significantly higher than 92%.

4.2 Non-medical Datasets

The non-medical datasets comprise seven data sets from different other domains: the credit card, glass, gene, horse, letter, mushroom and soybean dataset.

4.2.1 The credit card dataset

The credit card dataset comprises 690 examples collected from real credit card applications by customers with a good mix of numerical and categorical attributes. There are 51 attributes to be classified as credit card granted or not granted by the bank. 44% of the examples in the datasets are positive. The datasets also contain 5% missing values in the examples.

4.2.2 The Glass Dataset

The classification of glass dataset is used for forensic investigations. The datasets comprise 214 examples collected from chemical analysis of glass splinters. There are 70, 76, 17, 13, and 19 examples for 6 classes respectively. The datasets contain 9 attributes of continuous value to be classified into 6 classes.

4.2.3 The Gene Dataset

The gene dataset comprises 3175 examples of intron/exon boundaries of DNA sequences elements or nucleotide. A nucleotide is a four-valued nominal attribute and encoded binary i.e. {-1, 1}. There are 120 attributes to be classified into three classes: exon/intron (EI) boundary, intron/exon (IE) boundary, or none of these. EI boundary is called donor and IE boundary is called acceptor. 25% examples of the dataset are donors and 25% examples are acceptors.

4.2.4 The Horse Dataset

The horse dataset comprises 364 examples of horse colic. Colic is an abdominal pain in horses, which can result in death. There are 58 attributes collected from veterinary examination to be classified into three classes: horse will survive, die or euthanized. The dataset contains 62% examples of survival, 24% examples of death, and 14% examples of euthanized. About 30% of the values in the dataset are missing, which poses challenges in classification.

4.2.5 The Letter Recognition Dataset

Alphabet consists of 26 letters and recognition of letters is a large classification problem. It is a tough benchmark problem for the DEL algorithm. The dataset contains 20,000 examples of digitized patterns. Each example was converted into 16 numerical attributes (i.e. real valued vector), which are to be classified into 26 classes.

4.2.6 The Mushroom Dataset

The mushroom dataset comprising 8124 examples based on hypothetical observations of mushroom species described in a book. There are 125 attributes of the mushrooms collected based on the shape, color, odor, and habitat. 30% of the examples have one missing attribute value. 48% of examples are poisonous. The classifier has to categorize the mushrooms as edible or poisonous.

4.2.7 The Soybean Dataset

The soybean dataset comprises 683 examples collected from the descriptions of beans. The attributes are based on the normal size and color of leaf, the size of spots on leaf, hallow spots, normal growth of plant, the rooted roots, and the plant's life history, treatment of seeds, and the air temperature. There are 82 attributes to be classified into 19 diseases of soybeans. There are missing values of attributes in most of the examples.

Table 1: Summary of benchmark datasets

Dataset	No. examples	Attributes	Classes	Training set	Test set
Cancer	699	9	2	349	175
Diabetes	768	8	2	384	192
Heart	920	35	2	460	230
Thyroid	7200	21	3	3600	1800
Credit C	690	51	2	345	172
Glass	214	9	6	107	53
Gene	3175	120	3	1588	793
Horse	364	58	3	182	91
Letter	20000	16	26	16000	4000
Mushroom	8124	125	2	4062	2031
Soybean	683	82	19	342	171

4.3 Experimental Setup

Datasets are divided into training and testing sets and no validation set is used in the experimentation. The classification error rate is calculated according to:

$$C_i = 100 * \frac{T.T.P - C.P}{T.T.P} \quad (2)$$

where T.T.P denotes the total number of test patterns and C.P denotes the total number of correctly classified patterns. The numbers of examples in the training and test sets are chosen based on the reported works in the literature so that a comparison of results is possible. The size of the training and testing sets used in DEL are shown in Table 1.

4.4 Experimental Results

A summary of the experimental results of the DEL algorithm carried on 11 datasets described in Sections 4.1 and 4.2 is presented in Table 2. The classification error is defined as the percentage of wrong classifications in the test set defined by Eq. 2. Table 3 shows the comparison of DEL with its component individual networks in terms of classification error rates for glass dataset. It shows the error rates for glass datasets are relatively higher than the other datasets. This is due to the error rates of the individual NNs that led to higher error rate of the ensemble. Table 4(a) shows the accuracy of NNs and the common intersection and the diversity of the NNs of ensemble for the glass dataset is shown in Table 4(b). The accuracy Ω means the correct response sets of the individual NNs whereas the diversity ς means the number of different examples correctly classified by individual NNs. If S_i is the

correct response set of the *i*-th NN in the testing set, Ω_i is the size of S_i, and $\Omega_{i1,i2,\cdots,ik}$ is the size of the set $S_{i1} \cap S_{i2}, \cdots, \cap S_{ik}$, then the diversity ς of the ensemble is $\Omega_i = \Omega_{i1,i2,\cdots,ik}$. For the glass dataset, DEL produced an ensemble of four NNs (N_1, N_2, N_3, and N_4). The sizes of the correct responses are S_1=37, S_2=33, S_3=36, and S_4=36. The large variations in accuracies are caused by the incremental learning used by DEL. The ensemble started with N_1 and N_2 and trained them. When the two failed to achieve a successful ensemble, DEL added N_3 and N_4 at a final step. The size of $S_1 \cap S_2 \cap S_3 \cap S_4$ was only 29 resulting in diversity ς=8 among N_1, N_2, N_3, and N_4.

It is demonstrated here that the DEL uses a smaller number of training cycles to find the dynamic ensemble architecture with a small classification error. For example, for the glass dataset DEL with dynamic architecture produces a final ensemble with only four individual networks. Only five hidden nodes were added to individual networks training with constructive algorithm and two hidden nodes were deleted from individual networks' while training with a pruning algorithm. DEL achieved a classification error of 26.415% for this dataset. According to the comparison with other algorithms shown in Table 9, DEL achieves the lowest percentage of classification error.

To demonstrate how a hidden neuron's output changes during the entire training period, the hidden neurons' output for the cancer dataset is shown in Fig. 3. Constructive algorithm was used for training one network. The individual network started the training with one node in its first hidden layer and two nodes in its second hidden layer. During the training period four nodes were added to the first hidden layer of the network and nodes in second hidden layer were kept fixed at two nodes. The outputs stabilize and the convergence curve becomes smooth after about 100 iterations indicating that the learning may not require a very large number of iterations.

Fig. 4 shows the training error profile of ensemble for cancer, heart disease, glass, and soybean datasets. Two from medical and two from non-medical datasets are chosen. During the intermediate period of the training, individual networks were added to the ensemble by constructive strategy, hidden nodes were added as well as deleted from corresponding individual networks using a hybrid constructive−pruning strategy. For example, for cancer dataset in Fig. 4, the ensemble started with two individual networks with architecture (9–4–2–2) and (9–12–2–2). The NN architecture (9–4–2–2) has 9 inputs, two hidden layers with 4 and 2 neurons respectively and 2 outputs. The NN architecture (9–12–2–2) has 9 inputs, two hidden layers with 12 and 2 neurons respectively and 2 outputs. Constructive algorithms for individual network (9–4–2–2) and pruning algorithm for individual network (9–12–2–2) were applied during training. During the training, individual NNs with architectures (9–4–2) and (9–12–2) were added to the ensemble. Hidden nodes were added to individual networks (9–4–2–2) and (9–4–2) as constructive algorithms were used to train them. Hidden nodes were deleted from individual networks (9–12–2–2) and (9–4–2) as these two were trained using pruning algorithm. After addition of individual networks and hidden nodes by constructive strategy and deletion of hidden nodes by pruning strategy, the final ensemble with individual NN architectures of (9–8–2–2), (9–10–2–2), (9–8–2), (9–10–2) was attained.

Figs 5(a)-(b) show the training error profiles of individual NNs with constructive algorithm. For example, Fig. 5 (a) shows the curves of individual networks for which constructive algorithms were applied starting with architectures (9–4–2–2) (indicated by solid line) and (9–4–2) (indicated by dash line) for cancer dataset. At the intermediate period of training, hidden nodes were added to individual networks by the dynamic node creation (DNC) method until this node addition increased the performance of the ensemble. Finally, all these constructive networks in the ensemble completed training with (9–8–2–2) and (9–8–2) architectures. Solid lines indicate NNs with 2 hidden layers and dash lines indicate NN with single hidden layer from Fig 5 to Fig 8.

Fig. 6(a)-(b) show training error profiles of individual NNs with pruning algorithm. The pruning algorithm has an impact on error profiles which is visible from the non-smooth curves. Fig. 6(a) shows the curves of individual NNs applied to cancer dataset starting with (9–12–2–2) and (9–12–2) architectures. At the intermediate training period, hidden nodes were deleted from individual networks by the sensitivity calculation method until this node deletion increased the performance of the ensemble. Finally, all these pruning networks in the ensemble end up training with (9–10–2–2) and (9–10–2) architectures.

Fig. 7(a)-(b) show the curves of hidden nodes addition to the individual NNs training applying constructive algorithm. In this case, individual networks with small architecture started training and at the intermediate training period, hidden nodes were added to the first hidden layer of the individual network sensitivity by the dynamic node creation method. For example, Fig. 7(a) shows the curves of the hidden nodes addition to individual networks trained using constructive algorithm for cancer dataset. Here the individual network started training with (9–4–2–2) and (9–4–2) architectures and finally end up training with (9–8–2–2) and (9–8–2) architectures.

Fig. 8(a)-(b) show the curves of the hidden nodes deletion from the individual NNs training applying pruning algorithm. Individual networks with architecture larger than necessary started training in this case and at the intermediate training period, hidden nodes that deem not necessary were deleted from the first hidden layer of the individual network by the sensitivity calculation method.

Hidden node with the lowest sensitivity was deleted. If the deleted node does not posse lowest sensitivity, then the weights were restored. For example, Fig. 8(a) shows the curves of hidden nodes deletion from individual networks training by pruning algorithm for cancer dataset. Individual networks here

started training with (9–12–2–2) and (9–12–2) architectures and finally completed training with (9–10–2–2) and (9–10–2) architectures.

Table 2: Results obtained applying the proposed learning model for 11 benchmark datasets.

Dataset	Ensemble		Epoch	Error	
	Initial	Final		Training	Classification
Cancer	9-4-2-2 9-12-2-2	(9-8-2-2), (9-10-2-2), (9-8-2), (9-10-2)	113	0.01	0.571
Diabetes	8-4-4-2 8-9-4-2	(8-8-4-2), (8-7-4-2), (8-8-2), (8-7-2), (8-8-2), (8-7-2)	212	5.00	22.917
Heart Disease	35-9-2-2 35-13-2-2	(35-12-2-2), (35-11-2-2), (35-12-2), (35-11-2), (35-10-2)	85	4.00	15.652
Thyroid	21-8-3-3 21-18-2-3	(21-12-3-3), (21-16-2-3), (21-12-3), (21-16-3), (21-12-3), (21-16-3)	400	0.71	4.444
Credit Card	51-10-2-2 51-28-2-2	(51-13-2-2), (51-26-2-2), (51-13-2), (51-26-2), (51-13-2)	350	0.77	12.209
Glass	9-5-7-6 9-11-7-6	(9-10-7-6), (9-9-7-6), (9-10-6), (9-9-6)	300	3.74	26.415
Gene	120-14-5-3 120-18-6-3	(120-17-5-3), (120-16-6-3), (120-18-3), (120-16-3)	275	0.05	10.971
Horse	58-13-2-3 58-18-2-3	(58-17-2-3), (58-16-2-3), (58-17-3), (58-16-3), (58-17-3)	350	6.50	23.077
Letter Recognition	16-20 -23 -26 16-24 -23 -26	(16-23 -23 -26), (16-22 -23 -26), (16-23 -26), (16-22 -26)	215	0.004	12.2
Mushroom	125-1-2-2 125-7-2-2	(125-4-2-2), (125-6-2-2), (125-4-2)	95	0.002	0.591
Soybean	82-22-7-19 82-26-8-19	(82-25-7-19), (82-24-8-19), (82-25-19), (82-24-19), (82-25-19)	261	0.0006	4.094

Table 3: Comparison of ensemble's classification error with its component NNs for the glass database.

Ensemble / NN	Architecture	Classification error
Ensemble	(9–10–7–6), (9–9–7–6), (9–11–6), (9–9–6)	26.415
NN1	9–10–7–6	30.189
NN2	9–9–7–6	37.736
NN3	9–11–6	32.075
NN4	9–9–6	32.075

Table 4: For the test datasets of glass problem: (a) the accuracy and intersection of NNs; (b) the measure of diversity of these individual NNs. [61].

$\Omega_1 = 37$	$\Omega_2 = 33$	$\Omega_3 = 36$	$\Omega_4 = 36$
$\Omega_{12} = 33$	$\Omega_{13} = 34$	$\Omega_{14} = 34$	$\Omega_{23} = 30$
$\Omega_{24} = 30$	$\Omega_{34} = 34$	$\Omega_{123} = 30$	$\Omega_{124} = 30$
$\Omega_{134} = 33$	$\Omega_{234} = 30$	$\Omega_{1234} = 29$	

(a)Accuracy of NNs

$\varsigma_{12} = 4$	$\varsigma_{13} = 5$	$\varsigma_{14} = 5$	$\varsigma_{23} = 9$
$\varsigma_{24} = 9$	$\varsigma_{34} = 4$	$\varsigma_{123} = 7$	$\varsigma_{124} = 7$
$\varsigma_{134} = 5$	$\varsigma_{234} = 5$	$\varsigma_{1234} = 9$	

(b) Diversity of NNs in ensemble

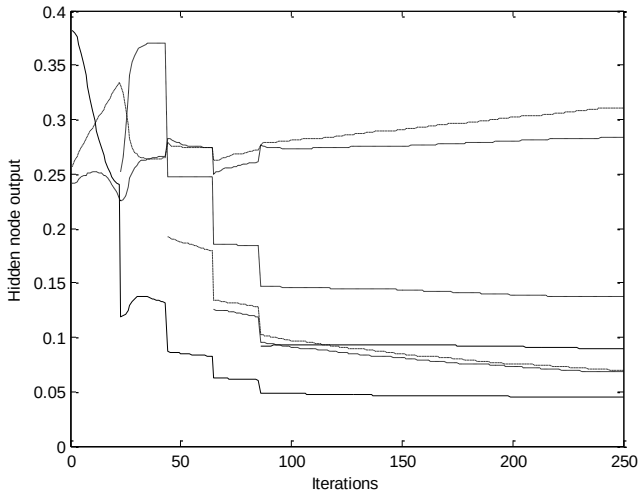


Fig 3. The hidden nodes output of a network with initial architecture (9–4–2–2) and final architecture (9–8–2–2) for the cancer datasets.

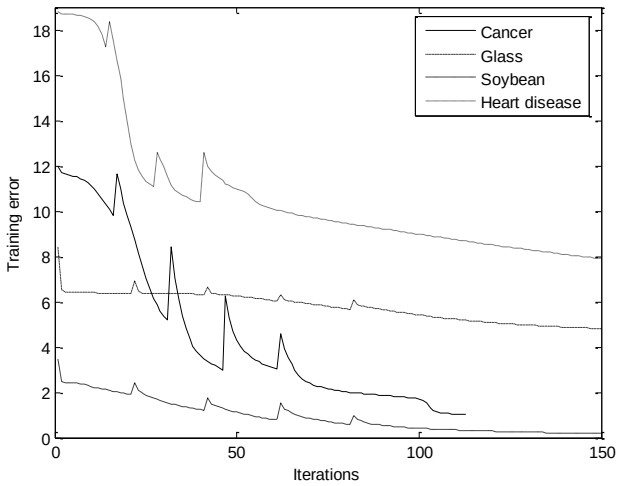
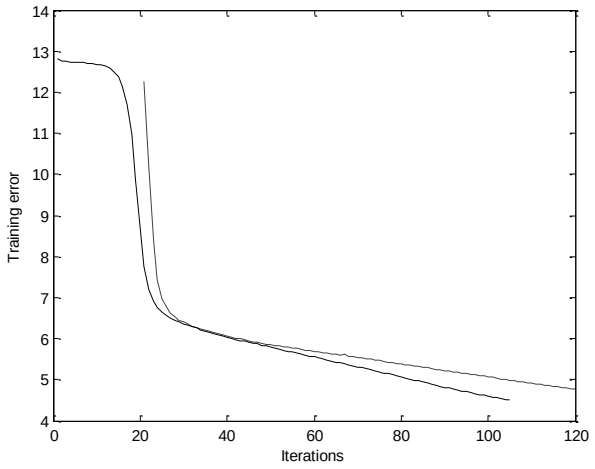
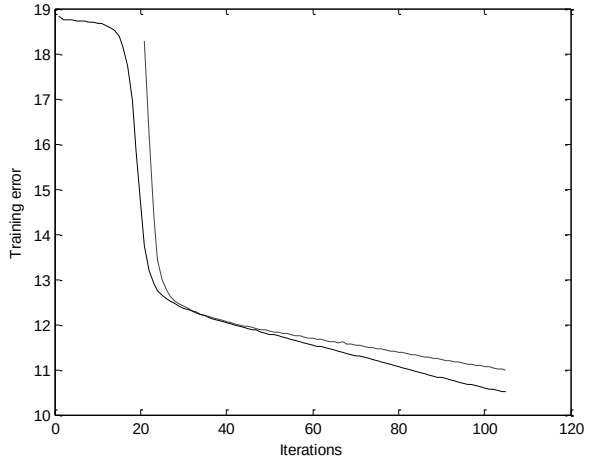


Fig 4. The error profile of the ensemble: cancer dataset; glass dataset; soybean dataset and heart disease dataset.

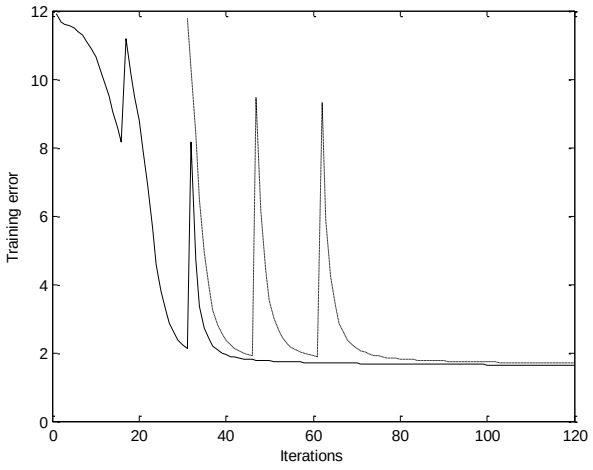


(a)

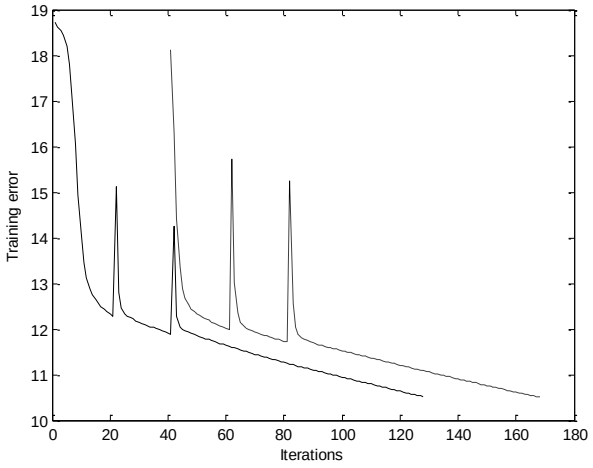


(b)

Fig. 5. The error of the individual networks for constructive algorithm: (a) for the cancer dataset; (b) for the heart disease dataset. Solid line indicates NN with 2 hidden layers and dash line indicates NN with single hidden layer (shown in Table 2).



(a)



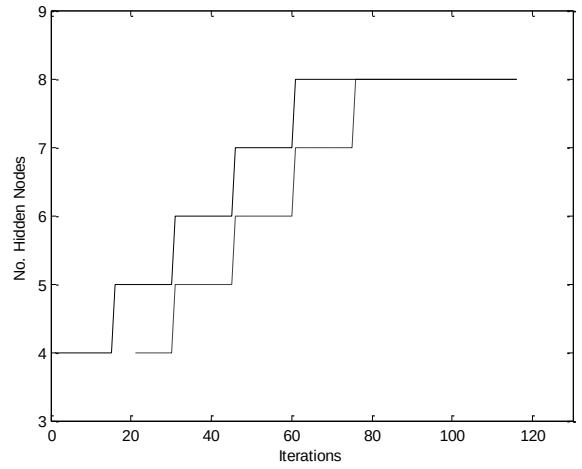
(b)

Fig. 6: The error of the individual networks for pruning algorithm: (a) for the cancer dataset; (b) for the heart disease dataset. Solid line indicates NN with 2 hidden layers and dash line indicates NN with single hidden layer (shown in Table 2).

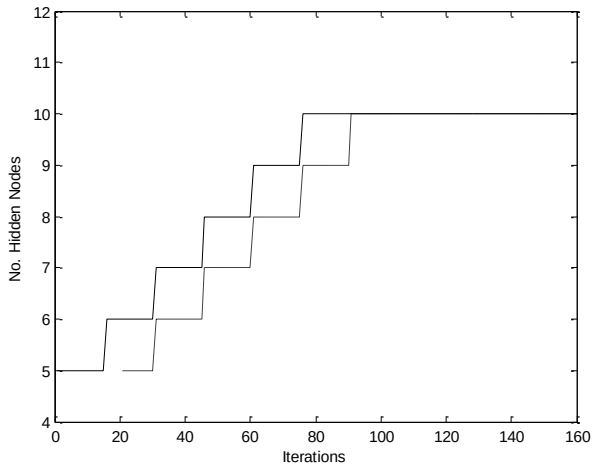
Figs 9(a)-(b) show the curves of individual networks addition during the training period. Individual networks were added to the ensemble applying constructive strategy. Initially the number of NNs in the ensemble was two. When addition increased the performance of the ensemble, the number was increased. For example, Fig. 9(a) shows the curve of individual network addition to the ensemble for cancer dataset. The curve shows that network addition to the ensemble completed training with four networks.

4.5 Correlations among the Individual NNs

In Tables 5-7, Cor_{ij} means correlation between individual networks j and i in the ensemble.

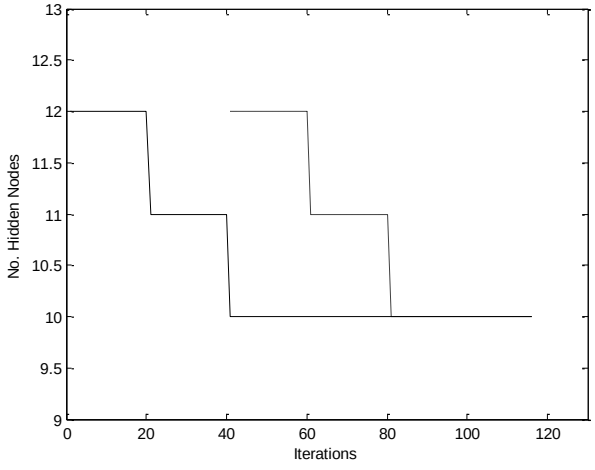


(a)

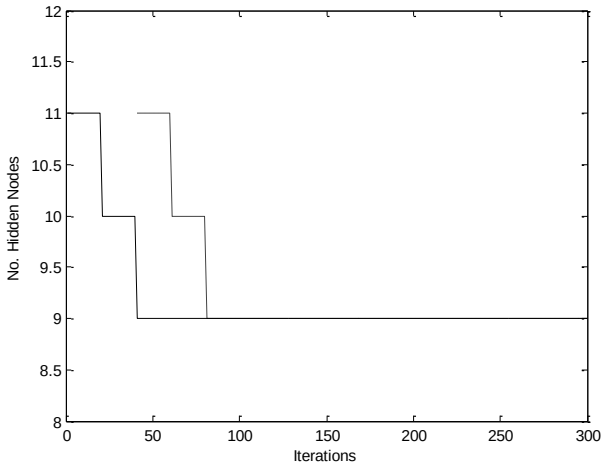


(b)

Fig. 7: Hidden nodes addition for individual networks training with constructive algorithm: (a) for the cancer dataset; (b) for the glass dataset. Solid line indicates NN with 2 hidden layers and dash line indicates NN with single hidden layer (shown in Table 2).

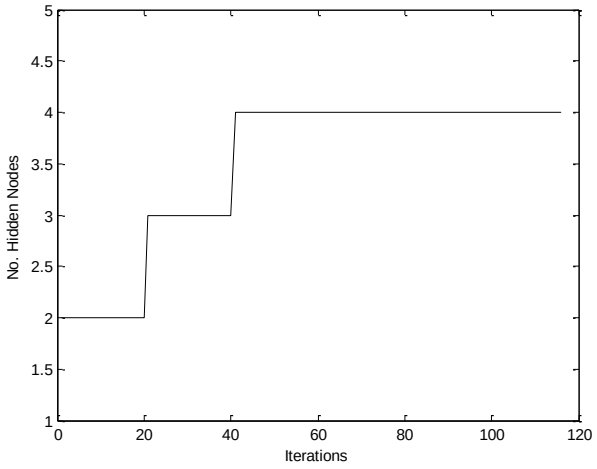


(a)

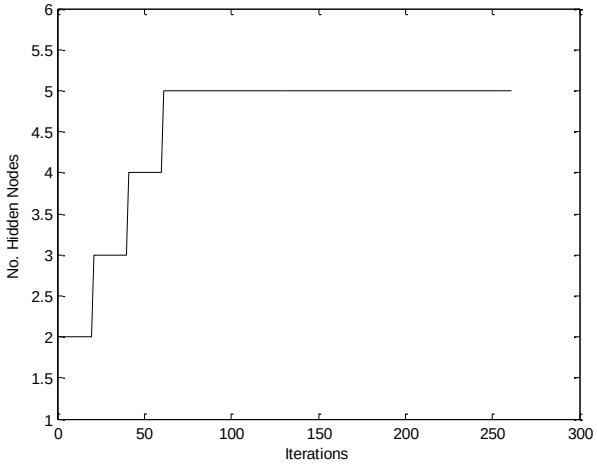


(b)

Fig. 8: Hidden nodes deletion for individual networks training with constructive algorithm: (a) for the cancer dataset; (b) for the glass dataset. Solid line indicates NN with 2 hidden layers and dash line indicates NN with single hidden layer (shown in Table 2).



(a)



(b)

Fig. 9: Individual network addition in the ensemble at the training period: (a) for the cancer dataset; (b) for the soybean dataset

Table 5 Correlation of Networks for the cancer dataset for η=0.1, λ=0.2. In this case iteration continued 116. In ensemble individual networks required is 4.

Cor_{12} = 0.018309	Cor_{13} = 0.021606	Cor_{14} = 0.019284
Cor_{23} = 0.017636	Cor_{24} = 0.015741	Cor_{34} = 0.018576

Table 6 Correlation of Networks for the cancer dataset for η = 0.1, λ = 1.0.

Cor_{12} = (–0.006913)	Cor_{13} = 0.013130	Cor_{14} = (–0.017219)
Cor_{23} = (–0.006320)	Cor_{24} = (–0.005632)	Cor_{34} = (–0.005960)

The distinguishable difference between Tables 5 and 6 is the negative correlation strength parameter λ = 0.2, so that the correlation between any two networks is positive in Table 5. But in Table 6, the negative strength correlation parameter is λ = 1.0 so that in almost all cases the value of correlation between any two networks is negative.

Table 7 Correlation of individual networks for the diabetes dataset for η=0.1, λ=0.3. In this case iteration continued 212. In the ensemble number of individual networks required to complete training was 6.

Cor_{12} = 0.018343	Cor_{13} = 0.016651	Cor_{14} = 0.015793
Cor_{15} = 0.017334	Cor_{16} = 0.015708	Cor_{23} = 0.022907
Cor_{24} = 0.021727	Cor_{25} = 0.023847	Cor_{26} = 0.021609
Cor_{34} = 0.019722	Cor_{35} = 0.021647	Cor_{36} = 0.019616
Cor_{45} = 0.020531	Cor_{46} = 0.018605	Cor_{56} = 0.020420

Table 8 Correlation of Networks for the diabetes dataset for η=0.1, λ=1.0.

Cor_{12} = 0.024974	Cor_{13} = 0.006464	Cor_{14} = 0.000086
Cor_{15} = 0.029325	Cor_{16} = (–0.015724)	Cor_{23} = 0.007256
Cor_{24} = 0.000096	Cor_{25} = 0.032919	Cor_{26} = (–0.017651)
Cor_{34} = 0.000025	Cor_{35} = 0.008520	Cor_{36} = (–0.004569)
Cor_{45} = 0.000113	Cor_{46} = (–0.000060)	Cor_{56} = (–0.020727)

The distinguishable difference between Tables 7 and 8 is the negative correlation strength parameter λ = 0.2 in Table 7 so that the correlation between any two networks is positive. But in the case of Table 8 the negative correlation strength parameter is λ = 1.0, which results in negative correlation between any two networks in many cases.

4.6 Comparison

To verify the performance of DEL algorithm, the results are compared with popular empirical study of ensemble network by Opitz and Maclin [62], a semisupervised ensemble learning algorithm i.e., SemiNCL by Chen et al. [50], and a fully semi-supervised ensemble approach to multiclass semi-supervised classification in two versions i.e. CBoost-Sup and CBoost-Semi by Soares et al. [51]. Opitz and Maclin have studied a number of networks such as a simple NN, an ensemble with varying initial weights, Bagging ensemble, and Boosting ensemble. They used resampling based on Arcing and Ada method. A confidence level of 95% can be achieved by an ensemble method than a single-component classifier [34]. Opitz and Maclin didn't apply Thyroid, Gene, Horse and Mushroom data sets in their experiments, therefore, the results are not available for comparison and marked as '-' in the table. Chen et al. [50] and Soares et al. [51] both have presented test errors by mean ± standard deviation % with 5%, 10%, and 20% of labeled data. They also didn't apply Cancer, Diabetics, Heart, Thyroid, Gene, Letter, Mushroom and Soybean data sets in their experiments, therefore, the results are not available for comparison and marked as '-' in the table.

5 Discussions

Most of the existing ensemble learning methods uses trail–and–error method to determine the number and architecture of NNs in the ensemble. Most of them use a two-stage design process for designing an ensemble. In the first stage, individual NNs are created and in the second stage these NNs are combined. In the ensemble, the number of NNs and the number of hidden neurons in the individual networks are predefined and fixed. These existing methods use two cost functions for designing the ensemble. One is for the accuracy and another is for diversity. In most of the existing ensemble methods, individual NNs are trained independently or sequentially rather than simultaneously, which lead to loss of interaction among NNs in the ensemble. In ensemble training, previously trained network is not affected.

In DEL, we presented a dynamic approach to determine the topology of an ensemble. This dynamic approach determines the number and architecture of the individual NNs in the ensemble. Such a dynamic approach is entirely new to designing NN ensemble. In DEL, better diversity among the NNs has also been maintained. In DEL, constructive strategy has been used for automatic determination of the number of NNs and constructive–pruning strategy has been used for automatic determination of the architecture of NNs in the ensemble. The hybrid constructive–pruning strategy has provided better diversity for the whole ensemble (Table 4b). NCL has been used for diversity of NNs in the ensemble encouraging individual networks to learn different regions and aspects of data space. But, if different NNs attempt to learn different regions with inaccurate architecture, learning will also be insufficient or improper by this attempt. Different training sets for individual networks are created which also help maintaining diversity among the NNs in the ensemble (Table 4b). In some cases, different training sets were created by variation of training examples and in other cases by random choice of the training examples. As NN is a kind of unstable learning, random redistribution of the training samples has provided better learning in the case of an unstable learning. Both three and four layered individual networks were used to design the ensemble.

DEL uses a minimum number of parameters i.e. only one correlation strength parameter λ. An incremental training approach has been used in DEL because even after choosing the appropriate architecture of the ensemble, DEL has to be trained several times for finding the correct value of the learning rate parameter and the correlation strength parameter λ. DEL uses only one cost function (the ensemble error *E*)

during training, not two cost functions, one for accuracy and the one for diversity used in some other ensemble method in the literature. DEL uses a one-stage design process. Individual networks are created and combined at the same design stage. The advantage of DEL is that it does not need any separate gating block. DEL uses the parameter λ as a balancing mechanism for bias–variance–covariance tradeoff. Since DEL generates uncorrelated networks in the ensemble, individual networks in this ensemble are well-diversified.

DEL algorithm uses both simple averaging and majority voting combination methods. For some problems simple averaging method performed better and for some other problems majority voting method performed better. Though problem dependent, the choice of the correlation strength parameter λ is important in DEL. To delete hidden nodes from individual networks in an ensemble, initially a network larger than necessary is considered. But, assessing the initial size of the NN is challenging, which is still an unknown parameter in DEL algorithm.

6 Conclusions

DEL is a new algorithm for designing and training NN ensembles. Traditional way of ensemble designing is still a manual trial-and-error process, whereas DEL is an automatic design approach. The number of NNs and their architectures are determined by DEL algorithm.

The major benefits of the proposed DEL algorithm compared to existing ensemble algorithms are 1) automatic creation of ensemble architectures; 2) preservation of accuracy and diversity among the NNs in the ensemble; and 3) minimum number of parameters specified by designer.

DEL emphasizes both accuracy and diversity of NNs in ensemble to improve the performance. Constructive and constructive-pruning strategies are used in DEL to achieve the accuracy of individual NNs. To maintain diversity of NNs, NCL and different training sets are used. The performance of DEL algorithm was confirmed on benchmark problems. In almost all cases, DEL outperformed the others. However, the performance of DEL needs to be evaluated further on some regression and time series problems.

Compliance with ethical standards

Conflict of interest: The authors declare that they have no conflict of interest.

Table 9 Test set error rates for the datasets: comparison of DEL with results of (i) a single NN classifier (Stan); (ii) an ensemble created by varying random initial weights (Simp); (iii) an ensemble created by Bagging method; (iv) an ensemble created by Arcing method, (v) an ensemble created by Ada method [63], (vi) a semisupervised ensemble learning algorithm i.e., SemiNCL [50], and (vii) a cluster-based boosting (CBoost) ensemble in two versions i.e. CBoost-Sup and CBoost-Semi [51].

Dataset	DEL	Stan	Simp	Bag	Arc	Ada	(with % of labeled data)								
							SemiNCL [45]			CBoost-Sup [46]			CBoost-Semi [46]		
							5%	10%	20%	5%	10%	20%	5%	10%	20%
Cancer	0.571	3.4	3.5	3.4	3.8	4.0	–	–	–	–	–	–	–	–	–
Diabetes	22.917	23.9	23.0	22.8	24.4	23.3	–	–	–	–	–	–	–	–	–
Heart d	15.562	18.6	17.4	17.0	20.7	21.1	–	–	–	–	–	–	–	–	–
Thyroid	4.444	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Credit C	12.209	14.8	13.7	13.8	15.8	15.7	15.87 ± 3.73	14.68 ± 2.84	14.10 ± 2.58	21.47 ± 3.47	19.96 ± 2.67	18.35 ± 3.03	18.67 ± 1.26	16.18 ± 2.73	15.82 ± 3.44
Glass	26.415	38.6	35.2	33.1	32.0	31.1	–	–	–	38.96 ± 12.01	19.01 ± 7.31	18.84 ± 6.88	36.31 ± 10.36	19.54 ± 4.57	20.32 ± 7.66
Gene	10.971	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Horse	23.077	–	–	–	–	–	36.59 ± 6.30	33.82 ± 4.97	32.80 ± 5.22	25.87 ± 6.27	23.45 ± 5.23	29.11 ± 4.99	26.23 ± 6.17	22.52 ± 5.19	29.12 ± 5.04
Letter	12.2	18.0	12.8	10.5	5.7	6.3	–	–	–	–	–	–	–	–	–
Mushroom	0.591	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Soybean	4.094	9.2	6.7	6.9	6.7	6.3	–	–	–	–	–	–	–	–	–

References

[1] Y. Li, B. Wei, Y. Liub, L. Yao, H. Chena, J. Yu, and W. Zhu, "Incorporating knowledge into neural network for text representation," *Expert Systems with Applications*, Vol. 96, pp. 103–114, 2018.

[2] S. Hooshdar and H. Adeli, H., "Toward Intelligent Variable Message Signs in Freeway Work Zones: A Neural Network Model", Journal of Transportation Engineering, ASCE, Vol. 130(1), pp. 83-93, 2004.

[3] D. L. Yu and J. B. Gomm, "Enhanced Neural Network Modelling for a Real Multi-variable Chemical Process," *Neural Computing & Applications*, Vol. 10(4), pp 289–299, 2002.

[4] C. Cengiz and E. Köse, "Modelling of color perception of different eye colors using artificial neural networks," *Neural Computing & Applications*, Vol. 23(7-8), pp. 2323-2332, 2013.

[5] A. Panakkat and H. Adeli, "Neural Network Models for Earthquake Magnitude Prediction using Multiple Seismicity Indicators," *International Journal of Neural Systems*, Vol. 17(1), pp. 13-33, 2007.

[6] Z. Ahmad and J. Zhang, "Bayesian selective combination of multiple neural networks for improving long-range predictions in nonlinear process modelling," *Neural Computing & Applications*, Vol. 14(1), pp. 78-87, 2005.

[7] A.R. Tashakori and H. Adeli, "Optimum Design of Cold-Formed Steel Space Structures Using Neural Dynamic Model," *Journal of Constructional Steel Research*, Vol. 58(12), pp. 1545-1566, 2002.

[8] A. Gotsopoulos, H. Saarimaki, E. Glerean, I. P. Jaaskelainen, M. Sams, L. Nummenmaa, J. Lampinen "Reproducibility of importance extraction methods in neural network-based fMRI classification," *NeuroImage*, Vol. 181, pp. 44–54, 2018.

[9] J. J. M. Sá Junior, A. R. Backes, O. M. Bruno, "Randomized neural network-based descriptors for shape classification," *Neurocomputing*, Vol. 312, pp. 201–209, 2018.

[10] J. A. R. Vargas, W. Pedrycz, E. M. Hemerly, "Improved learning algorithm for two-layer neural networks for identification of nonlinear systems," Neurocomputing, Vol. 329, pp. 86–96, 2019.

[11] F. Fourati, "Multiple neural control and stabilization," *Neural Computing & Applications*, Vol. 29(12), pp. 1435-1442, 2018.

[12] F. Masulli and G. Valentini, "Effectiveness of error correcting output coding methods in ensemble and monolithic learning machines," *Formal Pattern Analysis & Applications*, Vol. 6(4), pp. 285–300, 2004.

[13] R. Srinivasana, C. Wang, W. K. Ho, K. W. Lim, "Neural network systems for multi-dimensional temporal pattern classification," *Computers and Chemical Engineering*, Vol. 29, pp. 965–981, 2005.

[14] T. A. Choudhury, C. C. Berndt, Z. Man, "Modular implementation of artificial neural network in predicting in-flight particle characteristics of an atmospheric plasma spray process," *Engineering Applications of Artificial Intelligence*, Vol. 45, pp. 57–70, 2015.

[15] N. E. Sharkey, and A. J. Sharkey, "An analysis of catastrophic interference," *Connection Science* 7, pp.301–329, 1995.

[16] A. Gepperth, and C. Karaoguz, "A bio-inspired incremental learning architecture for applied perceptual problems," *Cognitive Computation* 8(5), pp. 924–934, 2016.

[17] M. McCloskey, and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psych. of Learning & Motivation* 24:109–165, 1989.

[18] R. M. French, "Catastrophic Forgetting in Connectionist Networks," *Trends in Cognitive Sciences*, Vol. 3(4), pp. 128–135, 1999.

[19] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, "Measuring Catastrophic Forgetting in Neural Networks," The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), February 2–7, 2018, New Orleans Riverside, New Orleans, Louisiana, USA, pp. 3390-3398, 2018.

[20] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science* 7(2), pp. 123–146, 1995.

[21] B. Ren, H. Wang, J. Li, and H. Gao, H. "Life-long learning based on dynamic combination model," *Applied Soft Computing* 56, pp. 398–404, 2017.

[22] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proc. of the National Academy of Sciences*, Vol. 114 (13), pp. 3521-3526, 2017.

[23] R. Coop, A. Mishtal, and I. Arel, "Ensemble learning in fixed expansion layer networks for mitigating catastrophic forgetting," *IEEE Transaction on Neural Networks and Learning Systems*, Vol. 24(10), pp.1623–1634, 2013.

[24] A. J. C. Sharkey, "On combining artificial neural nets," *Connection Science*, Special Issue on Combining Artificial Neural: Ensemble Approaches, Vol. 8 (3&4): 299–313, 1996.

[25] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 12(10), pp. 993–1000, 1990.

[26] P. M. Granitto, P. F. Verdes, and H. A. Ceccatto, "Neural Network Ensembles: Evaluation of Aggregation Algorithms," *Artificial Intelligence*, Vol. 163, pp. 139-162, 2005.

[27] R. A. Jacobs, "Bias / Variance Analyses of Mixtures−of−experts Architectures," *Neural Computation*, Vol. 9, pp. 369–383, 1997.

[28] T. Hancock and H. Mamitsuka, "Boosted Network Classifiers for Local Feature Selection," *IEEE Transaction on Neural Networks and Learning Systems*, Vol. 23(11), pp. 1767–1778, 2012.

[29] M. M. Islam, X. Yao, S. M. S. Nirjon, M.A. Islam, and K. Murase, "Bagging and Boosting Negatively Correlated Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, Vol. 38(3), pp. 771–784, 2008.

[30] Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. "Learning representations by back-propagating errors", *Nature*, 323(6088), pp. 533–536, 1986.

[31] N. H. Siddique and M. O. Tokhi, "Training Neural Networks: Backpropagation vs Genetic Algorithms," Proceedings of the International Joint Conference on Neural Networks (IJCNN'01), 15-19 July 2001, Washington, DC, USA, 2001, pp. 2673-2678, 2001.

[32] Y. Liu, X. Yao and T. Higuchi, "Evolutionary Ensembles with Negative Correlation Learning," *IEEE Transactions on Evolutionary Computation*, vol. 4, pp. 380–387, 2000.

[33] H. Chen and X. Yao, "Multiobjective neural network ensembles based on regularized negative correlation learning," *IEEE Transaction on Knowledge and Data Engineering*, vol. 22(12), pp. 1738–1751, 2010.

[34] Y. Liu and X. Yao, "Ensemble Learning via Negative Correlation," *Neural Networks*," Vol. 12(10), pp. 1399–1404, 1999.

[35] G. Giacinto and F. Roli, "Design of effective neural network ensembles for image classification purposes," *Image Vis. Comput.*, vol. 19(9–10), pp. 699–707, 2001.

[36] S. Hashem, "Optimal linear combinations of neural networks," *Neural Networks*, vol. 10(4), pp. 599–614, 1997.

[37] E. K. Tang, P. N. Suganthan, and X. Yao, "An analysis of diversity measures," *Machine Learning*, vol. 65(1), pp. 247–271, 2006.

[38] G. Brown, J. L. Wyatt, and P. Tino, "Managing diversity in regression ensembles," *Journal of Machine Learning Research*, vol. 6, pp. 1621–1650, 2005.

[39] M. L. Zhang and Z. H. Zhou, "Exploiting unlabeled data to enhance ensemble diversity," *Data Mining and Knowledge Discovery*, vol. 26(1), pp. 98–129, 2013.
[40] B. Rosen, "Ensemble learning using de-correlated neural networks," *Connection Science*, Special Issue on Combining Artificial Neural: Ensemble Approaches, Vol. 8 (3-4), pp. 373–384, 1996.
[41] Y. Liu and X. Yao, "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 29, pp. 716–725, 1999.
[42] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, 6(2), pp. 182–197, 2002.
[43] R. Mousavi and M. Eftekhari, "A new ensemble learning methodology based on hybridization of classifier ensemble selection approaches", *Applied Soft Computing*, Vol. 37, pp. 652–666, 2015.
[44] M. M. Islam, X. Yao and K. Murase, "A constructive algorithm for training cooperative neural network ensembles," *IEEE Transactions on Neural Networks*, 14 (4), pp. 820–834, 2003.
[45] M. Shahjahan and K. Murase, "A Pruning Algorithm for Training Cooperative Neural Network Ensembles" *IEICE Transaction on Information & Systems*, Vol.E89–D, No.3, pp. 1257–1269, 2006.
[46] X. C. Yin, K. Huang, and H. W. Hao, "DE^2: Dynamic ensemble of ensembles for learning non-stationary data", *Neurocomputing*, Vol. 165, pp. 14–22, 2015.
[47] R. M. O. Cruz, R. Sabourin, G. D. C. Cavalcanti, T. I. Ren, "META-DES: A dynamic ensemble selection framework using meta-learning", *Pattern Recognition*, Vol. 48, pp. 1925–1935, 2015.
[48] H. Chen and X. Yao, "Regularized negative correlation learning for neural network ensembles," *IEEE Transaction on Neural Networks*, vol. 20(12), pp. 1962–1979, 2009.
[49] K. Chen and S. Wang, "Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 33(1), pp. 129–143, 2011.
[50] H. Chen, B. Jiang, and X. Yao "Semisupervised Negative Correlation Learning," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29(11), pp. 5366-5379, 2018.
[51] R. G. Soares, H. Chen, and X. Yao, "A cluster-based semi-supervised ensemble for multiclass classification," *IEEE Transactions on Emerging Topics in Computational Intelligence*, Vol. 1(6), pp.408-420, 2017.
[52] M. H. Rafiei, and H. Adeli, "A New Neural Dynamic Classification Algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 28:12, 2017.
[53] A. S. Britto, R. Sabourin, L. E.S. Oliveira, "Dynamic selection of classifiers—A comprehensive review," *Pattern Recognition*, Vol. 47(11), pp. 3665-3680, 2014.
[54] Y. Ren, L. Zhang and P. N. Suganthan, "Ensemble Classification and Regression-Recent Developments, Applications and Future Directions," *IEEE Computational Intelligence Magazine*, vol. 11(1), pp. 41-53, 2016.
[55] R.M.O. Cruz, R. Sabourin, G.D.C. Cavalcanti, "Dynamic classifier selection: recent advances and perspectives," *Information Fusion*, 41, pp. 195-216, 2018.
[56] T.Y. Kwok, and D.Y. Yeung, "Constructive algorithms for structure learning in feed forward neural networks for regression problems," *IEEE Transactions on Neural Networks*, Vol. 8, pp. 630–645, 1997.
[57] T. Ash, "Dynamic node creation in backpropagation networks", *Connection Science*, Vol. 1(4), pp. 365-75, 1989.
[58] R. Reed, "Pruning algorithms–a survey," *IEEE Transactions on Neural Networks*, Vol 4(5), pp 740–747, 1993.
[59] L. Prechelt, "Automatic early stopping using cross validation: Quantifying the criteria," *Neural Networks*, vol. 11(4), pp. 761–767, 1998.
[60] M. Lichman, "UCI Machine Learning Repository http://archive.ics.uci.edu/ml", School of Information and Computer Science, University of California, Irvine, CA, 2013.
[61] L. I. Kuncheva and C. J. Whitaker, "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy," *Machine Learning*, 51, pp. 181–207, 2003.
[62] D. Opitz and R. Maclin, "Popular Ensemble Methods: An Empirical Study," *Journal of Artificial Intelligence Research*, Vol. 11, pp. 169–198, 1999.
[63] A. J. C. Sharkey and N. E. Sharkey, "Combining diverse neural nets," Connection Science, *The Knowledge Engineering Review*, Vol. 12(3), pp. 231–247, 1997.