INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2016, VOL. 62, NO. 4, PP. 401-408
Manuscript received April 19, 2016; revised November, 2016. DOI: 10.1515/eletel-2016-55

Classification of EEG Signals Using Quantum Neural Network and Cubic Spline

Mariam Abdul-Zahra Raheem and Ehab AbdulRazzaq Hussein

***Abstract*— The main aim of this paper is to propose Cubic Spline-Quantum Neural Network (CS-QNN) model for analysis and classification of Electroencephalogram (EEG) signals. Experimental data used here were taken from seven different electrodes. The work has been done in three stages, normalization of the signals, extracting the features by Cubic Spline Technique (CST) and classification using Quantum Neural Network (QNN). The simulation results showed that five types of EEG signals were classified with an average accuracy for seven electrodes that is 94.3% when training 70% of the features while with an average accuracy of 92.84% when training 50% of the features.**

***Keywords*—EEG Signals, ERP Signals, Cubic Spline, Neural Networks, Quantum Neural Networks**

I. INTRODUCTION

THE biomedical engineering interested dramatically in the automatic classification of Electroencephalogram (EEG) signals. Because the biomedical signals, inherently unstable and randomly change over time depending on the change and mental health conditions and situations of tension for the same person, and one of these signals is brain signal that varies according to the psychological state of the person himself and changed depending on the circumstances, all of this has paid great attention to the analysis of brain signals. The EEG is the registration of electrical activity on the scalp. Current flow due to firing of nerve cells in the brain results in a voltage wiggle that measured as EEG [1]. Measuring the brain's response to a stimulus is called event-related potential (ERP). The stimulus can be motor, sensory, or cognitive naturally. Human ERPs are usually recorded from electrodes placed on the human scalp.

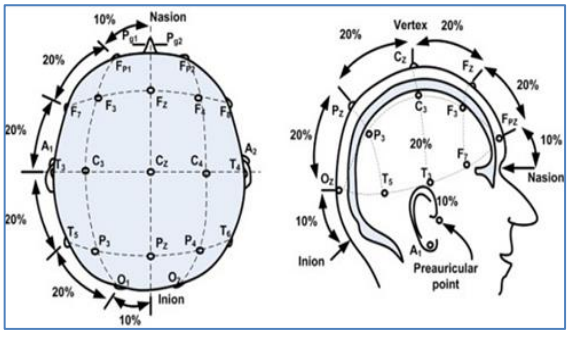

Fig. 1. The placement of electrodes on the scalp according to the international (10-20) standard.

Authors are with University of Babylon, College of Engineering, Electrical Engineering Department, Iraq (email: {ehabalhialy; mariam.raheem} @yahoo.com).

The placement of electrodes on the scalp according to the international (10-20) standard is shown in fig. 1 [2]. The popular way of analyzing event-related EEG signals is the computation of ERPs. This can be done by repeating an event of interest such as a visual stimulus of a computer screen and analyzing a small fraction of the EEG activity that is evoked by this event [3]. The feature extraction technique of EEG signals provides an accurate features in which would to classify between any event-related potentials of the brain using QNN. Five types of EEG signals are used. Each type of these signals is a mental task assigned to a particular person to perform it. These tasks are (baseline, multiplication, letter composing, rotation, counting).

II. FEATURE EXTRACTION AND SELECTION

A feature is a distinct or characteristic measurement, change, basic component that can extract from a slice of a signal. The features used to represent the signals without losing the important information about these signals. The feature extraction process is the determination of the feature or the feature vector from the signal. So as to make signal processing problems can be solved, need to convert signals to features which will become abbreviated representations of the signals, which only contain important information for the signal. The aim of this section is to identify convenient input feature vectors which would discriminate between the event-related potentials of the brain. In this work, EEG signal analysis is divided into five parts: normalization, knot extraction, knot selection, feature extraction and classifier.

A. Normalization

The normalization is a process of removing, the difference in the amplitude between the signals in each kind or category. In order to make the data has a zero mean; the computed mean of data will be subtracted from the raw EEG signal. Normalization steps can be summarized as follows:

1. For EEG signals, mean value of data is stable to zero value. Thus, the offset will be removed from the signal.
2. Subtract the mean value from the raw EEG signal.
3. After subtraction, the mean value of the original EEG signal, must be zero or nearby zero.

B. Knot extraction

The accurate analyses of the EEG signal have special importance in this system. Where extract the features of the EEG signal depends firstly and dramatically on determining characteristic points sites, whenever these sites were accurate,

the extracted features of the signals were accurate too. A technique that is used to find the knots largely depended on the shape of the EEG signal. These knots which are a part of the data can found by sorting data in ascending order after the data is divided into many parts and then to find the upper and lower values in every part of the data parts. This process will be repeated multiple times and in every time the number of the extracted knots will be increasing and will be saved in a matrix and will remove the duplicate points meaning that knots formerly extracted will be removed in order to prevent repetition.

C. Knot Selection

The extracted knots may not represent significant changes in the signal, or may be convergent between values, so that, in order to solve this problem, each extracted knots will be considered as a new data and will insert into the same technique previously used and thus will get only the distinctive knots that represent significant changes of the signal and will delete excess of the knots (the last knots) without losing the basic information of the signal.

D. Feature extraction using Cubic Spline Technique

Cubic spline interpolation is a useful technique to interpolate between known data points due to its stable and smooth characteristics [4]. The objective of the cubic spline interpolation is to get an interpolation formula, which is continuous in both first and second derivatives, both inside the intervals and at the interpolating knots. This will give function interpolation more smoothly. Generally, if the function to be approximate was smooth, then the cubic splines will do what is better than the piecewise linear interpolation [5].

The function S that consists of n-1 cubic polynomial pieces will be constructed as:

$$S(x) = \begin{cases} s_{1(x)} & if \quad x_1 \le x < x_2 \\ s_{2(x)} & if \quad x_2 \le x < x_3 \\ \vdots & \\ s_{n-1(x)} & if \quad x_{n-1} \le x < x_{n-1} \end{cases} \quad (1)$$

Where S_i is a third degree polynomial defined by:

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 - c_i(x - x_i) + d_i \quad (2)$$

Where n is the number of extracted knots. The cubic spline will need to conform to the following stipulations [6]:

1. The piecewise function $S(x)$ will interpolate all knots.
2. $S(x)$ will be continually on the interval $[x_1, x_n]$.
3. $S'(x)$ will be continually on the interval $[x_1, x_n]$.
4. $S''(x)$ will be continually on the interval $[x_1, x_n]$.

As the piecewise function $S(x)$ will interpolate all of the knots, we can conclude that [7]:

$$S(x_i) = y_i \quad (3)$$

For i = 1, 2, ..., n-1. Since $x_i \in [x_i, x_{i+1}]$

$$S(x_i) = s_i(x_i) \quad (4)$$

The mathematical expression for *Natural Cubic Spline (NCS)* condition is:

$$S''(x_1) = S''(x^n) \quad (5)$$

Two additional conditions are needed to determine the natural cubic splines which they are $e_1 = e_n = 0$ for every subinterval $[x_i, x_{i+1}]$ and the other values of e_i were not yet known. Then the curvatures are linear in an interval and denoting values at the points x_i as:

$$e_i = s_i''(x_i), \quad i = 1,2, \dots, n. \quad (6)$$

They comprise a linear spline for the data set $(x_i, e_i, i = 1,2, \dots, n)$ therefore:

$$s_i''(x) = e_{i+1}\frac{x - x_i}{x_{i+1} - x_i} + e_i\frac{x_{i+1} - x}{x_{i+1} - x_i}, \; i = 1,2, \dots, n-1 \quad (7)$$

By integrating over x twice, will get:

$$s_{i(x)} = \frac{e_{i+1}}{6h_i}(x - x_i)^3 + \frac{e_i}{6h_i}(x_{i+1} - x)^3 + c(x - x_i) + d(x_{i+1} - x) \quad (8)$$

Where

$$h_i = x_{i-1} - x_i. \quad (9)$$

And c and d are constants of integration. The terms of interpolation $s_i(x_i) = y_i$ And $s_{i+1}(x_{i+1}) = y_{i+1}$

$$\frac{e_i}{6h_i}(x_{i+1} - x)^3 + d(x_{i+1} - x) = y_i \quad (10)$$

And

$$\frac{e_{i+1}}{6h_i}(x_{i+1} - x)^3 + c(x_{i+1} - x) = y_{i+1} \quad (11)$$

Applying the C^0 conditions, will get:

$$c = \frac{y_{i+1}}{h_i} - \frac{e_{i+1}h_i}{6}; \quad d = \frac{y_i}{h_i} - \frac{e_i h_i}{6} \quad (12)$$

From the C^1 continuity condition, the other values of e_i can be found and by differentiating equation (8) after compensating c & d, will get:

$$s_i'(x) = \frac{e_{i+1}}{2h_i}(x - x_i)^2 - \frac{e_i}{2h_i}(x_{i+1} - x)^2 + \left(\frac{y_{i+1}}{h_i} - \frac{e_{i+1}h_i}{6}\right) - \left(\frac{y_i}{h_i} - \frac{e_i h_i}{6}\right) \quad (13)$$

And

$$s_i'(x_i) = -\frac{e_i}{2h_i}{h_i}^2 + \left(\frac{y_{i+1}}{h_i} - \frac{e_{i+1}h_i}{6}\right) - \left(\frac{y_i}{h_i} - \frac{e_i h_i}{6}\right) \quad (14)$$

$$s_i'(x_i) = -\frac{e_{i+1}h_i}{6} - \frac{e_i h_i}{3} + b_i \quad (15)$$

Where

$$b_i = \frac{y_{i+1} - y_i}{h_i}, \; i = 1,2, \dots, n-1 \quad (16)$$

Also

$$s_{i-1}'(x_i) = \frac{e_i}{2h_{i-1}}h_{i-1}^2 + \left(\frac{y_i}{h_{i-1}} - \frac{e_i h_{i-1}}{6}\right) - \left(\frac{y_{i-1}}{h_{i-1}} - \frac{e_{i-1}h_{i-1}}{6}\right) = \frac{e_{i-1}h_{i-1}}{6} + \frac{e_i h_{i-1}}{3} + b_{i-1} \quad (17)$$

$$s_{i-1}'(x_i) = s_i'(x_i) \quad (18)$$

By setting equation (18) at all interior points, will obtain:

$$h_{i-1}e_{i-1} + 2(h_{i-1} + h_i)e_i + h_i e_{i+1} = 6(b_i - b_{i-1}) \quad (19)$$

Which is a tridiagonal system of equations in terms of the unknowns e_i's. There are n unknowns and $n-2$ equations. These equations can be rearranged as:

$$\begin{aligned} &e_1 = 0 \\ &h_{i-1}e_{i-1} + u_i e_i + h_i e_{i+1} = v_i \quad i = 2,3, \dots, n-1 \\ &e_n = 0 \end{aligned} \quad (20)$$

Where

$$u_i = 2(h_{i-1} + h_i) \quad (21)$$

$$v_i = 6(b_i - b_{i-1}), \quad i = 2,3, \dots, n-1 \quad (22)$$

By using the tridiagonal solver, the values of $(e_2, e_3, \dots\ e_n)$ can be obtained. These values in addition to $e_1 = e_n = 0$ can then be used to find out the splines by the compensation in equation (8). Figure 2 shows the flowchart of cubic spline technique.

The cubic spline technique approach has many steps can be described as follows:

1. The extracted knot for each signal of EEG signals is considered as interpolation points of the cubic spline.

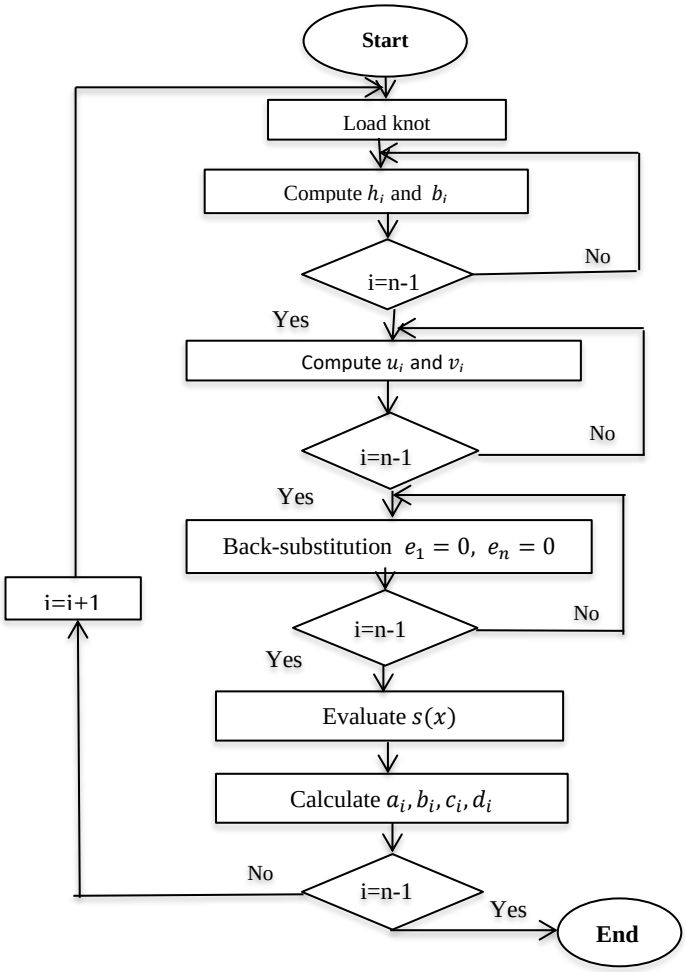

Fig. 2. Flowchart of CST approach.

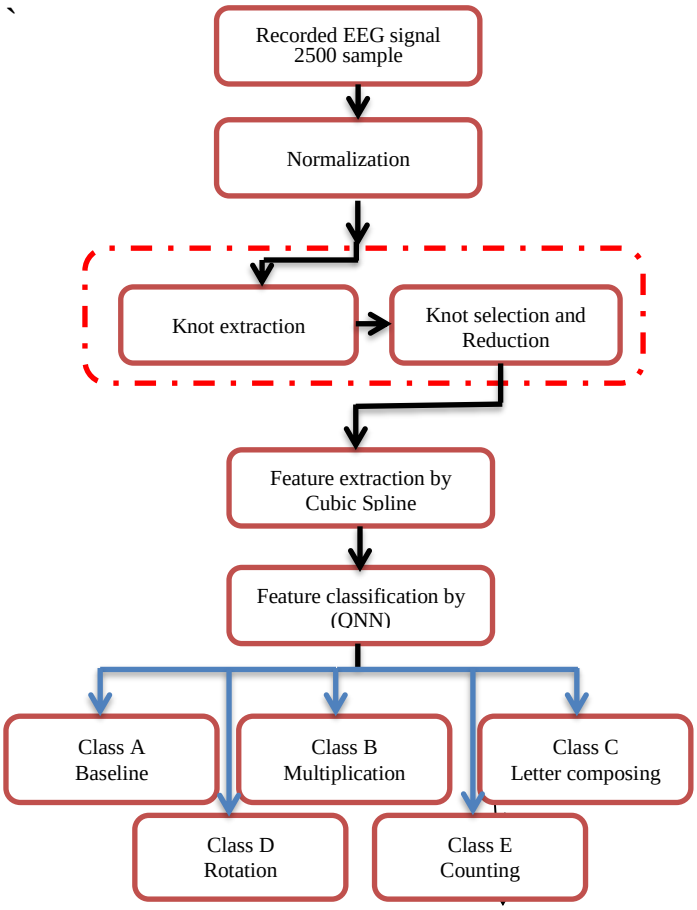

Fig. 3. Structure of EEG Classification System.

2. Forty coefficients are extracted for each beat and these coefficients are rotated to represent one column.
3. Between each neighboring two knots, four coefficients (features) are extracted representing the coefficients of the third degree equation.

III. EEG SIGNAL CLASSIFICATION SYSTEM PROPOSED

The main goal of the proposed system is to analyze the EEG signals depending on the cubic spline as a different way to extract the features of the signals and use QNN for classification. Figure 3 shows the general schematic diagram of the proposed system. Every one signal of EEG signals is formed by 2500 samples, which represents a vector pattern. These vectors in the data set will be normalized. After normalization, knots are extracted and then select the best knot, that is represents the basic characteristics of EEG signal form and its distinctive points. By cubic spline algorithm, the features have been extracted based on these knots. And then after forming the feature vectors, they will be applied to the QNN for classification or training purposes. A three layers (L) QNN used as a classifier for EEG signal, which is use the extracted features obtained by CST for training and testing purposes. The training process was conducted until reached a maximum iteration. QNN contains (5) output neurons (n_o) in the output layer, (40) input neurons (n_i) in the input layer.

IV. QUANTUM NEURAL NETWORK

Quantum Neural Network (QNN) is an energetic science based on the combination of quantum computing and artificial neural network. It also combines the benefits of fuzzy theoretic principles and neural modelling. Are similar to classical neural networks, QNNs has an inherently fuzzy architecture that can degrade the sample information to discrete levels of certainty or uncertainty. The transfer functions of quantum neuron able to form graded sections instead of crisp linear sections of the feature space. If the feature vectors lie at the boundary between interlaced classes, QNN will assign it partly to all related classes. If no certainty exists, QNN will assign it to the corresponding class [8].

The back-propagation algorithm used to modify the parameters for the desired output. The procedures of the algorithm are the error at the output layer which spread backward down to the input layer passing the hidden layer in the network in order to get the final desired outputs. The gradient descent method is utilized to calculate the phase variable θ_j^s and phase controlled factor $dw_i^k$of the network and adjusts them to minimize the output error [9].

V. LEARNING ALGORITHM FOR THE QNN

The gradient descent method is used to train the QNN of multi-layer excitation function. In each training cycle, the training algorithm revises both the connection weight between

the different level neuron and quantum intervals of the hidden layer [10].

There are two steps to train the QNN. The first step is making the input sample data compatible with the relevant class spaces by updating the connecting weights. The second step is embodying the uncertainty of data by updating the quantum intervals of quantum neurons in the hidden layer [11].

A. Update the synaptic weights in QNN

First should train the weights which involve presenting all the training set to the network and a forward pass and back propagation like a normal neural network. Let $d^k = [d_1^k d_2^k d_3^k \dots d_{n_o}^k]^T$ be the desired output vector for the x^k input feature vector $x^k = [{x_1}^k, {x_2}^k, {x_3}^k, \dots, {x_{n_i}}^k]^T$. Let $y_i^k = [y_1^k y_2^k y_3^k \dots y_{n_o}^k]^T$ be the actual output [12]. A gradient descent based algorithm for learning the synaptic weights of the QNN can be derived by minimizing the quadratic error function sequentially for each k.

$$E^k = \frac{1}{2}\sum_{i=1}^{n_o}(d_i^k - y_i^k)^2 \qquad k = 1,2,\dots,m \tag{23}$$

Where E: mean square error functions, m is the total number of the training pattern. The synaptic weights are adjusted so as to minimize this error. This can be accomplished by adjusting or changing each synaptic weight by an amount proportional to the gradient of E^k with regard to this specific synaptic weight.

The process of weights updating starting with the initial guess of the weight (which may be selected at random) and then a series of weights are generated using the following formulas:
This update could be found via calculate the derivative of E^k with respect to w_{ji} as:

$$w_{ji}(r+1) - w_{ji}(r) = -\eta\frac{\partial \mathrm{E}^k}{\partial w_{ji}} = \eta\sum_{i=1}^{n_o}(d_i^k - y_i^k)\frac{\partial y_i^k}{\partial w_{ji}} \tag{24}$$

Where $w_{ji}(r+1) and\ w_{ji}(r)$ are the values representing a w_{ji} after and before the modification for the kth inputs and η is a small positive number called the learning rate, $0 < \eta < 1$.

$$\frac{\partial y_i^k}{\partial w_{ji}} = y_i^{k\prime}\ \breve{H}_j^k \tag{25}$$

Where

$$y_i^{k\prime} = \beta_o y_i^k(1 - y_i^k) \tag{26}$$

$$\frac{\partial \mathrm{E}^k}{\partial w_{ji}} = \beta_o e_i^k y_i^k(1 - y_i^k)\ \breve{H}_j^k \tag{27}$$

Substituting equation (27) into equation (24), the update equation is obtained as:

$$w_{ji}(r+1) - w_{ji}(r) = \eta\beta_o e_i^k y_i^k(1 - y_i^k)\ \breve{H}_j^k \tag{28}$$

Since

$$w_{ji}(r+1) = w_{ji}(r) - \eta dw_i^k *\ \breve{H}_j^k \tag{29}$$

Where

$dw_i^k = \beta_o e_i^k y_i^k(1 - y_i^k)$, e_i^k=$d_i^k - y_i^k$, $\breve{H}_j^k$ is the output of the jth hidden neuron.

For synaptic weight v_{lj} that connecting between the jth hidden unit and the kth input unit, the update equation will be derived:

$$\mathrm{v}_{lj}(r+1) - \mathrm{v}_{lj}(r) = -\eta\frac{\partial \mathrm{E}^k}{\partial v_{lj}} = \eta\sum_{i=1}^{n_o}(d_i^k - y_i^k)\frac{\partial y_i^k}{\partial v_{ji}} \tag{30}$$

Where $\mathrm{v}_{lj}(r+1)$ and $\mathrm{v}_{lj}(r)$ are the values of v_{lj} before and after the adaptation.

$$\frac{\partial y_i^k}{\partial v_{ji}} = y_i^{k\prime}\sum_{i=1}^{n_o} w_{ji}\frac{\breve{H}_j^k}{\partial v_{ji}} \tag{31}$$

With $y_i^{k\prime}$ as defined in equation (26), the definition of $\breve{H}_j^k$ is:

$$\frac{\breve{H}_j^k}{\partial v_{ji}} = \beta_h\frac{1}{n_s}\sum_{s=1}^{n_s} h_j^{s,k}\left(1 - h_j^{s,k}\right)x_l^k \tag{32}$$

Substituting equation (31) and equation (32) in equation (30), the update equation becomes:

$$\frac{\partial \mathrm{E}^k}{\partial v_{lj}} = \beta_h\beta_o\sum_{i=1}^{n_o} e_i^k y_i^k(1 - y_i^k)\, w_{ji}\frac{1}{n_s}\sum_{s=1}^{n_s} h_j^{s,k}\left(1 - h_j^{s,k}\right)x_l^k \tag{33}$$

Result:

$$\mathrm{v}_{lj}(r+1) - \mathrm{v}_{lj}(r) = \eta\sum_{i=1}^{n_o} dw_i^k\, w_{ji}\frac{1}{n_s}\sum_{s=1}^{n_s} h_j^{s,k}\left(1 - h_j^{s,k}\right) * \beta_h\, x_l^k \tag{34}$$

Since

$$\mathrm{v}_{lj}(r+1) - \mathrm{v}_{lj}(r) = \eta dv_i^k * \beta_h\, x_l^k \tag{35}$$

Where

$$dv_i^k = \sum_{i=1}^{n_o} dw_i^k\, w_{ji}\frac{1}{n_s}\sum_{s=1}^{n_s} h_j^{s,k}\left(1 - h_j^{s,k}\right) \tag{36}$$

B. Updating the quantum intervals

First, QNN must be trained in order to recognize transitions occurring between classes. The synaptic weights of the QNN must be updated to enable the network to learn the class boundaries of the feature space. After that training the quantum intervals θ_j^s, before the jump-positions are updated, the training set is presented to the network. Once again to calculate $<\hat{\mathrm{H}}_j^{c_i}>$ (It's the sum of outputs of hidden neurons for all the inputs that belong to class c_i divided by the number of samples in that class). This is seemed as a kind of forward pass, then $\theta_j^s$is updated [13].

The idea of adjustments to quantum intervals is to achieve the minimum output miscellaneous of hidden layer neuron depending on the same class of sample data, basically, it's also depends on the negative gradient algorithm, and the samples at the vague boundary that do not belong to the same class can be assigned to different classes by this algorithm.

The variance of the output of the jth hidden nodes for ith class is

$$\sigma_{j,i}^2 = \sum_{x^k \epsilon c_i}(<\hat{\mathrm{H}}_j^{c_i}> - \hat{\mathrm{H}}_j^k)^2 \tag{37}$$

Where $<\hat{\mathrm{H}}_j^{c_i}> = \frac{1}{C_i}\sum_{x^k \epsilon c_i}\hat{\mathrm{H}}_j^k$ is the cardinal number of C_i , i is the class number of patterns.

The amendment of quantum intervals θ_j^s will be achieved by minimizing the objective function G.

$$G = \frac{1}{2}\sum_{j=1}^{n_h}\sum_{i=1}^{n_o}\sigma_{j,i}^2 = \frac{1}{2}\sum_{j=1}^{n_h}\sum_{i=1}^{n_o}\sum_{x^k \epsilon c_i}(<\hat{\mathrm{H}}_j^{c_i}> - \hat{\mathrm{H}}_j^k)^2 \tag{38}$$

The update equation for θ_j^s can be obtained by setting the change in θ_j^s, say $\Delta\theta_j^s$ proportional to the gradient of G with respect to θ_j^s as:

$$\Delta\theta_j^s = -\eta_\theta\frac{\partial G}{\partial \theta_j^s} = \sum_{i=1}^{n_o}\sum_{x^k \epsilon c_i}(<\hat{\mathrm{H}}_j^{c_i}> - \hat{\mathrm{H}}_j^k)\left[\frac{\partial <\hat{\mathrm{H}}_j^{c_i}>}{\partial \theta_j^s} - \frac{\hat{\mathrm{H}}_j^k}{\partial \theta_j^s}\right] \tag{39}$$

Where η_θ is the learning rate, $0 < \eta_\theta < 1$.

The derivative of Equation (37) gives:

$$\frac{\partial <\hat{\mathrm{H}}_j^{c_i}>}{\partial \theta_j^s} = \frac{1}{C_i}\sum_{x^k \epsilon c_i}\frac{\partial \hat{\mathrm{H}}_j^k}{\partial \theta_j^s} \tag{40}$$

$$\frac{\partial \hat{H}_j^k}{\partial \theta_j^s} = \frac{-\beta_h}{n_s}\sum_{s=1}^{n_s} h_j^{s,k}\left(1-h_j^{s,k}\right) = \frac{-\beta_h}{n_s}\sum_{s=1}^{n_s} v_j^{s,k} \quad (41)$$

Where $v_j^{s,k} = h_j^{s,k}\left(1-h_j^{s,k}\right)$ $, < v_j^{s,c_i} > = \frac{1}{C_i}\sum_{x^k \epsilon c_i} v_j^{s,k}$

Substituting equation (40) and equation (41) into equation (39) gives the update equation as:

$$\Delta\theta_j^s = \eta_\theta \frac{\beta_h}{n_s}\sum_{i=1}^{n_o}\sum_{x^k \epsilon c_i}(< \hat{H}_j^{c_i} > - \hat{H}_j^k)\ (< v_j^{s,c_i} > - v_j^{s,k} \quad (42)$$

The quantum interval can be obtained by:

$$\theta_j^s(r+1) = \theta_j^s(r) + \Delta\theta_j^s \quad (43)$$

Where $\eta_\theta \in (0,1)$ is the learning rate of θ_j^s , n_o is the number of nodes in the output layer, and represents the total number of classes, n_s represents the quantum interval of layers, x^k: $x^k \epsilon c_i$ this means from all samples belong to the class c_i . $< \hat{H}_j^{c_i} >$ Is the sum of outputs of hidden neurons for all the inputs that belong to class c_i divided by the number of samples in that class, to the kth feature vectors x^k, $\tilde{H}_j^k$ the response of the jth multi-level hidden unit.

VI. DATA DESCRIPTION

In this work, the raw EEG signals that have been used consisting of 2500 samples within 10 seconds, which is about five categories (baseline, multiplication, letter composing, rotation, counting) and each category contains 7 signals for seven persons [14].

VII. SIMULATION RESULTS

A. *Knot Extraction Result*

After normalization, knot extraction achieved depending on the shape of each signal from EEG signals. Firstly, all intersection points will found and then find the maximum and minimum values by sort the data on the signal in ascending hence these totally points will represent the sections of data. Then, repeat the process of extracting knots of each part of the signal. After extracting all the knots will reduce these knots using the same technique, and these extracted knots will be considered as original or new data and by the same technique, most change points will extract, without losing the basic information of the signal. Figure 4 to figure 8 shows the knot extracting for five categories before and after reduction.

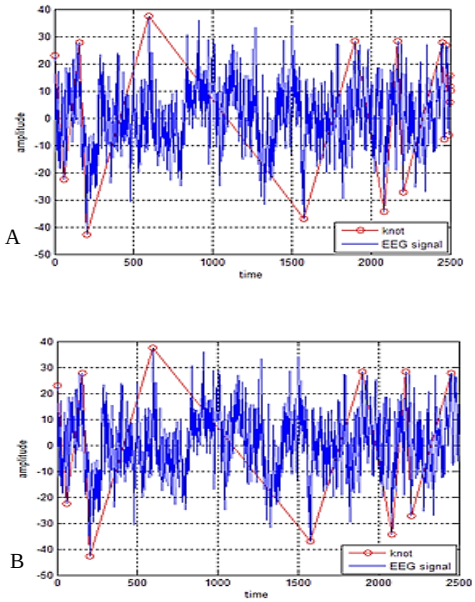

Fig.4. Knots in EEG signal for Baseline task before and after reduction A. Before reduction, B. After reduction

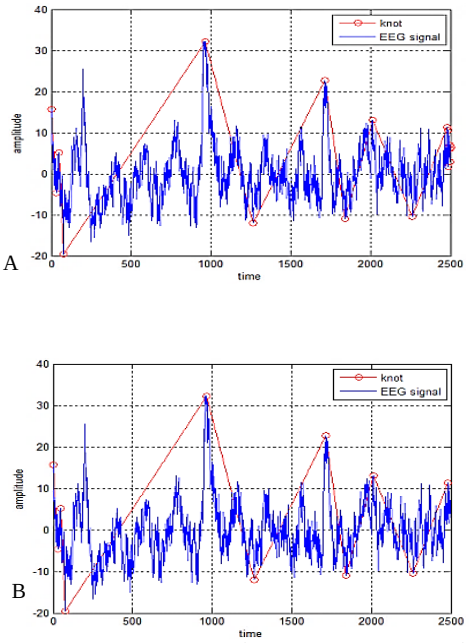

Fig.5. Knots in EEG signal for Multiplication task before and after reduction A. Before reduction, B. After reduction

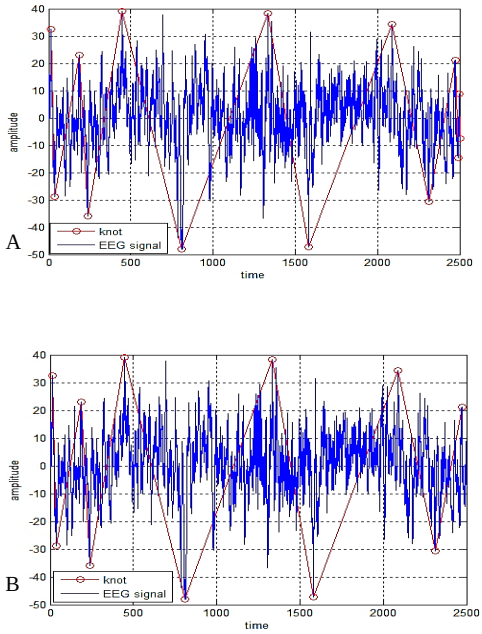

Fig.6. Knots in EEG signal for Letter Composing task before and after reduction A. Before reduction, B. After reduction

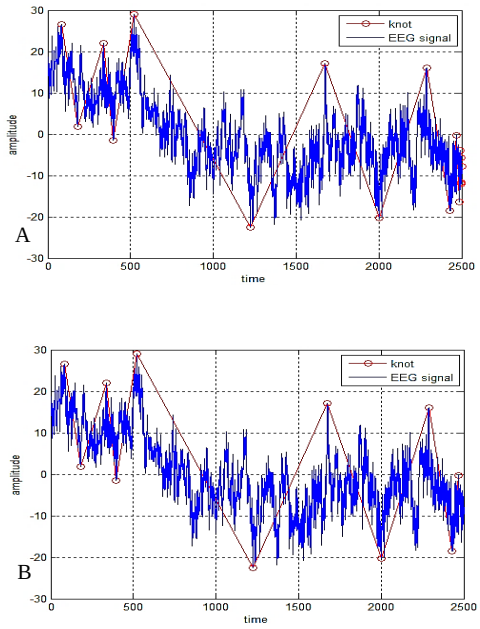

Fig.7. Knots in EEG signal for Rotation task before and after reduction A. Before reduction, B. After reduction

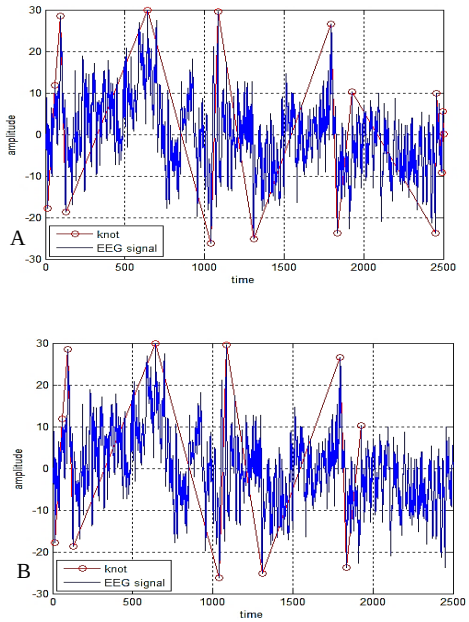

Fig.8. Knots in EEG signal for Counting task before and after reduction A. Before reduction, B. After reduction

B. *Feature Extraction Result*

In each signal of EEG signals, 11 knots were extracted. In CST, between each two neighboring knots, there are four coefficients, thus for each one signal of EEG there will be 40 coefficients. These coefficients represent the basic features of the EEG signal. Work has been achieved for each class. So the number of extracted features will be 40 columns and 35 rows. Figures 9 to 10 show the results of feature extraction for five classes.

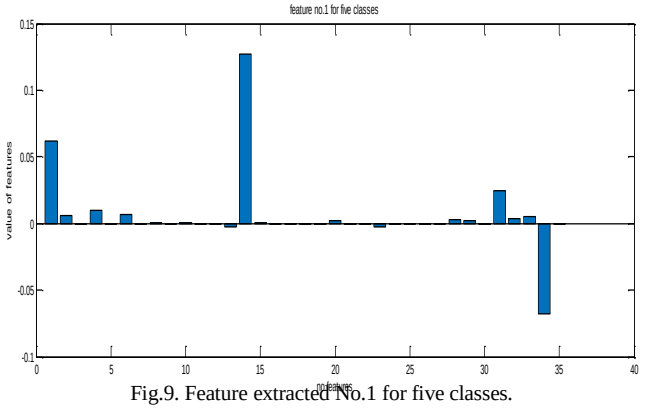

Fig.9. Feature extracted No.1 for five classes.

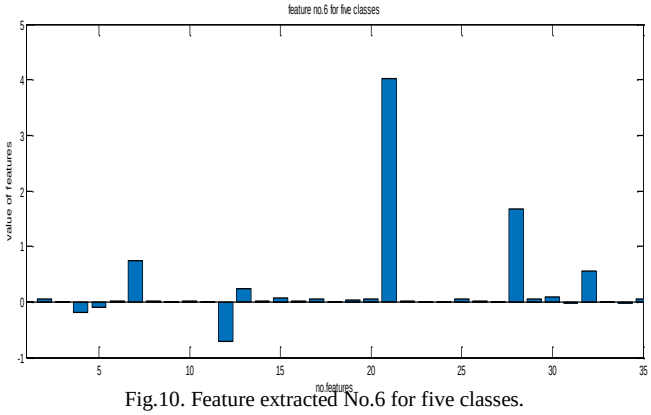

Fig.10. Feature extracted No.6 for five classes.

C. *Classification Results of EEG Signals*

The proposed system used five types of EEG signals. Each type of these signals is a mental task assigned to a particular person to perform it. These tasks are (baseline, multiplication, letter composing, rotation, counting). The number of the extracted features for each signal are 40 features, means that the entries that will enter the classification network is a matrix within 40 rows and 35 columns for five classes (in each case of the seven electrodes).

The extracted features were divided into two parts, training part and testing part. Each part consists of a set of features extracted. Randomly, 70% of samples were selected as training samples and 30% of samples for testing. Again, 50% of samples were selected as training samples and 50% of samples for testing.

In this work, all the graphics and results were generated by M-file MATLAB software program (Version 8.3, R2014a). In order to find out the performance of the classification network that is carried out by this software program, in each case, the classification accuracy evaluated by using the equation below:

$$Accuracy = \frac{Total\ number\ of\ correct\ classifications}{Total\ number\ of\ samples} * 100\%$$

The accuracy of classification of five classes of EEG signals due to the selection of 70% training and 30% testing data for seven electrodes with different iterations according to best iteration that gives the highest accuracy of classification is shown in table I and figure 11.

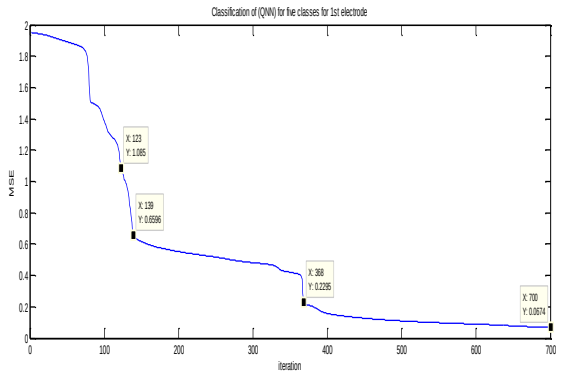

Fig.11. Classification of (QNN) for five classes with training (70% randomly) of data for the 1st electrode.

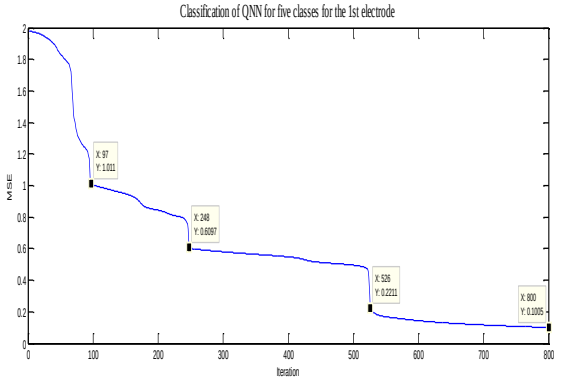

Fig.12. Classification of (QNN) for five classes with training (50% randomly) of data for the 1st electrode.

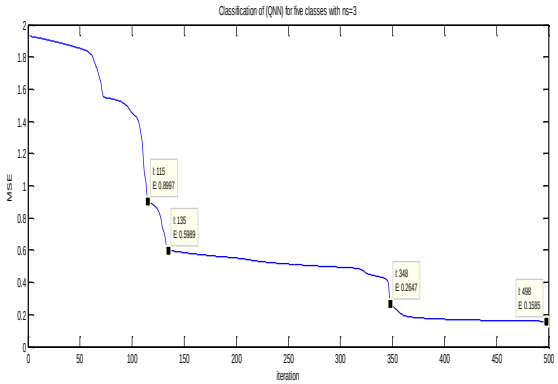

Fig.13. Classification of QNN with different quantum intervals (n_s).

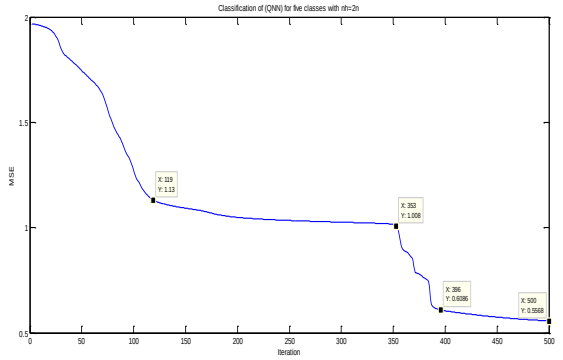

Fig.14. Classification of QNN with different number of nodes in the hidden layer (n_h).

TABLE I
THE ACCURACY OF CLASSIFICATION OF FIVE CLASSES OF EEG SIGNALS WITH TRAINING (70% RANDOMLY) OF DATA, FOR SEVEN ELECTRODES

Electrode	Iterations	MSE for Training	MSE for testing	Accuracy (%)
1st elec.	700	0.0674	0.9817	96%
2nd elec.	700	0.0873	0.9852	92%
3rd elec.	500	0.1081	1.1277	96%
4th elec.	700	0.0998	0.9500	92%
5th elec.	500	0.0793	1.1310	92%
6th elec.	500	0.1240	1.3059	96%
7th elec.	500	0.0746	1.1878	96%
Average	586	0.0915	1.095	94.3%

TABLE II
THE ACCURACY OF CLASSIFICATION OF FIVE CLASSES OF EEG SIGNALS WITH TRAINING (50% RANDOMLY) OF DATA, FOR SEVEN ELECTRODES

Electrode	Iterations	MSE for Training	MSE for testing	Accuracy (%)
1st elec.	800	0.1005	1.22	94.4%
2nd elec.	600	0.0794	1.227	88.88%
3rd elec.	500	0.1058	1.246	88.88%
4th elec.	500	0.0831	1.0708	94.4%
5th elec.	700	0.0459	1.2511	94.44%
6th elec.	700	0.177	1.1367	88.88%
7th elec.	700	0.0432	0.0909	100%
Average	643	0.0907	1.034	92.84%

TABLE III
THE CLASSIFICATION RESULTS OF QNN WITH DIFFERENT QUANTUM INTERVALS

No. of n_s	Iterations	MSE for Training	MSE for testing
ns=3	500	0.1584	1.0749
ns=4	500	0.1210	1.5730
ns=6	700	0.1225	1.0839

TABLE IV
THE CLASSIFICATION RESULTS OF QNN WITH DIFFERENT NUMBER OF NODES IN THE HIDDEN LAYER

No. of n_h	Iterations	MSE for Training	MSE for testing	Accuracy (%)
$n_h = n_i/2$	600	0.1876	1.1287	76%
$n_h = 2n_i$	500	0.556	1.542	52%
$n_h = 3n_i$	600	1.357	2.79	20%

The accuracy of classification of five classes of EEG signals due to te selection of 50% training and 50% testing data for seven electrodes with different iterations according to best iteration that gives the highest accuracy of classification is shown in table II and figure 12.

By changing the value of basic parameters, several tests have been conducted on the classification network. Such as changing the number of quantum intervals (ns), and the other test was implemented by changing the number of nodes in the hidden

layer (n_h). Table III and fig.13 show the classification results of QNN with different quantum intervals (n_s).

Table IV and figure 14 shows the classification results of QNN with different numbers of nodes in the hidden layer.

VIII. DISCUSSION

The proposed system used five types of EEG signals. All the graphics and results were generated by M-file MATLAB software program (Version 8.3, R2014a). Feature extraction accuracy depends on knot extraction accuracy. The extracted features for the signals in each class differ from each other and from the extracted features in other classes, as shown in figures 9 to 10.

In case of selecting 50% of data for training the network, classification results and MSE differ from each electrode to another and according to maximum iterations that gives highest accuracy and differs from the results obtained in case of selecting 70% of data for training, by comparing tables I and II, the results illustrate that the accuracy of classification which obtained in case of using 70% of data for training is higher than the accuracy of classification which obtained in case of using 50% of data for training. Except one case, that is when training 50% randomly of data from the 7th electrode, the accuracy obtained is 100% and MSE is 0.0432 which is the highest accuracy and fewer MSE obtained.

Changing the number of quantum intervals (n_s) for n_s=5 Whether to increasing or decreasing will decrease the accuracy as well as increase the MSE as shown in table III and the same result obtained when changing the number of the nodes in the hidden layer (n_h) from n_h=n_i as shown in table IV.

IX. CONCLUSION

CST used for the first time to extract the features for five types of EEG signals. The proposed system largely depended on the shape of the signal so it is very important to normalize the EEG signal in order to reduce the effects on it. The technique that used to extract the knots was very precise and has many benefits that are; reducing the features by selecting the best knots without need to reduce the features after extraction, and the process of, extracting and selecting the knots has been implemented by the same technique. CST, which is based on the extracted knots proved effective in extracting the features accurately. The accuracy of classification was obtained as an average from the seven electrodes which are 94.3% with 0.0915 MSE when training 70% of features and 92.84% with 0.0907 MSE when training 50% of the features. The number of iterations that gives the best accuracy was different from electrode to another, thus the number of iterations was not fixed.

REFERENCES

[1] Saeid Sanei and Jonathon Chambers, “EEG signal processing”, Cardiff University, England, 2007.

[2] Terence W. Picton and others, “The recording and analysis of event-related potentials”, Elsevier Science, Germany, Vol. (10), 1995.

[3] David Friedman and Ray Johnson, “Event-Related Potential (ERP) Studies of Memory Encoding and Retrieval: A Selective Review”, Microscopy Research and Technique, New York, 2000.

[4] Sky McKinley and Megan Levine, “Cubic Spline Interpolation,” CiteSeer, February, 1999.

[5] CJC Kruger, “Constrained Cubic Spline Interpolation for Chemical Engineering Applications,” M.S.C. Thesis, 2002.

[6] S. Fredenhagen, H. J. Oberle, & G. Opfer, “On the Construction of Optimal Monotone Cubic Spline Interpolations,” Journal of Theory, ELSEVIER, February, p.p.182-201, 1999.

[7] Professor J. Zhang, “Approximation by Spline Functions”, lecture 6, Dep. of Computer Science, University of Kentucky Lexington, KY 402060046, November 1, 2010.

[8] Maojun Cao, Fuhua Shang, “Quantum Neural Networks with application in adjusting PID parameters”, IEEE, 2009.

[9] Dianbao Mu and others, “Learning Algorithm and Application of Quantum Neural Networks with Quantum Weights,” International Journal of Computer Theory and Engineering, Vol. (5), 2013.

[10] Gopathy Purushothaman and Nicolaos B. Karayiannis, “Quantum Neural Networks (QNN’s): Inherently Fuzzy Feedforward Neural Networks”, IEEE, VOL. (8), 1997.

[11] Shicai Yu and Ning Ma, “Quantum Neural Network and its Application in Vehicle Classification”, IEEE, 2008.

[12] Daqi Zhu and Rushi Wu, “A Multi-layer Quantum Neural Networks Recognition System for Handwritten Digital Recognition”, IEEE, 2007.

[13] Ajith Abraham, “Handbook of Measuring System Design”, Oklahoma State University, Stillwater, OK, USA, 2005.

[14] Zachary A. Keirn, “Alternative modes of communication between man and machine,” M.S.C Thesis, Purdue University, 1988.