The Use of Output Combiners in Enhancing the Performance of Large Data for ANNs

Mumtazimah Mohamad, *Member, IAENG,* Md Yazid Mohd Saman, and Muhammad Suzuri Hitam

***Abstract*— Deriving classification information from large databases presents several challenges. The current methods used to classify a large dataset have the disadvantage of requiring long computational time and high complexity. In addition, most of the methods can only deal with selected features of the data while some of the methods can only deal with categorical or numerical attributes. This paper proposes large data solutions by defining the strategy to classify large data with local processors of Artificial Neural Networks (ANNs). A combination technique for reordered ANNs is proposed in modeling the combination of multiple ANNs as part of framework approach. Several repeated experiments with different techniques tested with the MNIST dataset show good percentage of performance and reduction of errors. The results obtained are in line with the importance of good performance achieved with the use of combiner for a large data solution.**

***Index Terms*—Large data, neural network, multi-classifier, output fusion, combiner**

I. Introduction

MANY real-world applications require the identification of a problem solution for a large collection of data in high dimensional space. The enormous size of these datasets poses very challenging problems in recognition tasks for efficient processing [1, 2]. As a result, the enhancement in recognition method with high efficiency and accuracy to support high dimensional data access has become an active research area. The term "large scale" can refer to different kinds of problems such as, problem with a large number of features, a large number of samples or a large number of categories [3]. The challenge of training a classifier with a large number of samples lies in the computational complexity of the training task. This is because the variability of the categories is high and several examples from each category are necessary for a correct representation of the data.

Classifiers use the samples several times in order to optimize the internal parameters such as the weights of the neural networks. This would result in longer training time due the large amount of data. A large number of samples in the training data also affect the complexity of a classification process. The model's hyper-plane representation based directly on the samples with a large number of reference vectors is impractical [4]. Strategies to select smaller subsets of data are needed in order to reduce the training times. Appropriate selections are very critical, since the strategies are usually heuristic method that does not guarantee a better performance for all cases.

The challenge of using Artificial Neural Networks (ANNs) for large data lies at the exclusion of features and the difficulties of adjustment that affect variations in position, orientation and scale. Hence, it requires a more intelligent and invariant feature selection and extraction mechanisms. Our contribution is on the strategy of partitioning using reordering technique based on [5] for parallel ANNs training to cope with large data. The solution of the problem to be tackled is a little bit different from previous research because it is assumed that each data and features in the dataset is important. Since the recognition process for large data is usually very complex and requires high computational time, a large ANNs size with ensemble technique is needed to accomplish the task. The performance

This rest of this paper is organized as follows. Section II describes the related works in detail. Section III describes the first strategies in for partitioning data as well as the diversity technique to prepare the data sequence for partitioning. Section IV describes the second strategy to combine the output of each classifier. Section V presents the experimental task for training and testing for the proposed strategies. Section VI and VII shows the result discussion and conclusion respectively.

II. Related Works

The partitioning of data is trending solution to large dataset [6]. With a large Multi Layer Perceptron (MLP) network, online ANNs could lead to thousands of epochs that require up to months to process [7, 8]. Previous research on large data have suggested several techniques such as the use of AdaBoost technique [9] , Rule Extraction technique [10], ANNs partitioning [11], and Support Vector Machine (SVM) [12]. The related issues in large datasets are classification tasks and accuracy of the machine learning used [13, 14]. Recognizing large dataset requires computational time that grows rapidly in relation to the data size, making the time required to be too long when solving large dataset [15]. ANNs is suitable for highly complex learning concepts for sufficiently large data in training times

Manuscript received July 18, 2013; revised February 12, 2014.

Mumtazimah Mohamad is with the Faculty Science & Technology, University Malaysia, Terengganu, Malaysia, on leave from University Sultan Zainal Abidin, Terengganu, Malaysia (phone: 6019-934-8822, fax: 609-6155-722; e-mail: ummurifqi09@gmail.com).

Mohd Yazid Md Saman is with the Faculty Science & Technology, University Malaysia Terengganu, Malaysia (e-mail: yazid@umt.edu.my).

Muhammad Suzuri Hitam is with the Faculty Science & Technology, University Malaysia Terengganu, Malaysia (e-mail: suzuri@umt.edu.my).

that scales linearly with the data size [16]. Essentially, ANNs with a larger dataset requires more time to generalize the learning and minimize the error function with additional hidden nodes. Additionally, it would be more time consuming if the datasets hold more than 50 thousand rows multiclass patterns with multidimensional attributes. SVM performs faster classification but not suitable for large dataset classification. It is because it needs to solve the quadratic programming problem in order to find a separation hyper-plane, which greatly increases the computational complexity [17].

A large number of categories can cause the training procedure unfeasible for many types of classifiers. Applications such as speech or handwritten character recognition or other problems containing thousands of different classes still remain a challenge [18]. ANNs with thousands of output neurons are impossible to be trained, as the errors associated with the samples would be too small for the direction of the gradient to be properly defined[2]. Strictly binary ANNs, which depend on being arranged in sub ANNs, would contain millions of ANNs or thousands of large scale binary problems to be solved. Moreover, data elimination cannot be used, as all samples are considered as important for a complete training. Regarding large dataset, the use of parallel processor is practical since it is based on scalable features and due its ability to accomplish simultaneous tasks [7, 8]. Due to this facts, the use of combiners is feasible as they use all the available information for all available ANNs, hence contributing to better and more robust solution in most application [19].

The use of a single ANN usually leads to unstable learner; and in fact it is sensitive to the initial conditions and works differently for different training data [20]. Therefore, a combiner technique has to be employed in order to preserve the capability of ANNs. Kittler et.al (1998) [6] had investigated the need for a theoretical framework to describe the combinations of classifiers (or networks). He then proposed a technique to be used in output combiner strategy, called the parallel combinations of classification. The outputs from multiple ANNs are required to be in diverse conditions [21, 22]. It is because when different areas of input spaces have been learned by some classifiers, they become specialized in specific areas of the input spaces, and consequently have fewer errors in those areas. Furthermore, the architecture of neural networks itself determined by trial and error and it is not unique. Thus, integrating different neural network using output combiner strategy is an elegant and effective way to solve the variety yielded network's output and it is rather easy [4, 12].

The proposed framework in this paper involved disjoint subsets of data that can be clustered in parallel. This data can either be used independently or merged to allow clustering to scale for large datasets. After individual training, each ANNs has its own diverse results. The results are then combined in order to get an aggregated output. In this paper, several methods to combine outputs of each classifier have been investigated. Individual classifier performances are not related significantly with combined performance as they missed out the information about the team strength of the classifiers [21, 23]. Therefore, the delegation of ANNs tasks should take into account the diversity principle that must be applied onto all existing ANNs. Diversity is important since individual ANNs have their own identities that are different to each other. Previous researchers have proven that multiple ANNs can outperform their base ANNs model since individual ANNs tends to make errors on different examples [4, 5]. However, there is no advantage to combine a set of identical ANNs if all ANNs generalize in the same way. Therefore, in order for this process to be effective, individual experts must exhibit some level of diversities among themselves [24].

There are a few techniques that can be used by each of the multiple ANNs to make different errors. Those diversity techniques are [25]: a) different initial conditions, b) different network architectures, c) different training data and d) different training algorithm. The first technique may involve an ANNs initialization with random weights, the learning rate and the momentum or the combination of them. The second technique is to vary the network architecture by changing the number of hidden layers or number of hidden nodes to set up each of ANNs with different architecture. The next technique is used to diversify the training data where re-sampling or pre-processing data methods such as bagging, noise injection, cross-validation, stacking, boosting and input decimation can be used. The last technique is by forcing each individual ANNs to run with a different ANNs learning approach from each other.

There are two main strategies that contribute to the core subjects in this paper. They are data partitioning and combination technique. The data partitioning strategy involves the pattern reordering and partitioning of the dataset. The detail of technique will be discussed in Section III.

III. Data Partitioning

Data partitioning is a popular algorithm for solving complex problems. It allows problems that are more difficult than the standard classifiers to be easily solved. Although the algorithms are naturally implemented as recursive procedures, they can also be implemented in a non-recursive way that stores the partial sub-problems in some explicit data structures. Most solutions designed for pattern classification composed of the following steps [26]: (i) breaking the problem into sub-problems, (ii) solving the trivial cases of the sub-problems and (iii) combining the sub-problems to form the original problem.

Data partitioning provides a way to design efficient algorithms. For example, if a base classifier has a complexity proportional to $O(\iota^2)$, where ι is the number of samples, then by dividing the problem to R smaller sub problems, the average sub-problem's complexity becomes $O((l/R)^2)$. If $R > 1$, the new algorithm reduces the complexity to $R^{-1}O(l^2)$. This reduction can also be observed as an advantage on the memory usage as memory is a very critical limiting factor to any computational process. Another advantage to this technique, it can be easily adapted for parallel processing. The sub problems are usually independent and can run on different processors without requiring complex strategies for communication between the different processes [9].

For the non-recursive method, instead of forming trivial cases, simpler versions of the original problem is solved in order to find the final solution [27]. This approach is suitable for the classification scenario where powerful methods for normal-sized problems are available, but are unable to solve large scale problems. The data partitioning technique in this paper has been encapsulated with the technique for creating diversity which will be described in following sub-sections.

A. Creating Diversity

Many researchers believe that the success of classifier ensembles not only depends on a set of appropriate classifiers, but also on the diversity being inherent in from each classifier [28-30]. A diversity technique is adopted in assigning vectors for worker processors in order to make sure the training data is diverse among processors. A classifier that is diverse would have the ability to find the extent of diversity among classifiers and estimate the improvement or deterioration inaccuracy of individual classifiers when they have been combined [31]. Essentially, there is no advantage to combine a set of identical ANN if all ANN generalize in the same way. In addition, the diversity also has been raised up by [13] as a factor of increased of ensemble.

In order to have a significant improvement, the individual classifiers must exhibit some level of diversity of own identity among themselves [19]. Therefore, there are a few methods for each of multiple ANNs in making different errors. They are either with different initial conditions, different topology, different training data or different algorithm.

The input data can be presented as DS X d pattern matrix or DS X DS different matrix. Suppose DS is a set samples of input data. A selected reordering is run on each of the samples of DS that results in B partitions $P=\{Tr_1, Tr_2, \ldots Tr_{rm}\}$.

Different Initial Conditions

A set of ANN could be varied by random weight initialization, learning rate, different momentum rate for each multiple ANNs. Different initial weight to different networks can converge to different local minima and independent errors.

Different Topology

By varying the topology or the architecture, with different hidden layer number or hidden nodes number or different neural network with different architecture are useful for diversity; the error made by two modular systems with different internal structure might well be uncorrelated.

Different Training Data

This method is the most popular used in ensemble methods, which involve in altering the training data. Different datasets in each network can lead to a different space of possible classifiers. It also mean it allows individual classifiers to generate different decision boundaries [9]. Each data can be varied using different re-sampling or pre-processing data. By using pre-processing technique, the data can be extracted from the raw data for a different feature sets. Either, the input data for ANN could be distorted in different ways. The data smoothing or normalization could be employed in order to make sure smooth and learnable ensemble learning. In certain cases, there would be an issue in establishing different training datasets. In specific, for low number of patterns in a particular datasets, the subset of training dataset would appear similarly with other subset and it could caused the lower degree of diversity and impractical.

Different Training Algorithm

Different algorithm or different multiple ANNs can be employed, e.g, feedforward ANNs, RProp training, conjugate gradient, recurrent network and etc. Associate with varying data, this training algorithm could be used to assemble a set of potential multiple ANNs. This paper adopts two techniques, (i) different initial condition technique and (ii) different training data. The initial condition is based on the ANN parameter and weight setting while the later is the reordering technique. The reordering procedure has been discussed in next sub-section.

B. Reordering Procedure

Reordering procedure is important to make the different training data for multiple ANN using parallel processors. Reordering procedure alters sequence of patterns to apply the concept of diversity[4]. The individual network in each processor is desperately needed to use batch learning mechanism with respect to scalability of large dataset. Furthermore, maintaining the original sequence can let all ANN falls in the same or very similar configuration and the training condition is very low. Fig. 1 shows the original ordering of datasets.

Algorithm: Original Reordering

Input: original dataset DS, number of processor P
Output: The new training subsets $\{Tr_1, Tr_2, \ldots Tr_m\}$

Begin
  Initial weight
  for t=1 to P
    for i=1 to N pattern
      Select S_i from DS
    end for
  end for
  Output the final training subsets $\{Tr_1, Tr_2, \ldots Tr_m\}$
End

Fig. 1. Original ordering of learning algorithm

This network is impractical for multiple ANN because there would be no improvement to the classifiers if the involved training data is the small and similar to the other network in parallel processors [32]. The reordering is altered with simple reordering resampling with replacement.

Another reordering technique is Bagging algorithm as in Fig.2. The bagging algorithm is widely used and has been proved for data sampling with replacement and efficiently constructs a reasonable size for training data [12]. A bagging sampling algorithm such as in Fig.4, adapted from [9] creates a unique training set with replacement over a uniform probability distribution on the original data. The sampling process with replacement means that each sample

values are independent where the covariance between two samples is zero.

Input: original dataset DS, boostrap value S
Output: The new training subsets $\{Tr_1, Tr_2, \ldots Tr_m\}$

```
Begin
  for t=1 to N
    for i=1 to N
      RandRow =S*rand()
        if RandRow <= P
          St (i, all columns)=
              DS(randRow,AllColumns)
        End if
    Next i
  Next i
 Output the final training subsets {Tr1,Tr2,...Trm}
End
```

Fig. 2. Bagging algorithm as [3] for data re-sampling in generating different data.

C. Delegating Training Tasks to Parallel Processors

Parallel learning plays an important role since the size of the dataset is expected to grow faster than the capacity calculation [8]. Although, parallelism could manage long training time for sequential ANNs within minimal time, it has been discovered that the accuracy deteriorated when multiple processing took place [14]. Distributing the training part to more than one processor enables large scale computations that are economical and reliable [26]. Parallelism can minimize the usually long training time in sequential ANNs. However, this paper focuses on the method to reduce training using parallel training.

In multilabel ANNs classification, m problem can be described as with respect to a given d-dimensional feature space, Ω, and training dataset $\Omega_{tr} \subset \Omega$. Each element $\bar{x}$ in Ω_{tr} is associated with class label, $l \in \text{label} = \{l^1, l^2, \ldots, l^m\}$, where $l^j \neq l^h$ for all $h \neq j$ and $m > 2$. An ANNs system F can be trained on Ω_{tr} such that for any given feature vector $\bar{x} \in \Omega$, $F(\bar{x}) \in l$. F can be ANNs system or single ANN where the weights are determined by a ANNs algorithm.

The training dataset T is partitioned into equal parts $T_1, T_2, T_3, .., T_N$, where N is the number of parallel processors used. Each processor has a whole copy of the network. Each of them presents a different pattern block for each training cycle as in (1). Each of the processors has their own ANNs training. Equation (1) shows the first phase which is to get the individual network response after representing the pattern data into the network. This feed forward phase can be formulated based on [5, 33] where N is the number of nodes of the corresponding error while w_{j3} is the weight of node j of the hidden layer to the nodes. w_{ij} and x_i are the weights for input node to the node of hidden layer and the output values. T_j is the threshold for the output nodes.

$$y = F_3\left(\sum_{j=1}^{N} W_{j3}\left(F_2\left(\sum_{i=1}^{N} W_{ij}x_i - T_j\right)\right) - T\right) \quad (1)$$

The error signal is based on the weights of the neural network [4] as in (2). E represent sum square error which describes the difference between the output y the desired output t.

$$E = \frac{1}{2}\sum_{p=1}^{p}(t-y)^2 \quad (2)$$

The error E, is minimized by after the weight is calculated. The gradient descent weight update starts from the output layer to the hidden layer by propagating the error to each layer. In this parallel case, the weight is updated locally. The component gradient is the result from individual processors that each represents some part of the batch of the training pattern as summarized in (3). $\bar{w}$ represents the set of all weights. $\Delta B^{\bar{w}}$ represents the weight changes in batch B to the corresponding sum of the component gradient $\bar{g}^{<q>}$.

$$\Delta B^{\bar{w}}(n+1) = \eta \sum_{p \in B_q} \frac{\partial E_p}{\partial \bar{w}} + \alpha \Delta B^{\bar{w}}(n)$$

$$= \eta \sum_{q=0}^{p-1} \bar{g}^{<q>} + \alpha \Delta B^{\bar{w}}(n) \quad (3)$$

The component gradients $\bar{g}^{<q>}$ from each processor are synchronized using (4). Equation (4) describes the modification that involved only once in performing the collection and summation of all component gradients of $\bar{g}^{<q>}$, instead of each pattern in the sequential algorithm.

$$\bar{g}^{<q>} = \sum_{p \in B_q} \frac{\partial E_p}{\partial \bar{w}} \quad (4)$$

The component gradient in batch B communicates using neighbor configuration of P processors. Each P processor will send its component gradient and will receive p-1 processor's component gradient and implicitly exchange the component gradient with the predecessor. This process is repeated until the sum square value is smaller than the threshold defined.

After vectors of pattern have been generated and assigned to multiprocessors, the master processor has to wait for each processor's response in order to combine the outputs. The communication of distribution of vectors involved the parallelization is implemented using Message Passing Interface (MPI). MPI is a parallel standard application using message-passing paradigm. It provides portability, and flexibility in managing and carrying parallel algorithms with optimized code for all parallel architectures. Fig.3 shows that the master processor has to identify and allocate vectors of pattern for available individual processors. This procedure involves assigning vectors of pattern for each processor in order to delegate task. The master processors process feeds the queue with tasks at one end, and the tasks are shifted down to the queue of worker processors. When a worker processor process, the neighbor processor sending a tasks shifted down to workers and the process is idle. Then, the tasks are to be shuffled to the left so the space held by the task is filled into the left side end of the queue. When the master processor receives all weights and output vector from worker processors, the ensemble combination of outputs will take place. Detail on the combination technique is to make sure the best single output obtained from each processor is discussed in Section IV.

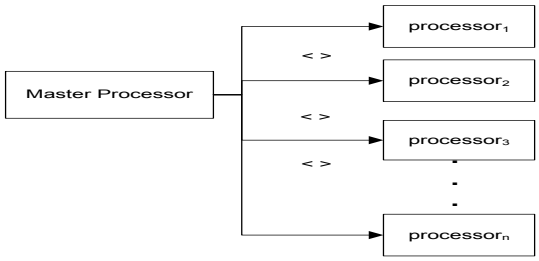

Fig. 3. Delegating tasks to processors

In worker processor part, after receiving vector of patterns from master processor; each processor has its own time with different training pattern for ANNs learning. Before ANNs training started, random weights have to be initialized. The initialization considers the number of processors include master processor ranging from -1 to 1 floating numbers. The weights initialization is also considered to be one of diversity technique for classifier for each worker processor. The worker processor adopts (1) for batch training in to complete the hidden and output nodes for feed-forward ANNs and error term for correcting errors. The errors is as in (2) are iteratively calculated until the minimum errors found. The weights updates are performed locally based on the errors based on (3). The weights are then being saved after the entire data patterns vector complete. After the vectors of pattern completes, the worker processors are considered as a classifier and capable to classify patterns. All the worker processors weights and outputs will be sent to master processor for next procedure. At this point, the worker has their own output classifiers and be ready for combination process. The combiner technique will take place to aggregate and combine all the outputs of multiprocessors and will be discussed in Section IV.

IV. Combining Multiple ANNs Outputs

The underlying idea of obtaining the best output among system is the utilization of the valuable information from neural networks [20]. There are several important requirements for both classifiers and combining techniques to be considered to ensure that they can achieve high classification performance. Each individual network should have enough training data and each of the members of the multiple ANNs must have a complementary set of classifier [34]. The number of training for each pattern and the network size are the two important factors in measuring the performance of neural networks [30]. A combiner can be developed based on the modification of the training algorithm or on the modification of learning set. The output of after combiner procedure for multiple ANNs is often more accurate than any independent network output. It is because the independent network within multiple ANNs network can have potentially different weights and techniques in creating the diversity of the networks. Approaches that use high resolution representations usually have n number of examples for each clustering solution in the ANNs environment. Hence, affecting its computational time and memory complexity [35].

Numerous techniques of ensemble are used to model the combination of multiple outputs such as linear combiner, output combiner and voting combiner. There are basic rules for choosing specific combiner in ensuring consistent behavior among classifier and data. The linear combiner is dedicated to be used for real-valued number of output model such in regression or classification combiner. Each component partition in P is a subset of dataset $P_i = \{C_1^i, C_2^i, C_3^i, \dots, C_{k(i)}^i\}, Tr = C_1^i \cup C_2^i \cup, C_3^i, \dots, C_{k(i)}^i \quad \forall P_i$, and k_i is the number of clusters in the i^{th} partition. The probability of class y is estimated based on [24] by using input x of a model which is $f_t(y|x)$ and combiner $t = \{1, \dots T\}$ as (5). The weights of each combiner represented by $w_t = \frac{1}{T}, \forall t$, which uses normalization Z function for a valid distribution as in (5).

$$\overline{f_t}(y|x) = \frac{1}{Z}\prod_{t=1}^{T} f_t\,(y|x)^{w_t} \qquad (5)$$

A simple homogeneous averaging of the probability estimates possibilities of non-homogeneous weighted average. A non-homogeneous combination could give a lower error than a homogeneous combination if the classifiers have different accuracies. In practice, it is difficult to estimate the parameters w without over-fitting and it will lead to a small gain. Combination of classifiers by mixing of each expert is non-homogeneous but it dependant to the input x. It is suggested to use the output combiner as proposed by [5]. The class probability estimates is assumed to be independent to make sure the determination of possible and reliable weights w. The linear combination and output combination allows for continuum of combining strategy. The linear and output combiners are applicable when the output of classifier is real-value numbers. In order to aggregate the output of each classifier and also to achieve the objectives of this research; the following techniques for ANNs combination have been investigated.

A. Output Average combiner

Output Average combiner is the combiner that utilizes the average obtained from each classifier processor. The final output for this simple average is given by [5]. All classifier's output is summed and averaged as in (6).

$$\hat{y}_c(x) = \frac{1}{N_{networks}}\sum_{net=1}^{N_{networks}} y^{net}\,(x) \qquad (6)$$

Then the class c, yielding maximum of the averaged values, $\hat{y}_c$ is assigned to the pattern. Any argmax returns the argument, class c, with highest output value $\hat{y_c}(x)$ in (7).

$$(x) = argmaxC = 1, \dots, Nclasses \;\; yc(x) \qquad (7)$$

Output Average is a simple type of combiner where it takes the averages the individual classifiers the average of $N_{networks}$. The notation of super index shown in (7) is referring to the multiple ANNs network $N_{networks}$.

B. Weighted average combiner

This technique combines average and weighted majority voting in which the weights are applied not only to class

labels, but also to continuous outputs. This kind of combination rule can be categorized as trainable or non-trainable combination rule, depending on how the weights are obtained. In non-trainable combination rule, the weights are obtained as a part of regular training during the combination process just as in AdaBoost. While, for trainable combination rule, the weights are obtain from separate training such as in a mixture of experts' model.

The weights that are generated for each classifier or class of output, from the training performances are represented by T weight, w_T. The total support for class w_j is:

$$\mu_j(x) = \sum_{t=1}^{T} w_{t,j} d_t, j \ (x) \quad (8)$$

where $w_{t,j}$ is the weight of the t_{th} classifier for classifying class w_j instances.

C. Majority voting combiner

Majority voting is the simplest technique for combining classifiers. The class that receives the largest number of votes is selected as the final classification decision of output. Majority voting can be in any class whether when almost all the classifiers agree [9]. At least more than half of classifiers, or the highest vote (whether exceeded the 50% votes or not) will nominate the selected output as in (9).

$$vote_c^{net}(x) = \begin{cases} 1 & if\ y_c^{net}\ is\ the\ highest\ value\ in\ y^{net} \\ 0 & otherwise \end{cases} \quad (9)$$

All in all, the votes for class c, is calculated according to (10).

$$v_c(x) = \sum_{net=1}^{N_{nets}} vote_c^{net} \ (x) \quad (10)$$

Lastly, the class which is most often voted by the classifiers, which is the highest v_c (x), is assigned to the pattern, x, as in (11).

$$\text{Class}(x) = \arg\max_{c=1,\ldots,N_{classes}} (v_c \ (x)) \quad (11)$$

The weakness with this kind of voting is that the information provided by the network is reduced to a single vote so the probabilistic information related to each output is omitted.

V. Experimental Study

Evaluation of performance based on strategies of partitioning and ensemble method are discussed in this section. The details of the used datasets are given in the subsequent section. Then the experimental setting and the result are presented.

A. Dataset

The proposed method is examined over one large benchmark dataset which is the Modified National Institute Science & Technology (MNIST) data. The MNIST dataset is a standard handwritten digit classification and recognition benchmark. It is originally developed by the National Institute of Standards and Technology [3]. The dataset contains the original black and white images of digits 0 to 9. To date, there are 60 000 training samples and 10 000 test samples, with a resolution of 28 × 28 pixels and a depth of 256 bits, that can be used to test and verify the effectiveness of various machine learning algorithms. These images are normalized using a bipolar technique while preserving their aspect ratio. The dataset is stored in vectors and multidimensional matrices. The integer data are stored in high-endian format used by most non-Intel processors. The image pixels are organized in row-wise values of 0 to 255. The value of 0 represents the background color (white) while 255 represent the foreground color (black). The network is trained using the bipolar values instead of the grey scale values. Bipolar in this case refers to the normalization of both network input and output.

B. Experimental Settings

The pixel intensity of the original data of MNIST is in gray scale images which ranges from 0 (background) to 255 (maximum foreground intensity). They has been normalized to their real values as in [11] in (12).

$$\frac{pixel\ intensity}{127.5} - 1.0 \quad (12)$$

A prototype simulator has been developed to simulate the training of ANNs on parallel computers and for sequential ANNs. Table I shows the network parameter setup for processing the MNIST data.

TABLE I. NETWORK SETUP

Network Parameter	Values
Number of Patterns	60000
Number of Inputs	784
Number of Outputs	1
Number of Class	30
Hidden Nodes	30
Number of weights	23581

To get more generalized network, the data was normalized and randomly distributed from all classes related to its class. The dataset were divided into two groups: the training (70%) and the testing datasets (30%). The training parameter values, such as the learning rate and the momentum were fixed throughout the training. The cluster was set up in seven nodes using Intel Dual Core Machines. The algorithm was developed using C language while the MPI was implemented using MPICH2 and was compiled using Visual Studio 2005. The sequential training was simulated with the used of combiner and the exclusion of combiner part as well as for parallel training. The parallel training results obtained were significant where major decrease of time in training were observed with decreasing trends. The parallel training executed for two mode which is for bagging reordering (as in Fig. 1) and simple reordering from the original ordering of datasets (as in Fig. 2). To be more fair and general, all experiments are averaged up to 10 independent runs. There are seven partitions with respect to seven processors have been used for this experiment.

C. Performance measure

The performance of multiple ANNs was measured based on i) the improvement comparison and ii) reduced error. The Improved of Performance (IP) was calculated for both a

single ANNs and multiple ANNs that use combiner. The IP calculation was based on (13).

$$IP = Performance_{ensemble} - Performance_{singleNet} \quad (13)$$

Equation (13) refers to corrected classified patterns of the test set for multiple ANNs and single ANNs. On the other hand, the Percentage of Error Reduction (PER) is calculated afterwards based on [6]. PER is calculated based on (14).

$$PER = 100 \cdot \frac{Error_{singleNet} - Error_{ensemble}}{Error\_singleNet} \quad (14)$$

where Error is:

$$Error = 100 - Performance \quad (15)$$

Error is refers to total errors of recognition by single and multiple ANNs. Any negative values for calculations in (13) and (14) will indicate that multiple ANNs for the particular technique perform better than a single ANNs.

VI. Result and Discussions

The combination techniques were tested to compare their performance for large dataset. The dataset were normalized using bipolar technique while preserving their aspect ratio. The dataset was stored in vectors and multidimensional matrices. The parallel ANNs was successfully developed using D & C method and by using the combination phase at the end of the parallel session. For seven processors, it was observed that the time taken for a single ANNs learning was 250 hours while it only took 10 hours for parallel ANNs learning. The speedup factor affecting the performance as similarly studied by [36]. The performance of three combiner techniques has been measured. The techniques were Output Average (OA), Weighted Average (WA) and Majority Voting (MV) techniques.

The first network has been tested using ANNs with the various combiner methods with simple modification of ordering. Fig. 4 shows the mean percentage for Increased of Performance (IP) for Original reordering ANNs with three variety of network combiner. The Majority Voting technique recorded the highest increase of performance that is nearly up to 9% for seven processors compared to a single classifier. Meanwhile Output Average recorded 8.6% and Weighted Average scores 3.14% of improvement respectively.

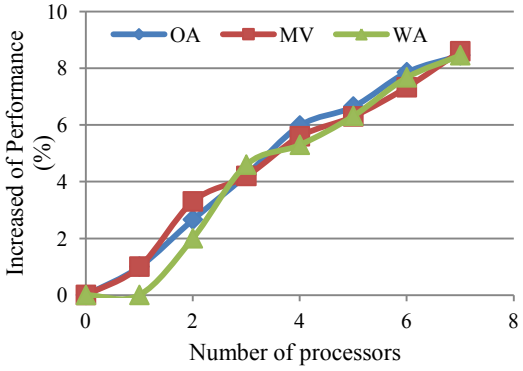

Fig. 4. Increased of Performance for original ordering

Fig. 5 shows that the mean percentage of reduced error for the Output Average technique increased nearly to 53%. Meanwhile, the Majority Voting method scores 40% error reduction and Weighted Average method scores 19%. The reduced error for this network is not more than 53% but shows the differences among each ensemble methods especially when processor increased.

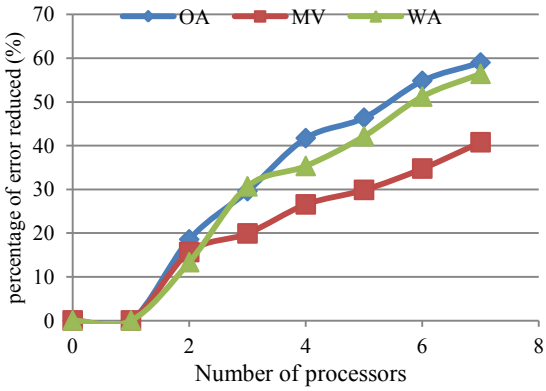

Fig. 5. Percentages of error reduction for original ordering

Bagging reordering has been tested using ANNs with the various combiner methods. Fig. 6 shows the mean percentage for Increased of Performance (IP) for Bagging reordering Artificial Neural Network with three variety of network combiner. The Output Average technique recorded the highest increase of performance that is nearly up to 10% for seven processors compared to a single classifier. Both Majority Voting and Bayesian method scores more 8% of improved performance. Meanwhile, Weighting Average method in this case increased 4% of performance.

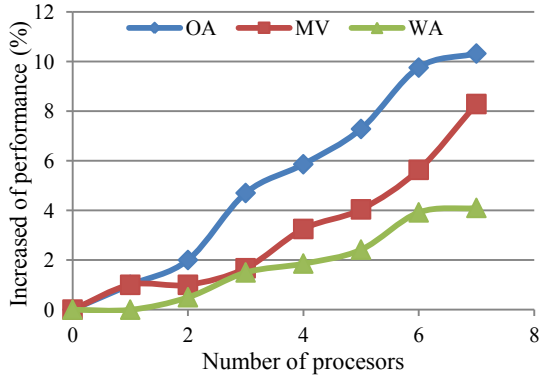

Fig. 6. Increased of Performance for Bagging reordering

Fig. 7 shows that the mean percentage of reduced error for the Output Average technique increased nearly to 69%. This technique has reduced the error in 50% from the use of single network of Bagging reordering. Meanwhile, the Majority Vote technique scores more than 48% and the Weighted Average technique scores 41%. The reduced error for Majority Voting and Weighted Average scores has been slightly similar for two to three processors. However, it shows the significant differences when the processor increased in Bagging reordering network. This result is similar to the result in [5, 24] in aspect of error reduced error when the number of classifier increased. Although it is also

shown that the Output Average technique demonstrated the better performance, both Majority Vote and Weighted Average techniques were capable of providing good results for any size of multiple ANNs with good accuracy scores.

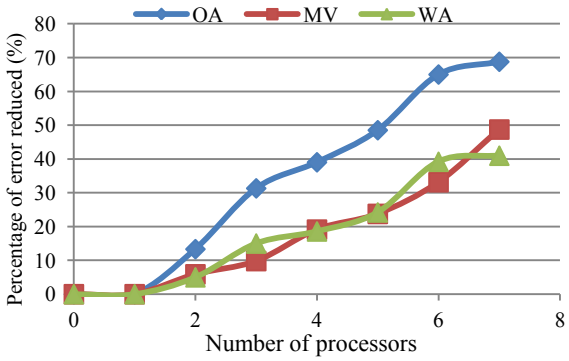

Fig. 7. The percentage of error reduced (PER) for Bagging reordering

The simple original reordering shows the increased of (IP) performance between each method compared to Output Average is so small and negative. This situation can be seen in Table II. Table II shows the differences of IP values for Output Average method across Weighted Average and Majority Voting method for simple reordering network. The performance of Output Average method improves has been improved compared to other two methods. Negative values from Table II show the facts that error for Output Average has been reduced for most number of processors when compared to other methods except for Weighted Average.

TABLE II. IP DIFFERENCES COMPARED TO OA FOR SIMPLE REORDERING

		Increased of Performance (IP)	
		WA	MV
Num of Processors	1	0	-1
	2	0.64	-0.66
	3	-0.04	0.36
	4	-0.38	-0.68
	5	-0.35	-0.33
	6	-0.53	-0.18
	7	0.14	-1.4E-14

The PER values for Original ordering networks in Table III also shows the reduction of error for other method are in small scores and negative especially when compare to Output Average method. This result shows that the Output Average outperformed other methods reordering with no improvement by other methods. Output Average method can be considered as the best ensemble method. The following investigations are about to find the significant of Output Average method compared the other methods.

TABLE III. PER DIFFERENCES COMPARED TO OA FOR SIMPLE REORDERING

		Percentage of Error Reduction (PER)	
		WA	MV
Num of Processors	1	0	0
	2	-2.9097	-18.5495
	3	-9.66243	3.875105
	4	6.57188	-15.1612
	5	-4.12638	-16.516
	6	-5.47551	-20.0724
	7	-7.394	-2.59582

Table IV shows the differences of IP for Output Average method in comparing with Weighted Average and Majority Voting method in Bagging reordering network. Small and negative values indicate that the alternative technique have not been better than Output Average technique. On the other hand, any positive IP values indicate the opposite. It can be observed that the differences IP scores with respect to Output Average are mostly small and negative. This means that the result of increased of performance percentage provide by Output Average technique is slightly better than the results provided from other methods.

TABLE IV. IP DIFFERENCES COMPARED TO OA FOR BAGGING REORDERING

		Increased of Performance (IP)	
		WA	MV
Num of Processors	1	0	-1
	2	-1	-1.5
	3	-3.03333	-3.2
	4	-2.6075	-4
	5	-3.24	-4.86
	6	-4.12	-5.83333
	7	-2.03429	-6.22

The PER values for Bagging reordering networks in Table V also shows the reduction of error for other methods is small and negative exclude the scores for Weighted Average method. Weighted Average outperforms Output Average method for some particular processors. This can be seen by the occurrence of positive numbers for Weighted Average method for four to seven processors. Output Average has been slightly improved but has been sometime not performed when compared to Weighting Average in this case. But with Majority Voting method, Output Average is slightly performed.

TABLE V. PER DIFFERENCES COMPARED TO OA FOR BAGGING REORDERING

		PERCENTAGE OF ERROR REDUCTION (PER)	
		WA	MV
Num of Processors	1	0	0
	2	-7.45098	-13.3333
	3	-21.5294	0.882353
	4	4.803922	-19.9324
	5	4.117647	-24.7686
	6	5.189706	-31.885
	7	8.937255	-27.8

The pair t-test has been applied in order to statistically compare the simple original reordering and bagging reordering multiple ANNs. This test has is conducted to verify the variability of the performance of Bagging with respect to simple reordering to rank the ensemble method. These two measurements of IP and PER have been checked whether they are independent and distributed normally.

Table VI shows the statistical result for pair t-test of IP mean value for Bagging and simple reordering using three ensemble methods. This result shows the equal variance among network for all ensemble methods. Therefore, the t-test for IP value for improved ensemble methods is statistically significant. However, there a few cases for Bagging in which most of value of α are above than 0.05. The Bagging reordering is statistically better than simple reordering of the original datasets especially for classifier that employed Output Average (OA) method. The simple reordering relates significantly for single processor and for

five processors (Weighting Average) and 6 processors (Majority Voting).

TABLE VI. T-TEST FOR MEAN OF IP FOR BAGGING VS SIMPLE REORDERING

Bagging vs Simple reordering		OA		WA		MV	
		t value	α	t value	α	t value	α
Num of Processors	1	-0.06	0.51	-0.01	0.31	-0.04	0.93
	2	0.2	0.32	0.01	0.22	0.3	0.21
	3	0.411	0.62	0.06	0.52	0.52	0.46
	4	0.510	0.81	0.12	0.41	0.67	0.43
	5	0.621	0.11	-0.59	0.31	1.03	0.29
	6	0.97	0.90	0.87	0.10	-0.62	0.65
	7	1.03	0.27	1.03	0.67	1.56	0.53

Table VII shows the statistical result for pair t-test of PER mean for Bagging and simple original reordering network. This result also shows the equal variance among network for three ensemble methods. The t-test values of PER for improved ensemble methods is statistically significant. Bagging reordering technique in this case has been statistically dispersed compared to simple reordering for error reduction but with some worse score in all methods especially for Majority Voting.

TABLE VII. T-TEST FOR MEAN OF PER FOR BAGGING VS SIMPLE REORDERING

Bagging vs Simple reordering		OA		WA		MV	
		t value	α	t value	α	t value	α
Num of Processors	1	-0.01	0.31	-0.07	0.43	0.30	0.41
	2	0.22	0.22	-0.06	0.21	-0.07	0.52
	3	0.42	0.52	0.07	0.46	-0.14	0.12
	4	0.79	0.41	0.34	0.43	1.23	0.31
	5	0.921	0.31	0.7	0.29	1.35	0.51
	6	-0.62	0.10	1.1	0.65	-0.07	0.60
	7	1.03	0.67	1.25	0.53	1.03	0.67

The result shows that, with Bagging reordering pattern diversity, the large dataset classification using multiple ANNs improves with better performance with the use of Output Average method. This finding is consistent with the finding of past research by [37] which suggest output averaging can outperform other methods. This result is supported by the t-test that aligned with first result. On the other hand, the t-test scores shows smaller error reduction for Bagging reordering compared with most processors used for simple reordering. The statistical scores are average among all methods for reduction of error where t-test score are negatives for some processors.

Generally, the results provided by proposed ensemble strategy are significant with the use of reordering technique where there are some specific cases where a combiner performs better on a few subsets. This finding is consistent with past study by [21], which there is no combiner technique that outperforms other combiner techniques consistently. In addition, the result that have been obtained shared a similar view of combiner technique comparison with what has been demonstrated in [32]. In this paper, two or more combiners can reach a similar general value while one is more suitable for a set of data, and the other one fits better on another subset of classification problem. The overall finding from this research suggests that, the large data is better managed in multiple ANNs training where the ANNs workload can be reduced whilst maintaining its accuracy.

VII. CONCLUSION

In this paper, a new strategy is proposed which is based on diversity of training data and the ensemble combiner methods. The alternative representation of ANNs for large data is also suggested. This strategy that has been conducted for ANNs tasks to manage large data associated with recognition tasks. The dataset has been normalized to ensure the smooth training for each ANNs. The partitioning with the reordering techniques has been introduced to distribute and shuffle the order of training pattern. The combination of multiple ANNs outputs has been worked in compiling all the outputs of from various ANNs in multiple processors to get the best result.

Employing several techniques in classification task for large data requires several integrated strategies. The partitioning and distributing the different data vector to different ANNs processor as a good consensus to delegate tasks from large data is practical. The use of reordering employed herein is the other alternative in creating diversity among ANNs processor show performance improvement among networks tested. The ensemble method to combine all the output from ANNs processors as a final consensus function result representation is a worthwhile task for large data. The sense of using reordering is worthwhile for the large datasets with significant performance. Although, Majority Voting and Weighted Average have been provided good general result and might perform better in specific cases.

The proposed strategies show significant improvement for the large dataset in classification task for generalization ability and accuracy contribute by multiple classifiers. The results demonstrate that most of strategy are dataset dependent but good for large dataset. However, this result also depends on normalization for both input and output. Otherwise, it performs quite worst than single ANNs. The need for variety of normalization techniques should be imposed in future. The study will be extended to consider on more combiners including the selection and pooling of output aggregation technique.

REFERENCES

[1] L. Xiaoou, J. Cervantes, and W. Yu, "Fast Classification using Large Datasets via Random Selection Clustering and Support Vector Machines", *Intelligent Data Analysis,* 12:16, 2012.

[2] L. Bottou, "Large-scale Machine Learning with Stochastic Gradient Descent", in *19th International Conference on Computational Statistics*, Paris: Physica-Verlag HD, 2010, pp. 177-186.

[3] L. Bottou and Y. L. Cun, "Large Scale Online Learning", *Advances in neural information processing systems,* 16:217, 2004.

[4] N. C. Oza and K. Tumer, "Classifier Ensembles: Select Real-world Applications", *Information Fusion,* 9:1, pp. 4-20, 2008.

[5] J. Torres-Sospedra, C. Hernández-Espinosa, and M. Fernández-Redondo, "Introducing Reordering Algorithms to Classic Well-Known Ensembles to Improve Their Performance", in *Neural Information Processing*. 7063: Springer Berlin Heidelberg, 2011, pp. 572-579.

[6] V. Turchenko, L. Grandinetti, and A. Sachenko, "Parallel Batch Pattern Training of Neural Networks on Computational Clusters", in *International Conference on High Performance Computing and Simulation (HPCS)*, Madrid, 2012, pp. 202-208.

[7] A. R. M. Kattan, R. Abdullah, and R. A. Salam, "Training Feed-Forward Neural Networks Using a Parallel Genetic Algorithm with the Best Must Survive Strategy", in *International Conference on Intelligent Systems, Modelling and Simulation*, Liverpool, 2010, pp. 96-99.

[8] S. Babii, "Performance Evaluation for Training a Distributed BackPropagation Implementation", in *4th International Symposium on Applied Computational Intelligence and Informatics*, Timisoara 2007, pp. 273-278.
[9] R. Polikar, "Ensemble Learning", in *Ensemble Machine Learning: Methods and Applications*: Springer US, 2012, pp. 1-34.
[10] N. De Silva and N. Thurairajah, "Architecture of Ensemble Neural Networks for Risk Analysis", in *ASC 48th Annual International Conference*, Birmingham City University, U.K. , 2012.
[11] S. Cheng, L. Li, D. Chen, and J. Li, "A Neural Network Based Ensemble Approach for Improving the Accuracy of Meteorological Fields Used for Regional Air Quality Modeling", *Journal of Environmental Management,* 112:1, pp. 404-414, 2012.
[12] X. Ceamanosa, B. Waske, J. A. Benediktsson, J. Chanussot, M. Fauvele, and J. R. Sveinsson, "A Classifier Ensemble Based on Fusion of Support Vector Machines for Classifying Hyperspectral Data", *International Journal of Image and Data Fusion,* 1:3, pp. 293–307, 2010.
[13] Y. Bi, "The Impact of Diversity on the Accuracy of Evidential Classifier Ensembles", *International Journal of Approximate Reasoning,* 53:4, pp. 584-607, 2012.
[14] M. Mohamad, M. Y. M. Saman, and M. S. Hitam, "Divide and Conquer Approach in Reducing ANN Training Time for Small and Large Data", *Journal of Applied Sciences,* 13:1, pp. 133-139, 2013.
[15] D. C. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column Deep Neural Networks for Image Classification", presented at the *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* Providence, RI 2012.
[16] B. M. Wilamowski, "Neural Network Architectures and Learning Algorithms", *IEEE Industrial Electronics Magazine,* 3:4, pp. 56-63, 2009.
[17] J. K. Bradley and R. Schapire, "Filterboost: Regression and Classification on Large Datasets", in *Advances in neural information processing systems*, Vancouver, 2008, pp. 185-192.
[18] J. Cervantes, X. Li, W. Yu, and K. Li, "Support Vector Machine Classification for Large Data Sets via Minimum Enclosing Ball Clustering", *Neurocomputing,* 71:4, pp. 611-619, 2008.
[19] L. Yue, T. Zaixia, and Z. Bofeng, "Computational Grid Based Neural Network Ensemble Learning Platform and Its Application", in *International Conference on Management of e-Commerce and e-Government (ICMECG '09)*, Nanchang, 2009, pp. 176-181.
[20] L. Jain, M. Sato-Ilic, M. Virvou, G. Tsihrintzis, V. Balas, C. Abeynayake, and T. Windeatt, "Ensemble MLP Classifier Design", in *Computational Intelligence Paradigms*. 137: Springer Berlin Heidelberg, 2008, pp. 133-147.
[21] Z.-H. Zhou and N. Li, "Multi-information Ensemble Diversity", in *Multiple Classifier Systems*. 5997, Cairo: Springer Berlin Heidelberg, 2010, pp. 134-144.
[22] M. Salkhordeh Haghighi, A. Vahedian, and H. Sadoghi Yazdi, "Making Diversity Enhancement Based on Multiple Classifier System by Weight Tuning", *Neural Processing Letters,* 35:1, pp. 61-80, 2012/02/01 2012.
[23] H. Alizadeh, H. Parvin, and S. Parvin, "A Framework for Cluster Ensemble Based on a Max Metric as Cluster Evaluator", *IAENG International Journal of Computer Science,* 39:1, pp. 10-19, 2012.
[24] L. Kuncheva and J. Rodríguez, "A Weighted Voting Framework for Classifiers Ensembles", *Knowledge and Information Systems,* 38:2, pp. 259-275, 2012/12/01 2014.
[25] M. W. Shields and M. C. Casey, "A Theoretical Framework for Multiple Neural Network Systems", *Neurocomputing,* 71:7–9, pp. 1462-1476, 2008.
[26] S. Bhagat, "Divide and Conquer Strategies for MLP Training", presented at the *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Vancouver, 2006.
[27] H. Parvin, H. Alinejad-Rokny, and S. Parvin, "Divide and Conquer Classification", *Australian Journal of Basic and Applied Sciences,* 5:12, pp. 2446-2452, 2011.
[28] K. Li and Y. Han, "Study of Selective Ensemble Learning Method and its Diversity Based on Decision Tree and Neural Network", in *Chinese Control and Decision Conference (CCDC)*, Xuzhou, 2010, pp. 1310-1315.
[29] K. W. Hsu and J. Srivastava, "Diversity in Combinations of Heterogeneous Classifiers", in *Advances in Knowledge Discovery and Data Mining*. 5476: Springer Berlin Heidelberg, 2009, pp. 923-932.
[30] H. K. Butler, "The Relationship Between Diversity and Accuracy in Multiple Classifier Systems," Air Force Inst of Tech Wright-Patterson AFB Graduate School of Engineering & Managementt, 2012.
[31] C. Fernández, C. Valle, F. Saravia, and H. Allende, "Behavior Analysis of Neural Network Ensemble Algorithm on A Virtual Machine Cluster", *Neural Computing & Applications,* 21:3, pp. 535-542, 2012.
[32] B. Minaei-Bidgoli, H. Parvin, H. Alinejad-Rokny, H. Alizadeh, and W. F. Punch, "Effects of Resampling Method and Adaptation on Clustering Ensemble Efficacy", *Artificial Intelligence Review,* 41:1, pp. 1-22, 2014.
[33] C.-L. Liu and H. Fujisawa, "Classification and Learning Methods for Character Recognition: Advances and Remaining Problems Machine Learning in Document Analysis and Recognition", in *Machine Learning in Document Analysis and Recognition* vol. 90, S. Marinai and H. Fujisawa, Eds., ed: Springer Berlin Heidelberg, 2008, pp. 139-161.
[34] P. Hanafizadeh, E. S. Parvin, P. Asadolahi, and N. Gholami, "Ensemble Strategies to Build Neural Network to Facilitate Decision Making", *Journal of Industrial Engineering International,* 4:6, pp. 32-38, 2008.
[35] T. Windeatt, "Accuracy Diversity and Ensemble MLP Classifier Design", *IEEE Transactions on Neural Networks,* 17:5, pp. 1194-1211, 2006.
[36] N. P. Khanyile, J.-R. Tapamo, and E. Dube, "An Analytic Model for Predicting the Performance of Distributed Applications on Multicore Clusters", *IAENG International Journal of Computer Science* 39:3, pp. 312-320, 2012.
[37] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An Overview of Ensemble Methods for Binary Classifiers in Multi-class Problems: Experimental Study on One-vs-One and One-vs-All Schemes", *Pattern Recognition,* 44:8, pp. 1761-1776, 2011.

Mumtazimah Mohamad (M'2013) is from Malaysia. She received her BSc. (2000) in Information Technology from University Kebangsaan Malaysia, Malaysia, and her MSc in Software Engineering (2005) from University Putra Malaysia. She is currently towards her PhD degree in computer science at University Malaysia Terengganu, Malaysia. Her research interests include ensemble learning for large datasets in parallel processing.

Prof. Mohd Yazid Md Saman received his B.Sc (1981) in computer science from Essex University, U. K and M. Sc.(1987) in computer science from University Teknologi Malaysia. He received his Ph.D (1993) from Loughborough University of Technology, UK, also in computer science. His joined University Putra Malaysia in 1981 and had employed as an Associate Professor in 1997. He then joins University Malaysia Terengganu in 2001 where he is currently employed as a Professor since 2004. His research spans various aspects of technology and engineering, software development, distributed and parallel computing, computer networks, simulation and performance modeling.

Dr Muhammad Suzuri Hitam received his B.Tech. Hons. in Quality Control and Instrumentation Technology from University Sains Malaysia, Malaysia and his Ph.D from Leeds University, U.K. He is currently an associate professor and the director of Information Technology Management Center, University Malaysia Terengganu, Malaysia. His main research interests are in image processing, soft-computing and robotics.