# Ensemble of Single-Layered Complex-Valued Neural Networks for Classification Tasks

# Ensemble of Single-Layered Complex-Valued Neural Networks for Classification Tasks

Md. Faijul Amin[1], Md. Monirul Islam[1, 2], and Kazuyuki Murase[1, 3]

[1]Department of Human and Artificial Intelligence Systems, Graduate School of Engineering, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan

[2]Department of Computer science and Engineering, Bangladesh University of Engineering and Technology, Dhaka - 1000, Bangladesh

[3]Research and Education Program for Life Science, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan

**Acknowledgement:** This work was supported by grants from Japanese Society for Promotion of Science (JSPS), Yazaki Memorial Foundation, and University of Fukui. Md. Monirul Islam is supported by a JSPS fellowship.

**Correspondence**: Dr. Kazuyuki Murase
Professor
Department of Human and Artificial Intelligence Systems,
University of Fukui
3-9-1 Bunkyo, Fukui 910-8507, Japan
Tel: +81(0)776 27 8774
Fax: +81(0)776 27 8420
E-mail: murase@synaspe.his.fukui-u.ac.jp

# Ensemble of Single-Layered Complex-Valued Neural Networks for Classification Tasks


## Abstract

This paper presents ensemble approaches in single-layered complex-valued neural network (CVNN) to solve real-valued classification problems. Each component CVNN of an ensemble uses a recently proposed activation function for its complex-valued neurons (CVNs). A gradient-descent based learning algorithm was used to train the component CVNNs. We applied two ensemble methods, negative correlation learning and *bagging*, to create the ensembles. Experimental results on a number of real-world benchmark problems showed a substantial performance improvement of the ensembles over the individual single-layered CVNN classifiers. Furthermore, the generalization performances were nearly equivalent to those obtained by the ensembles of real-valued multilayer neural networks.



**Keywords** – activation function, classification, complex-valued neural network, ensemble.


## 1. Introduction

Complex numbers are inevitable from both the theoretical and application perspectives. In order to process such information by artificial neural networks, researchers have developed various complex-valued network (CVNN) models, such as feed-forward and recurrent CVNNs [11,12,21], complex-valued self-organizing map [6], and complex-valued associative memories [14,20]. Recent developments are compiled in [7]. It is very natural that the CVNN would find its applications on the areas, such as telecommunications, speech recognition, image processing, and others, where data to be processed are complex-valued. However, some researchers recently have also applied CVNN to real-valued classification problems by representing and solving the problems in the complex domain.

Researchers have investigated and found that a complex-valued neuron (CVN) could achieve better classification ability than a real-valued neuron (RVN). One of the earlier works [19] studied the discrimination ability of a complex perceptron on Boolean functions up to four inputs. It was shown that the complex perceptron could achieve twice the discrimination ability of a real perceptron. In [22], the real-valued inputs and

the class labels were encoded by complex numbers, and then were processed by a CVN to solve the XOR and the symmetry detection problem. In [17], real-valued inputs were encoded by the phases of unity magnitude complex numbers. Depending on the magnitude of the CVN's output, the class label of an input pattern was determined. The CVN could achieve an improvement of 135% over a real-valued neuron (RVN) for the three-input Boolean functions.

There has been another recent approach that used multilayer feed-forward architecture of multi-valued neurons [1]. The approach also encoded the inputs by the phases of unity magnitude complex numbers, but the class labels were encoded by the roots of unity in the complex plane. They showed that their feed-forward multilayer network could successfully solve the parity $n$ ($2 \leq n \leq 9$) problem and the two spirals problem, and could perform better in the "sonar" benchmark and the Mackey–Glass time series prediction problems.

Most of the aforementioned approaches, however, have some shortcomings. The CVN models of [19] and [22], for example, require careful settings of the target outputs ($\{0, 1\}$ in [19] and $\{1, 0, 1+i, i\}$ in [22]) for different classes. Choosing an arbitrary output encoding scheme (0 = class *A* and 1 = class *B*, or the reverse setting) will not work for some problems since it was reported in [19] that the CVN could not realize several three-input Boolean functions, while the complementary functions were realized. Assigning output values to different classes is even more complicated for the CVN model of [22]. The same problem exists in [17]. Moreover, the learning algorithm of the CVN [17] may suffer from instability due to a reciprocal of derivative, namely when the derivative approaches to zero.

In order to minimize the shortcomings, we proposed a new class of activation functions [2], whose role is similar to the conventional RVN in the classification tasks. The functions combine the real and imaginary part of the complex numbers, and map complex values into bounded real values. Due to the differentiability of the activation functions, a gradient-descent learning algorithm can be easily derived (see [2] for the learning algorithm). We showed that a CVN with these activation functions could successfully

solve several Boolean classification problems (including linear and nonlinear problems). We further studied the generalization ability of single-layered CVNN on several real-world multiclass problems, and showed that the performance was comparable to that of the multilayer real-valued neural networks (RVNNs).

It should be mentioned that a CVN or a single-layered CVNN (only one layer of computing neurons, each representing one class) can not match all possible problems' complexity because of the fixed structure (determined by the number of inputs and number of classes). To improve the performance, one possibility is to use multiple layers of neurons as was done in [1]. However, it is well known that an ensemble of classifiers can achieve better classification ability than that of an individual classifier, provided that the individual classifiers do not make error on the same part of the data [24].

In this study, therefore, we investigate the ensemble methods in the single-layered CVNNs that were developed in our earlier work [2]. Among the ensemble creation methods, we examined two methods, negative correlation learning [15] and *bagging* [3]. The former is an explicit method, while the latter is an implicit method [4].

We show here that the ensemble methods can enhance the performance of single-layered CVNNs to a considerable extent. Furthermore, experimental results on various real-world benchmark problems show a comparable generalization performance of the single-layered CVNN ensembles to that of the multilayer RVNN ensembles.

It is noteworthy that any ensemble methods can be easily applied to our CVN model due to its gradient based learning rule. For example, the negative correlation learning requires each member in the ensemble to be trained with gradient-descent based learning [15]. So it is difficult to apply the negative correlation learning to the CVN models which can not be trained with a gradient based learning rule (e.g., the CVN model of [1]).

The remainder of the paper is organized as follows. In Section 2, we briefly discuss the CVN model used in the ensembles of this study, along with the classification ability of a single CVN on some Boolean problems. Two methods for creating the ensembles of single-layered CVNNs, i.e., negative correlation learning and *bagging*, are discussed in

Section 3. Experimental results on a number of real-world benchmark problems are presented in Section 4. Finally, we give our concluding remarks in Section 5.

# 2. CVN Model and Its Classification Ability

This section briefly discusses the CVN model and its classification ability, which we presented in our earlier work [2]. The discussion includes the representation of real-valued input data to a CVN, the role of the activation function, and the classification ability of a CVN on some Boolean problems.

## *2.1 Input data representation*

To present complex-valued information to a CVN, we encoded the real-valued data by the phases, between 0 and π, of the unity magnitude complex numbers. For example, if a real-valued number $x \in [a, b]$, where $a, b \in R$, then the corresponding complex number $z = e^{\mathbf{i}\pi(x-a)/(b-a)}$, where $\mathbf{i} = \sqrt{-1}$. Clearly, in this representation, when the real-valued variable $x$ moves along a line from $a$ to $b$, the corresponding complex variable $z$ moves over the upper half of a unit circle on the complex plane. In order to process the Boolean data, the values TRUE and FALSE were represented by $e^{\mathbf{i}\pi}$ and $e^{\mathbf{i}0}$, respectively.

## *2.2 Activation function*

Our motivation for designing the activation function came from the role of the activation function in a real-valued output neuron for the classification tasks. The neuron has essentially two functional parts, an aggregation part and an activation part. The aggregation part maps a multidimensional input into a one dimensional output by multiplying each of the inputs to the neuron by the connection weights and then by summing up the weighted inputs. The other part, i.e., activation function does a threshold operation on the output given by the aggregator. As for instance, consider a threshold function given by

$$y(v) = \begin{cases} \text{class } A, \text{if } v \geq 0 \\ \text{class } B, \text{otherwise,} \end{cases}$$

where $v = \mathbf{w}^T\mathbf{x} + b$, $\mathbf{w}$ and $\mathbf{x}$ being the weight and the input vectors, and $b$ is the bias of the neuron. Clearly, the threshold function divides its one dimensional domain into two disjoint parts; each part denotes one of the two classes.

Thus the role of an activation function, in a real-valued output neuron, is to divide the function's domain (defined by the output of the aggregator) into disjoint sets or regions for representing the corresponding classes. Hence forward, we call the output of the aggregator part as the net-input of a neuron, and the domain of an activation function as the net-input space of the neuron.

Motivated by the role of an activation function in a real-valued output neuron, we formulated a family of activation functions for CVN in [2]. The functions map complex-valued net-inputs into bounded real-values by combining the real and imaginary parts of the net-inputs. The purpose of such mapping is to divide the net-input space into different regions to represent the classes.

Figure 1 shows one of the functions which we have used in this study of the ensembles. The function maps complex-values into bounded real values and has the form of $f_{C\to R}(u + \mathbf{i}\, v) = (f_R(u) - f_R(v))^2$ where $f_R(x) = 1/(1+e^{-x})$, $u, v, x \in R$, and $\mathbf{i} = \sqrt{-1}$. As can be seen from Fig. 1, the function saturates in four regions, $R_1$, $R_2$, $R_3$, and $R_4$. Among the regions, $R_1$ and $R_3$ denote one class, while $R_2$ and $R_4$ denote the other class. Note that the function is differentiable with respect to real and imaginary part of the net-input individually. Since the cost function (mean squared error) to be minimized is real-valued, this kind of differentiability is sufficient to derive a gradient-descent based learning algorithm by considering the real and imaginary parts of the weight parameters individually. See [2] and also the Section 3.1 for the details of the learning algorithm.

Liouville's theorem states that there is no complex-valued function which is bounded and differentiable in the entire complex domain except the constant. We can avoid this constraint by combining the real and imaginary part of the net-input meaningfully for the classification tasks, and this was our main objective of designing the activation functions.

## 2.3 Classification ability of a CVN

We studied the classification ability of a CVN with the new activation functions on several Boolean problems in [2]. Here we briefly discuss those results. It was shown that a CVN could solve all the two-input Boolean problems, which include both the linear (e.g., OR function) and nonlinear (e.g., XOR function) problems. As an illustration, we show how a CVN could solve the OR and the XOR problem in Fig. 2.

In case of three-input Boolean functions, our CVN (with the activation function of Fig. 1) could solve 254 functions out of 256. Although in [2], we reported 253 functions, we later found that with a different random initialization, a CVN could solve 254 functions. The remaining two unsolved functions are complement to each other, and they form the three-input parity problem. Note that from the classification point of view, each Boolean function and its complement form a single classification tasks.

For the three-input Boolean functions, Michel and Awwal [17] have reported that their CVN model could solve 245 functions out of 256, whereas our model could solve 254 functions. Nemoto and Kono [19] showed that their complex perceptron could solve 127 different dichotomies out of 128 (counting each function and its complement as one classification task or dichotomy). They, however, reported that several functions of the 127 dichotomies could not be solved by their complex perceptron, while the complements were solved. In comparison, our CVN could solve both the functions and the complements of the 127 dichotomies.

Symmetry detection problem in binary sequences is a linearly non separable problem, and thus can not be solved by an RVN. We have tested our CVN model on the symmetry problem from 2 to 10 bits, and a CVN could solve all the cases. Nitta [22] showed an example solution to the six bit symmetry detection problem by a CVN and mentioned that the other cases could be solved in the similar way.

The above descriptions indicate that a CVN has better classification ability than an RVN. The reason might be due to the mapping ability of a CVN. As discussed in section 2.2, a neuron has essentially two functional parts, an aggregation part and an activation part.

The aggregation part does a mapping from a multidimensional input space to the neuron's net-input space which is one dimensional for an RVN and two dimensional for a CVN. The other part, i. e., activation function divides the net-input space into disjoint sets representing different classes. In the mapping by the aggregator, each input is multiplied by a connection weight and then added. If we consider $\Im_R$ as the set of all possible such mappings for an RVN and $\Im_C$ the set of all possible mappings for a CVN it can be seen that $\Im_R \subset \Im_C$. It is because a complex multiplication scales and rotates an input by any arbitrary amount, while a real multiplication does a scaling by an arbitrary amount but a rotation by only 0 or $\pi$. In other words, the mapping ability of a CVN is superior to an RVN and this might be reason for better classification ability of a CVN.

We further studied the performance of single-layered CVNN on several real-world benchmark problems in [2]. The single-layered CVNN was composed of multiple CVNs, each represented one class in a *winner-takes-all* fashion. It was shown that such CVNNs could achieve comparable generalization ability to that of the multilayer real-valued neural networks (RVNNs).

## 3. Ensemble of Single-Layered CVNN

Ensemble of classifiers has been widely used to improve the performance of the individual classifier. The key point of the ensemble method is to take the opinions from the experts, before making the final decision. The motivation came from the decision making process in our society. For example, we have parliaments, juries, committees, board of directors, and so on. It is trusted that taking opinions from a group of experts results in a better decision than the decision from an individual expert. Though the idea is a very simple one, researchers have proved its extensive benefit in the automated decision making applications. Ensemble is also known as other names, such as multiple classifier systems, committee of classifiers, or mixture of experts.

Since our CVN model and single-layered CVNN showed impressive classification ability, we would like to investigate their performance enhancement by the ensemble methods. There are several methods for creating the ensembles, such as *bagging* [3], boosting [26], negative correlation learning [15], stacked generalization [28], mixture of experts [9,10],

etc. Each of these methods has also different variations. The Design, implementations, and the issues of the ensemble methods can be found in [25].

The main goal of the ensemble methods is to create diversity among the classifiers, so that the classifiers do not make error on the same part of the data, i.e., one's error can be compensated by the other members. Ensemble methods can be broadly divided into two categories, explicit and implicit methods [4]. The explicit methods incorporate a diversity measure into the cost function directly, while the implicit methods create diversity by presenting different data subsets to the member classifiers. In this study, we applied two methods, negative correlation learning [15] and *bagging* [3]. The former is an explicit method and the latter is an implicit method. We briefly discuss these two methods in the next subsections.

## *3.1 Negative correlation learning*

Negative correlation learning (NCL) for designing the ensembles is used, especially, when the base learning systems are neural networks [15]. It has shown several empirical successes [5,8,16]. The key idea behind the NCL is to encourage the individual networks in an ensemble to learn different parts or aspects of a training data, so that the ensemble can learn the whole training data better. The NCL does so by introducing a correlation penalty term into the cost function.

To formally represent the cost function, let there be $M$ individual networks in an ensemble. Then, the loss function for each network $j$ ($1 \leq j \leq M$) is defined by

$$e_j = \frac{1}{N}\sum_{n=1}^{N} e_j(n) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}(d(n) - f_j(n))^2 + \frac{1}{N}\sum_{n=1}^{N}\lambda p_j(n) \ , \qquad (1)$$

where $e_j(n)$ is the error of the network $j$ for $n$-th training pattern, and $N$ is the size of training set. Similarly, $d(n)$, $f_j(n)$, and $p_j(n)$ are the desired output, actual output, and correlation penalty function for the $n$-th training pattern, respectively. In Eq. (1), the parameter $\lambda$ is used to adjust the strength of the penalty.

The penalty function $p_j$ has the following form

$$p_j(n) = (f_j(n) - \overline{f}(n)) \sum_{k \neq j} (f_k(n) - \overline{f}(n)) \\ = -(f_j(n) - \overline{f}(n))^2 \,, \tag{2}$$

where $\overline{f}(n)$ is the average over the outputs of individual networks, also known as the ensemble output when simple averaging is used to combine the individual network's output. Second part of Eq. (2) follows from the fact that $\sum_{j=1}^{M} (f_j(n) - \overline{f}(n)) = 0$.

Minimization of Eq. (1) implies that each network has to minimize the difference between the target output and its actual output, as well as the penalty term. Note that minimization of the penalty term in Eq. (1) results in the maximization of the distance of individual network's output from the average value. It can be clearly understood from the second part of Eq. (2) since there is a negative sign before the distance term. Thus the penalty term encourages each network to be functionally different, and we can expect to get useful diversity among the networks. Since the NCL incorporates a diversity measure directly into the cost function, it is considered as an explicit method [4].

Learning refers to the weight adjustment of connection weights in the neural network paradigm. Supervised learning adjusts the weights by the optimization of some objective function. It is seen that in the practical implementation of NCL, a gradient-descent learning is used to minimize the cost function of Eq. (1) [5,8,15,16]. Since the CVN model of our earlier work can be trained by a gradient-descent based learning method [2], NCL can easily be applied to the ensemble of CVNNs. From Eqs. (1) and (2), it can be seen that the partial derivative of $e_j(n)$ with respect to the output of network $j$ is:

$$\frac{\partial e_j(n)}{\partial f_j(n)} = -(d(n) - f_j(n)) - 2\lambda(1 - \frac{1}{M})(f_j(n) - \overline{f}(n)) \tag{3}$$

Now for any complex-valued weight $w = w^R + \mathbf{i}\, w^I$, the update of the weight is given by

$$\begin{aligned}\Delta w &= -\eta\frac{\partial e_j(n)}{\partial w^R} - \mathbf{i}\,\eta\frac{\partial e_j(n)}{\partial w^I} \\ &= -\eta\frac{\partial e_j(n)}{\partial f_j(n)}\frac{\partial f_j(n)}{\partial w^R} - \mathbf{i}\,\eta\frac{\partial e_j(n)}{\partial f_j(n)}\frac{\partial f_j(n)}{\partial w^I},\end{aligned} \tag{4}$$

where $\eta$ is the learning rate. In the above equation, $\frac{\partial e_j(n)}{\partial f_j(n)}$ can be substituted by Eq. (3), while $\frac{\partial f_j(n)}{\partial w^R}$ and $\frac{\partial f_j(n)}{\partial w^I}$ can be computed from the functional form of the activation function.

As an illustration of how an ensemble can improve the classification ability, we present a solution of three-bit parity problem by an ensemble of three CVNs in Table 1. Note that this parity problem could not be solved by a CVN. When the ensemble was trained with the NCL algorithm, as can be seen from Table 1, each of the CVNs made error with one input pattern, which is shown as shaded. However, the errors of the CVNs are diverse, i.e., the CVNs made error on a single but different patterns. As a result one's error can be compensated by the other CVNs. In other words, a *majority voting* or a simple average of the outputs can solve the three-bit parity problem completely. We further studied the parity *n* problem for $4 \le n \le 9$, and found that an ensemble of CVNs could solve all the problems successfully. Higher-input parity problems, however, required higher number of CVNs in the ensemble. For example, the eight and nine-input parity problems required 10 and 19 CVNs, respectively.

It was shown by Ueda and Nakano [27] that the generalized error of an ensemble can be decomposed into bias, variance, and covariance terms. The NCL tries to reduce the covariance term without affecting the bias and the variance terms [5]. The expected result of making the outputs of the members negatively correlated is that the individual members will make error diversely. As explained for the three-bit parity problem, such diversity lets an ensemble achieve better classification ability.

### *3.2 Bagging*

It is the short name for *bootstrap aggregating*, and one of the earliest algorithms for constructing an ensemble proposed by Breiman [3]. In this algorithm, diversity among the individual members is obtained by the bootstrapped replicas of the training data. Different training data subsets are randomly drawn, with replacement, from the entire training data. Each training data subset is used to train a different classifier. Individual classifiers are then combined usually by some combining scheme, such as *majority voting* or averaging the outputs (when the outputs are continuous and/or have an interpretation of *posterior probabilities*) of the classifiers.

The algorithm for creating an ensemble of single-layered CVNN by *bagging* is shown in Fig. 3. The algorithm takes training data $D$, number of classifier $T$, and a fraction $F$ as inputs. In each iteration $t = 1 \ldots T$, a subset of training data $D_t$ is created by randomly drawing, with replacement, $F$ fraction of training samples from the original training data $D$. Then, each single-layered $\text{CVNN}_t$ is trained with the data subset $D_t$, and added to the ensemble. Since each of the data subsets are likely to be varied from each other, the individual members in the ensemble are likely to be diverse. In other words, it can be expected that the individual classifiers will make error on different parts of the data. Thus, the *bagging* algorithm, unlike the NCL, creates diversity implicitly by data sampling method, without incorporating any diversity term in the cost function.

## 4. Experimental Studies

To evaluate the performance enhancement of single-layered CVNN by ensemble methods, we carried out experiments on 13 real-world benchmark problems. The problems were taken from the UCI machine learning data repository. The data sets of our experiments vary in their characteristics; for example, the number and types of the features, the number of output classes, and the number of examples in the data sets. Table 2 shows the characteristics of the data sets. The details of the data sets can be found in the URL: http://archive.ics.uci.edu/ml/datasets.html.

## 4.1 Experimental settings

In the classification problems, one is interested on the prediction ability of the classifier model over unseen data, which is also known as generalization performance. We measured the generalization performance in terms of classification error on the unseen data. This means the lower the classification error the better the generalization performance.

Since our purpose is to investigate the performance enhancement of single-layered CVNN using ensemble methods, we first applied single classifiers, i.e., single-layered CVNNs, to the problems. Then the NCL and the *bagging* algorithms were applied for creating the ensembles. We took ten CVNNs in the ensembles for both the NCL and the *bagging* algorithms. In a single-layered CVNN, the number of CVNs was similar to the number of classes for a given problem, and the CVN with the highest output designated the predicted class.

To combine the decisions from the individual single-layered CVNNs, we used a simple average of the output values since the output of the activation function used in our experiments was continuous and bounded real valued in the range of (0, 1). This type of combining scheme is generally used when the outputs are continuous and/or have an interpretation of *posterior probabilities* [13]. After taking the averages of the single-layered CVNNs (neuron by neuron basis) of the ensemble, the highest average value designated the predicted class.

In case of *bagging*, when the data subsets were selected for the individual classifiers, we drew 75% examples randomly, with replacement, from the original training data set. For example, if a training data set was consisted of 100 items, 75 items were randomly drawn, with replacement, to create a subset. Such kind of fractions is generally used in the implementation of *bagging* algorithm as it does not allow much overlap among the data subsets [25].

During the training process we set the learning rate fixed at 0.15, and initialized the real and imaginary parts of the complex-valued weights by random numbers taken from a

uniform distribution $U(-0.5, 0.5)$. In case of NCL, we have an extra parameter $\lambda$ or the penalty coefficient. We set $\lambda$ as 0.5 in our experiments. Training epochs for each of the data sets are listed in Table 2.

## *4.2 Experimental results*

In this section, we report the testing error rates of single-layered CVNN ensembles for the data sets presented in Table 2. Results were averaged over five standard 10-fold cross validation experiments. For each 10-fold cross validation, the data sets were first partitioned into 10 equal or nearly equal (since the number of examples may not be a multiple of 10) sized sets, and then each set was used in turn as the test or unseen data. Single classifiers (single-layered CVNNs) and the ensembles were trained on the remaining nine sets.

Table 3 shows the testing error rates of single classifiers (single-layered CVNNs) and their ensembles trained with NCL. The error rates are presented in terms of percentage of misclassifications. We also show the error reductions by the ensembles in the rightmost column. The error reductions are presented in percentage relative to the error of a single classifier. For example, an error reduction from 2.5 to 1.25 indicates a 50% reduction, just as a reduction from 5 to 2.5 would also be a 50% reduction. It can be seen from Table 3 that the ensembles could reduce the errors for all the data sets, except for the data set, *iris*. In many cases, the reductions were impressive. For example, in case of the *glass*, *ionosphere*, *satellite*, *segmentation*, *sonar*, and *soybean* problems, the classification errors were reduced by 13.0%, 19.3%, 18.7%, 22.1%, 19.4%, and 40.0%, respectively.

Similar results for the *bagging* algorithm are shown in Table 4. Here the ensembles of single-layered CVNNs could reduce the testing error rates for all the data sets. Error reductions were noteworthy for the data sets *glass*, *hepatitis*, *iris*, *segmentation*, *sonar*, and *soybean*. Quantitatively, these reductions were 11.6%, 10.7%, 15.2%, 11.8%, 10.0%, and 35.6%, respectively. These results suggest that the ensemble methods can be applied to the CVNN for its performance improvement.

We also compare the generalization abilities of the single-layered CVNN ensembles with that of the conventional multilayer RVNN ensembles. The comparison is made with the RVNN ensemble results reported in [23], where the authors evaluated the generalization abilities of multilayer RVNN ensembles and decision tree ensembles. Our purpose of comparing with the reported result is to see how much generalization can be achieved by the single-layered CVNN ensembles in comparison to that of the multilayer RVNN ensembles for the benchmark data sets.

Table 5 shows the testing error rates of the single-layered CVNN ensembles and multilayer RVNN ensembles for both the NCL and *bagging* algorithms. The *bagging* results of the RVNN ensembles were taken from [23], while the NCL results of RVNN ensembles were taken from our experiments, since NCL results were not available in [23]. However, for the NCL results of RVNN ensembles, we used the similar experimental settings of [23], such as the number of hidden units in each RVNN of the ensembles, learning rate (0.15), momentum term (0.9), and initial weights (randomly between -0.5 and 0.5). Table 6 shows the number of weight parameters of single-layered CVNNs and multilayer RVNNs used in the ensembles. Since for the multilayer RVNN ensembles we followed the neural networks' structure of [23], the number of parameters of the single-layered CVNNs was not same as that of multilayer RVNNs. Under this setting, Table 6 shows that except for the *soybean* data set, multilayer RVNNs have more parameters than those of single-layered CVNNs.

The results of single-layered CVNN ensembles, in Table 5, show that between the NCL and *bagging* methods, NCL could perform better for the data sets *ionosphere*, *satellite*, and *sonar*, while the *bagging* algorithm performed better for the *iris* data set. For the remaining most cases, the performances were more or less similar. Therefore, no method seems to be superior to the other, although each of the methods could reduce the error of the single classifiers.

If we take the best result of single-layered CVNN ensembles between the NCL and *bagging* algorithms and the best result of multilayer RVNN ensembles similarly, for each of the data sets from Table 5, and then compare, we see that the performance of single-

layered CVNN ensembles is almost similar to that of the multilayer RVNN ensembles for most of the data sets. Moreover, for the data sets *glass* and *sonar*, CVNN ensembles could perform even better. This comparison indicates that the single-layered CVNN ensembles, even with only one computational layer in the individual classifiers, could achieve comparable performance to that of the multilayer RVNN ensembles.

It is mentionable that from the computer simulation perspective, learning cost of single-layered CVNN ensembles is expensive due to the reason that a complex multiplication requires four real multiplication and two real addition/subtraction operations. Exact leaning cost of single-layered CVNN ensemble and multilayer RVNN ensemble for one learning step (forward signal propagation and backward error propagation) depends on the number of connection weights of individual single-layered CVNN and multilayer RVNN, respectively, in the ensembles. Even if the number of connection parameters is equal for a single-layered CVNN (counting each complex-valued weight as two parameters due to its real and imaginary part) and a multilayer RVNN, the required number of real multiplication or division operations will be nearly twice for the single-layered CVNN, assuming the computation of sigmoid function is done by lookup table. The number of addition/subtraction operations is also nearly twice in the single-layered CVNN for equal number of parameters. Here, extra addition and subtraction operations for the single-layered CVNN come from the complex multiplication operation. However, hardware implementations may reduce the learning cost of single-layered CVNN ensembles significantly [18].

## 5. Conclusion

We studied the ensembles of single-layered CVNNs in this paper. Although we applied two ensemble methods, NCL and *bagging*, any ensemble method can be easily applied to our CVN model, due to its gradient-descent based learning rule.

We explained how an ensemble of CVNs could solve the three-input parity problem, which could not be solved by a single CVN. Solving the parity problem up to nine inputs, by the ensembles of CVNs, has also been reported. This indicates a substantial performance improvement by the ensembles. The essence of the performance

improvement is that the individual classifiers should not make error on the same part of the data.

For further investigation, two ensemble methods, NCL and bagging, were applied to several real-world multiclass problems. Experimental results showed that the ensemble methods could improve the performance of single-layered CVNN for almost all the data sets. However, none of the methods (NCL and *bagging*) seems to be superior to the other.

In general, a single-layered CVNN, which has a fixed structure for a given problem (determined by the input and output dimensions of the problem), will not match with all possible problems' complexities. Nevertheless, the generalization ability of the single-layered CVNNs and their ensembles on several real-world benchmark problems reveals their high information processing capabilities, even with only one computing layer. Comparison with the multilayer RVNN ensembles supports this fact. Specifically, single-layered CVNN ensembles could achieve almost similar generalization performance to that of the multilayer RVNN ensembles, and for some data sets even better.

Solving the real-valued classification problems in the complex domain requires more researches. Extending our single-layered CVNN to multilayer and studying their ensembles for solving the classification problems can be a future research direction.

## Acknowledgement

This work was supported by grants from Japanese Society for Promotion of Science (JSPS), Yazaki Memorial Foundation, and University of Fukui. Md. Monirul Islam is supported by a JSPS fellowship.

## Reference:

[1] Aizenberg, C. Moraga, Multilayer feedforward neural network based on multi-valued neurons and a backpropagation learning algorithm, Soft Computing 11 (2) (2007) 169–183.

[2] M.F. Amin, K. Murase, Single-layered complex-valued neural network for real-valued classification problems, Neurocomputing (2008), doi:10.1016/j.neucom.2008.04.006.

[3] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.

[4] G. Brown, X. Yao, On the effectiveness of negative correlation learning, First UK Workshop on Computational Intelligence, Edinburgh, Scotland, 2001.

[5] G. Brown, Diversity in neural network ensembles, PhD dissertation, University of Birmingham, January, 2004.

[6] T. Hara, A. Hirose, Plastic mine detecting radar system using complex-valued self-organizing map that deals with multiple-frequency interferometric images, Neural Networks, 17 (8–9) (2004) 1201–1210.

[7] A. Hirose, Complex-Valued Neural Networks, Springer, Berlin, 2006.

[8] M.M. Islam, X. Yao, K. Murase, A constructive algorithm for training cooperative neural network ensembles, IEEE Transactions on Neural Networks 14 (4) (2003) 820–834.

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

[10] M.J. Jordan, R.A. Jacobs, Hierarchical mixtures of experts and the EM algorithm, Neural Computation, 6 (2) (1994) 181–214.

[11] T. Kim, T. Adali, Fully complex multilayer perceptron for nonlinear signal processing, Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology 32 (2002) 29–43.

[12] M. Kinouchi, M. Hagiwara. Memorization of melodies using complex-valued recurrent neural network, in: A. Hirose (Ed.), Complex-Valued Neural Networks: Theories and Applications, vol. 5, World Scientific Publishing, Singapore, 2003, pp. 205–226.

[13] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley–Interscience, New Jersey, 2004.

[14] Y. Kuroe, A model of complex-valued associative memories and its dynamics, in: A. Hirose (Ed.), Complex-Valued Neural Networks: Theories and Applications, World Scientific Publishing, Singapore, 2003, pp. 57–79.

[15] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Networks, 12 (10) (1999) 1399-1404.

[16] Y. Liu, X. Yao, T. Higuchi, Evolutionary ensembles with negative correlation learning, IEEE Transactions on Evolutionary Computation, 4 (4) (2000) 380–387.

[17] H.E. Michel, A.A.S. Awwal, Enhanced artificial neural networks using complex numbers, in: Proceedings of IJCNN'99, International Joint Conference on Neural Networks, vol. 1, 1999, pp. 456–461.

[18]H. E. Michel, A. A. S. Awwal, D. Rancour, Artificial neural networks using complex numbers and phase encoded weights–Electronic and optical implementations, in: Proceedings of IJCNN 2006, International Joint Conference on Neural Networks, 2006, pp. 1213–1218.

[19]I. Nemoto, T. Kono, Complex neural networks, Systems and Computers in Japan, 23 (8) (1992) 75–84.

[20]I. Nemoto, Complex associative memory and complex single neuron model, in: A. Hirose (Ed.), Complex-Valued Neural Networks: Theories and Applications, World Scientific publishing, Singapore, 2003, pp. 107–130.

[21]T. Nitta, An extension of the backpropagation algorithm to complex numbers, Neural Networks, 10 (8) (1997) 1391–1415.

[22]T. Nitta, Solving the XOR problem and the detection of symmetry using a single complex-valued neuron, Neural Networks, 16 (8) (2003) 1101–1105.

[23]D. Opitz, R. Maclin, Popular ensemble methods: An empirical study, Journal of Artificial Intelligence Research, 11 (1999) 169–198.

[24]N. C. Oza, K. Tumer, Classifier ensembles: Select real-world applications, Information Fusion, 9 (1) (2008) 4–20.

[25]R. Polikar, Ensemble based systems in decision making, IEEE Circuits and Systems Magazine, 6 (3) (2006) 21–45.

[26]R.E. Schapire, The strength of weak learnability, Machine Learning, 5 (2) (1990) 197–227.

[27]N. Ueda, R. Nakano. Generalization error of ensemble estimators, in: Proceedings of IJCNN’96, International Joint Conference on Neural Networks, 1996, pp. 90–95.

[28]D.H. Wolpert, Stacked generalization, Neural Networks, 5 (2) (1992) 241–259.

**Table 1**
Input-output relationship of three-bit parity problem and the outputs of three CVNs. The rightmost column shows the averages of the outputs.

| Input pattern | Output | $f_1$ | $f_2$ | $f_3$ | $f_{ens} = \overline{f}$ |
|---|---|---|---|---|---|
| 0 0 0 | 1 | 0.9959 | 0.9586 | 0.9685 | 0.9743 |
| 0 0 1 | 0 | 0.0078 | 0.0037 | 0.0141 | 0.0085 |
| 0 1 0 | 0 | 1.0000 | 0.0042 | 0.0012 | 0.3351 |
| 0 1 1 | 1 | 0.9959 | 0.9600 | 0.9640 | 0.9733 |
| 1 0 0 | 0 | 0.0000 | 0.0201 | 0.0015 | 0.0072 |
| 1 0 1 | 1 | 0.9966 | 0.9920 | 0.0000 | 0.6629 |
| 1 1 0 | 1 | 0.9966 | 0.0000 | 0.9933 | 0.6633 |
| 1 1 1 | 0 | 0.0000 | 0.0151 | 0.0354 | 0.0168 |

Shaded outputs indicate misclassifications.

**Table 2**
Characteristics of data sets and number of epochs.

| Data Set | Cases | Class | Features | | Epochs |
|---|---|---|---|---|---|
| | | | Continuous | Disc. | |
| cancer | 699 | 2 | 9 | – | 60 |
| credit-a | 690 | 2 | 6 | 9 | 60 |
| credit-g | 1000 | 2 | 7 | 13 | 60 |
| glass | 214 | 6 | 9 | – | 100 |
| heart-c | 303 | 2 | 8 | 5 | 80 |
| hepatitis | 155 | 2 | 6 | 13 | 100 |
| ionosphere | 351 | 2 | 34 | – | 70 |
| iris | 150 | 3 | 4 | – | 100 |
| satellite | 6435 | 6 | 36 | – | 60 |
| segmentation | 2310 | 7 | 19 | – | 60 |
| sonar | 208 | 2 | 60 | – | 100 |
| soybean | 683 | 19 | – | 35 | 60 |
| vehicle | 846 | 4 | 18 | – | 60 |

**Table 3**
Average test set error rates (percentage of misclassifications on unseen data) of single-layered CVNNs and their ensembles trained with negative correlation learning algorithm. Averages were taken over five standard 10-fold cross-validations. The rightmost column shows percentage of error reduction by the ensemble with respect to a CVNN's error.

| Data Sets | CVNN | CVNN ENSEMBLE (NCL) | Error Reduction |
|---|---|---|---|
| cancer | 3.6 | 3.5 | 2.8 |
| card-a | 14.3 | 13.7 | 4.2 |
| card-g | 26.2 | 24.5 | 6.5 |
| glass | 34.5 | 30.0 | 13.0 |
| heart-cleveland | 17.1 | 15.7 | 8.2 |
| hepatitis | 19.6 | 17.8 | 9.2 |
| ionosphere | 10.9 | 8.8 | 19.3 |
| iris | 3.3 | 3.3 | 0.0 |
| satellite | 13.4 | 10.9 | 18.7 |
| segmentation | 6.8 | 5.3 | 22.1 |
| sonar | 16.0 | 12.9 | 19.4 |
| soybean | 9.0 | 5.4 | 40.0 |
| vehicle | 21.3 | 20.3 | 4.7 |

**Table 4**
Average test set error rates (percentage of misclassifications on unseen data) of single-layered CVNNs and their ensembles trained with *bagging* algorithm. Averages were taken over five standard 10-fold cross-validations. The rightmost column shows percentage of error reduction by the ensemble with respect to a CVNN's error.

| Data Sets | CVNN | CVNN ENSEMBLE (Bagging) | Error Reduction |
|---|---|---|---|
| cancer | 3.6 | 3.4 | 5.6 |
| card-a | 14.3 | 13.7 | 4.2 |
| card-g | 26.2 | 24.6 | 6.1 |
| glass | 34.5 | 30.5 | 11.6 |
| heart-cleveland | 17.1 | 15.7 | 8.2 |
| hepatitis | 19.6 | 17.5 | 10.7 |
| ionosphere | 10.9 | 10.3 | 5.5 |
| iris | 3.3 | 2.8 | 15.2 |
| satellite | 13.4 | 12.3 | 8.2 |
| segmentation | 6.8 | 6.0 | 11.8 |
| sonar | 16.0 | 14.4 | 10.0 |
| soybean | 9.0 | 5.8 | 35.6 |
| vehicle | 21.3 | 20.3 | 4.7 |

**Table 5**
Test set error rates of RVNN (having one hidden layer) ensembles and single-layered CVNN ensembles for two ensemble methods, NCL and *bagging*.

| Data Sets | CVNN ensemble | | RVNN ensemble | |
|---|---|---|---|---|
| | NCL | Bagging | NCL | Bagging |
| cancer | 3.5 | 3.4 | 3.3 | 3.4 |
| card-a | 13.7 | 13.7 | 13.9 | 13.8 |
| card-g | 24.5 | 24.6 | 24.9 | 24.2 |
| glass | 30.0 | 30.5 | 32.2 | 33.1 |
| heart-cleveland | 15.7 | 15.7 | 15.5 | 17.0 |
| hepatitis | 17.8 | 17.5 | 17.9 | 17.8 |
| ionosphere | 8.8 | 10.3 | 8.8 | 9.2 |
| iris | 3.3 | 2.8 | 2.7 | 4.0 |
| satellite | 10.9 | 12.3 | 10.9 | 10.6 |
| segmentation | 5.3 | 6.0 | 5.5 | 5.4 |
| sonar | 12.9 | 14.4 | 15.1 | 16.8 |
| soybean | 5.4 | 5.8 | 5.7 | 6.9 |
| vehicle | 20.3 | 20.3 | 20.4 | 20.7 |

**Table 6**
Number of parameters of single-layered CVNNs and multilayer RVNNs (one hidden layer) used in the ensembles. Parameters include the connection weights and biases. For CVNN, each weight and bias was counted as two parameters as it has real and imaginary parts.

| Data Sets | Input | Output | CVNN | RVNN | |
|---|---|---|---|---|---|
| | | | Parameters | Hidden units | Parameters |
| cancer | 9 | 2 | 40 | 5 | 62 |
| card-a | 51 | 2 | 208 | 10 | 542 |
| card-g | 63 | 2 | 256 | 10 | 662 |
| glass | 9 | 6 | 120 | 10 | 166 |
| heart-cleveland | 35 | 2 | 144 | 5 | 192 |
| hepatitis | 32 | 2 | 132 | 10 | 352 |
| ionosphere | 34 | 2 | 140 | 10 | 372 |
| iris | 4 | 3 | 30 | 5 | 43 |
| satellite | 36 | 6 | 444 | 15 | 651 |
| segmentation | 19 | 7 | 280 | 15 | 412 |
| sonar | 60 | 2 | 244 | 10 | 632 |
| soybean | 82 | 19 | 3154 | 25 | 2569 |
| vehicle | 18 | 4 | 152 | 10 | 234 |

# Figure captions

**Fig. 1.** Activation function with contour diagram. The function has the form $f(u + \mathbf{i}\, v) = (f_R(u) - f_R(v))^2$ where $f_R(x) = 1/(1+e^{-x})$, $u, v, x \in R$, and $\mathbf{i} = \sqrt{(-1)}$, and maps complex values into bounded real values in range of [0, 1]. It saturates in four regions $R_1$, $R_2$, $R_3$, and $R_4$. Among the regions, $R_1$ and $R_3$ denote one class, while $R_2$ and $R_4$ denote the other class. The function is drawn elevated to show the regions on the net-input space.

**Fig. 2.** Net-inputs (sum of weighted inputs) of a trained complex-valued neuron for two-input Boolean patterns: (a) the OR problem, and (b) the XOR problem. Contour lines show the output values of the activation function. An input pattern yielding an output value greater than or equal to 0.5 is assigned to class 1, otherwise class 0 is assigned.

**Fig. 3.** Bagging algorithm for creating an ensemble of single-layered CVNN.

**Figure 1**

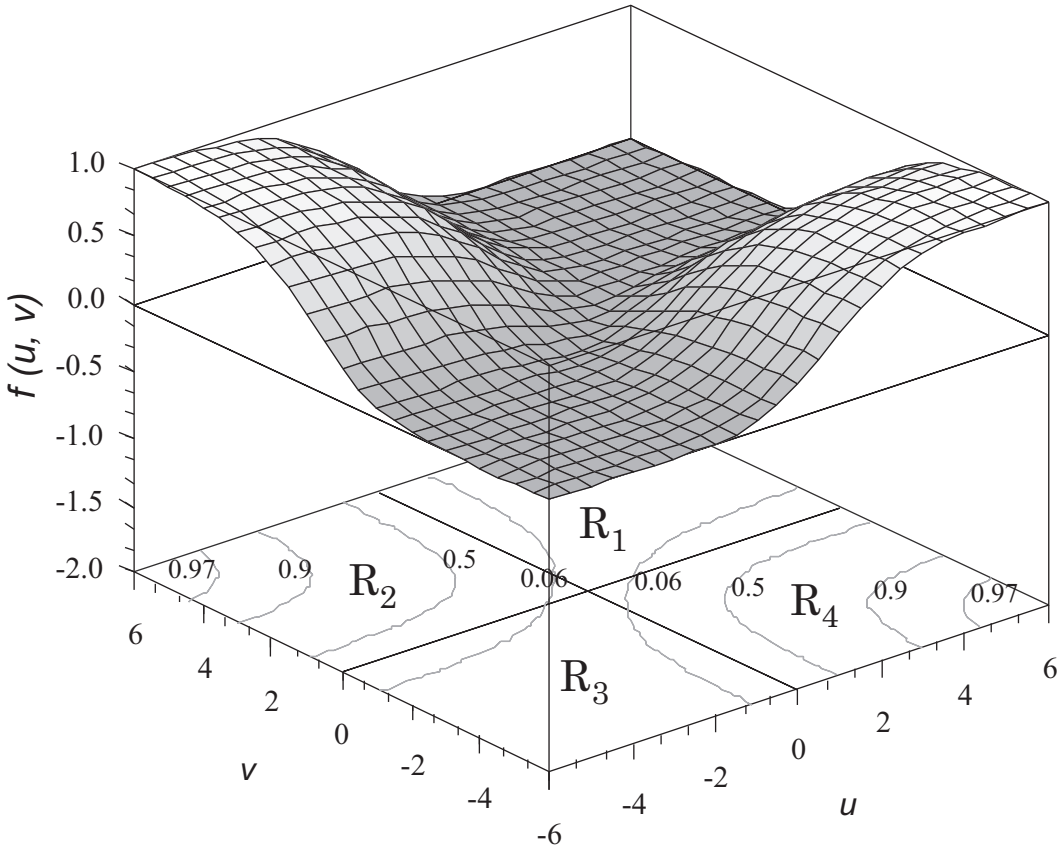
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
f (u, v)
0.97
0.9
0.5
0.06
0.06
0.5
0.9
0.97
R1
R2
R3
R4
6
4
2
0
-2
-4
-6
v
u

**Figure 2**

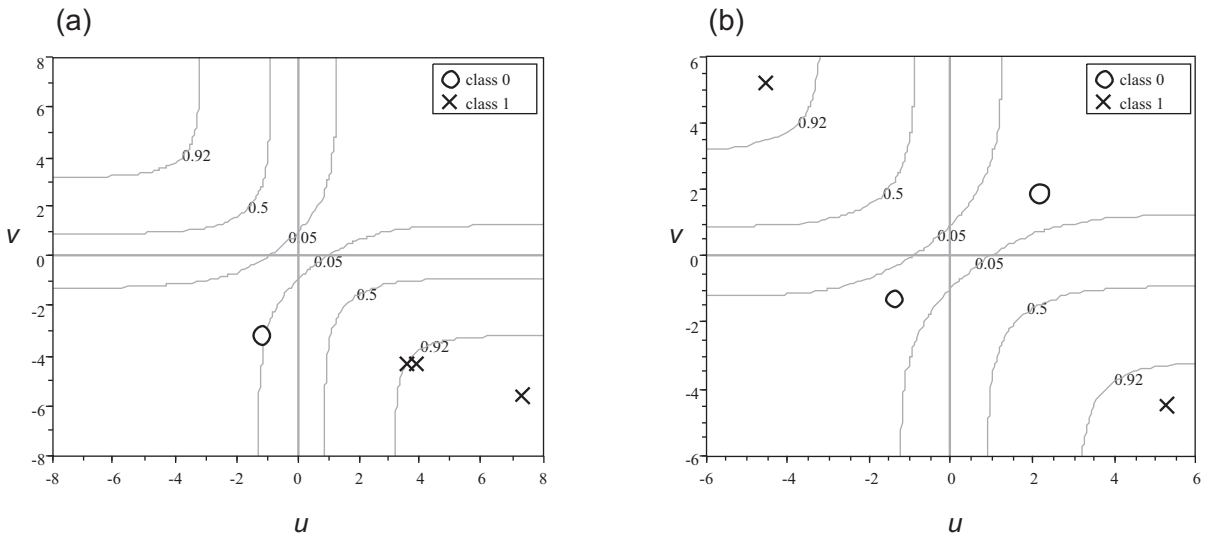
(a)
class 0
class 1
0.92
0.5
0.05
0.05
0.5
0.92
v
u
(b)
class 0
class 1
0.92
0.5
0.05
0.05
0.5
0.92
v
u

Figure 3

***Algorithm: Bagging***

**Input**:

- Training data $D$ with correct labels $\omega_i \in \Omega = \{\omega_1, \ldots, \omega_C\}$ representing $C$ classes
- Integer $T$ specifying number of iterations
- Percentage $F$ to create bootstrapped training data

**Do** $t = 1, \ldots, T$

1. Take a bootstrapped replica $D_t$, by randomly drawing, with replacement, $F$ percent of training patterns from $D$.

2. Create and train $CVNN_t$ with the training subset $D_t$.

3. Add $CVNN_t$ to the ensemble.

**End**