Ensembles of Artificial Example based Neural Networks

M. A. H. Akh and P. C. Shill
Dept. of Computer Science and Engineering, Khulna University of Engineering & Technology (KUET), Bangladesh
Email: (akhand, pintu)@cse.kuet.ac.bd

K. Murase
Dept. of Human and Artificial Intelligence Systems, University of Fukui, Japan
Email: murse@synapse.his.u-fukui.ac.jp

***Abstract*—Ensembles with several neural networks are widely used to improve the generalization performance over a single network. A proper diversity among component networks is considered as an important parameter for ensemble construction so that the failure of one may be compensated by others. Data sampling, i.e., different training sets for different networks, is the most investigated technique for diversity than other approaches. This paper presents a data sampling based neural network ensemble method where the individual networks are trained on the union of original training set and a set of some artificially generated examples. Generated examples are different for different networks and are the element to produce diversity among the networks. After each network is trained, the method checks whether the trained network is suitable to ensemble or not, and absorbs the network based on the ensemble performance with it. The effectiveness of the method is evaluated on a suite of 20 benchmark classification problems. The experimental results show that the performance of this ensemble method is better or competitive with respect to the existing popular methods.**

***Index Terms*—Artificial training example, diversity, generalization, neural network ensemble, bagging, AdaBoost, negative correlation learning.**

I. INTRODUCTION

The goal of ensemble construction with several classifiers is to achieve better generalization ability over a single classifier. The inspiration for building an ensemble is the same as for establishing a committee of people: each member of the committee should be as competent as possible, but the members should be complementary to one another. If the members are not complementary (i.e., always agree), the committee is unnecessary as any member performs like the committee. If the members are complementary, then when one or a few members make an error, there is a high probability that the remaining members can correct the error. Thus, to achieve better generalization by an ensemble, proper diversity among component classifiers is an important factor to compensate failure one by others.

Artificial Neural Networks (NNs) and decision trees (DTs) are the two popular tools among classifiers. Starting in the early 90s a number of ways have been investigated, for creating diverse NNs or DTs for ensembles, and among them data sampling, i.e., different data/examples for different classifiers, is found to be the most effective approach [1-2]. This is because it is the training data that determines the function of a classifier. The most popular algorithms that explicitly or implicitly use different training data for different classifiers in an ensemble are the bagging [3], AdaBoost [4], and Negative Correlation Learning (NCL) [6-7].

Both bagging and AdaBoost algorithms explicitly manipulate the original training data to create a separate training set for each classifier in an ensemble [10]. Bagging creates the separate training set by forming bootstrap replicas of the original training data, while AdaBoost creates it by the same method but with adaptation [4, 8-9]. Both methods use different training sets to build different DTs or to train different NNs for constructing ensemble of DTs or NNs.

Although bagging and AdaBoost are the general techniques that can be used with any type of classifier (i.e., DTs or NNs), NCL is applicable only to construct ensembles of NNs. Like bagging and AdaBoost, NCL [6-7] does not create separate training sets explicitly for NNs in an ensemble. The NCL rather uses a correlation penalty term in the error function of the NNs by which networks can maintain training time interaction. The training method used in NCL is simultaneous where all NNs in the ensemble are trained on the same original training data at the same time. Since NCL provides training time interaction among NNs, it can produce diverse NNs for the ensemble.

A new scheme, called DECORATE algorithm [5], recently has been proposed for DT that sequentially produces a relatively large number of DTs to select several DTs for constructing an ensemble. It first creates a DT based on the original training set and selects it for ensemble. The algorithm uses different training sets for other DTs; a particular training set is the union of the original training data and randomly created artificial patterns. The method considers a DT when ensemble's performance does not degrade with this DT. The aim of using artificial training data is to create diverse DTs for the ensemble. To generate an artificial example, it randomly generates a set of input feature and then defines the class label of the pattern. It first checks the class probability of the features set passing through the

doi:10.4304/jcp.5.12.1819-1827

existing ensemble and then inversely defines class probability to use in training. Due to inverse relabeling, the method is called Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples (DECORATE).

Based on artificial training examples, the DECORATE algorithm is found to be an effective ensemble method when compared to other DT based ensemble methods, such as bagging and AdaBoost. There is no Neural Network Ensemble (NNE) method based on such artificial training examples according to the best of our knowledge. As a classifier, NNs have been extensively applied across numerous domains and many cases NNs show better classification ability when compared to DTs [17]. In some situations a single NN is a better classifier than an ensemble of DTs [5]. Therefore, investigation of artificial training examples for NNE construction might be interesting and is a current demand. But the idea of DECORATE cannot be applied directly on an ensemble of NNs because NN differs from DT on the basis of various points of view. NNs work on preprocessed training data that contain all information in numeric form. The focus of this study is to evaluate a neural network ensemble method based on artificial training examples.

The rest of the paper is organized as follows. Section II gives a description of the artificial training example creation for neural networks and training of the networks for the ensemble. Section III presents an experimental study. Section IV concludes the paper with a brief summary.

II. ARTIFICIAL TRAINING EXAMPLE BASED NEURAL NETWORK ENSEMBLE

Proper diversity among component NNs is an important parameter for ensemble construction so that the failure of one may be compensated by others. Artificial training examples, that are different for different NNs, may motivate different NNs towards different functional spaces and may promote diversity among the NNs. But the random artificial examples might not be always positive to improve overall NNE performance though they promote diversity. For this reason, a selection scheme requires to build NNE with appropriate NNs.

Fig. 1 and Fig. 2 show the functional block diagram and the pseudo code, respectively for the Artificial Training Example based Ensemble (ATEE) method. First NN is trained with the original training set (T) and by default the NN is considered for final NNE as shown in Fig.1. For each of other NNs, the method generates a unique set of training examples, i.e., the artificial training set. The training set on which a NN is trained therefore is the union of the original training set and the generated examples. After training a NN is finished, it checks the NNE's performance including the NN and discards the NN if performance degrades. Therefore, to include more NNs in the final ensemble, it follows a trial-and-test with a relatively large number of trained NNs. Fig. 2 shows the ATEE algorithm to construct an NNE of maximum M networks from N_{max} trained NNs. To observe NNE's performance, the algorithm measures ensemble error, i.e., the number of examples on which NNE gives wrong decision. When ensemble error decreases for a network,

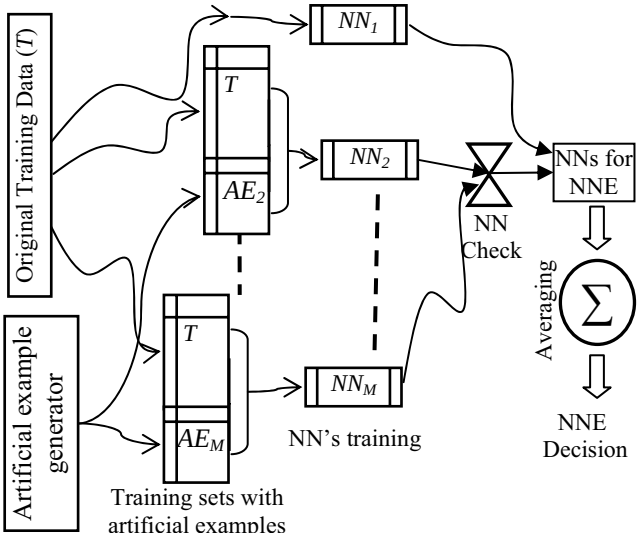


Figure 1. Functional block diagram of ATEE

1. Let N_{max} be the number of NNs to be trained for an NNE and M the desired number of NNs.

   Take original training set $T= \{(x(1), d(1)),\ldots\ldots, (x(N), d(N))\}$ with class label $d(n) \in K = \{1,2,\ldots,k\}$

   Take R_{Size} -factor that determines number artificial examples for a network

   i=1 (for NN in NNE) and *trials* =1 (for trial NN)

   Train network NN_i with T (i.e., first NN is trained with original training data)

   Initialize ensemble, $NNE = \{NN_i\}$

   Compute ensemble error, $\varepsilon = \left(\sum_{(x(n),d(n)) \in T: NNE(x(n)) \neq d(n)} 1 \right) / N$

2. *while trials* < N_{max} and $i < M$

   {

   a. Generate $R_{Size} \times |T|$ artificial training examples, AE_i

   b. Label examples in R with probability of class labels inversely proportional to predictions of *NNE*

   c. Prepare training set T_i, $T_i = T \cup AE_i$ and train network NN_i with T_i

   d. $NNE = NNE \cup \{NN_i\}$

   e. Compute ε' based on Step 1

   f. *if* $\varepsilon' \leq \varepsilon$ *then* $i = i + 1$ *and* $\varepsilon = \varepsilon'$, *otherwise* $NNE = NNE - \{NN_i\}$

   g. *trials = trials +1*

   }

3. Ensemble decision is made in simple average way

Figure 2. Pseudo code of ATEE

the NN is considered for final ensemble. Ensemble decision is made averaging individual networks' decisions for simplicity.

The artificially generated training examples in the training of different NNs are the main attraction of ATEE. The diversity among the NNs solely depends on the artificial examples; the size of artificial examples set (*AE*) maintains the level of diversity. The size of *AE* is defined by the factor R_{Size} i.e., the size ratio of *AE* to the original training set (*T*).

To create an artificial training example, ATEE first generates the Gaussian number for each input attribute with the mean and the standard deviation of the attribute values of *T*. To define the class label of the pattern, it first checks the class probability of it by passing through the existing NNE and then inversely defines class probability to use in training. Thus, the desired output value for an artificial example might be any continuous value in between 0.0 - 1.0 such as 0.1, 0.2, 0.6, etc. On the other hand, the desired output value of an original training example is a discrete value 0 or 1. For several output nodes for a multiclass problem, one output value sets as 1 instance to the desired class while other output values remain 0.

The main problem of this algorithm seen from the above description is that it trains relatively large number of NNs for an ensemble. Also, it uses relatively larger training set (i.e., original training set plus generated examples) to train a NN. Both the factors increase training time. However, it might not be guaranteed to build NNE with desired number of NNs due to selection scheme of NNs.

III. EXPERIMENTAL STUDIES

This section first gives description of benchmark problems on which ATEE was applied to evaluate its performance. It is also compared with the popular methods bagging, AdaBoost and NCL on the basis of achieved performance. Finally, the effect of different parameters on the performance of ATEE is investigated.

A. Benchmark Data and General Experimental Methodology

Twenty real-world classification problems from the University of California, Irvine (UCI) machine learning benchmark repository were used for experiments. The characteristics of the problems are summarized in Table I, and show a considerable difference in the number of patterns, input features and classes. These problems, therefore, provide a suitable experimental test bed. The UCI web site (http://www.ics.uci.edu/~mlearn/) contains detailed descriptions of these problems.

The UCI benchmark repository contains only raw data that require preprocessing to use in any NN. We followed the benchmark methodology suggested in [13] for preprocessing the data sets of different problems. In this work, the continuous input feature values were rescaled between 0 and 1 with a linear function. For discrete features, the number of inputs was selected as the number of distinct values in the data set in general. As an example, the ACC problem has 6 continuous and 9 discrete input features and after manipulation it was fed to a neural network as 51 inputs.

Basic three layered feed-forward architectures [18] were used for NNs in ensembles. The number of nodes in the input layer was equal to the number of processed inputs of a given problem; the number of nodes in the output layer was set as the number classes of the corresponding problem. We chose the number of hidden nodes based on the number of input and output nodes [9], one hidden node for every output node plus one hidden node for every 10 input nodes. The minimum number of hidden nodes was set at 5. The common logistic sigmoid function, $\phi(y) = 1/(1+\exp(-y))$, was used as the activation function for nodes in the hidden and output layers.

TABLE I.

CHARACTERISTICS OF BENCHMARK DATASETS

Dataset	Exa-mple	Class	Input Features		NN Architecture	
			Cont.	Disc.	Input	Hidd. Node
Australian Credit Card (ACC)	690	2	6	9	51	10
Breast Cancer Wisconsin (BCW)	699	2	9	-	9	5
Car (CAR)	1728	4	-	6	21	10
Diabetes (DBT)	768	2	8	-	8	5
Heart Disease Cleveland (HDC)	303	2	6	7	35	5
Hepatitis (HPT)	155	2	6	13	19	5
House Vote (HSV)	435	2	-	16	16	5
Hypothyroid (HTR)	7200	3	6	15	21	5
Ionosphere (INS)	351	2	34	-	34	10
Iris Plants (IRP)	150	3	4	-	4	5
King+Rook vs. King+Pawn (KRP)	3196	2	-	36	74	10
Lymphography (LMP)	148	4	-	18	18	10
Lungcancer (LNG)	32	3	-	56	56	10
Page Blocks (PGB)	5473	5	10	-	10	5
Promoters (PRM)	106	2	-	57	228	10
Soybean(SBN)	683	19	-	35	82	25
Segmentation(SGM)	2310	7	19	-	19	10
Sonar (SNR)	208	2	60	-	60	10
Waveform (WVF)	5000	3	21	-	21	10
Zoo(ZOO)	101	7	15	1	16	10

The learning rate of back-propagation (BP) is a parameter on which the performance of any BP based algorithm may vary [11]. The value of a learning rate was selected 0.1 - 0.15 for the experiment after few trial runs. The number of NNs for ensembles was set to 20 for all problems. It has been reported that an ensemble consisting of 20 NNs is sufficient to reduce the testing set error [9]. The initial weights of NNs were set randomly

between - 0.5 and 0.5.

B. Experimental Results

This section presents experimental results of ATEE along with bagging, AdaBoost and NCL. The results were compared based on the testing error rate (TER). The TER refers to the rate of wrong classifications produced by the trained ensemble on the testing set; the lower TER value represents the better generalization ability. In this paper, we also considered a method without data sampling called simple NNE (sNNE) for comparative analysis. In case of sNNE, only initial random weight sets are different for different NNs and all the NNs are trained with the same original training set. A previous study revealed that in some cases, sNNE also gives results comparable to data sampling based methods [9]. By default; a data sampling based method also randomly initializes weights of a NN which makes for variation in weight sets in different NNs. In general, sNNE is considered as a base line; therefore, ATEE is compared with sNNE as well as with the popular existing methods.

The total number of NNs trained for ensemble construction was 20, except ATEE. To be comparable to other methods, the maximum number of NNs per NNE in ATEE was defined as 20 and maximum trial NNs as 30. The number of epochs for training NNs was set 50. It was observed that a minimum TER was found with a particular value of R_{Size} for ATEE. This value was found not to be the same for all problems. We, therefore, selected the value of R_{Size} for a particular problem after some trial runs, and finally this value was selected in the range of 0.5 to 1.

The number of training epochs per NN is a component determining the better TER. However, better TER is shown in different epochs for different problems as well as different methods. Therefore, an ensemble method was tested for a problem using three different training epochs per NN; 50, 75 and 100. For a specific problem the best

Table II (a).
TERS COMPARISON OVER FIVE STANDARD 10-FOLD CROSS-VALIDATION RUNS. A PLUS (OR MINUS) SIGN INDICATES THAT TER IS FOUND SIGNIFICANTLY BETTER (OR WORSE) THAN SNNE BY *T*-TEST. A SINGLE AND DOUBLE PLUS/MINUS IS FOR 95% AND 99% CONFIDENCE INTERVAL, RESPECTIVELY.

Problem	sNNE	Bagging	AdaBoost	NCL	ATEE(NN/NNE)
ACC	0.1519	0.1409 +	0.1556	0.1432	**0.1356(5.62)** ++
BCW	0.0333	0.0331	0.0333	**0.0278** ++	0.0319(5.38)
CAR	0.1128	0.0995 ++	**0.0799** ++	0.1036 ++	0.1203(2.00)
DBT	0.2379	0.2321	**0.2305**	0.2308 ++	0.2342(1.14)
HDC	0.1613	0.1607	0.1613	0.16	**0.1507(6.92)** +
HPT	0.1587	0.1533	0.18 -	0.1547	**0.152(1.02)**
HSV	0.0489	**0.0372** ++	0.0437 ++	0.0396 +	0.0442(5.68)
HTR	0.0531	0.0518 ++	**0.0263** ++	0.0522 ++	0.0528(1.02)
INS	0.1343	0.1297	0.1034 ++	0.1366	**0.0606**(12.9) ++
IRP	**0.0267**	0.0293	0.028	**0.0267**	**0.0267**(1.00)
KRP	0.076	**0.0214** ++	0.0894	0.0765	0.0557(9.14) ++
LMP	0.1571	0.1543	0.1829 -	0.1571	**0.1357(4.96)** ++
LNG	0.5	**0.4067** +	0.4533	0.4933	0.4667(16.5
PGB	0.0582	0.0563	**0.0499** ++	0.0589	0.0577(1.02)
PRM	0.068	0.068	0.072	0.068	**0.066**(20.00)
SBN	0.0541	**0.0517**	0.0535	0.0544	0.0559(10.5) -
SGM	0.07	0.0648	**0.0432** ++	0.0677 ++	0.0761(3.14) -
SNR	0.195	0.194	0.181	0.195	**0.166(**7.58) ++
WVF	0.1327	**0.1297** +	0.132	0.1312	0.1308(4.18)
ZOO	0.11	0.1	**0.038** ++	0.11	0.098(16.52) +
Average	**0.127**	**0.1157**	**0.1169**	**0.1244**	**0.1159**

Table II (b).
A PAIR WISE WIN/DRAW/LOSS SUMMARY AND METHOD BEST/WORST SUMMARY.

NNE Method	Pair Wise Win/Draw/Loss Summary				
	sNNE	Bagging	AdaBoost	NCL	ATEE
sNNE	-	18/1/1	12/2/6	11/5/4	16/1/3
Bagging		-	9/0/11	4/1/15	10/0/10
AdaBoost			-	10/0/10	11/0/9
NCL				-	12/1/7
Best/Worst	1/9	5/1	6/7	2/4	8/3

TER among the three individual runs was used to compare to TERs of other ensemble methods. The built-in parameters of ATEE and NCL are also found to have an important effect on TER. To minimize the effect of these parameters on a problem, ATEE was tested with R_{Size} values of 0.5, 0.75 and 1; NCL was tested with λ values of 0.25, 0.5 and 0.75. Thus, for a particular problem, the best result for ATEE and NCL among the nine individual runs was used to compare with the other methods. On the other hand, the results presented for sNNE, bagging and AdaBoost is the best among three independent runs i.e., for training epochs per NN equal 50, 75 and 100.

It is known that a different arrangement of patterns in training and testing sets may produce different performance, even when the numbers of patterns are kept the same for the training and testing sets. Therefore, 10-fold cross validation was used for creating the training and testing sets with different arrangement of patterns. In 10-fold cross validation, the patterns of the original data set were divided into 10 equal or nearly equal sets. By rotation, one set was reserved for testing while the remaining nine sets were used for training.

Table II (a) shows the average TERs over five standard 10-fold cross-validation (i.e., 5*10=50) runs. The best TER among the five methods is shown in bold-face type for each problem. Considering sNNE as a base line, pair two tailed *t*-test was conducted between sNNE and other NNEs individually to determine the significance in the variation of results for each problem. If TER of an NNE method is found significantly better than sNNE by *t*-test, it is marked with a plus (+) sign with TER. On the other hand, a minus (-) sign indicates TER of an NNE method is significantly worse than sNNE for a particular problem. Table II (b) shows a summary of the results presented in the Table II (a).

According to Table II (a), no data sampling based ensemble method outperformed others including sNNE for all the problems. However, on the basis of average TER for all the problems, any data sampling based method (i.e., bagging/AdaBoost/NCL/ATEE) is better than sNNE. sNNE achieved the best TERs jointly with others for only one problem out of 20 problems. On the other hand, bagging, AdaBoost, NCL and ATEE have the best TERs for 5, 6, 2 and 8 problems, respectively. In this point of view ATEE is the best among the methods. Also on the basis of average TER over 20 problems, ATEE is better than sNNE, NCL and AdaBoost and competitive to bagging. Average TER achieved by ATEE is 0.1159 whereas the TERs for sNNE, bagging, AdaBoost and NCL are 0.127, 0.1158, 0.1169 and 1244, respectively as shown in Table II (a).

Bootstrap based data sampling (i.e., bagging or AdaBoost) creates different training sets for different NNs from the original available training examples. The problems where bagging or AdaBoost performed better generally contain sufficient training examples such as, CAR, DBT, and SGM (Table I). With respect to sNNE, bagging and AdaBoost outperformed in 18 and 12 problems, respectively; the results are found significant by *t*-test for seven for each of bagging and AdaBoost. Besides, explicit creation of training sets for different NNs, NCL trains all the NNs in an NNE with the same original training examples. However, NCL encourages different NNs to produce different hypothesis adding a penalty term into the error function of a NN. On the basis of Table II (b), NCL appears better than sNNE for 11 problems (in which in six cases the TERs were significantly better).

ATEE creates some artificial training examples that are unique for individual NNs. Due to example creation; it might not suffer from the limitation of examples like bootstrap sampling based methods. From Table II (a) it can be seen that ATEE is shown the best suited NNE method for problems with a limited number of examples, such as HPT, INS, LMP, and PRM problems. LMP has 148 training examples and the TER achieved by ATEE is 0.1357; however, TERs for sNNE, bagging and AdaBoost are 0.1571, 0.1543, and 0.1829, respectively. With respect to sNNE, out of 20 problems ATEE is better in 16 problems, and among these, the TERs for seven problems are significantly better.

ATEE builds adaptive ensembles with different sizes for different problems due to the NN selection scheme. To select a second NN for the final ensemble, ATEE trains NNs with an artificial training set having the opposite class definition of the first NN. Therefore the second NN might show higher diversity than the first one and combining it with the first one might degrade NNE performance, rather than improving or maintaining previous performance (i.e., the condition to add a NN in the NNE). This might be the reason ATEE failed to build an ensemble and return a single NN for several problems. When it built ensembles, ATEE selected few NNs from a large number of trained NNs. Out of 20 problems in only 10 cases did it build NNEs with more than five NNs and only once (i.e., for PRM) meet the desired NNE size.

The selection scheme of ATEE guaranteed to improve the ensemble performance from a single NN. Therefore, the problems in which ATEE built NNE with a relatively larger number of NNs showed a better TER. For example, for the INS problem ATEE built an ensemble with 12.9 NNs on average and achieved the best TER among the five methods. The achieved TER of ATEE for this problem was 0.0606; however, TERs were 0.1343, 0.1297 and 0.1034 for sNNE, bagging and AdaBoost, respectively. ATEE has the best TER for eight problems out of 20 problems. At a glance, ATEE is an effective NNE method for the small sized problem and also competitive with other methods for large problems.

C. Experimental Analysis

The main features of ATEE are the use of artificial training example set to train predefined number (i.e., N_{Max}) of NNs and selection of NNs for final NNE. The parameter R_{Size} maintains the size artificial example set. Therefore, this section presents empirical studies regarding these matters.

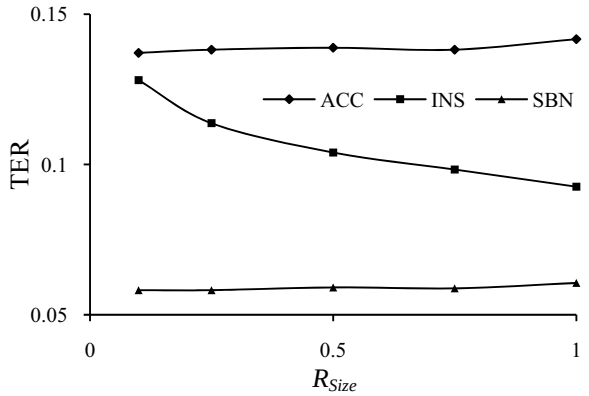
C1. Effect of R_{Size}

The Performance of a method may vary depending on its parameter values, and better TER may be achieved at a certain range or point. For better TER, it is necessary to maintain proper diversity among component NNs, as we said earlier. This subsection investigates the effect of R_{Size} on TER and diversity of ATEE.

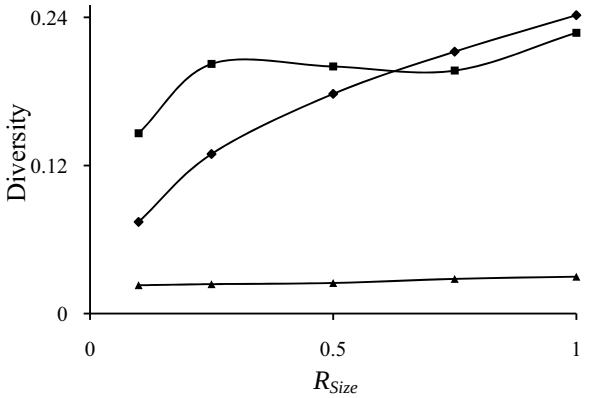
Simply, diversity means disagreement among component NNs. Many diversity measuring techniques have been proposed. Among them the pairwise plain disagreement technique [12] is the most popular and is the one employed in this study. For two NNs *i* and *j*, the plain disagreement is equal to the proportion of the patterns on which the NNs make different class predictions. It can be expressed as

$$div_{i,j} = \frac{1}{N}\sum_{n=1}^{N} Diff\ (C_i(n), C_j(n)), \quad (1)$$

where N is the number of patterns in the testing set and $C_i(n)$ is the class assigned by the NN i for the pattern n. $Diff(a, b) = 0$, if $a = b$; otherwise $Diff(a, b) = 1$. The diversity for a NNs pair varies from 0 to 1. This measure is equal to 0, when the NNs return the same classes for each pattern, and it is equal to 1 when the predictions are always different. The NNE diversity is the average of all the NN pairs' diversity.



(a). R_{Size} vs TER



(b) R_{Size} vs diversity

Figure 3. Effect of R_{Size} on ATEE

Fig. 3 presents experimental results over five 10-fold cross validation runs for three problems ACC, INS, and SBN. R_{Size} was varied from 0.1 to 1.0. For NNE construction, 30 NNs (i.e., N_{max}) were trained to select a maximum of 20 NNs. In an NNE each NN was trained for 50 epochs. According to Fig. 3, TER and diversity varied for the variation of R_{Size}. The effects of R_{Size} are more visible for ACC and INS (those are two-classed problems) than for SBN that have 19 classes. For the ACC and INS problems diversity increased greatly with respect to R_{Size} but TER did not improve in the same manner. The lowest TER for a particular problem is found at a particular value of R_{Size} and the value is different for different problems. For the INS problem the lowest TER occur at R_{Size}=1.0. On the other hand, the minimum TER for SBN problem occurs when R_{Size} is equal to 0.25. For this reason, R_{Size} is required to tune for better TER.

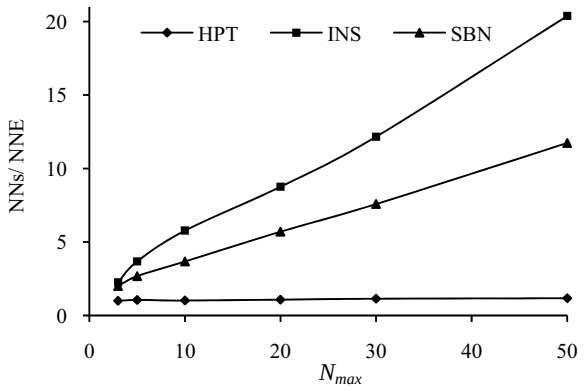
C2. Effect of Number of Trial NNs (N_{Max})

This section presents an empirical study due to variation of the parameter N_{Max} (i.e., number of trial NNs) for ATEE. N_{Max} was varied from 3 to 50 NNs; and TER, diversity and final NNE size were measured for the three selected problems. The value of R_{Size} was fixed at 0.5 for the experiment and each NN was trained with 50 epochs.

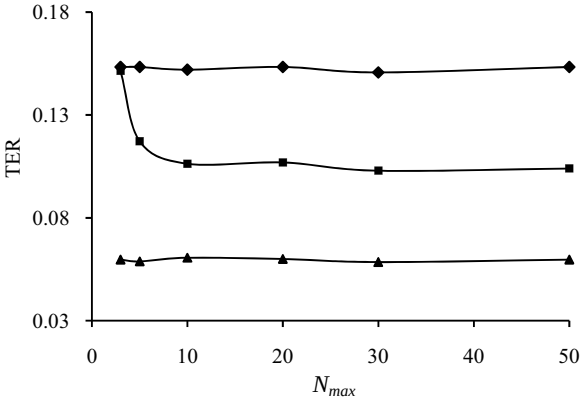
Fig. 4 presents experimental results over five 10-fold cross validation runs for HPT, INS, and SBN problems for with and without NN selection. Final NNE considered all the trained NNs i.e., N_{Max} for without NN selection case. NNs selection for final NNE was found problem dependent when N_{Max} value was increased from 3 to 50 NNs. For HPT problem NNE was almost invariant and for INS and SBN problems number of selected NNs increased due to increase N_{Max} value.

There is a relationship between the achieved TER and the NNE size as it is seen from Fig. 4. TER for HPT is found almost invariant as ATEE failed to select NNs for this problem though N_{Max} value increased up to 50. On the hand, for INS problem the selected NNs were 2.26 and 20.38 for N_{Max} 3 and 50, respectively. As a result TER improved from 0.1514 to 0.104 for this problem. Similar observation is also available for SBN problem.

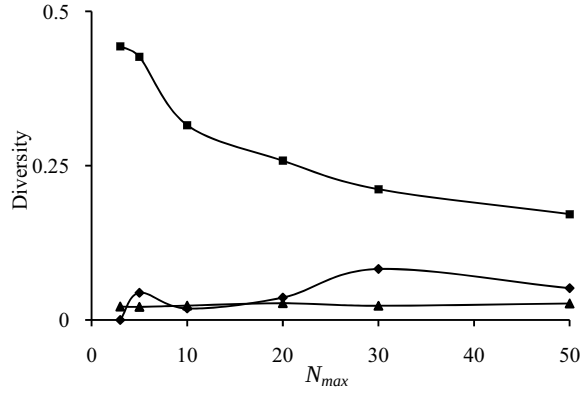
Another observation from the Fig. 4 is that diversity achievement due to N_{Max} variation is problem dependent. Diversity was almost invariant for HPT and SBN problems. For the INS problem, on the other hand, diversity was very high for very low value of N_{Max} and reduced up to a certain level when N_{max} value increased. Due to diversity variation, TER variation for INS was more visible than HPT and SBN problems. For INS problem TER and diversity were 0.1514 and 0.4432, respectively for N_{Max}=3. On the other hand, these values were 0.104 and 0.1714 for the same problem at N_{Max}=50. This result replies that the better TER does not appear at very high diversity point but requires a certain level of diversity. On the other hand the very low diversity gave larger TERs for HPT problem. Therefore, there is trade-off between TER and diversify as it is also studied in the previous studies [14-15].



(a). N_{Max} vs NNs/NNE.



(b). N_{Max} vs TER.



(c). N_{Max} vs Diversity.

Figure 4. Effect of trial NNs (i.e., N_{Max}) variation on ATEE.

C3. Effect of NN Selection

This section investigates the importance of NN selection for final NNE in ATEE. NN selection can have both a positive and negative effect. Selection may reply concise NNE with appropriate NNs. On the other hand, selection demands a relatively larger numbers of NNs that follows ATEE. To observe the effect of NN selection, the following section compare TER between standard ATEE (i.e., with NN selection) and ATEE without NN selection, a deviation of the standard method.

Table III presents a comparison between standard ATEE and ATEE without NN selection on the basis of achieved TER and networks in the final NNE. The experiment was done on 10 selected problems. The value of R_{Size} was fixed at 0.5 for the experiment. In both cases 30 NNs were trained and each NN was trained with 50 epochs. NNE considered all the 30 NNs in without selection case, and the number of NNs selected for a particular problem in the standard ATEE method is shown in the table.

Table III reveals that NN selection is important for ATEE. Among the 10 problems, seven cases standard

TABLE III.

TERS AND NNE SIZE COMPARISON BETWEEN ATEE WITH AND WITHOUT NN SELECTION OVER FIVE STANDARD 10-FOLD CROSS-VALIDATION RUNS.

Problem	Standard ATEE (With NN Selection)		ATEE Without NN Selection	
	TER	NNs/NNE	TER	NNs/NNE
ACC	0.1388	6.46	0.1385	30
BCW	0.0319	5.60	0.031	
DBT	0.2408	1.12	0.2826	
HDC	0.156	6.30	0.1593	
HPT	0.152	1.06	0.1973	
HSV	0.0498	2.88	0.0572	
INS	0.104	12.68	0.1029	
SBN	0.0591	7.58	0.0632	
SNR	0.23	3.98	0.236	
ZOO	0.116	18.64	0.124	
Average	**0.1278**	**6.63**	**0.1392**	**30**

ATEE with NN selection is shown to achieve better (i.e., lower) TER with concise NNE. In some cases without selection (i.e., NNE with 30 NNs) is shown very bad results. As an example, for DBT problem TER with 30 NNs was 0.2826. For the same problem TER for standard ATEE was 0.2408.

D. Incremental R_{Size} for adaptive NN selection

It was observed in the previous section that the value of R_{Size} is an important factor to achieve better performance for a particular problem. The parameter R_{Size} provides an opportunity to tune its value for better results. But tuning its value is time consuming and sometimes difficult for large problems. Therefore, investigation for an adaptive scheme for better performance is interesting.

For ATEE, the resulting TER is a sort of a similarity to the final NNE size, i.e., selected networks for final NNE. When it was able to select several networks the final NNE gave a better TER. For network addition (i.e., a NN selection for NNE), much trial was required for the second network due to the high diversity with the first one; the problem was common for large R_{Size} values. To overcome this problem it is possible to increase the value of R_{Size} from a minimum level.

To understand the effectiveness of such an adaptation scheme we performed two different experiments on some selected problems: one with fixed R_{Size} and another with incremental R_{Size}. In the case of incremental value selection, initially R_{Size} was set at 0.05 and when a network is added in the final NNE the value increased by 0.05. In this way, to select a third network the R_{Size} value was was taken as 0.1, and the highest value was taken as 0.75.

Table IV compares achieved TER and number of NNs in the final NNE for fixed R_{Size}= 0.5 and incremental R_{Size}. In both cases each network was trained with 50 epochs and maximum trial networks were 30. It can be seen from Table IV that due to incremental R_{Size} the average number of networks in the NNE increased but not very much. For the selected eight problems the average number of networks was 2.91 and 4.77 for fixed R_{Size}=0.5 and incremental case, respectively. The effect of NNE size enlargement is shown to improve NNE performance, and average TER was reduced from 0.1257 to 0.1226. Although TER was not reduced for all cases, this comparison indicates that adapting R_{Size} may produce better TER. Therefore, more effective adaptive schemes may give better results for selecting appropriate networks in the final NNE in ATEE.

TABLE IV.

TERS AND NNE SIZE COMPARISON BETWEEN ATEE WITH FIXED AND INCREMENTAL R_{SIZE} OVER FIVE STANDARD 10-FOLD CROSS-VALIDATION RUNS.

Problem	Fixed R_{Size} (0.5)		Incremental R_{Size}	
	TER	NNs/NNE	TER	NNs/NNE
BCW	0.0319	5.6	0.0304	7.12
DBT	0.2408	1.12	0.2429	2.92
HPT	0.152	1.06	0.1413	5.24
HSV	0.0498	2.88	0.0447	4.74
PGB	0.0773	1.20	0.073	2.10
SGM	0.0941	3.60	0.0904	6.82
SNR	0.23	3.98	0.226	4.86
WVF	0.1299	3.84	0.132	4.32
Average	**0.1257**	**2.91**	**0.1226**	**4.77**

IV. CONCLUSIONS

In this study an artificial training example based NNE (called ATEE) is investigated. ATEE is tested on a large number of benchmark problems and is compared to the popular NNE methods such as bagging, AdaBoost and NCL. ATEE constructed problem dependent adaptive NNE that was smaller than the other methods. ATEE is showed the best TERs with respect to bagging and Adaboost for small sized problems; bagging and boosting suffer shortage of examples for small sized problems. Also for any other problem, ATEE was found competitive.

ATEE trains predefined number (i.e., N_{Max}) of NNs one after another where each NN is trained with the union of available training examples and the artificially generated examples. After training each NN, ATEE checks whether the trained NNs is suitable to NNE or not. The parameter R_{Size} maintains the size of artificial example set and motivates ATEE to achieve the better performance. Also, the variation of N_{Max} reveals that better TER demands certain level of diversity. Network selection scheme is also found important and ATEE performed better when relatively large number of NNs selected for an NNE. An adaptive scheme using incremental R_{Size} is also investigated in this study regarding to select more NNs and found to improve performance slightly.

There are several future directions following this study. Artificial training example generation in the other ways might be interesting. A different adaptive scheme of R_{Size} may produce better result. In addition, other selection techniques based on pruning or thinning [16] can also be of great interest.

ACKNOWLEDGEMENTS

The authors would like to thank Mr. Rushdi Shams, Dept. of CSE, KUET and Shaniur Alam Sarkar, Dept. of Humanities (English), KUET for proofreading this paper. The authors are also grateful to the anonymous reviewers for their constructive comments.

REFERENCES

[1] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity Creation Methods: A Survey and Categorization," *Information Fusion*, vol. 6, pp. 99-111, 2005.

[2] T. G. Dietterich, *Ensemble Learning, The Handbook of Brain Theory and Neural Networks*, 2002, pp. 405-408.

[3] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 23-140, 1996.

[4] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," *in Proc. of the 13th Int. Conf. on Machine Learning*, 1996, pp. 148-156.

[5] P. Melville and R. J. Mooney, "Creating diversity in ensembles using artificial data," *Information Fusion*, vol. 6, pp. 99-111, 2005.

[6] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural Networks*, vol. 12, pp. 1399-1404, 1999.

[7] Y. Liu and X. Yao, "Simultaneous Training of Negatively Correlated Neural Networks in an Ensemble," *IEEE Trans. Systems*, Man, and Cybernetics – Part B, vol. 29, pp. 716-725, 1999.

[8] E. Bauter and R. Kohavi, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," *Machine Learning*, vol. 36, pp. 105-142, 1999.

[9] D. W. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169-198, 1999.

[10] R. Polikar, "Bootstrap Inspired Techniques in Computational Intelligence," *IEEE Signal Processing Magazine*, vol. 24, pp. 59-72, 2007.

[11] S. Haykin, *Neural Networks – A Comprehensive Foundation*, 2nd ed., Prentice Hall, 1999.

[12] A. Tsymbal, M. Pechenizkiy and P. Cunningham, "Diversity in search strategies for ensemble feature selection," *Information Fusion*, vol. 6, pp. 83-98, 2005.
[13] L. Prechelt, Proben1- *A Set of Benchmarks and Benching Rules for Neural Network Training Algorithms*, Tech. Rep. 21/94, Fakultat fur Informatik, University of Karlsruhe, Germany, 1994.
[14] M. A. H. Akhand, Md. Monirul Islam, K. Murase "Progressive Interactive Training: A Sequential Neural Network Ensemble Learning Method", *Neurocomputing*, vol. 73, pp. 260-273, 2009.
[15] M. M. Islam, X. Yao, and K. Murase, "A Constructive Algorithm for Training Cooperative Neural Network Ensembles", *IEEE Transactions on Neural Networks,* vol. 14, pp. 820-834, 2003.
[16] R. E. Banfield, L. O. Hall, K. W. Bowyer and W. P. Kegelmeyer, "Ensemble diversity measures and their application to thinning", *Information Fusion*, vol. 6, pp. 49-62, 2005.
[17] Fariba Shadabi, Dharmendra Sharma and Robert Cox, "Learning from Ensembles: Using Artificial Neural Network Ensemble for Medical Outcomes Prediction", *Innovations in Information Technology*, pp. 1-5, 2006.
[18] Mackin, K.J. Yamaguchi,T. Nunohiro, E. Jong Geol Park Hara, K. Matsushita, K.Ohshiro, M.; Yamasaki, K. Systems, Man and Cybernetics, "Ensemble of artificial neural network based land cover classifiers using satellite data", *ISIC, IEEE International Conference*,vol.7, Issue.10, 2007, pp. 1653-1657.

M. A. H. Akhand received the B.Sc. degree in Electrical and Electronic Engineering from Khulna University of Engineering and Technology (KUET), Bangladesh in 1999, the M.E. degree in Human and Artificial Intelligence Systems in 2006, and the Doctoral degree in System Design Engineering in 2009 from University of Fukui, Japan. He joined as a lecturer at the Department of Computer Science and Engineering at KUET in 2001, and is now an Assistant Professor. He is a member of the Bangladesh Computer Society (BCS) and Institution of Engineers, Bangladesh (IEB). His research interest includes artificial neural networks, evolutionary computation and other bio-inspired computing techniques.

P. C. Shill received the B.Sc. degree in Computer Science Engineering (CSE) from Khulna University of Engineering and Technology (KUET), Bangladesh in 2003 and M.Sc. degree in Computer Engineering from Politecnico di Milano, Italy in 2008. He joined as a lecturer at the Department of CSE, KUET in 2004 and currently he is serving as an Assistant Professor. His research interest includes evolutionary computation, fuzzy logic and artificial neural networks.

K. Murase is a Professor at the Department of Human and Artificial Intelligence Systems, Graduate School of Engineering, University of Fukui, Fukui, Japan, since 1999. He received ME in Electrical Engineering from Nagoya University in 1978, PhD in Biomedical Engineering from Iowa State University in 1983. He Joined as a Research Associate at Department of Information Science of Toyohashi University of Technology in 1984, as an Associate Professor at the Department of Information Science of Fukui University in 1988, and became the professor in 1992. He is a member of The Institute of Electronics, Information and Communication Engineers (IEICE), The Japanese Society for Medical and Biological Engineering (JSMBE), The Japan Neuroscience Society (JSN), The International Neural Network Society (INNS), and The Society for Neuroscience (SFN). He serves as a Board of Directors in Japan Neural Network Society (JNNS), a Councilor of Physiological Society of Japan (PSJ) and a Councilor of Japanese Association for the Study of Pain (JASP).