Hierarchical Incremental Class Learning with Reduced Pattern Training

[1]Sheng-Uei Guan, [2]Chunyu Bao [2]and Ru-Tian Sun

[1]School of Engineering and Design

Brunel University, Uxbridge, Middlesex, UB8 3PH, UK

[2]Department of Electrical and Computer Engineering

National University of Singapore

10 Kent Ridge Crescent, Singapore 119260

Abstract—Hierarchical Incremental Class Learning (HICL) is a new task decomposition method that addresses the pattern classification problem. HICL is proven to be a good classifier but closer examination reveals areas for potential improvement. This paper proposes a theoretical model to evaluate the performance of HICL and presents an approach to improve the classification accuracy of HICL by applying the concept of Reduced Pattern Training (RPT). The theoretical analysis shows that HICL can achieve better classification accuracy than Output Parallelism [1]. The procedure for RPT is described and compared with the original training procedure. RPT reduces systematically the size of the training data set based on the order of sub-networks built. The results from four benchmark classification problems show much promise for the improved model.

Index Terms—classifier systems, Output Parallelism, instance selection, reduced pattern training, hierarchical learning

1 Introduction

Neural network (NN) is a computational module, which simulates the human brain behavior in a fundamental manner. NN classifiers have some merits over traditional pattern recognition systems with respect to the capability of adaptive learning, generalization ability with noisy or sparse learning data, and feasibility for hardware implementation [2]. Current NN classifiers suffer from major drawback of high internal interference because of the strong coupling among their hidden-layer weights [3]. Usually there are regions in the class feature space, which show high overlapping due to the resemblance of two or more classes. Meanwhile, there are other regions, which show little or even no overlapping, according to the "uniqueness" of the classes therein. High coupling among hidden nodes will then result in partial over- and under-learning in the different regions [4]. Enlarging the network, increasing the number and quality of training samples, and techniques for avoiding local minima will not "stretch" the learning capacities of the NN classifier beyond a certain limit as long as hidden nodes are tightly coupled, and hence, "cross-talking" during learning [5] and causing "catastrophic interferences" [6].

A modular neural network attempts to solve these problems via task decomposition. It generally decomposes large-sized tasks into several sub-tasks; each one is handled by a simple, fast and efficient module. Then, sub-solutions are integrated via a multi-module decision making strategy [2]. Many task decomposition methods have been proposed in the literature ([5,7-19]). These decomposition methods can be classified into two major categories:

domain decomposition and class decomposition.

Domain Decomposition is a category of decomposition methods based on the characteristics of input data space. Sharkey found that it is effective to partition the input data space into several subspaces and train each module to fit the local data in each subspace [8]. Jacobs et al. proposed that with the mixture of expert architecture, expert networks could be used to learn subspaces and then integrated via a gating network [3]. In the multi-sieving neural network, patterns are classified by a rough sieve at the beginning and they are reclassified further by finer ones in the subsequent stages [11].

Another category of decomposition methods based on the characteristics of output space is *Class Decomposition*. A common class decomposition approach is to split a K-class problem into K two-class sub-problems and each module is trained to learn a "yes or no" problem for one class. Each two-class sub-problem is learnt independently [12]. Hence, each sub-problem forms a module that is independent from the others. The final overall solution is obtained by integrating all the trained modules' solutions together. A powerful extension to the above class decomposition method, output parallelism, is proposed in [1][19]. Using output parallelism, a complex problem can be divided into several sub-problems, each of which is composed of the whole input vector and a fraction of the output vector. Each module is responsible for producing a fraction of the output vector of the original problem. These modules can be grown and trained in parallel.

Task decomposition methods reduce the internal interference among hidden layers, consequently, improve performance and accuracy. However, the above task decomposition methods do not accommodate sharing of information between modules. The correlation between classes or sub-networks is ignored. A sub-network can only use the local information restricted to the classes involved in it. Sub-networks cannot exchange information already learnt by them with each other. The global information between classes that can be positive to the learning of sub-networks is missing as well as internal interference between them.

We adopt a task decomposition approach named hierarchical incremental class learning (HICL) proposed initially in 2002 [13]. In this approach, a K-class problem is divided into K sub-problems. The sub-problem is learnt sequentially in a hierarchical structure with K sub-networks. Each sub-network takes the output from the sub-network immediately below it. And this output is fed into the sub-network above it as shown in Figure 1.

In this paper, we present a theoretical model to evaluate the performance of the HICL network. Output Parallelism [1] can be regarded as a typical task decomposition approach. Through the analysis, we can see HICL performs better than Output Parallelism. Moreover, we propose a new method to improve the performance of HICL network by applying the concept of Reduced Pattern Training (RPT). We notice that later sub-networks in HICL do not need to recognize classes already classified by earlier sub-networks. With RPT, systematic instance selection is used to diminish subsets of the original training data set for each sub-network. For the purpose of comparison with HICL, we call the improved HICL version as RPT-HICL in the following.

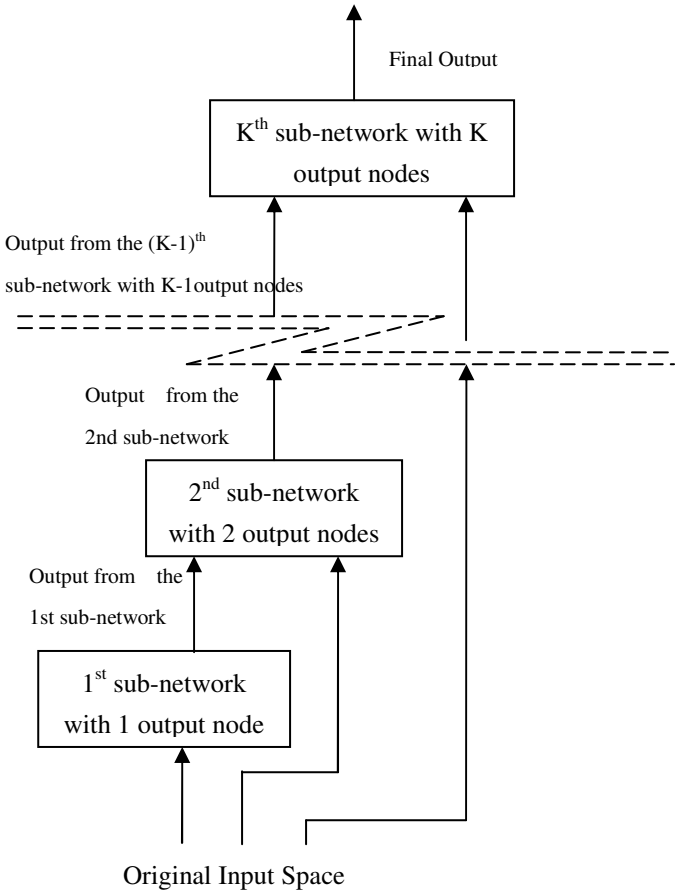


Figure 1: Overview of HICL

In this paper, we introduce the structure of HICL in section 2. In section 3, the theoretical model of HICL is introduced and comparison between HICL and Output Parallelism is presented. The motivation for RPT is given is section 4. In section 5, we explain why RPT-HICL could achieve better classification accuracy than the original HICL. In section 6, the experimental results of HICL with RPT are presented and discussed. Finally, the conclusion is drawn.

2 HICL in its Original Form

In the original HICL method [13], the original *K*-class problem is solved using a hierarchical modular neural network (HMNN) consisting of *K* sub-networks. After a sub-network is added and trained, a new sub-network is constructed on top of it. The new sub-network accepts the output from the old sub-network, together with the original input as its input. The output space of the new sub-network is one dimension larger than that of the old sub-network. For classification problems, this means the output space of the new sub-network includes one more class than the old sub-network. It should be mentioned that in the RPT-HICL method, the new sub-network will not accept the output from the old sub-network and it will just accept some subset of patterns form the old one as will be described later.

In HICL, the functionality of the first sub-network is to classify the training samples belonging to the first output class. This is a localized computation associated with the output attribute representing the specified class only, which is the same as one single module in class decomposition. Because internal interference is removed, the output from this sub-network tends to be more accurate. The functionalities of sub-networks other than the first one are more complex. Because each sub-network needs to deal with more than one class simultaneously, the correlation between different classes is taken into consideration automatically. There are two functions for each sub-network.

- The minor function is to perform reclassification to the classes learnt previously. If the lower sub-networks produce no error, they provide the present sub-network with linear-separable inputs. Due to the strong bias of these inputs, the reclassification process

is most likely to follow the decision boundaries built by lower sub-networks and simply repeats the results from them.

- The major function is to classify samples belonging to the newly added class from all the other classes. This function is a local computation relative to the new class in the sub-network, which is the same as a sub-network in class decomposition. However, it should be noted that some of the classes are already classified in the lower sub-networks. Again, if this pre-classified information contains no error, it is not necessary to classify the patterns belonging to the new class from those belonging to the classes learnt in the lower sub-networks.

3 Theoretical Analysis for HICL and Output Parallelism

Here we present a model to compare the performance of the HICL network and the Output Parallelism network [1]. We mentioned that the major function of a sub-network is to classify the samples belonging to the newly added class from all the other classes. The reclassification to the classes learnt previously has minor contribution to the final classification accuracy. So we ignore the minor function of the sub-networks. We consider that a sub-network just accepts the outputs from the earlier one and simply passes them to the next sub-network without any change.

Assume our classification problem has r output classes. Using HICL, it is divided into r sub-networks and each sub-network classifies a new class from the rest. Before RPT is

applied, each sub-network is learned using all the training patterns. After the whole network is learned, we use a data set to evaluate the classification accuracy of the HICL network which is called test set. Assume there are N instances in the test data set and N_j represents the pattern number in class j (here $j=1, 2, \ldots, r$). Thus, define p_j to represent the probability of error when sub-network j processes the test data set. For the first sub-network, it processes all the instances and chooses instances belonging to the first class. After that it passes the results to the second sub-network. Now it is the second sub-network's turn to classify these instances. It accepts the results from the first sub-network. For those instances that have been classified into the first class, it simply passes them through. The second sub-network tries to choose the instances belonging to the second class from the remaining data set. After that it passes the results to the third sub-network. This process is continued until all the instances are classified.

Define E_j as the number of the instances which are wrongly classified by the sub-network j. We note that if an instance is misclassified in a lower sub-network, the upper sub-networks are not likely to correct this error. We have:

$$E_1 = N \cdot p_1 \tag{1}$$

$$E_2 = \left[N - N_1 - (N - N_1) \cdot p_1\right] \cdot p_2 = (N - N_1)(1 - p_1) p_2 \tag{2}$$

In most situations, p_j is a small number compared with 1. Thus, we ignore the terms $1 - p_1$ and we get

$$E_2 \approx (N - N_1) p_2 \tag{3}$$

Similarly, In sub-network j

$$E_j \approx (N - N_1 - N_2 - \cdots - N_{j-1}) p_j \tag{4}$$

By integrating the results from all the sub-networks, we have

$$E^{HICL} = \sum_{j=1}^{r} E_j = \sum_{j=1}^{r} \left(N - \sum_{k=1}^{j-1} N_k \right) \cdot p_j \tag{5}$$

Next we compare the classification error of the HICL network with that of the Output Parallelism network. Assume we also divide the output space into r sub-networks using Output Parallelism. Each sub-network in Output Parallelism tries to classify the instances of some specific class from all the instances. Thus, the sub-networks in Output Parallelism have similar roles compared to the sub-networks in HICL. We note that each sub-network in HICL has the same task as the corresponding sub-network in Output Parallelism and they are learnt using the same training data set. So the probability of error of sub-network *j* processing the test set in Output Parallelism is equal to or at least is close to that in HICL. For simplicity, we consider they are equal. Thus, p_j also represents the probability of error for sub-network *j* in Output Parallelism processing the test set.

In Output Parallelism, define F_j as the number of the test instances which are wrongly-classified by sub-network *j*.

$$F_j = N \cdot p_j \tag{6}$$

Then we integrate the results from all the sub-networks. We can not simply add all the F_j, because some instances may be classified wrongly in two or more sub-networks. So the final results will be as follows:

$$\begin{aligned} E^{OP} = N\big[& p_1 + p_2 + \cdots + p_r - (p_1 p_2 + p_1 p_3 + \cdots + p_{(r-1)} p_r) \\ & + (p_1 p_2 p_3 + \cdots + p_{(r-2)} p_{(r-1)} p_r) - \cdots + (-1)^{r-1} (p_1 p_2 \cdots p_r) \big] \end{aligned} \tag{7}$$

We mentioned earlier that p_j is a small number in most situations. While the term $N \cdot (p_1 + p_2 + \cdots + p_r)$ palys a major role in E^{OP}. So we ignore the influence from the other terms. We have

$$E^{OP} \approx N \cdot (p_1 + p_2 + \cdots + p_r) \quad (8)$$

It is obvious that E^{OP} is larger than E^{HICL}. Intuitively, most sub-networks in HICL need not process as many test instances as those in Output Parallelism. Thus, the number of instances classified wrongly is reduced using HICL.

4 Applying Reduced Pattern Training in HICL

HICL facilitates information transfer regarding classes between sub-networks during training. The later sub-networks can obtain learnt information from the earlier sub-networks. This aids learning in the upper sub-networks. However, there are areas in which the model can be improved based on two observations.

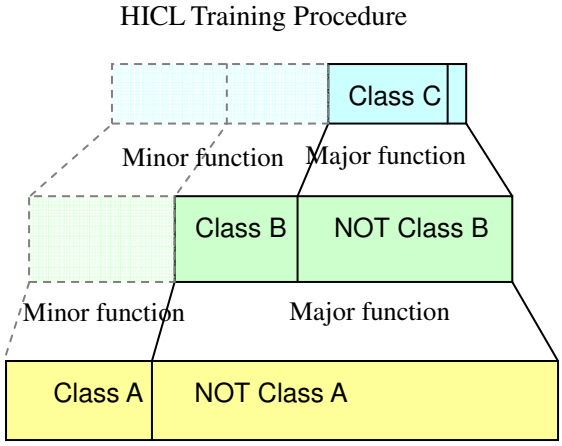
Observation 1: Learning in HICL involves every sub-network being trained with the full training data; this can be changed to leverage on the advantages of reduced training data.

Observation 2: Inherited partitions are unlikely to change thus the resources used for reclassification in subsequent sub-networks could possibly be put to better use. Moreover, inherited partitions have more inputs than the original problems. These extra inputs may

introduce *internal interference* to the following sub-networks.

Using these two observations, the improved training procedure – RPT-HICL is proposed. In RPT-HICL, systematic instance selection is used to produce a training subset from the original training data set for each sub-network. The higher in the hierarchy the sub-network is, the smaller the training set. Later sub-networks do not need to recognize classes already classified by earlier sub-networks. That is to say, the minor function of sub-networks in HICL is done away with.

Hence in the modified training procedure, RPT-HICL, only the first sub-network is trained with all patterns. For later sub-networks, they are trained with *a subset of the previous sub-network's training set*. Patterns that belong to classes already classified in previous sub-networks are excluded. As a result of using reduced training sets, training will be increasingly efficient as the network is built.



Sub-network	Training Patterns	
	Original	RPT
3	Class A	
	Class B	
	Class C	Class C
2	Class A	
	Class B	Class B
	Class C	Class C
1	Class A	Class A
	Class B	Class B
	Class C	Class C

Figure 2: A three-class HICL training example (RPT: reduced training set)

A three-class example (Figure 2) is given to illustrate the difference in the training procedure. Given Class A, B and C, there will be three sub-networks built altogether. For the original HICL method all three sub-networks are trained with all training patterns. The first sub-network will separate Class A from the rest. The second sub-network will separate Class A and Class B from the rest. The third sub-network will separate all three classes. The results are collated from the output nodes of the last sub-network.

For the RPT-HICL method, the first sub-network is trained with all patterns. The second sub-network is trained with fewer patterns – those belonging to Class A are excluded (sub-network 1 has already learnt to identify Class A and a repetition of the job is not desired). The third sub-network is trained with even fewer patterns – those belonging to Class A and Class B are excluded. A slight modification is made to the collation of results. For RPT-HICL, the value from the last output node from each sub-network is saved. The results from these nodes are merged to form the output space (Figure 3).

The reason for this is because later sub-networks are not trained to recognize classes identified by earlier sub-networks. Hence the identification of any one test pattern must be checked against each sub-network in order. Once an output is generated by any sub-network in the hierarchy, the test pattern will be deemed as belonging to that sub-network and will not be presented to the rest of the network.

From Figure 3, we can see that only the last output node from each sub-network will contribute to the new output space in RPT-HICL. This last output node indicates whether the

pattern belongs to this sub-network (class) or not.

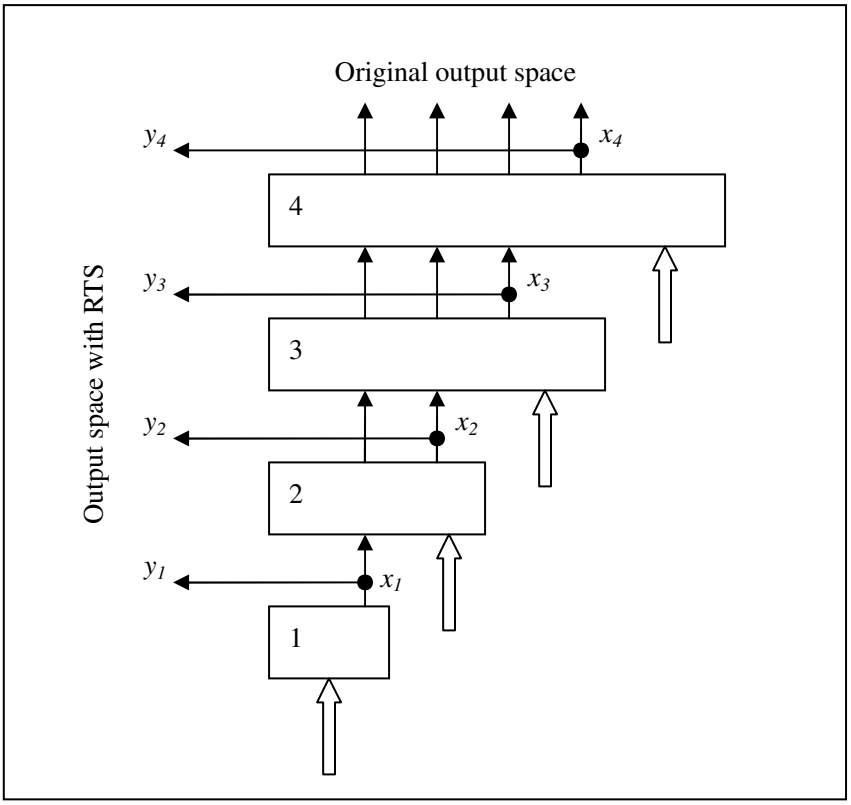


Figure 3: Modification to the collation of results

When a test pattern is presented to the network, it first enters sub-network 1. Sub-network 1 will have an output, x_1, of value between o_{max} and o_{min}. With a simple threshold function, we can decide whether the pattern belongs to Class 1 or NOT Class 1. Mathematically described,

$$y_1 = \begin{cases} 1, & x_1 \geq 0.5 \\ 0, & x_1 < 0.5 \end{cases} \tag{9}$$

If the pattern does not belong to Class 1, it will be presented to sub-network 2. y_2 can be evaluated in the same way. And so on.

5 Explaining Why RPT-HICL is Better than Original HICL

Now we explain why RPT-HICL could lead to better classification accuracy compared with

the original HICL. In the HICL methods, each sub-network tries to solve a two-class problem. Sub-network *j* tries to decide if a pattern belongs to class *j* or not. Normally, the number of the patterns belonging to class *j* is far smaller than that not belonging to class *j*. In other words, sub-network *j* tries to solve a two-class problem in which patterns have imbalanced distribution. Now we show that imbalanced pattern distribution may bring adverse effect for the training and removing some patterns from the crowded side can improve the classification accuracy of neural networks if useful information is not removed.

Consider a two-class classification problem and one class has much more patterns than the other one. Assume the two classes could be divided by a curve or a curved surface in the input space.

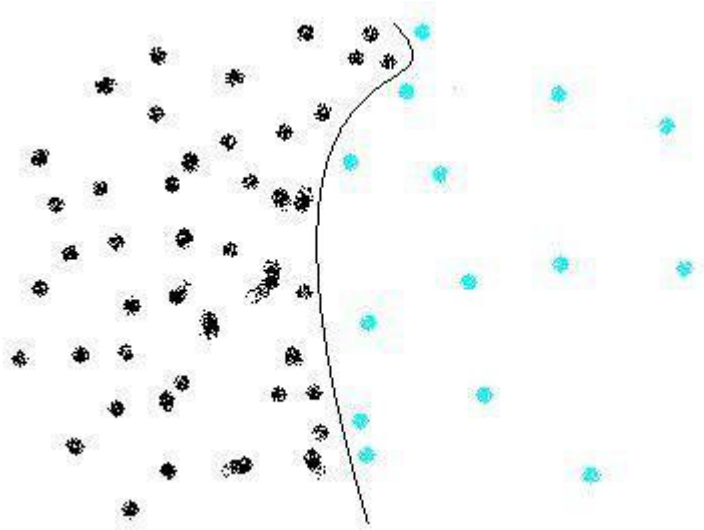


Figure 4: The real border in a two-class problem with imbalanced pattern distribution

Figure 4 shows the pattern distribution of the training set and the real boundary between two classes. In back-propagation NNs, the training objective is to minimize the error *E* (Of course,

early stopping criteria [1][23-24] is used to prevent from overtraining), instead of finding (or approaching) the real border between two classes. Here

$$E = 100 \cdot \frac{o_{\max} - o_{\min}}{K \cdot P} \sum_{p=1}^{P} \sum_{k=1}^{K} (o_{pk} - t_{pk})^2 \qquad (10)$$

where $o_{\max}$ and $o_{\min}$ are the maximum and minimum values of output coefficients in the problem representation. Logistic function (Equation 11) is used as the activation function. *P* is the pattern number and *K* is the output number in NN. In a two-class NN, we can choose *K=1*.

$$\varphi(v) = \frac{1}{1 + \exp(-v)} \qquad (11)$$

For an unseen pattern, $\varphi(v)$ is computed. If $\varphi(v) \geq 0.5$, it would be considered belonging to class 1, otherwise class 2. In other words, $\varphi(v) = 0.5$ is the computed border.

Equation (10) can be rewritten as:

$$E = 100 \cdot \frac{o_{\max} - o_{\min}}{P} \sum_{p_1=1}^{P_1} (o_{p_1} - t_{p_1})^2 + 100 \cdot \frac{o_{\max} - o_{\min}}{P} \sum_{p_2=1}^{P_2} (o_{p_2} - t_{p_2})^2 \qquad (12)$$

where P_1 is the number of patterns in class 1 and P_2 is the number of patterns in class 2.

Assume that the left side in Figure 4 consists of patterns in class 1. In order to minimize the first term in Equation 12, the patterns in class 1 would like the computed border to be far away from them, in other words, they try to push the border to the right. Similarly, the patterns in class 2 try to push the border away to the left, to minimize the second term in Equation 12. According to our assumption, there are much more patterns in class 1 than those in class 2. The patterns in class 1 will have dominated influence in the error function *E*.

Therefore, the computed border would shift to the right side of the actual border.

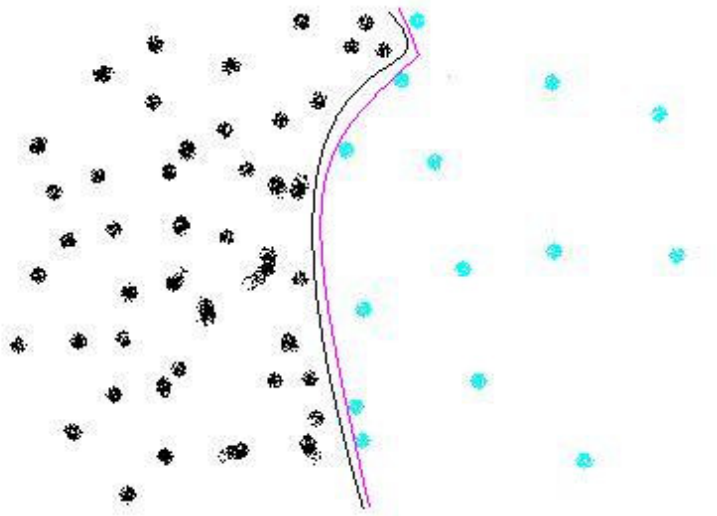


Figure 5: The computed border is on the right side of the real border.

In Figure 5, the light-colored line represents the computed border. The training error *E* with respect to the computed border is smaller than the one with respect to the actual border, because there are much more patterns sitting in the left side of the border. Obviously, the light-colored line is biased. When unseen testing patterns are presented to the trained NN, the NN may classify wrongly patterns in class 2 into class 1 and increase the classification error of the test set. If we remove some patterns from class 1, the computed border would be closer to the real one.

We examine the training patterns in those HICL sub-networks. Later sub-networks need not be trained to recognize classes identified by earlier sub-networks, so the patterns belonging to the classes identified by earlier sub-networks are useless. Removing these patterns will not lose useful information. Moreover, it can improve the classification accuracy. Thus, RPT-HICL networks could achieve better classification accuracy than the original HICL networks.

6 Experiments and Analysis

6.1 Experimental Scheme

Constructive Backpropagation (CBP) algorithm was used to train the network in the experiments [20]. CBP can reduce the excessive computational cost significantly and it does not require any prior knowledge concerning decomposition. In this paper, RPROP is used with the following parameters*:* $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_0 = 0.1$, $\Delta_{max} = 50$, $\Delta_{min} = 1.0e\text{-}6$, with initial weights selected from *–0.25…0.25* randomly [21]. In order to avoid large computational cost and overfitting, a method called early stopping based on validation set is used to set the stopping criteria [1][23-24] .

The set of available patterns is divided into three sets: a training set is used to train the network, a validation set is used to evaluate the quality of the network during training and to measure overfitting, and a test set is used at the end of training to evaluate the resultant network. The size of the training, validation, and test set is 50%, 25% and 25% of the problem's total available patterns.

Four benchmark classification problems, namely Glass, Segmentation, Vowel and Pen-Based Recognition were used to evaluate the performance of the network. These classification problems were taken from the PROBEN1 benchmark collection [22] and University of California at Irvine (UCI) repository of machine learning database. In the set of experiments undertaken, the classification problems were conducted 20 times. All the hidden units and output units use the sigmoid activation function and E_{th} is set to 0.1. When a hidden unit

addition was required, 8 candidates were trained and the best one selected. All the experiments were simulated on a Pentium IV – 2.4GHZ PC. The sub-problems were solved sequentially and the CPU time expended was recorded respectively.

In the set of experiments undertaken, simulations were run for three different orders of classes for each of the four problems. Although there are many permutations for each problem set, we limited our experiments to just three different ordering configurations due to time and resource constraints.

6.2 Experimental Results

A. Glass

This data set is used to classify glass types. The data set consists of 9 inputs, 6 outputs, and 214 patterns (divided into 107 training patterns, 54 validation patterns, and 53 test patterns). The patterns were normalized and scaled so that each component lies within [0, 1].

Ordering of classes	Training Procedure	Training Time (s)	Classification Error (%)
none	Output Parallelism	25.1	36.1321
5,4,3,6,1,2	Original HICL	62.4	35.6604
	RPT-HICL	**24.9**	**22.4528**
6,1,4,5,2,3	Original HICL	91.4	34.3396
	RPT-HICL	**13.3**	**25.0943**
1,2,3,4,5,6	Original HICL	102.5	34.5283
	RPT-HICL	**16.1**	**27.7358**

Table 1: Experimental results for Glass problem

From Table 1, we can see that the HICL networks in different ordering achieved better classification accuracy compared to the Output Parallelism network. It matches with our

analysis in Sec. 2. We also find that RPT-HICL has significant improvement on classification accuracy for the orders tested in this problem. Also training time is decreased. This problem shows an average increase in classification accuracy of 27% using RPT-HICL compared to that using the original HICL.

B. Segmentation

This data set consists of 18 inputs, 7 outputs, and a total of 2310 patterns (1155 training patterns, 578 validation patterns, and 577 test patterns). The patterns were normalized and scaled so that each component lies within [0, 1].

Ordering of classes	**Training Procedure**	**Training Time (s)**	**Classification Error (%)**
none	Output Parallelism	1719.6	5.18198
7,2,6,1,3,4,5	Original HICL	5660.8	3.37955
	RPT-HICL	**1513.3**	**2.27036**
7,2,6,1,3,5,4	Original HICL	5433.1	3.39688
	RPT-HICL	**1459.6**	**2.54766**
1,2,3,4,5,6,7	Original HICL	2975.3	4.26343
	RPT-HICL	**1585.5**	**2.68631**

Table 2: Experimental results for Segmentation problem

From Table 2, we can see that the HICL networks achieved better classification accuracy compared to the Output Parallelism network. It matches with our analysis in Sec. 2. We also find that RPT-HICL has improvement on classification accuracy for all the orders tested in this problem. Also training time is decreased. This problem shows an average increase in classification accuracy of 42% using RPT-HICL compared to that using the original HICL.

C. Vowel

The input patterns of this data set are 10 element real vectors representing vowel sounds that belong to one of 11 classes. It has 990 patterns in total (they were divided into 495 training patterns, 248 validation patterns, and 247 test patterns). The patterns were normalized and scaled so that each component lies within [0, 1].

Ordering of classes	**Training Procedure**	**Training Time (s)**	**Classification Error (%)**
none	Output Parallelism	513.9	25.5466
1,3,10,2,4,8, 7,5,11,9,6	Original HICL	5831	15.3846
	RPT-HICL	**462.4**	**8.78543**
1,10,8,4,3,2, 5,9,6,7,11	Original HICL	4395.3	17.8138
	RPT-HICL	**473.6**	**12.7935**
1,2,3,4,5,6, 7,8,9,10,11	Original HICL	5833.6	20.2024
	RPT-HICL	**439.6**	**10.2834**

Table 3: Experimental results for Vowel problem

From Table 3, we can see that the HICL networks achieved better classification accuracy compared to the Output Parallelism network. It matches with our analysis in Sec. 2. We also find that RPT-HICL has improvement on classification accuracy for all the orders tested in this problem. Also training time is decreased. This problem shows an average increase in classification accuracy of 40% using RPT-HICL compared to that using the original HICL.

D. Pen-Based Recognition

This data set consists of 16 inputs and 10 outputs. It has 7494 patterns in total (they were divided into 3747 training patterns, 1873 validation patterns, and 1874 test patterns). The patterns were normalized and scaled so that each component lies within [0, 1].

Ordering of classes	Training Procedure	Training Time (s)	Classification Error (%)
none	Output Parallelism	4382.0	4.45571
5,7,1,4,8,9,3,2,10,6	Original HICL	32658.5	1.85433
	RPT-HICL	**1450.8**	**1.41409**
7,3,1,5,4,9,6,2,10,8	Original HICL	42274.5	1.78762
	RPT-HICL	**1726.3**	**1.32871**
10,9,8,7,6,5,4,3,2,1	Original HICL	23090.5	1.79427
	RPT-HICL	**2425.4**	**1.49413**

Table 4: Experimental results for Pen-Based Recognition problem

From Table 4, we can see that the HICL networks achieved better classification accuracy compared to the Output Parallelism network. It matches with our analysis in Sec. 2. We also find that RPT-HICL has improvement on classification accuracy for all the orders tested in this problem. Also training time is decreased. This problem shows an average increase in classification accuracy of 26% using RPT-HICL compared to that using the original HICL.

6.3 Discussions

From the experimental results, we could see that different ordering leads to different classification accuracy. Two ordering methods, MSEF-CDE and MSEF-FLD, were developed in the earlier paper (Guan & Li, 2002). Based on these two methods they evaluated the classification error in each sub-network and ordered these sub-networks in HICL based on those errors, from the smallest to the greatest. Using them, the original HICL tends to get better classification accuracy than the other ordering. However, we should note that MSEF-CDE and MSEF-FLD are based on the whole training set. While in RPT-HICL, most

sub-networks are learned just using portions of the training set. So, MSEF-CDE and MSEF-FLD are not applicable to RPT-HICL. Our experiment results confirmed this observation. How to get the best order for RPT-HICL? It is one of the challenging tasks for our future research.

7 Conclusions

In this paper, a task decomposition method called HICL was introduced. A theoretical model was presented to compare the performance of HICL with a typical task decomposition approach - Output Parallelism. Our analysis showed that HICL can get better classification accuracy than Output Parallelism and the experimental results confirmed this. Also, an improved training method for the HICL network, RPT-HICL, was proposed and experiments were conducted to test its viability. The results from four benchmark problems showed that network performance in terms of classification accuracy and training time is enhanced with hierarchical learning.

More work has to be done to investigate the minimum output dimensionality of the problem for the benefits of RPT-HICL to be reaped. Alternatively the extent of training set size reduction can be explored to check the existence of optimal training set. The next phase of the research would be to investigate exactly how much reduction of patterns will give arise to an optimal solution.

Due to resource constraints, we have only been able to conduct experiments on four benchmark problems. It would be interesting to conduct some real world applications of RPT-HICL, to investigate issues like scalability and robustness of the method proposed.

References

[1] Guan, S.-U. and Li, S.: Parallel growing and training of neural networks using output parallelism, *IEEE Transaction on Neural Networks*, **13** (2002), 542 –550.

[2] Auda, G., Kamel, M. and Raafat, H.: Modular neural network architectures for classification, *IEEE International Conference on Neural Networks,* **2** (1996), 1279–1284.

[3] Jacobs, R.A., Jordan, M.I., Nowlan, M.I. and Hinton, G.E.: Adaptive mixtures of local experts, *Neural Computation*, **3** (1991), 79–87.

[4] Auda, G., Kamel, M. and Raafat, H.: A new neural network structure with cooperative modules, *World Congress on Computational Intelligence,* **3** (1994), 1301-1306.

[5] Jacobs, R., Jordan, M., and Barto, A.: Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks, *Cognitive Science*, **15** (1991), 219–250.

[6] Murre, J.: Learning and categorization in modular neural networks, *Harvester-Wheatcheaf.* (1992).

[7] Romaniuk, S.G. and Hall, L.O.: Divide and conquer neural networks, *Neural Networks,* **6** (1993), 1105-1116.

[8] Sharkey, A.J.C.: Modularity, combining and artificial neural nets, *Connection Science*, **9** (1997), 3–10.

[9] Feldman, J.: Neural representation of conceptual knowledge, edited by Nadel et al., *Neural connections, mental computation*, Cambridge, Massachusetts, USA: MIT Press (1989).

[10] Anand, R., Mehrotra, K., Mohan, C.K., and Ranka, S.: Efficient classification for multiclass problems using modular neural networks, *IEEE Transaction on Neural Networks,* **6** (1995), 117-124.

[11] Lu, B.L., Kita, H.,and Nishikawa, Y.: A multisieving neural network architecture that decomposes learning tasks automatically, *Proceedings of IEEE Conference on Neural Networks,* Orlando, FL,

(1994), 1319-1324.

[12] Lu, B.L., and Ito, M.: Task decomposition and module combination based on class relations: A modular neural network for pattern classification, *IEEE Transaction on Neural Networks*, **10** (1999), 1244-1256.

[13] Guan, S.-U. and Li, P.: A Hierarchical Incremental Learning Approach to Task Decomposition, *Journal of Intelligent Systems*, **12** (2002), 194-205.

[14] Guan, S.-U. and Zhu, F.: Class Decomposition for GA-based Classifier Agents – A Pitt Approach, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **34** (2004), 381-392.

[15] Guan, S.-U., Neo, T.N. and Bao, C: Task Decomposition Using Pattern Distributor, *Journal of Intelligent Systems*, **13** (2004), 123-150.

[16] Guan, S.-U., Li, S.C. and Tan, S.K.: Neural Network Task Decomposition Based on Output Partitioning, *Journal of the Institution of Engineers Singapore*, **44** (2004), 78-89.

[17] Guan, S.-U., and Li, P.: Incremental Learning in Terms of Output Attributes, *Journal of Intelligent Systems*, **13** (2004), 95-122.

[18] Guan, S.-U. and Zhu, F.: A Class Decomposition Approach for GA-based Classifier Agents, *Engineering Applications of Artificial Intelligence*, **18** (2005), 271-278.

[19] Guan, S.-U. and Li, S.C.: An approach to parallel growing and training of neural networks, *Proceedings of 2000 IEEE International Symposium on Intelligent Signal Processing and Communication Systems*, Honolulu, Hawaii, USA (2000).

[20] Lehtokangas, M.: Modeling with constructive backpropagation, *Neural Networks,* 12, (1999), 707-716.

[21] Riedmiller, M. and Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: the PRPOP Algorithm, *Proceedings of the IEEE International Conference on Neural Networks*, (1993), 586-591.

[22] Prechelt, L.: PROBEN1: A set of neural network benchmark problems and benchmarking rules, Technical Report 21/94, Department of Informatics, University of Karlsruhe, Germany (1994).

[23] C. S. Squires and J. W. Shavlik, "Experimental analysis of aspects of the cascade-correlation learning architecture," *Machine Learning Research Group Working Paper 91-1*, Computer Science Department, University of Wisconsin-Madison, 1991.

[24] G. Auda, M. Kamel and H. Raafat, “Modular neural network architectures for classification,” in *IEEE International Conference on Neural Networks*, Vol. 2, 1996, pp.1279-1284.