Neural Network Ensemble Construction Fusing Multiple Popular Methods

M. A. H. Akhand and K. Murase

Abstract—**The goal of an ensemble construction with multiple neural networks (NNs) is to achieve better generalization ability in comparison with a single network. Proper diversity among component networks is considered to be an important aspect of neural network ensemble (NNE) construction, in that a failure of one network may be compensated for by other networks. Conventional methods first produce a multiplicity of diverse networks and then combine their decisions for the overall decision of the ensemble. In general, they do not check whether each network is essential for the ensemble, or whether a subset of the networks might perform better. If a particular network performs poorly, it may not be possible for the remaining networks to compensate effectively, resulting in a poorly performing NNE. Although a number of techniques have been investigated in the last few years, no single technique has been discovered that performs well on all possible problems. While a certain solution may outperform alternatives for a subset of problems, the method may perform worse on other problems. This paper discusses Ensembles Fusing Multiple Popular Methods (EMPM), which demonstrate a relatively good performance on all possible problems. EMPM first produces a network pool using several popular methods, and then it selects networks for an ensemble using a proposed forward selection scheme. EMPM has shown better performance with a compact ensemble over conventional methods when tested on a suite of 25 benchmark problems. Experimental analyses have revealed that a heterogeneous network pool of EMPM (using multiple popular methods) is more effective than a pool of any individual conventional method for NN selection. Forward selection has also been found to be a more effective method with a variety of benefits when compared to conventional genetic algorithm-based selection.**

Index Terms — **diversity, generalization ability, neural network ensemble, network selection.**

I. Introduction

The goal of an ensemble construction with multiple neural networks (NNs) is to achieve better generalization ability in comparison with a single neural network. The inspiration for building an ensemble is the same as that 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, that is, if they always agree, then the committee is unnecessary, since any one member could perform the task of the entire committee. On the other hand, if the members are complementary, there is a high probability that the remaining members may be able to correct a potential error caused by one or a few members. Thus, for neural network ensemble (NNE) construction, proper diversity among component networks is considered to be an important aspect, in that a failure of one network may be compensated for by other networks [1], [2].

There are a variety of ways of constructing an NNE composed of diverse NNs, such as by using different training sets, architectures and learning methods. Since the functionality of an NN depends on its training data, data sampling (i.e., NNs training using different data) has been shown to be more effective for achieving diversity than other approaches [3]-[5]. A number of techniques employing data sampling have been investigated for creating diverse NNs. However, no single technique has been found to be ideal for all possible problems [3], [4]. While a certain approach may outperform alternatives for a subset of problems, the method may perform worse on other problems. One possible reason for this phenomenon might be the wide variety of attributes of real world problems (i.e., complexity, number of examples, number of input attributes, output classes, etc.) [5]; any technique that adheres to a single method fails to maintain sufficient diversity for all possible problems. Nevertheless, bagging, AdaBoost, and negative correlation learning are popular ensemble methods based on data sampling which show better overall performance in comparison with other methods such as DECORATE, class label switching, random subspace methods and smearing [4].

Since no single NNE method has been found to be effective for all possible problems, in this study we aim to construct an NNE that can either outperform or perform comparably well relative to the existing optimal solution for any given problem. Here, we consider a method of fusing of multiple popular NNE methods to achieve this goal. The proposed method works in two different phases. First, it creates a pool of networks using several popular methods (called "fusion"), and second, it selects a subset of NNs from the pool for constructing an NNE. We believe that an appropriate selection process is likely to result in a good selection of NNs, and that an NNE combining this selection may outperform or perform comparably relative to the best alternative used.

The remainder of this paper is organized as follows. Section II outlines existing conventional ensemble methods. Section III details a proposed ensemble construction fusing multiple popular NNE methods. Section IV presents experimental results of the new method, and compares performance with conventional methods. Finally, Section V

Manuscript received April 14, 2010.

M.A.H. Akhand is with the Dept. of Computer Science and Engineering (CSE), Khulna University of Engineering & Technology (KUET), Khulna-9203, Bangladesh (phone: +880-41-774318; fax: +880-41-774403; e-mail: akhand@cse.kuet.ac.bd; website: www.kuet.ac.bd/cse/akhand).

K. Murase is with the Dept. of Human and Artificial Intelligence Systems, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan (e-mail: murase@synapse.his.fukui-u.ac.jp).

concludes the paper with a brief summary and some future possible directions for research following this study.

II. Conventional Popular NNE Construction Methods

The step of creating diverse networks is commonly followed by ensemble methods for achieving better performance. Among existing alternatives, data sampling, that is, using different training data for different NNs, is commonly considered to be most effective for achieving diversity. The three most popular algorithms that explicitly or implicitly use different training sets for different NNs in an ensemble are bagging [7], AdaBoost [8] and Negative Correlation Learning (NCL) [9].

The bagging algorithm [7] explicitly creates different training sets for different NNs to achieve diversity. It creates a particular training set for a given NN by forming bootstrap replicates of the original training set. Given an original training set T consisting of m patterns, a new training set T' is constructed by drawing m patterns uniformly from T, and replacing the patterns with bootstrap replicates. Due to random selection, on average, each training set contains only approximately 63.2% unique patterns from T [11]. As a result, some patterns appear multiple times, while others are left out. For an overall NNE decision, a voting process in which all the trained NNs are considered in a final NNE decision where the outputs of the NNs are given equal voting strength is used.

The training set T' for each component NN in AdaBoost [8] is chosen from the original training set T using a bootstrap method, similar to the case of bagging, but with adaptation. Training patterns that were incorrectly classified by previous component NN(s) are chosen more frequently for creating a new training set than patterns that were correctly classified. AdaBoost thus increases focus on the previously misclassified patterns by increasing their presence in the training set of future NN(s). For an ensemble decision, decisions of individual NNs are assigned weights, where the weight of a decision of a component network in the final weighted vote depends on its accuracy [3], [11]. Therefore, combination in AdaBoost [8] seems to be based on implicit NN selection: A network has no effect when its voting strength is zero or nearly zero. However, in the final NNE, all the NNs remain, resulting in a comparatively bulky NNE.

Like bagging and AdaBoost, NCL does not create separate training sets explicitly for NNs in an ensemble. For diversity, NCL rather adds a penalty term to the backpropagation (BP) [17] error function. The error, $e_i(n)$, of a network i for the n-th training pattern in BP is given by the formula

$$e_i(n) = \frac{1}{2}(f_i(n) - d(n))^2, \quad (1)$$

where $f_i(n)$ and $d(n)$ are the actual and desired outputs for the n-th training pattern, respectively. The problem with this error function is that it returns similar hypotheses for different NNs when training on the same training data. Therefore, this outcome results in relatively less diversity than that of other solutions, and is consequently not suitable for performance as is in an ensemble [9], [10]. The NCL algorithm, therefore, introduces a penalty term in the error function to establish training time interaction among NNs in the ensemble. The error of the i-th NN in the ensemble for the n-th training pattern [9] is given by the formula

$$e_i(n) = \frac{1}{2}(f_i(n) - d(n))^2 + \lambda(f_i(n) - f(n))\sum_{j \neq i}(f_j(n) - f(n)) \quad (2)$$

where $f(n)$ is the actual output of the ensemble for the n-th training pattern, and λ is a scaling factor that controls the penalty term. The ensemble output is generally obtained by averaging the outputs of all the component NNs.

Due to the correlation penalty term in the network error function, the component NNs can interact with one another during training, and different NNs are motivated to give different hypotheses implicitly. Therefore, NCL promotes diversity even though all component NNs train using the same original training data. NCL trains a predefined number of NNs, and considers outputs of all the trained NNs for the decision of the final NNE.

Since the early 1990s, a number of NNE methods have been proposed, and the aforementioned three methods have appeared as the most popular methods. In a comparative analysis, each of the three methods of this group has individually outperformed alternative approaches on certain problems. However, among the three, there is no ideal method that outperforms all other methods for all possible problems [4]. Furthermore, a particular popular method may actually show the worst relative performance on certain problems. An NNE that either outperforms or at least performs comparably relative to the best popular methods on every possible problem would be of interest. For such an NNE, we consider a fusion of the popular three methods (i.e., bagging, AdaBoost and NCL) in this study. The next section describes the method in detail.

III. An Ensemble Fusing Multiple Popular Methods (EMPM)

There is no ideal conventional NNE method that performs well on all possible problems. One possible reason is the amazing variety of attributes of real world problems. Any technique that adheres to a single method fails to maintain proper diversity for all possible problems. Moreover, existing NNE methods consider outputs of all produced or trained NNs for an ensemble without checking whether each one is essential. If a particular NN performs very poorly, it may not be possible for the remaining networks to compensate effectively, resulting in a poorly performing NNE. Finally, it is important to construct an NNE with appropriately selected NNs to achieve better generalization ability. Commonly, the quality of the selected NNs increases in proportion to the diversity of those available. Accordingly, a pool of diverse NNs fusing multiple popular methods tends to function well in an ensemble.

There are two vital factors to keep in mind when constructing an Ensemble fusing Multiple Popular Methods (EMPM) for a better-performing NNE: the creation of a network pool using multiple popular methods, and the selection of NNs from the pool. For a pool having a total of N networks, the number of NNs in the final NNE may be less than or equal to N. The following subsections discuss pool creation and the selection scheme.

Diversity in the original pool of NNs is important for selecting appropriate networks for a good NNE. Since any

conventional method produces diverse networks, one may consider any such method for pool creation. However, it is worth noting that a selection of NNs from a given such pool may not perform well, and the constructed NNE may not outperform or perform comparably well relative to others due to similarity in approaches followed for producing the network pool. On the other hand, if a pool is produced using a variety of different methods, an NNE with appropriately selected NNs will always perform better than or comparably compared to the best individual method. Since bagging, AdaBoost and NCL are the most popular methods, diverse NN pool creation using these methods may result in better performance. Results show that NCL performs relatively better for small-sized problems, while bagging and AdaBoost exhibit superior performance for large-sized problems. For very large problems, AdaBoost has shown very good results in some cases [3], [4], [11]. A network pool constructed using the above three methods seems helpful for compensating for the limitations of individual methods, and an NNE constructed with appropriately selected NNs from the original pool will perform better. Therefore, we have chosen to investigate bagging, AdaBoost and NCL for creating a heterogeneous network pool in this study.

The purpose of the selection scheme is to discover the optimal NN subset in the pool for constructing an NNE that performs well. For a pool containing m NNs, a total of 2^m subsets may need to be checked to determine the optimal NNE. Reportedly, a total of 10 to 20 NNs are standard for an NNE with good generalization [3], [4]. If m=15, then the number of total subsets that must be checked is 2^{15}=32768 for determining the optimal NNE. This number is relatively large, and requires significant time for checking. Therefore, this paper also investigates an efficient network selection scheme called "forward selection" for reducing the time required for selecting the NN subset.

The idea of forward selection is simple: Select the best NN from the pool, and try to minimize its error by the compensation of other NNs. An NNE shows better performance when the failure of one component NN is compensated for by other component NNs. Compensation is easy when the error produced by an NN is small. Therefore, NNs in an NNE should be as accurate and diverse as possible to achieve better performance [4]. Therefore, in obtaining the overall result, selecting the most accurate NN of the pool and then trying to compensate any error using other NNs makes sense, and is likely to result in a good selection of appropriate NNs for an NNE.

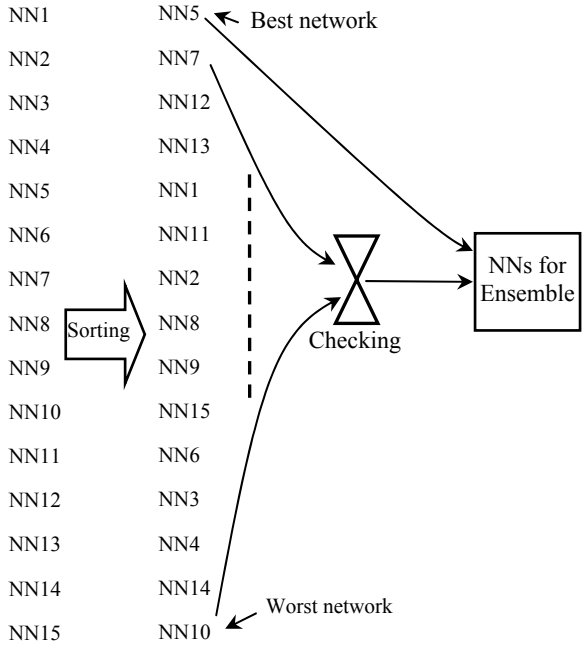

Fig. 1: Forward selection of networks.

The proposed method selects the most accurate NN as the base of the NNE. First, it sorts all the NNs in descending order according to their accuracy. Then, it selects the most accurate NN for an NNE by default. Next, it chooses the second best NN from the sorted list and adds this NN to the previously selected best NN. If overall NNE performance using these two NNs is found to be inferior to that of using the best NN alone, then discard the second best NN. Similarly, other NNs are added from those sorted to construct an NNE with an optimal selection of NNs. Given a pool with m NNs, forward selection evaluates each NN's performance individually, and measures the performance of each of the m-1 combinations of the NNs.

A forward selection scheme for a pool size of 15 NNs is shown in Fig. 1. One may consider either diversity among the NNs in addition to accuracy of the NNE, or simply accuracy alone, as selection criteria. For simplicity, a simple average combination of NNs is used for an NNE decision. Because of the fewer combinations of NNs required for checking, using forward selection is more efficient than checking combinations of all NNs.

IV. Experimental Studies

First, this section provides a detailed description of benchmark problems, and presents an experimental methodology. Then, performance of the proposed EMPM is compared to that of conventional NNE methods in finding solutions to benchmark problems. Finally, an experimental analysis is presented for examining the performance of the proposed method in detail.

A. Benchmark Problems and General Experimental Methodology

Twenty-five real world classification problems were employed in this study. The origin of these problems is the University of California, Irvine (UCI) machine learning benchmark repository; detailed descriptions are available at the UCI website [6]. Table I shows the characteristics of problems which show considerable variety in the number of examples, input features and classes. Thus, these problems provide a suitable experimental test bed.

UCI contains raw data and requires preprocessing for use in NNs. We have followed benchmark methodology [13] for preprocessing data of the problems. The number of output nodes was equal to the number of classes for a given problem. 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 general.

The structure of an individual NN in the NNE was the standard three-layered network having one hidden layer. We chose the number of hidden units based on the number of

Table I: Characteristics of benchmark datasets.

Dataset	Exa-mple	Class	Input feature		NN input	Hidd. node
			Cont.	Disc.		
Australian Credit Card	690	2	6	9	51	10
Auto Imports	205	6	16	8	24	10
Breast Cancer Wisconsin	699	2	9	-	9	5
Balance	625	3	-	4	20	10
Car	1728	4	-	6	21	10
Diabetes	768	2	8	-	8	5
Echocardiogram	131	2	7	1	8	5
Ecoli	336	7	7	-	7	10
German Credit Card	1000	2	7	13	63	10
Hepatitis	155	3	6	13	19	5
House Vote	435	2	-	16	16	5
Hypothyroid	7200	3	6	15	21	5
Ionosphere	351	2	34	-	34	10
King+Rook vs King+Pawn	3196	2	-	36	74	10
Lymphography	148	4	-	18	18	10
Low Resolution Spectrometer	531	10	101	-	101	20
Page Blocks	5473	5	10	-	10	5
Soybean	683	19	-	35	82	25
Segmentation	2310	7	19	-	19	10
Sonar	208	2	60	-	60	10
Splice Junction Gene Sequences	3175	3	-	60	60	10
Satellite	6435	6	36	-	36	10
Wine	178	3	13	-	13	5
Waveform	5000	3	21	-	21	10
Zoo	101	7	15	1	16	10

input and output units, which was one hidden unit for every output unit, plus one hidden unit for every 10 input units[3], [4]. The minimum number of hidden units was set to 5. The most popular logistic sigmoid function, $\phi(y) = 1/(1+\exp(-y))$, was used as an activation function for the nodes in the hidden and output layers. The initial random weights of component NNs were set to between -0.5 and 0.5. The learning rate of backpropagation (BP) was set to 0.1, and each network was trained for 50 equal epochs. The scaling factor (λ) of NCL was chosen to be between 0.25 and 0.75. The selected parameter values are not optimal values, but were chosen for simplicity as well as for fairness in observation.

In order to evaluate the performance of an NNE, generalization is measured on the testing set that was reserved from available data and not used by any NN in training and selection. The testing error rate (TER) is the common measure of generalization; the lower its value, the better the generalization is. Note that the aim of any NNE method is to minimize the TER.

B. Experimental Results of EMPM and Comparison with Conventional Methods

This section evaluates the performance of the proposed EMPM on the benchmark problems. The EMPM fuses three data sampling-based NNE methods (bagging, AdaBoost and NCL); the standard forms of these three methods have been implemented and their results have been compared to those of EMPM. For a more comprehensive understanding, the simple NNE (sNNE) method, where initial weight sets are only different in different NNs, is also considered. sNNE without data sampling exhibits less diversity, and is considered as the baseline point of NNE construction [3], [4].

It can be seen that the TER may vary due to the variation of the testing set data, even if the size of the data set remains the same. Therefore, standard 10-fold cross validations have been used for result presentation. In the cross validation, initially available training examples were partitioned into ten equal or nearly equal sets, and for each turn, one set was reserved as a testing set, while the remaining nine sets were used for training.

The TER achieved by EMPM was compared to that for conventional methods over five standard 10-fold cross validation (i.e., 5*10=50) runs in Table II. In the table, for each problem, the best TER is indicated in bold among the five methods. Thirty NNs were trained for an NNE in a conventional method, since the previous study revealed that this number was sufficient for better generalization [3], [4]. For fairness of comparison, the pool size of the EMPM was also set to be 30 NNs, for which each of the bagging, AdaBoost and NCL methods produced 10 NNs. A conventional method considered all 30 NNs for the final NNE, but the set of NNs in the EMPM was problem-dependent due to selection.

The results presented in Table II reveal the effectiveness of the proposed EMPM approach clearly. It is found that the performance of sNNE is worse compared to the other methods, and that of the EMPM is the best. sNNE exhibits the best TER for only one problem (i.e., Auto Imports), and is the worst for 16 out of 25 problems. For Auto Imports, NCL also achieved the same TER as sNNE, with a value of 0.44. The reason for sNNE having achieved the worst results is its training with same original training data for all NNs, where the initial random weight set variation was the only element to introduce diversity. The traditional data sampling based methods bagging, AdaBoost and NCL outperformed sNNE for 19, 18, and 14 different problems, respectively. Although AdaBoost was shown to be the best among the conventional methods based on average TER (i.e., 0.1351), it was worse than sNNE for five problems. On the other hand, EMPM outperformed sNNE for 24 problems, which was greater in number compared to any conventional method.

EMPM constructs an NNE by selecting networks from a pool in which NNs are trained with different data sampling methods. Hypothetically, for proper selection, it is guaranteed that EMPM outperforms sNNE if any conventional method (that has been employed in pool creation) outperforms sNNE. For the Zoo problem, the TERs for sNNE, bagging, AdaBoost and NCL were 0.126, 0.116,

Table II: Comparison of EMPM with conventional methods over five standard 10-fold cross validation runs. The bottom of the table contains a summary of the results.

	TERs of conventional methods				EMPM	
Problem	**sNNE (30 NNs/NNE)**	**Bagging (30 NNs/NNE)**	**AdaBoost (30 NNs/NNE)**	**NCL (30 NNs/NNE)**	**TER**	**NNs/NNE**
Australian Card	0.151	**0.1406**	0.1562	0.1516	0.1469	5.94
Auto Imports	**0.44**	0.458	0.462	**0.44**	0.455	5.18
Breast Cancer	0.0333	0.0322	0.0325	**0.0276**	0.0313	12.82
Balance	0.0842	0.079	**0.0165**	0.0842	0.0303	15.66
Car	0.1212	0.1158	**0.077**	0.1207	0.0936	15.96
Diabetes	0.2416	0.2387	0.2387	0.2416	**0.2339**	4.02
Echocardiogram	0.1308	0.1384	**0.1169**	0.1308	**0.1169**	6.46
Ecoli	0.3824	0.2394	0.2497	0.3491	**0.2248**	6.78
German Card	0.2502	**0.244**	0.2488	0.247	0.247	14.04
Hepatitis	0.1587	0.1533	0.1813	0.1587	**0.148**	9.72
House Vote	0.0437	**0.0358**	0.0428	0.0391	0.041	13.38
Hypothyroid	0.0592	0.0589	0.0328	0.0585	**0.0262**	4.16
Ionosphere	0.1931	0.1383	**0.1137**	0.1926	0.1206	7.06
King+Rook	0.1441	0.0237	0.1095	0.1419	**0.0224**	20.82
Lymphography	0.1571	0.1629	0.1914	0.1571	**0.1443**	7.94
Low Resolution	0.1143	0.1207	0.1132	0.1158	**0.1075**	7.56
Page Blocks	0.0789	0.0742	0.0597	0.0786	**0.0493**	6.78
Soybean	0.0591	0.0573	0.0591	0.0579	**0.0559**	15.00
Segmentation	0.0913	0.0799	0.0684	0.0912	**0.0639**	6.58
Sonar	0.225	0.228	0.225	0.225	**0.222**	10.76
Splice Junction	0.1604	0.1591	0.1531	0.1577	**0.151**	16.84
Satellite	0.1597	0.1528	0.1461	0.1576	**0.1408**	8.08
Wine	0.0223	0.0247	0.1141	0.0212	**0.0165**	19.56
Waveform	0.132	0.1301	0.13	0.132	**0.1296**	7.70
Zoo	0.126	0.116	**0.038**	0.132	0.048	8.82
Average	**0.1504**	**0.1361**	**0.1351**	**0.1484**	**0.1227**	**10.305**
Method Best/Worst	**1/16**	**3/3**	**5/6**	**2/4**	**16/0**	

	Pair wise win/draw/loss summary				
	sNNE	**Bagging**	**AdaBoost**	**NCL**	**EMPM**
sNNE	-	19/0/6	18/2/5	14/8/3	24/0/1
Bagging		-	13/1/11	9/0/16	22/0/3
AdaBoost			-	9/1/15	20/1/4
NCL				-	21/1/3

0.038 and 0.132, respectively, and the TER achieved by EMPM was 0.048, which was closest to that of the best conventional method, AdaBoost. On the other hand, Auto Imports is a small-sized problem on which the performance of both bagging and AdaBoost was worse than that of sNNE, and on which the performance of NCL was equal to that of sNNE; therefore, for EMPM to achieve a worse TER than that of sNNE for this problem is acceptable.

EMPM also outperforms the conventional methods used to produce a heterogeneous NN pool. It is better than bagging, AdaBoost and NCL for 22, 20 and 21 problems, respectively, according to Table II. This indicates that EMPM may construct NNEs with appropriate NNs. Again, due to the selection scheme, an additional benefit of EMPM is that it achieves a compact NNE that is smaller than that of the traditional methods which train a predefined number of NNs and in which all the networks are considered for NNE. In some cases, with a smaller NNE, EMPM achieved the best TERs. As an example, for the Hypothyroid problem, with 4.16 NNs/NNE, the TER for EMPM was 0.0262; on the other hand, with 30 networks, the TERs for bagging, AdaBoost and NCL were 0.0589, 0.0328 and 0.0585, respectively.

C. Experimental Analyses

This section first presents a modified version of EMPM, where a genetic algorithm is adapted for network selection, and compares the results with the standard EMPM. An empirical analysis for several methods of network pool creation is then analyzed to indicate the performance of the proposed method.

C.1. Forward Selection versus the Genetic Algorithm for NN Selection

The Genetic Algorithm (GA) is a search and optimization algorithm that works based on the process of natural selection [12]. GA has also been used for NN selection for NNEs in previous studies [14], [15], [19]. GASEN [14], [15] applied GA on a bootstrap sampling-based network pool for an ensemble with selected NNs. Selection on same bootstrap based NN (i.e., bagging) might not be effective for all cases, as discussed earlier. SNCL [19] also employed GA on an NCL-based network pool. In contrast, due to offline selection, it is possible to utilize any other method or combination of several methods to create the NN pool that is used in the proposed EMPM. This section investigates the utilization of GA instead of forward selection in the NN selection phase of EMPM. This new approach is called "GA-based EMPM."

A bit string encoding scheme is used in GA-based selection. The gene size was defined as the number of the total NNs in the pool. Each gene is indicated as an NNE with NNs having corresponding bits in the gene represented in binary as 1s. The fitness function was defined with the accuracy of the classification of the training set. The population size was set to 20 based on a number of trial runs; the crossover and mutation rates were 1 and 0.2, respectively. Initially some bits of each gene were randomly initialized to 1. If the overall fitness value of the population did not improve after several generations, an NNE was constructed based on the best-fitted gene.

GA evaluates NN subsets independently of the pool size to find a better subset. GA needs to check $P*G$ subsets of NNs from the pool, where the population size is P and the total number of generations is G. Although discovery of the best subset of NNs by GA is not guaranteed, it is cost-effective compared to checking all possible NNs subsets, that is, 2^m for m NNs, in the pool.

A gene bank is considered in GA to overcome the recalculation of fitness value if a similar gene has already appeared. It is observed that the fitness value (i.e., an NNE's performance based on a gene) calculation requires much more time relative to other operations. For the fitness value of a gene, the gene bank is searched, and the fitness value is retrieved if a similar gene exists; otherwise, the fitness value is calculated and is stored in the gene bank for later use. This process improves the running time of the GA-based selection scheme. However, checking $P*G$ subsets and other operations in GA require more time than the operations in forward selection.

Table III shows the average result of the standard EMPM and GA-based EMPM over five standard 10-fold cross validation (i.e., 5*10=50) runs. The NN pool was kept the same in both cases for fair observation. For the pool of 30 NNs, each of the bagging, AdaBoost and NCL methods produced 10 NNs. The TERs for the NNE with all 30 NNs are presented under the heading of 'Before selection'. A simple averaging technique was followed to obtain the output of the NNE from the output of individual NNs.

Table III: Experimental result of EMPM with GA-based selection and forward selection over five standard 10-fold cross validation runs.

Problem	Before selection	GA-based EMPM		EMPM (Forw. selec.)	
	TER (30 NNs/ NNE)	TER	NNs/ NNE	TER	NNs/ NNE
Australian Card	0.1458	0.1423	12.00	0.1469	5.94
Auto Imports	0.439	0.453	11.98	0.455	5.18
Breast Cancer	0.0331	0.0342	11.82	0.0313	12.82
Balance	0.0787	0.0423	11.68	0.0303	15.66
Car	0.1121	0.0935	11.88	0.0936	15.96
Diabetes	0.24	0.2342	12.06	0.2339	4.02
Echocardiogram	0.1246	0.1169	10.92	0.1169	6.46
Ecoli	0.2757	0.2527	10.86	0.2248	6.78
German Card	0.242	0.2468	11.80	0.247	14.04
Hepatitis	0.1613	0.1547	11.80	0.148	9.72
House Vote	0.0372	0.0396	10.06	0.041	13.38
Hypothyroid	0.0573	0.049	9.58	0.0262	4.16
Ionosphere	0.14	0.1257	12.46	0.1206	7.06
King+Rook	0.0318	0.0229	12.28	0.0224	20.82
Lymphography	0.16	0.1643	11.32	0.1443	7.94
Low Resolution	0.1143	0.1121	11.8	0.1075	7.56
Page Blocks	0.0701	0.0537	10.32	0.0493	6.78
Soybean	0.0529	0.0553	11.96	0.0559	15.00
Segmentation	0.0815	0.0701	12.08	0.0639	6.58
Sonar	0.245	0.229	12.34	0.222	10.76
Splice Junction	0.1551	0.1563	11.56	0.151	16.84
Satellite	0.1487	0.1422	12.14	0.1408	8.08
Wine	0.0153	0.0153	12.08	0.0165	19.56
Waveform	0.1282	0.1286	12.34	0.1296	7.70
Zoo	0.09	0.06	10.98	0.048	8.82
Average	**0.1352**	**0.1278**	**11.604**	**0.1227**	**10.305**

The selection of one scheme (i.e., forward selection or GA) improves the overall TER, as can be seen from Table III. In some cases, the improvement was impressive, such as for the Balance, Page Blocks, and Zoo problems. For the Page Blocks problem, the TER before selection was 0.0701; after selection, the TERs were 0.0537 and 0.0493 for GA and forward selection, respectively. Between the selection schemes, on the basis of the average TER for all 25 problems, forward selection was shown to be better than GA, although in some cases (e.g., the Australian Card and Waveform problems), GA achieved lower TERs. The average TERs for GA and forward selection were 0.1278 and 0.1227, respectively. In addition, the average number of selected NNs

for forward selection is less than that for GA; the numbers of selected NNs were 11.604 and 10.305 for GA and forward selection, respectively.

A remarkable observation that can be made from the results in Table III is that GA selects a similar number of NNs for all the problems. The reason for this result is that GA considers all the NNs in every generation. On the other hand, forward selection is shown to be more effective in choosing an NNE size specifically tailored for the problem. An issue with GA is time complexity, because it always works with a population of solutions (here, NNEs). The coming section will describe this issue using empirical results.

C.2. Effects of Different Pool Creation Methods and Different Pool Sizes

This section investigates various networks pool creation methods for NNE construction. Pools were created using four methods: bagging, AdaBoost, NCL and a combination of these three methods, that is, EMPM. In EMPM, one-third of the NNs were trained by each of bagging, AdaBoost and NCL as similar to in experiments in the previous sections. Four problems were selected for analysis having variation in the number of available examples, input features and output classes. For example, Hypothyroid and Zoo contain both continuous and discrete features; whereas the Ionosphere and Soybean problems have only continuous and discrete features, respectively.

In the experiment, one-third of the available examples were reserved as a testing set for measuring TER and diversity. The remaining two-thirds of the examples were used to train NNs for the pool, and to check status in the selection process. The pool size varied from 9 to 90 NNs for each method. TER, diversity, number of selected NNs in the final NNE, and required time for network pool creation plus the selection process were measured for analysis. Figs. 2–5

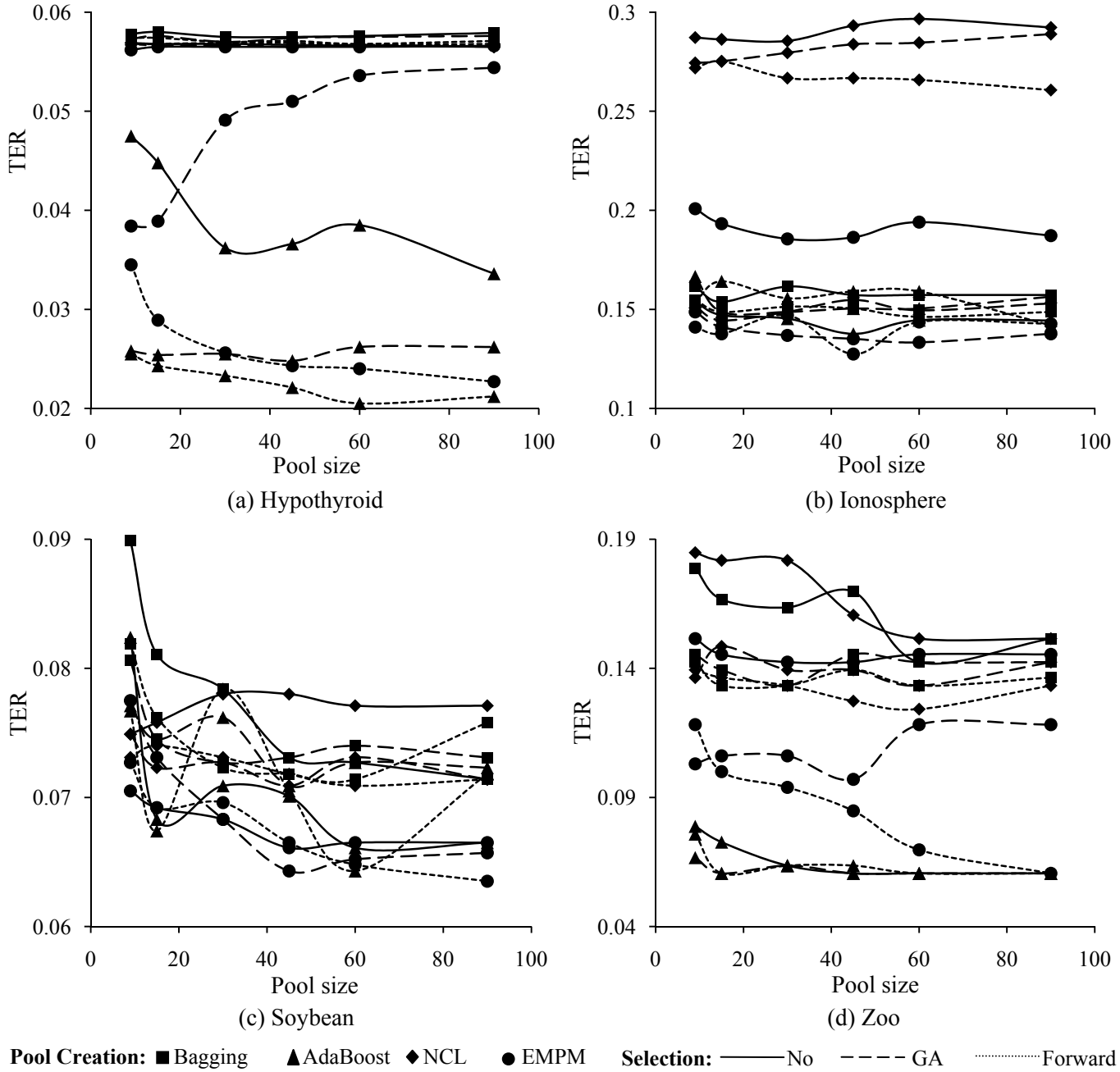

Fig. 2: Effect of pool size on TER.

present a summary of measures; the presented results are the average over ten independent runs.

The diversity indicates how predictions differ among component NNs on the testing set. To measure diversity, we employed the most commonly used pairwise plain disagreement measure technique [16]. For two NNs *i* and *j*, the plain disagreement is equal to the proportion of the examples on which the NNs make different predictions as given in the following formula:

$$div_{i,j} = \frac{1}{N}\sum_{k=1}^{N} Diff\,(C_i(x_k), C_j(x_k)), \quad (3)$$

where N is the number of examples in the testing set, $C_i(x_k)$is the class assigned by network i to example k, and $Diff(a,b)$ = 0, if $a=b$; otherwise $Diff(a,b)$ = 1. The diversity in the plain disagreement method varies from 0 to 1. This measure is equal to 0 when the NNs return the same classes for each example, and to 1 when the predictions are always different. The total NNE diversity is the average of all NN pairs in the NNE.

Figure 2 shows the effect of pool size on TER for both selection schemes. Bagging, AdaBoost or NCL without any selection represent the corresponding standard method. Performance due to selection on these methods is found to be problem-dependent. Selection on the pool created by bagging and NCL is found to achieve better (i.e., lower) TER for the Soybean and Zoo problems; however, the TERs for both selection schemes were almost identical to those for standard bagging and NCL for the Hypothyroid problem.

It is also observable from Fig. 2 that the performance of an NNE of selected NNs depends on the NN pool on which the selection applies. NCL showed a worse TER than other methods for the Ionosphere problem (Fig. 2(b)), and continued to exhibit a worse TER after selection, although

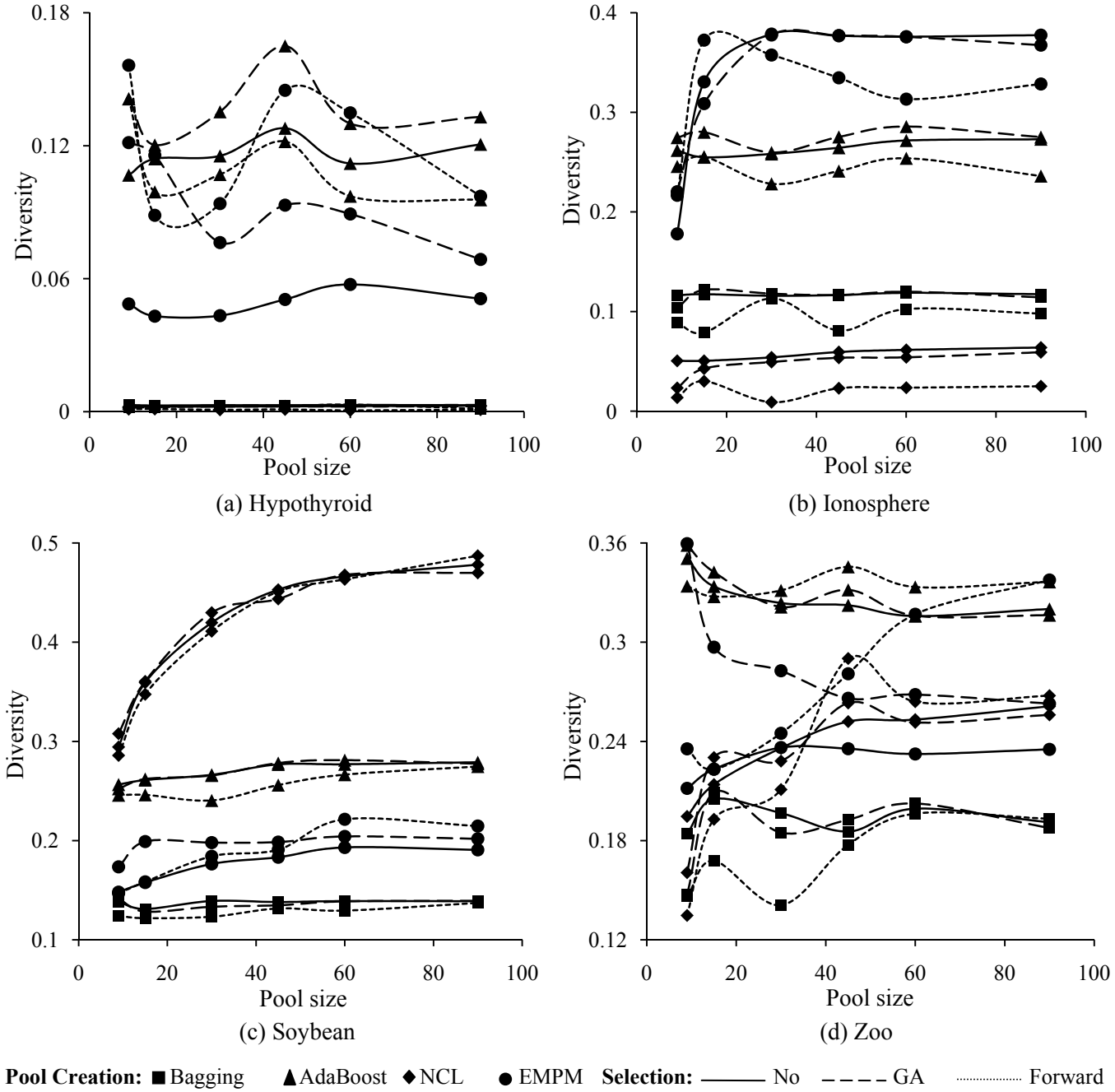

Fig. 3: Effect of pool size on diversity.

the TER improved due to selection. Similar observations for other cases lead to the conclusion that selection on a standard method having worse TER normally fails to achieve a better TER than a standard method having a good TER. Similarly, AdaBoost achieved the lowest TERs for the Hypothyroid (Fig. 2(a)) and Zoo (Fig. 2(d)) problems, and remained with the best TER after selection. Another observation for AdaBoost is that the TER increased after selection in some cases of the Ionosphere and Soybean problems. Each NN in AdaBoost was trained on a different training set based on error-based distribution of the original training data, and each NN took part in a weighted voting combination for a better NNE decision. Since selection schemes did not consider the weight of an NN, the performance degradation makes sense in some cases when the selection process left only one or a small number of important NN(s).

Although selection on the pool of a standard method (i.e., bagging, AdaBoost or NCL) does not result in a better TER in every case, selection on the pool of EMPM is effective. The pool of an EMPM contains heterogonous NNs produced by bagging, AdaBoost and NCL, and selection is shown to achieve a lower TER than the TER for the entire pool for any problem, as seen in Fig. 2. For the Hypothyroid problem, the TERs for the entire pool were 0.0565, 0.0565, and 0.0566 for pool sizes of 45, 60 and 90 NNs, respectively; on the other hand, TERs with selected NNs using forward selection were 0.0243, 0.0276, and 0.0227, respectively. It is worth mentioning here that for the same Hypothyroid problem, AdaBoost was the best among the standard methods (Fig. 2(a)), and that the TERs of EMPM with forward selection were close to those of AdaBoost. We can see that EMPM with forward selection used mostly AdaBoost trained NNs for this problem. A similar observation can also be made for the Zoo problem. The better TER of EMPM makes sense,

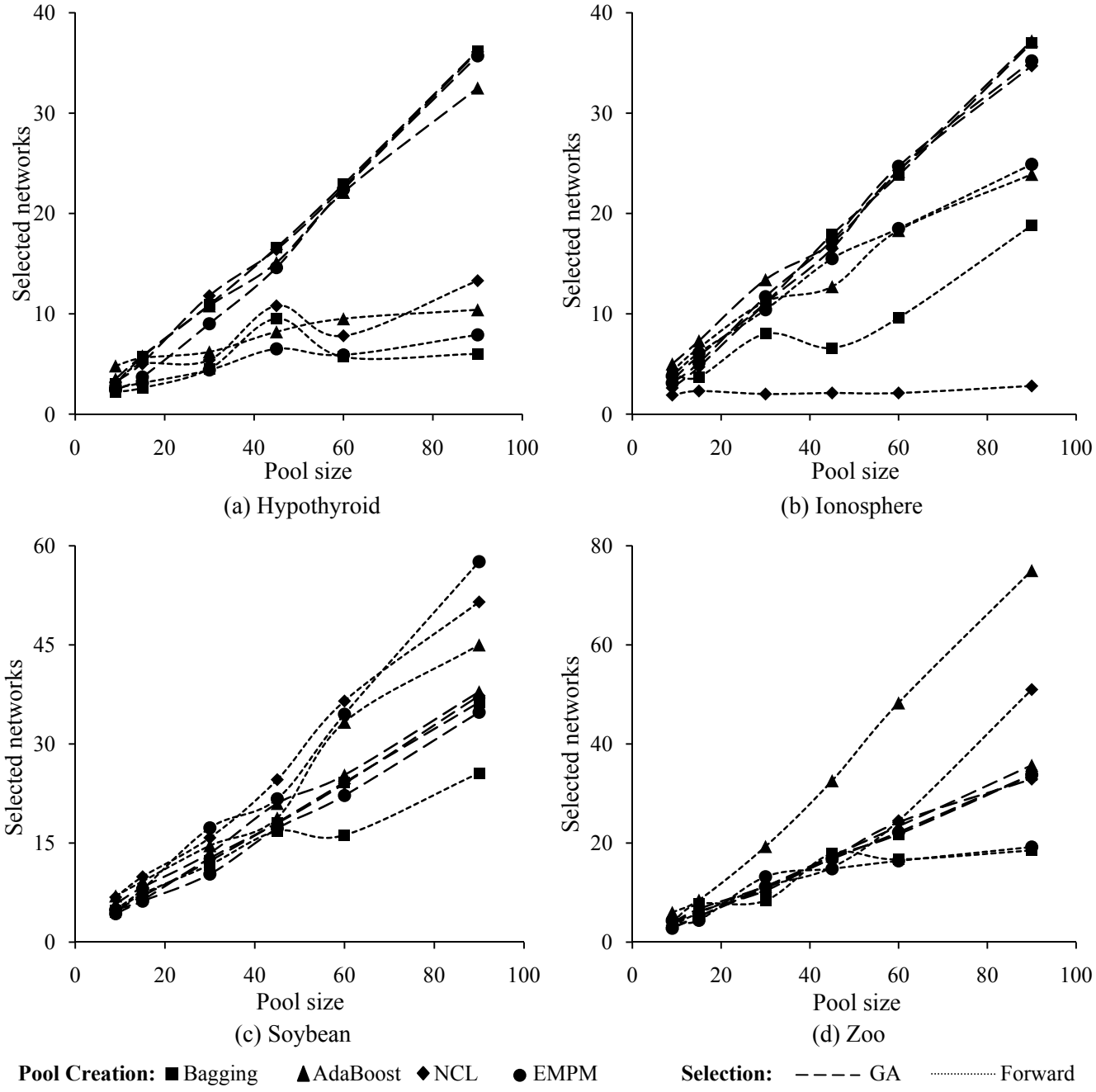

Fig. 4: Effect of pool size on selected networks for final ensemble.

because the pool contained a large variety of NNs in which it was easy to find the best suited NNs for an NNE.

It is also remarkable for the Ionosphere and Soybean problems (Fig. 2(b) and Fig. 2(c)) that EMPM without selection performed better, and that selection also improved the TER in most cases. For the Soybean problem with a pool size of 90 NNs, the TER of the NNE with the entire pool was 0.0665, and TERs for the forward selection and GA-based selection were 0.0635 and 0.0657, respectively; these three values of EMPM were the best values for this problem. The reason for the better results of EMPM may be understood better by measuring diversity.

Figure 3 shows the diversity of corresponding results presented in Fig. 2. From the figure, we can see that AdaBoost showed the best diversity for the Hypothyroid and Zoo problems. AdaBoost also showed the best TERs for both problems (Fig. 2). The diversity of EMPM for the Hypothyroid and Zoo problems was also found to be competitive to that of AdaBoost; therefore, the TERs for these problems were close to those of AdaBoost. The diversity of EMPM for the Ionosphere and Soybean problems for any pool size was better than that for others, and selection also maintained a better diversity. Thus, EMPM achieved the best TERs in most of the cases for the Ionosphere and Soybean problems, as can be seen in Fig. 2.

Figure 4 presents the number of NNs selected for the results shown in Figs. 2 and 3 for the four selected problems (i.e., the Hypothyroid, Ionosphere, Soybean and Zoo problems). The number of selected NNs increased with pool size; selection found more NNs that met criteria when the pool contained more NNs. For a pool size of 60 NNs, the selected number of NNs by GA for all four problems was about 20, and for a pool size of 90, the selected number of NNs was approximately 40. It is worth noting here that at a particular pool size, the GA selected, on average, the same number of NNs for different problems. The reason for this might be random initialization of genes and consideration of all NNs for each generation. On the other hand, forward selection was found to return a problem-dependent NNE size after selecting appropriate NNs from the pool. It selected 13.3, 2.8 and 51.5 NNs for the Hypothyroid, Ionosphere and Soybean problems, respectively, when the pool size was 90 NNs, using NCL.

The number of selected NNs using forward selection was far fewer than that using the GA based selection for some cases, as can be seen in Fig. 4. Also, in some cases, the GA had higher TERs than forward selection even with a larger number of NNs, and the TER difference also increased when the pool size increased. As an example, for the Hypothyroid problem (Fig. 4(a)), the EMPM-selected NNs in the cases of GA and forward selection were 35.7 and 7.9, respectively, with a pool size of 90 NNs, and the corresponding TERs were 0.0544 and 0.0227. These results clearly indicate the effectiveness of the proposed forward selection in comparison with GA. However, in some cases, such as the Soybean and Zoo problems, forward selection considered more NNs than GA. Requiring more NNs for problems having more classes, such as Soybean (19 classes) and Zoo (7 classes), and fewer NNs for problems having fewer classes, such as Hypothyroid (3 classes) and Ionosphere (2 classes), makes sense for forward selection.

Although forward selection returned more NNs than GA in some cases, it does not mean that forward selection took more time in those cases. Selection time mostly depends on the pool size, not on the number of selected NNs. To illustrate the time requirements, Fig. 5 presents an overview of the average time requirements in training NNs for pools and in selection schemes for the Hypothyroid and Soybean problems. Hypothyroid is a large-sized problem having 7200 examples; Soybean has large number of inputs (83) and output classes (19). Actual training time for any method depends on several factors, such as the machine and the environment. For consistency and to ensure fair comparison, all experiments were conducted on the same machine: a Dell OptiPlex 745 (CPU: Intel Core2 Duo 1.8 GHz; RAM: 2 GB).

The time required to train NNs for a pool is much greater than the time required for any selection scheme, as can be seen in Fig. 5. However, forward selection took less time

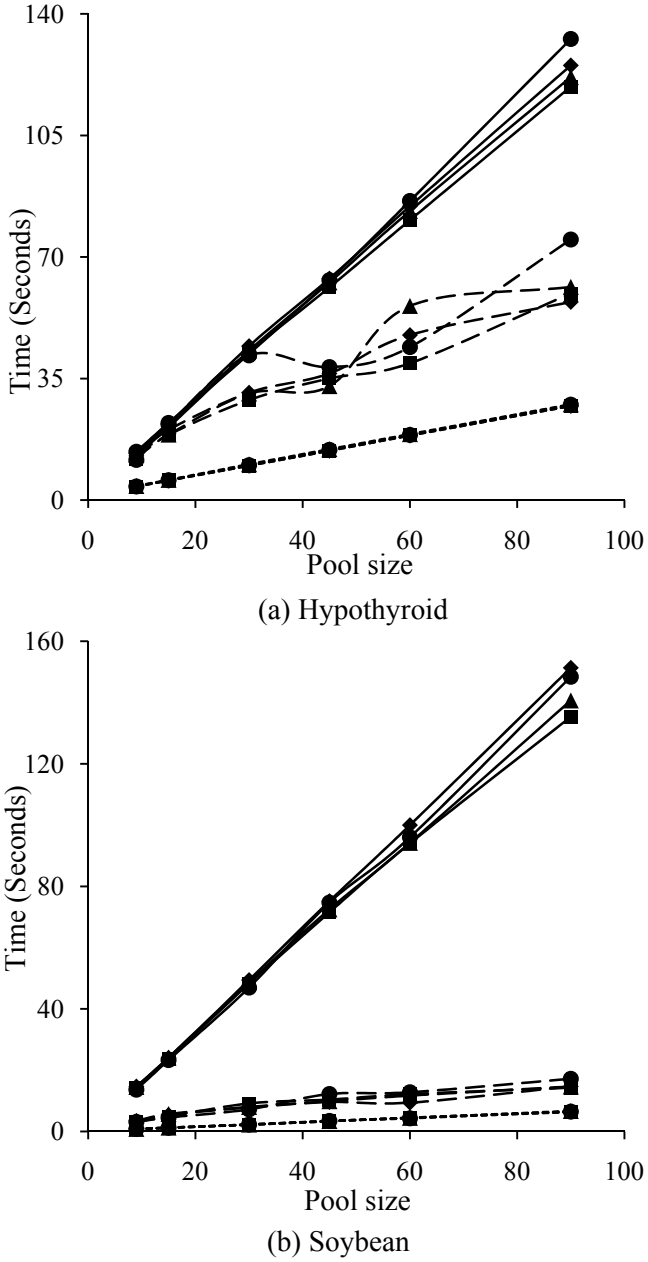

Fig.5: Effect of pool size on training and selection time.

compared to GA-based selection for all cases. For the Hypothyroid problem with an EMPM pool size of 90 NNs, the required NNs training time for the pool was 133 seconds; on the other hand, the selection times were 75 seconds and 27 seconds for GA and forward selection, respectively. The reason for faster operation of forward selection has already been explained in Section III. It is worth noting here, as can be seen in Fig. 5, that the time requirement for the EMPM pool creation (i.e., combination of several methods) is the same as that for any standard method for the same pool size. An EMPM pool is only a set of heterogeneous NNs that are trained using different standard NNE methods. Finally, EMPM with forward selection seems to be the most effective for NNE construction.

V. Conclusions

The purpose of building an ensemble with multiple neural networks is to achieve better performance for any given problem. Conventional ensemble methods train a predetermined number of networks, of which all are generally involved in ensemble decision. Since individual networks are the main task-handing elements in an ensemble, it is important to construct an ensemble with appropriate networks. Therefore, in order to get better performance, this study has presented an ensemble that employs selected networks.

There are two vital factors in constructing an ensemble with selected networks: the creation of a network pool, and the selection of networks. Considering both factors together, this paper proposes Ensembles fusing Multiple Popular Methods (EMPM) as an ensemble method for achieving better performance. Bagging, AdaBoost and NCL methods were also considered in relation to heterogeneous network pool creation for the EMPM, and a simple technique, called forward selection, was used to select appropriate networks for the ensemble.

Bagging, AdaBoost and NCL are the most popular ensemble methods; however, among them, no single solution has been shown to be better for all possible problems. In contrast, the proposed EMPM fusing the methods has been found to be better than the conventional methods with a compact ensemble when tested on a large number of benchmark problems. Forward selection has also been found to be an effective method with a variety of benefits when compared to GA-based selection.

Following this study, there are a number of possible directions for future research. In this study, a heterogeneous pool was created using three data sampling techniques; however, investigation into other methods is also necessary. For simplicity, only training set classification accuracy was only considered as a selection criterion here; nevertheless, including diversity in the selection criteria may result in better performance, although selection time may increase. Furthermore, other selection techniques based on pruning or thinning [18] may also prove to be of interest.

References

[1] A. J. C. Sharkey, "On combining artificial neural nets," *Connection Science*, vol. 8-3/4, pp. 299-314, 1996.

[2] A. J. C. Sharkey, and N. E. Sharkey, "Combining Diverse Neural Nets," *Knowledge Engineering Review*, vol. 12, no. 3, pp. 299-314, 1997.

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

[4] M. A. H. Akhand, Md. Monirul Islam and K. Murase, "A Comparative Study of Data Sampling Techniques for Constructing Neural Network Ensembles," *International Journal of Neural Systems*, vol. 19, no 2, pp.67-89, 2009.

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

[6] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz, "UCI Repository of Machine Learning Databases," Dept. of Information and Computer Sciences, University of California, Irvine, 1998. Available: http://www.ics.uci.edu/~mlearn/

[7] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123-140, 1996.

[8] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," *Proc. of the 13th International Conference on Machine Learning, Morgan kaufmann*, pp. 148-156, 1996.

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

[10] Y. Liu and X. Yao, "Simultaneous Training of Negatively Correlated Neural Networks in an Ensemble," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, no. 6, pp. 716–725, 1999.

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

[12] D. E. Goldberg, *Genetic Algorithms*, Addison-wesley, 1998.

[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] Z. Zhou, J. Wu and W. Tang, "Ensembling Neural Networks: Many Could Be Better Than All," *Artificial Intelligence*, vol. 137, pp. 239-263, 2002.

[15] Z. Zhou, J. Wu , W. Tang and Z. Chen, "Selectively Ensembling Neural Classifiers," *Proc. of the 2002 International Joint Conference on Neural Networks (IJCNN2002), Honolulu, HI, USA*, pp. 1411-1415, 2002.

[16] A. Tsymbal, M. Pechenizkiy and P. Cunningham, "Diversity in search strategies for ensemble feature selection," *Information Fusion*, vol. 6, pp. 83-98, 2005.

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

[18] 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.

[19] K. Tang, M. Lin, F. L. Minku and X. Yao, "Selective Negative Correlation Learning Approach to Incremental Learning," *Neurocomputing*, vol. 72, pp. 2796-2805, 2009.

M. A. H. Akhand received a B.E. degree in Electrical and Electronic Engineering from Khulna University of Engineering and Technology (KUET), Bangladesh, in 1999, an M.E. degree in Human and Artificial Intelligence Systems in 2006, and a Doctoral degree in System Design Engineering in 2009 from the University of Fukui, Japan.

Dr. Akhand joined the Department of Computer Science and Engineering at KUET as a lecturer in 2001, and is now an Assistant Professor. He is a member of the Institute of Engineers, Bangladesh (IEB). His research interests include artificial neural networks and bio-inspired computing techniques.

K. Murase received an M.E. in Electrical Engineering from Nagoya University in 1978, and a PhD in Biomedical Engineering from Iowa State University in 1983.

Dr. Murase has been a Professor of the Department of Human and Artificial Intelligence Systems, Graduate School of Engineering, University of Fukui, Fukui, Japan, since 1999. He Joined the Department of Information Science of Toyohashi University of Technology as a Research Associate in 1984, the Department of Information Science of Fukui University as an Associate Professor in 1988, and became a professor in 1992.

Dr. Murase 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 on the Board of Directors of the Japan Neural Network Society (JNNS), as a Councilor of the Physiological Society of Japan (PSJ), and as a Councilor of the Japanese Association for the Study of Pain (JASP).