Data Classification using Quantum Neural Network

Dr. Ghassan H. Abd Al- Majeed
Professor.
Computer center- Baghdad University
Baghdad, Iraq
Email: ghmajeed@gmail.com

Dr. Zainab T. Alisa
Assistant Professor.
Electrical Engineering- Baghdad University
Baghdad, Iraq
Email: zainab_alisa@yahoo.com

Hassan Saadallah Naji
Department of Electrical Engineering
College of Engineering
Baghdad University
Email: hassanmaster79@yahoo.com

ABSTRACT

In this paper, integrated quantum neural network (QNN), which is a class of feedforward neural networks (FFNN's), is performed through emerging quantum computing (QC) with artificial neural network(ANN) classifier. It is used in data classification technique, and here **iris flower data** is used as a classification signals. For this purpose independent component analysis (ICA) is used as a feature extraction technique after normalization of these signals, the architecture of (QNN's) has inherently built in fuzzy, hidden units of these networks (QNN's) to develop quantized representations of sample information provided by the training data set in various graded levels of certainty. Experimental results presented here show that (QNN's) are capable of recognizing structures in data, a property that conventional (FFNN's) with sigmoidal hidden units lack. In addition, (QNN) gave a kind of fast and realistic results compared with the (FFNN). Simulation results indicate that QNN is superior (with total accuracy of 97.778%) than ANN (with total accuracy of 93.334%).

Keywords: signal classification, artificial neural network, quantum computing, data analysis and fuzziness.

تصنيف البيانات بأستخدام الشبكة العصبية الكمية

أ.د. غسان حميد عبد المجيد
مركز الحاسبة / جامعة بغداد

أ.م.د. زينب توفيق باقر
قسم الهندسة الكهربائية / جامعة بغداد

حسن سعد الله ناجي
قسم الهندسة الكهربائية / جامعة بغداد

الخلاصة

يتطرق البحث الى الشبكة العصبية الكمية كاحدى اصناف الشبكات العصبية الاصطناعية والتي تتم من خلال دمج مجالين مهمين وهما مجال الاحتساب الكمي مع مجال الشبكة العصبية الاصطناعية . تستخدم هذه الدائرة في مجال تقنية تصنيف البيانات, ان بيانات ازهار السوسن (Iris flower database) قد تم اعتمادها هنا كأشارات تصنيف. ولهذا الغرض تم اعتماد تقنية تحليل المركبة الاساس (ICA) كتقنية استخلاص الخصائص من اشارات التصنيف بعد مرحلة المعالجة الاولية للبيانات لغرض تهيئة البيانات, ان الطبقة الوسطية المعروفة بالطبقة الخفية تتألف من عدد من الخلايا العصبية التي تكون متعددة المستويات لغرض

معالجة بيانات التدريب و الاختبار لتعطي بذالك مستويات متعددة من التأكيد. النتائج التجريبية تشير بأن الشبكة العصبية الكمية لها قدرة تمييز البيانات افضل مقارنة بالشبكة العصبية الصناعية و لنفس الغرض, بالاضافة لذالك النتائج تشير بأن الشبكة العصبية الكمية تعطي نتائج تمييز(تصنيف) اسرع و اكثر واقعية اي ان دوائر QNN تعتبر الافضل(بدقة تصل97.778%) مقارنة بالشبكة العصبية الاصطناعية(التي تصل دقتها 93.334%).

الكلمات المفتاحية:تصنيف الاشارات,الشبكة العصبية الصناعية,الاحتساب الكمي, تحليل البيانات, و المنطق المضبب.

1. INTRODUCTION

Quantum neural network (QNN's) is a promising area in quantum computation and quantum information field. In 1997, Lov K. Grover proposed a method that can speed up a range of search applications over unsorted data using Quantum mechanics, **Lov K. Grover, 1997**.

Several models have been proposed in the literature but most of them need clear hardware requirements to implement such models, one of the most exciting emerging technologies is quantum computation, which attempts to overcome limitations of classical computers by employing phenomena unique to quantum-level events, such as nonlocal entanglement and superposition. It is therefore not surprising that many researchers have conjectured that quantum effects in the brain are crucial for explaining psychological phenomena, including consciousness, **Abninder, 2006**.

Jarernsri. L. Mitrpanont, and Ananta Srisuphab, presented the approach of the quantum complex-valued backpropagation neural network or QCBPN, the challenge of their research is the expected results from the development of the quantum neural network using complex-valued backpropagation learning algorithm to solve classification problems, **Jarernsri, 2003**.

Independent component analysis (ICA) is essentially a method for extracting useful information from data. It separates a set of *signal mixtures* into a corresponding set of statistically independent component signals or *source signals*. ICA belongs to a class of *blind source separation* (BSS) methods for separating data into underlying informational components, **Isabelle, 2006**.

The mixtures can be sounds, electrical signals, e.g., electroencephalographic (EEG) signals or images (e.g., faces, and Functional Magnetic Resonance Imaging (FMRI) data). The defining feature of the extracted signals is that each extracted signal is statistically independent of all the other extracted signals, **James, 2004.**

2. METHODLOGY FOR INTEGRATED QNN SIGNAL CLASSIFIER SYSTEM

The overall block diagram that shows the structure of integrated QNN as signal classifier system is shown in **Fig.1**. Every single recorded input signal Iris signals database is depicted or formed by [1X50] discrete data matrix, and represents a vector pattern. Two different Iris data sets are formed for training and testing purposes. The discrete dataset has three different classes species, the structure of integrated QNN signal classifier system can be shown by the following principal steps**:**

2.1 Normalization

Normalization is a process to simplify data as feature extraction. It is usually affected by peak-to-peak magnitudes and offset of input data because of physiology conditions surrounding, psychological state, artifacts; therefore, normalization mainly required to decrease the effects of undesirable parameters and offset.

2.2 Feature Extraction

Feature extraction or dimensionality reduction is the process of extracting useful information from the signal, features are characteristics of a signal that are able to distinguish between different classes species. Feature extraction requires reducing the size of the data by selecting appropriate features, selected features should be minimally redundant and the expected results should maximally depend on these features, and preserve all information from the signal that is needed for classification.

In ICA, each signal is described as a scalar variable, and a set of signals as a vector of variables, and the process of obtaining signal mixtures from signal sources using a set of mixing coefficients, **Isabelle, 2006**.

$$x_1 = a_{11}s_1 + a_{12}s_2 \quad (1)$$

$$x_2 = a_{21}s_1 + a_{22}s_2 \quad (2)$$

Above equations can be rewritten using matrix –by-vector form as:

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (3)$$

$$\mathbf{S} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \quad (4)$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad (5)$$

Then **Eq. (1)** can be written in matrix by vector form as follow:

$$\mathbf{X = A\ S} \quad (6)$$

Where, (a_{11}, a_{12}, a_{21}, a_{22}), a set of mixing coefficients, (s_1, s_1) are original signals(source signals), and(x_1, x_2) set of "mixture" points which can be transformed back to the source signals (s_1, s_1) using a set of unmixing coefficients, which reverse the effects of the original geometric transformation from source signals to signal mixtures, **Joshua, 2000**.

$$s_1 = \mu x_1 + \sigma x_2 \quad (7)$$

$$s_2 = \gamma x_1 + \delta x_2 \quad (8)$$

Above equations can be rewritten using matrix –by-vector form as:

$$\begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} \mu & \sigma \\ \gamma & \delta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (9)$$

$$W = \begin{bmatrix} \mu & \sigma \\ \gamma & \delta \end{bmatrix} \quad (10)$$

Then **Eq. (9)** can be rewritten using matrix –by-vector form as:

$$S = W X \tag{11}$$

Where (μ, σ, γ, δ), a set of unmixing coefficients.

2.3 QNN and its Learning Algorithm

FFNNs must use the sample information as a mere reference for creating the internal representations, thus, it should not encode the sample information accurately into the internal representations. Such an exact or faithful encoding of the sample information results in the FFNN memorizing the "crispness" in the training data set. But an inherently fuzzy architecture should be capable of generalizing the sample information into various graded levels of certainty over the entire feature space. This may be possible if the architecture is capable of creating graded internal representations from the sample information. The QNN is as architecture capable of allowing the sample information to be encoded into certain levels grades of certainty/uncertainty only, **Jarernsri, 2003**.

One simple way of incorporating the ability to form consistent multilevel partitions in the hidden layer is to create hidden unit partitions with the property of "spreading-out" over regions of uncertainty in the feature space and collapsing-in over regions of certainty. If all the hidden unit transfer functions have the ability to form "graded" partitions instead of the crisp linear partitions, then these partitions can be "collapsed-in" or "spread-out" as required, using a suitable algorithm. Such an algorithm will not require that the fuzzy measures on the feature space be known, but will be a general procedure for learning the imprecision and uncertainty in the data set. This motivates the study of hidden units with *multilevel* transfer functions, **Jarernsri, 2003** and **Gopathy, 1996**.

Suppose the multilevel hidden unit has n_s discrete *states* or *levels*. Then its transfer function can be written as a superposition of ns sigmoidal functions, each shifted by θ^r. The output of this multilevel unit can be written as $(1/n_s)\sum_{r=1}^{n_s} \mathrm{sgm}(v^T x - \theta^r)$ where v is the connected weight matrix between input and the hidden units in hidden layers, and x is the input feature vector . The step widths of the multilevel transfer function, which may be called the quantum intervals, will be representative of discrete localized cells in the feature space consisting of feature vectors with approximately the same level of uncertainty as to their membership to the classes in the data set. These quantum intervals "jump-positions" θ^r unlike the step widths, step heights need not be learned by independent parameters because several sigmoid can be shifted to the same location and added together to give steps of desired heights, to an approximation which reduces total number of parameters to be learned by almost one-third, **Gopathy, 1996**.

QNN consists of n_i inputs, one hidden layer of n_h nodes, each one represents a multilevel units and n_o outputs. Output units can be linear or sigmoid. Let $\mathbf{x_k}=[x_{1,k}, x_{2,k}, \ldots, x_{n,k}]^T$, k=1,..,.m , be the k^{th} feature vector of the data set **X**. Then the input to the j^{th} hidden unit from the k^{th} feature vector is:

$$h^{-}_{j,k} = \sum_{l=0}^{n_i} v_{jl}\, x_{l,k} \tag{12}$$

$$h^{\sim}_{j,k} = \frac{1}{n_s}\sum_{r=1}^{n_s} h^{r}_{j,k} = \frac{1}{n_s}\sum_{r=1}^{n_s} \mathrm{sgm}(\beta_h(h^{-}_{j,k} - \theta^{r}_{j})) \tag{13}$$

Where $\tilde{h}_{j,k}$: the response of the j^{th} multilevel hidden unit from k^{th} is feature , $sgm(\tau) = \frac{1}{1+e^{(-\tau)}}$: is a sigmoid function (unipolar), β_h a slope factor for all multilevel hidden units in hidden layer, θ_j^r define the jump positions in transfer function, and n_s is the number of levels or sigmoids in the multilevel hidden unit, **Fig. 2 (a)** plots the response $\tilde{h}_{j,k}$ of j^{th} four level quantum neuron as a function of its input $\bar{h}_{j,k}$ with equal step heights and **Fig. 2 (b)** demonstrates multilevel transfer function with unequal step heights through simple shifting ,**Jarernsri, 2003**.

$$\bar{y}_{i,k} = \sum_{j=0}^{n_h} w_{ij}\, \tilde{h}_{j,k} \tag{14}$$

With $\tilde{h}_{0,k} = 1$, $\forall k$ therefore, the response of the i^{th} output unit for the k^{th} input feature vector can be written as:

$$\hat{y}_{i,k} = sgm(\beta_o\, (\bar{y}_{i,k})) \tag{15}$$

The major steps of QNN learning algorithm are summarized and presented as follows:

A) Update the synaptic weights:

Step 1: Select $\alpha, \alpha_\theta, \beta_h, \beta_o$ (by trail and error) and randomly initialize the weights (**W** & **V**) and values of jump positions θ_j^r.

Step 2: Present k^{th} input pattern and specify the desired output.

Step 3: Calculate actual output $\hat{y}_{i,k}$, using the present values of V_{jl} and W_{ij}.

Step 4: Find the error terms ($e^{o}_{i,k}$) and ($e^{h}_{j,k}$).

Calculate the output error.

$$e^{o}_{i,k} = (y_{i,k} - \hat{y}_{i,k})\; \hat{y}_{i,k}(1 - \hat{y}_{i,k}) \tag{16}$$

Calculate the hidden layer error term.

$$e^{h}_{j,k} = \left(\frac{1}{n_s}\sum_{r=1}^{n_s} h^{r}_{j,k}(1-h^{r}_{j,k})\right)\sum_{p=1}^{n_o} e^{o}_{p,k} w_{pj} \tag{17}$$

p is over all nodes in the layer above node j.

Step 5: Adjust the synaptic weights:

$$w_{ij,k} = w_{ij,k-1} + \alpha e^{o}_{i,k} \frac{1}{n_s}\sum_{r=1}^{n_s} sgm(\beta_h\,(\bar{h}_{j,k} - \theta_j^r) \tag{18}$$

$$v_{jl,k} = v_{jl,k-1} + \alpha\,\beta_h\,(\frac{1}{n_s}\sum_{r=1}^{n_s} h^{r}_{j,k}\,(1 - h^{r}_{j,k})\, x_{l,k} \tag{19}$$

Where, k + l , k , and , k - l index next, present, and previous respectively, and is a learning rate.

Step 6: Present another input pattern and go back to step 2
All of the training samples are presented cyclically.

B) Update the quantum intervals:
Step 1: In each training cycle, calculate the outputs $\hat{y}_{i,k}$, and $v^r_{q,k}$ for each hidden node.
Then, take the average values for each class, $\langle \tilde{h}_{j,c_m} \rangle$, $\langle v^s_{j,c_m} \rangle$, for m^{th} class during the training of QNN is as:

$$\langle \tilde{h}_{j,c_m} \rangle = \frac{1}{|c_m|} \sum_{x_k : x_k \in c_m} \tilde{h}_{j,k} \quad (20)$$

$$\langle v^r_{j,c_m} \rangle = \frac{1}{|c_m|} \sum_{x_k : x_k \in c_m} v^r_{j,k} \quad (21)$$

$|c_m|$ = The cardinality of m^{th} class

Step 2: Calculate the quantum interval adjustment $\Delta\theta^r_q$ for each level:

$$\Delta\theta^r_j = \alpha_\theta \frac{\beta_h}{n_s} \sum_{m=1}^{n_o} \sum_{x_k : x_k \in c_m} \left(\langle \tilde{h}_{j,c_m} \rangle - \tilde{h}_{j,k}\right)\left(\langle v^s_{j,c_m} \rangle - v^s_{j,k}\right) \quad (22)$$

Where α_θ **is** the learning rate.

Step 3: Update jump-positions by:

$$\theta^r_j = \theta^r_j + \Delta\theta^r_j \quad (23)$$

Step 4: Continue next cycle and go back to step 2.

Nomenclature:
n_o= number of output nodes.
n_i= number of input nodes.
n_s= number of quantum interval.
θ^r_j = value of r^{th} jump-position of hidden node j.
w_{ij} = the strength of connection between j^{th} hidden node and i^{th} output node.
v_{jl}= the strength of connection between l^{th} input node and j^{th} hidden node.
$\hat{y}_{i,k}$ = actual output at output node i for k^{th} input pattern.
$y_{i,k}$= desired output at output node i.
$e^o_{i,k}$ & $e^h_{j,k}$ = error terms of output node i and hidden node j for k^{th} input pattern.
$h^r_{j,k} = \text{sgm}\,(\bar{h}_{j,k} - \theta^r_j))$
$\bar{h}_{j,k} = \sum_{l=0}^{n_i} v_{jl}\, x_{l,k}$, the internal state of hidden node j for k^{th} input pattern.
$\tilde{h}_{j,k} = \frac{1}{n_s} \sum_{r=1}^{n_s} h^r_{j,k}$, the output of j^{th} hidden node.

$v^r_{j,k} = h^r_{j,k}(1 - h^r_{j,k})$.

3. RESULTS AND DISCUSSION

3.1 Results of using ANN Classifier

Three layer ANN was employed as classifier, for Iris data set signal, network structure is 4-16-3 namely input vector (n_i) is equal to (4) representing number of input variables. The 4 rows (sepal length, sepal width, petal length, and petal width) contained a single hidden layer, number of hidden units was chosen by trial and error; which revealed that number of neurons at hidden layer (n_h) is at least greater than twice of input nodes (n_i) correspond to features of each species, as an assumption ($n_h = 4*n_i$) for good performance ($n_h = 16$) traditional neuron with sigmoid activation function (unipolar), and output layer contains number of neuron equivalent to classes (species), ($n_o = 3$), structure of the ANN classifier is shown in **Fig.3** with number of nodes at each layer.

Three layers ANN was employed as classifier of Iris data signal, randomly select input feature vectors to achieve the uncertainty principle. Randomly selected 70% samples as training samples and 30% for testing samples. Learning rate (η) is chosen by trial and error for weigh adjusting(W_{ij}, V_{jl}) is set to (0.01) MATLAB programming test, the number of iteration (epochs) is set to 1500.

The performance of ANN classifier (for both training and testing phases) is shown in **Fig.4** and for training data set; ANN gave a Mean Square Error (MSE) of (0.0054) and accuracy of (100.00%).

For testing phase, ANN classifier showed a mean square error MSE of (0.1174) with accuracy of (93.334%). **Fig.4** shows the relationship between MSE and number of iteration for ANN classifier. As can be seen that MSE is decreasing with increasing number of iterations, which revealed that the network is converged with iteration number of (1500) and MSE of (0.0054).

3.2 Results of using QNN Classifier

Three layers QNN was employed as classifier of Iris data signal, the performance of the QNN was tested to perform classification on Iris data signal. Randomly selected 70% of the samples are used for training (training input feature vectors) and 30% for testing (testing input feature vectors), the structure is 4-8-3 for the neural network, input vector(n_i) is equal to (4).

The number of multi-level hidden units (which are used in the hidden layer of QNN rather than traditional neurons as in ANN) with the number of multi-level neurons (n_h) is no more than twice of the input nodes (n_i), in such structure of ($2*n_i$), i.e. (n_h=8) multi-level hidden units with sigmoidal activation function (unipolar), and the output layer contains neurons equivalent to number of species ($n_o = 3$). The major difference between QNN and ANN is that the QNN uses quantum neuron (multi-level neuron (graded) with sigmoid activation function), structure of QNN classifier is shown in **Fig.5**). It contains a single hidden layer with ($n_h = 8$) units, three output units (no. of species) and 4 input nodes, also identifies by existence of jump-positions (thetas' values) of multi-level hidden units of the QNN hidden units were chosen by trial and error.

The QNN is composed of multi-level hidden units with ($\mathbf{n_s} = 3$) (chosen by trial and error) levels for each hidden unit, the learning rate (learning ratio) (η) for weigh adjusting(W_{ij}, V_{jl}**)** is set to (0.07) training by MATLAB test, and the learning rate for quantum interval adjusting (eta_theta η_θ) is set to (0.001), and slop factor for unit at hidden layer $\beta_h = 2$, but for output layer slop factor $\beta_o = 1.5$, the number of iteration (epochs) is set to (300). Synaptic weights(W_{ij}, V_{jl}**)** are adjusted by

minimizing quadratic error function with respect to particular weights, in training itself; jump-positions of multi-level hidden units adjusted also.

Results are done with (70%) of input feature vectors for training phase and (30%) of input feature for testing phase, performance of QNN classifier shown in **Fig.6** and summarized with (1)MSE of (0.1258) and accuracy of (97.143%) for training phase (2) MSE (0.3780) with accuracy of (97.778%) for testing phase. **Fig.6** displayed the relationship between MSE and number of iteration for QNN classifier, in which reveals that MSE is decreasing with increasing number of iterations, but when the iteration number exceeded 150 the MSE decreases very slightly which means the network is converged with iteration number of (150) and MSE of (0.1258). As in **Fig.4**, **Fig.6** shows an inverse relationship between MSE and number of iterations.

Main difference between two figures is that QNN converges with less iteration, **Fig.6** shows convergence occurs at (150) in QNN classifier compared with (1500) for ANN.

3.3 QNN vs. ANN Classifiers

To discuss the results of ANN and QNN classification for Iris data signals (QNN vs. ANN classifiers), table 1 shows a comparison between them. Two issues can be concluded from table1, first one is that QNN classifier gave better accuracy for testing phase compared with ANN classifier (97.778% compared with 93.334%) and second issue is that QNN converged with less number of iterations (150 epochs for QNN compared with 1500 epochs for ANN), and this indicates that the time required for QNN convergence is about (90%) less than that of ANN.

The reason is that ANN is unable to correctly assign class membership to data samples belonging to regions of the feature space where there is overlapping among the classes. The reason for this is that FFNNs use sharp decision boundaries (due to crisp membership function) to partition the feature space. As a result, the outputs of trained ANNs cannot generally be interpreted as membership values. Also it can be found that QNN is more reliable than the ANN because QNN generates a more structured representation of the input data at the hidden layer than that of the ANN as QNN use multilevel hidden units, this is not surprising, given the fact that the jump-positions of the multi-level hidden units of the QNN are updated by minimizing some measure based on the class-conditional variances at the outputs of the hidden units.

Another advantage is that QNN systems are using quantum neuron instead of traditional neuron which is often able to learn faster and require less number of neurons in the hidden layer which could lead to a smaller number of weights and reduction of the number of neurons in the hidden layer which could lead to smaller number of weights, or it can be said generally that this means reducing the total number of parameters (input weights, output weights, jump position) to be learned by almost half, as an assumption to the total number for hidden units in hidden layer is with($4*n_i$) empirically for best accuracy in ANN classifier, while assumption to the total number for multilevel hidden units in hidden layer is with($2*n_i$) empirically for best accuracy in QNN classifier. This means that the total number of parameters (input weights, output weights, jump position) to be learned is reduced by almost half which represents another advantage for QNN.

Here the number of multilevel hidden units in the hidden layer was set to (8) by QNN but (16) by ANN, the input weights is (**V**(8x4)) and output weight (**W**(3x8)) and jump position with (8x3)matrix by QNN while the input weight is (**V**(16x4)) and output weight is (**W**(3x16)) for ANN.

4. CONCLUSSIONS

In this paper, a data classification system based on multi-level transfer function integrated Quantum Neural Networks (QNN) is proposed. The classification system methodology consists of ion signals which here iris data. First, the classification signals should be normalized then feature extraction is applied using independent component analysis technique then classification task is to be achieved firstly using artificial neural network classifier and secondly using integrated Quantum Neural Networks (QNN).

The architecture of (QNN's) has inherently built in fuzzy, hidden units of these networks (QNN's) to develop quantized representations of sample information provided by the training data set in various graded levels of certainty. Experimental results presented here show that (QNN's) are capable of recognizing structures in data, a property that conventional (FFNN's) with sigmoidal hidden units lack. In addition, (QNN) gave a kind of fast and realistic results compared with the (FFNN). Simulation results indicate that QNN is superior (with total accuracy of 97.778%) than ANN (with total accuracy of 93.334%).

REFRENCES

Abninder L., Chris E., Frederick W. K., Steven W., and Paul T.,2006, *Is the Brain a Quantum Computer?,* Cognitive Science Society, University of Waterloo, No.30, pp. 593–603.

Gopathy Purushothaman and Nicolas. B. Karayiannis, 1996, *Quantum Neural Networks (QNNs): Inherently Fuzzy Feedforward Neural Networks*, IEEE Transactions On Neural Networks, Vol.2, pp. 1085-1090.

Isabelle G., Steve ., Masoud N., and Lotfi A. Z., 2006, *Feature Extraction Foundations and Applications, USA and United Kingdom.*

James V. Stone, 2004, *Independent Component Analysis A Tutorial Introduction*, A Bradford Book The MIT Press Cambridge, Massachusetts London, England.

Jarernsri. L. Mitrpanont, and Ananta Srisuphab**,** 2003 *The Realization of Quantum Complex-Valued Backpropagation Neural Network in Pattern Recognition Problem*.The 9th. International Conference on Neural Information Processing (ICONIP'OZ) Vol. 1.

Joshua B. T., Vin de Silva, and John C. L., A, 2000 *Global Geometric Framework for Nonlinear Dimensionality Reduction, Science,* Vol.290, pp. 2319-2323, USA.

Lov. K. Grover, 1997 *Quantum Mechanics Helps in Searching for a Needle in a Haystack*, American Physical Society, Vol 79,Issue 2, pp. 325–328.

Xin-Yi T., Yu-ju J. Chen, S. C., and Rye C. Hwang, 2005, *Quantum NN vs. NN in Signal Recognition*, Proceedings of the Third IEEE International Conference on Information Technology and Applications.

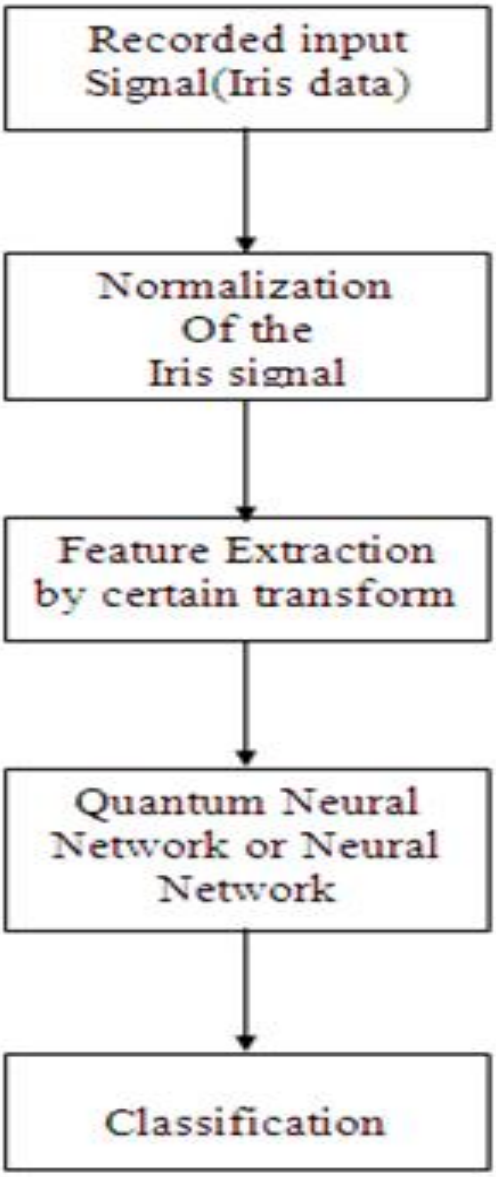
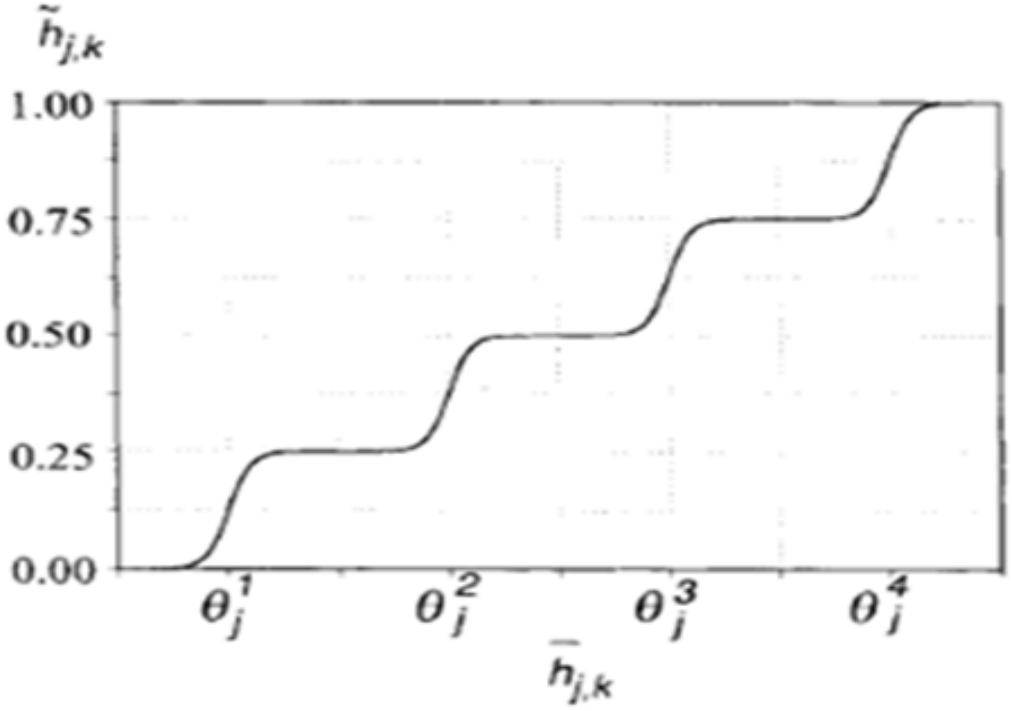
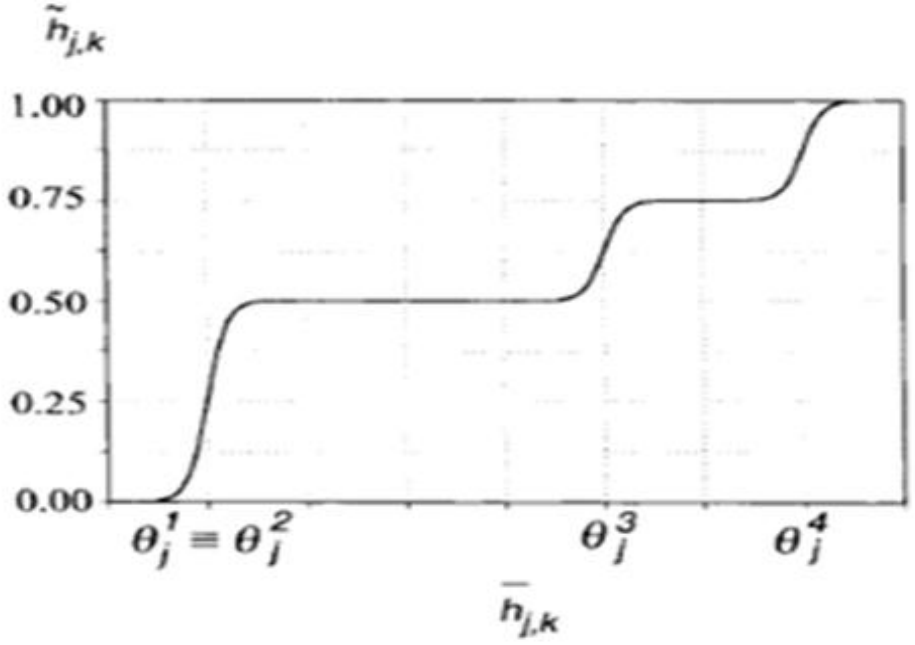


Figure 1. Block diagram of the methodlogy for signal classification system.



(a)



(b)

Figure 2. Multilevel transfers function with (a): equal step heights (b): unequal step heights.

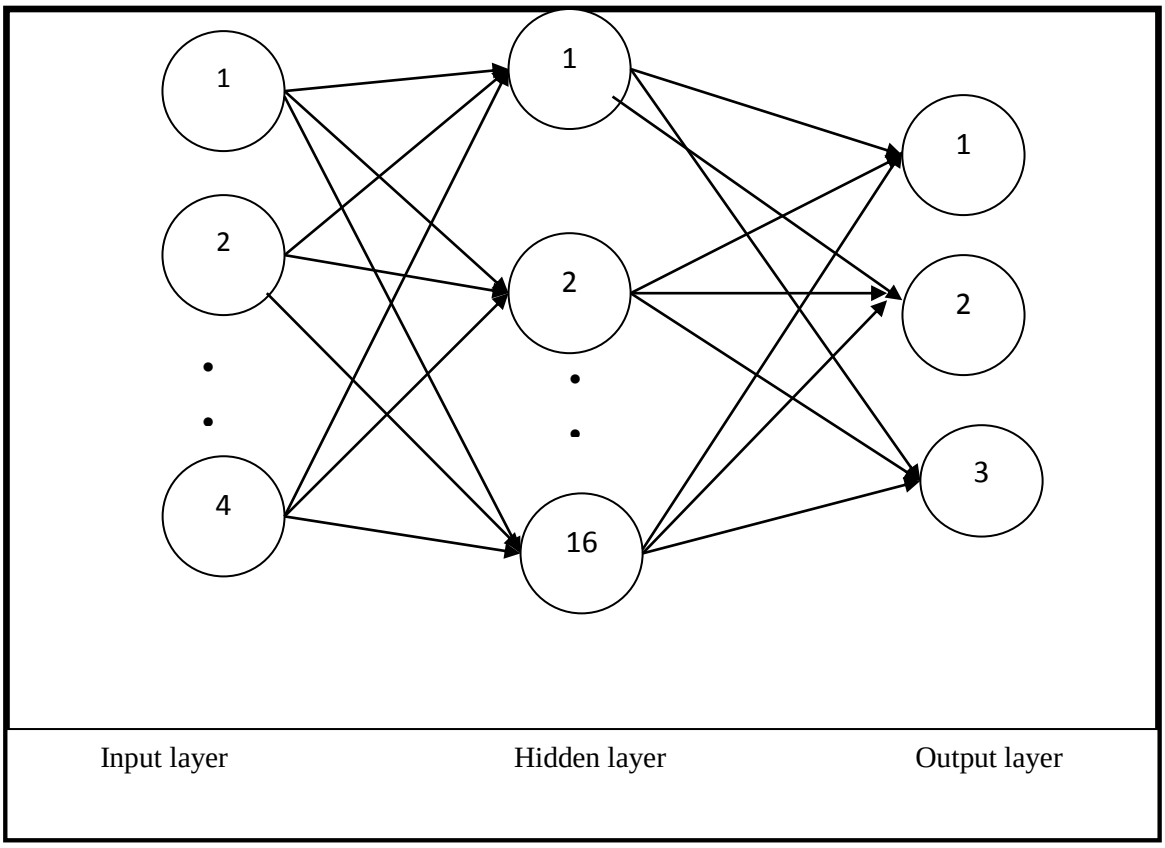


Figure 3. Structure of the Artificial Neural Network (ANN) classifier.

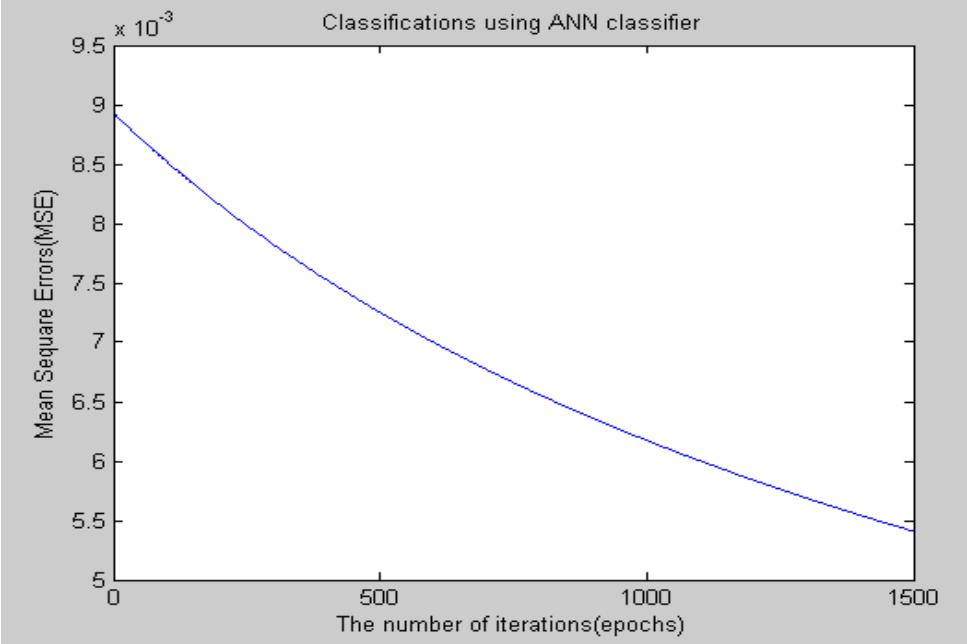


Figure 4. Classification of signals by ANN classifier.

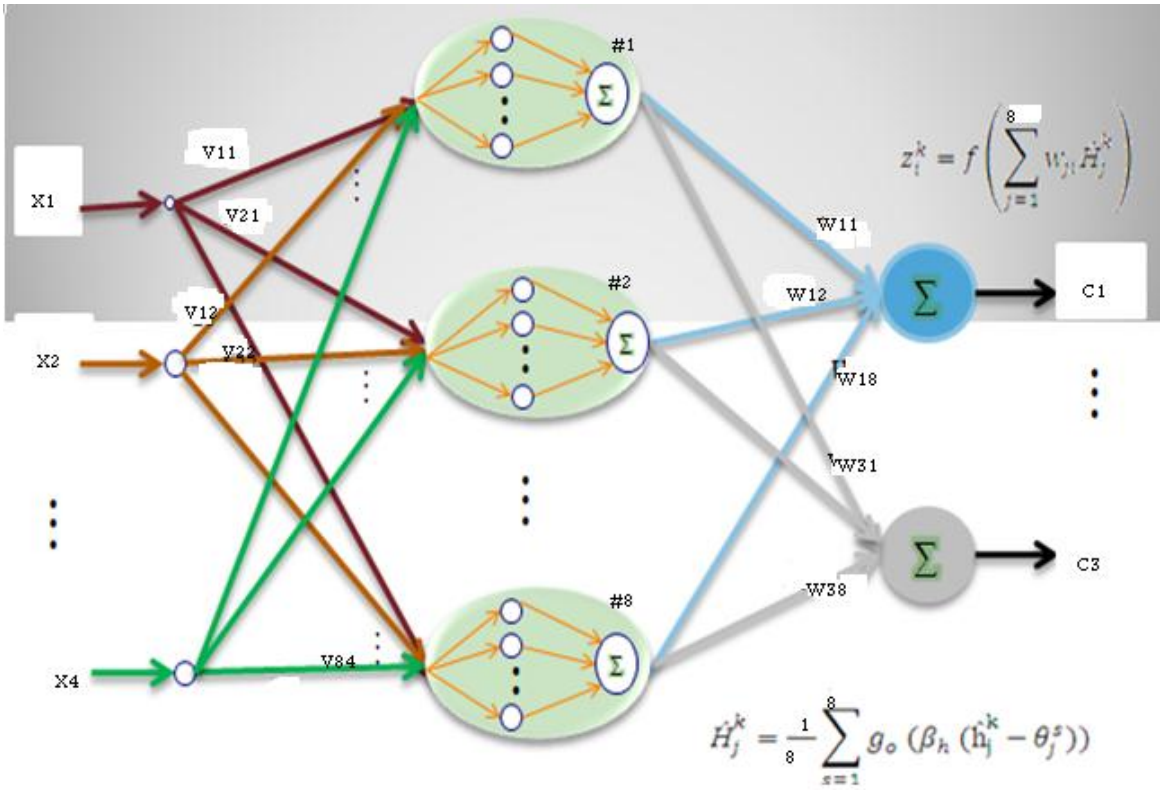


Figure 5. Structure of Quantum Neural Network (QNN) classifier.

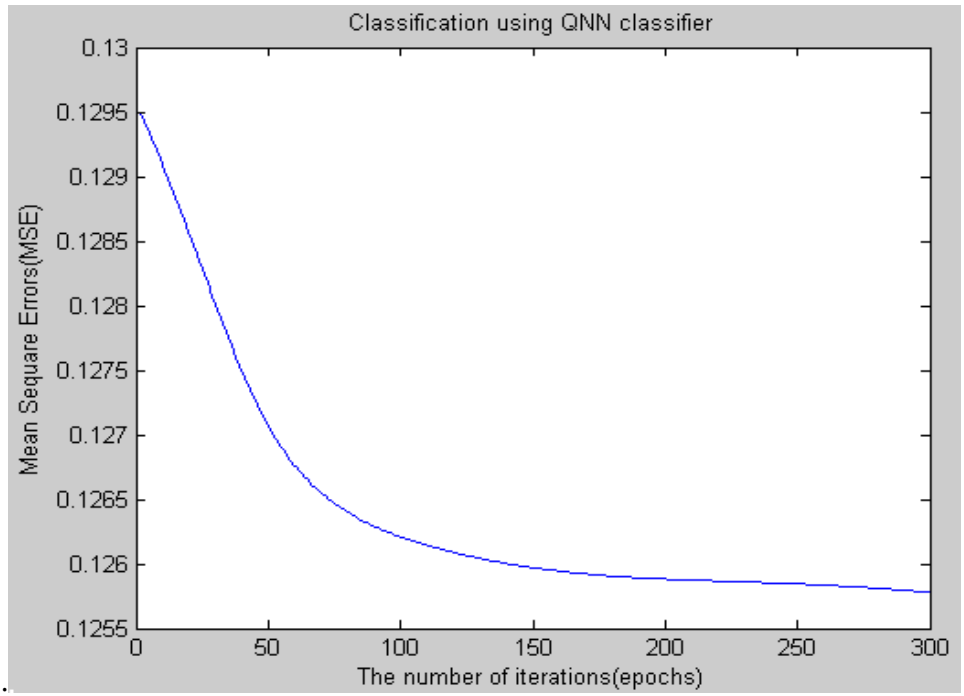


Figure 6. classification of signals by QNN classifier.

Table 1. Classification and performance using ANN and QNN classifiers.

Network classifier type	MSE for training set	Accuracy for training set (%)	No. of epochs for convergence	MSE for testing set	Accuracy for testing set (%)
ANN classifier	0.0054	100.00	1500	0.1174	93.334
QNN classifier	0.1258	97.143	150	0.3780	97.778