2015 IEEE Symposium Series on Computational Intelligence

Quantum based Neural Network Classifier and its Application for Firewall to Detect Malicious Web Request

Om Prakash Patel*, Aruna Tiwari†, Patel Vikram Dineshbhai‡, Ojas Gupta§
Department of Computer Science and Engineering
Indian Institute of Technology, Indore, India 453441
*
Email: oppatel13@gmail.com
† Email: artiwari@iiti.ac.in
‡ Email: cse1200123@iiti.ac.in
§ Email: cse1200121@iiti.ac.in

***Abstract*—In this paper, a quantum based neural network classifier is designed as a Firewall (QNN-F) to detect malicious Web requests on the Web. The proposed algorithm forms a neural network architecture constructively by adding the hidden layer neurons. The connection weight and threshold of the neurons are decided using the quantum computing concept. The quantum computing concept gives large subspace for selection of appropriate connection weights in evolutionary ways. Also, the threshold value is decided using the quantum computing concept. To enhance the performance of the system, a Web crawler is also proposed which finds objectionable URLs on the Web according to the objectionable keywords. The proposed algorithm is tested on Web data, to develop a firewall which detects malicious Web requests. Extensive testing on 2000 objectionable and non objectionable URLs are done which shows that proposed system works efficiently for detection of objectionable content. To judge the performance of the proposed classifier, it is compared with the Support Vector Machine, Back Propagation neural learning algorithm and quantum based classifier (Q-BNN). The comparison validates that, the QNN-F performs better than other compared algorithms.**

I. Introduction

Human brain has the great ability to unravel and classify the complex patterns of the real world. Inspiring from the brain anatomy, artificial neural network has introduced in 1943. Human brain needs training specific to the kind of task involved. Analogously, artificial neural network also needs training algorithm. Many models have been proposed like back propagation, perceptron and recurrent network which represent working of the human brain. These models have been successfully applied in several fields like economics, defense, stock market, engineering, medical, computer network and many more. However, performance of neural network in these mentioned areas depends on many parameters like, quality of input data set, number of hidden layer neurons, threshold of neurons, connection weights, etc. [1]. Deciding these factors for a particular application with optimal neural network is an open area of research. To optimize the network architecture, one of the initiative has been taken based on quantum computing concept by Lu in 2013 [2]. They proposed a method which uses quantum concept to decide connection and weights over it in evolutionary way. One more neural network algorithm is proposed for optimizing neural network by forming neural network constructively and connection weights have been decided using quantum computing concept [3]. In constructive neural network algorithm, the neurons are being added in the network during learning to corresponding samples [4].

There are many such applications in the area of computer network which needs pattern recognition, one of them is network security. Network security issues are becoming critical due to availability of huge data on Internet. Since, there is a lot of malicious content present on the Internet which might affect the users mental strength. There are methods develop by researchers ([5], [6]) in the field of computer network security by the make use of quantum computing concept. These methods uses idea of traditional genetic algorithm to makes use of the immune operator of immune clone strategy and quantum cryptography to exchange the encryption key with absolute security. However, there have been many researchers, in this area who have dealt with this network security problem. Feng [7], proposed solution based on URL and IP address based filtering results in blocking of many Websites. This method block objectionable URLs on the basis of IP address and URLs name. However, the approach failed to provide desired results due to the fact that many Websites changes its content and IP address dynamically. Everyday thousands of objectionable Web sites are hosted which make this solution unfeasible. On the other hand, image based classification solves this issue to some extent, but fails as 85 % of the information present is in the form of text [8], [9]. There are some researchers who have combined these methods together to get higher efficiency [10]–[12]. These method uses simple text parsing techniques, and images processing using features of objectionable images. These method solve the problem to some extend only because not analyzing text semantically may produce false results. Still, there are several issues to be taken care of in blocking the objectionable content and classifying the extracted features from the Web. For example, if a Website containing some hyperlink, which leads to some objectionable content, then the firewall must block particular

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.20

link rather than whole Website which contain objectionable content link. Also blocking of the Website on the basis of having objectionable keywords in it and may also block some education sites like stop suicide, sex education etc. Therefore, semantic analysis of sentences having objectionable keywords is required here. To overcome the issues discussed here, in this paper quantum based neural network learning algorithm is presented for developing a network firewall which detects malicious Web request (QNN-F). The proposed algorithm forms a neural network classifier constructively by the adding the neuron at hidden layer whereas, the connection weight and threshold of neurons is decided using the quantum computing concept. In the proposed algorithm some unique features have been added for accurate classification of objectionable Web content. This system mainly consist of five components: Web crawler, requested Web page analyzer, hyperlinked analyzer, feature extractor and quantum neural network.

The remaining part is organized as follows. In the Section II some preliminaries are discussed which are required to understand the proposed approach. In the Section III the proposed algorithm is discussed in detail. In Section IV QNN-F architecture is explained. Experiment with real world data is conducted and detailed testing is described in Section V. Finally, proposed method is concluded in Section VI.

II. Preliminaries

In this paper a quantum based neural network learning algorithm is proposed with firewall as its application which detects the malicious Web request. This proposed algorithm has two major components, the first is quantum based learning algorithm and the other is computer networking part for firewall. In this section some basic necessary concepts are discussed in brief, which helps to illustrate the proposed algorithm. Initially, preliminaries related to quantum computing is explained then subsequently preliminaries for firewall is discussed.

A. Quantum Neural Network

This method forms a neural network structure, which consists of three layers, input layer, hidden layer and output layer. Let X=$(X_1, X_2, X_3, ..., X_u)$ denote the input samples, where u is the number of input samples and X_l=$x_1^l, x_2^l, x_3^l, ..., x_n^l$ where n is number of attributes in one instance of input sample. Therefore, the number of input layer node is equalled to n. For j^{th} hidden layer neuron, connection weights are denoted as follows:

$$W_j^{real} = (w_{j1}, w_{j2}, w_{j3}, ..., w_{jn}) \tag{1}$$

Here, the number of neurons in hidden layer are decided constructively by adding neuron one by one. The Output layer contains only one neuron, which decides whether the content is objectionable or not to form the neural network structure. The step function has been used for learning and it is given as follows:

$$net_j = \sum_{i=1}^{n} w_{ji} \times x_i \tag{2}$$

$$f(net_j) = \begin{cases} 1 & if \quad net_j \leq Threshold \\ \\ 0 & if \quad net_j > Threshold \end{cases} \tag{3}$$

To find, whether the input sample is objectionable or not, the value of net_j is compared with the threshold. In the proposed neural network, the connection weights (W_j) and threshold (λ_j) of neuron is decided using quantum computing concept. The required preliminaries for the quantum computing concept is presented subsequently.

The core idea of quantum inspired algorithm is to present solution of a problem in terms of so-called quantum bits Q rather than in classical bits. The weight matrix corresponding to Eq (1) and threshold $(\lambda_j^{'})$ of j^{th} hidden layer neuron in the form of quantum bit Q can be represented as:

$$W_j^{'} = (Q_{j1}, Q_{j2}, Q_{j3}, ..., Q_{jn}); \tag{4}$$

$$\lambda_j^{'} = (Q_j^{Th}); \tag{5}$$

The quantum bit (Q_j) of weight matrix and threshold Q_j^{Th} is being decided by quantum concepts, where any quantum bit (Q_j) is represented by several qubits (q).

$$Q_j = (q_{j1}|q_{j2}|......|q_{jk}) \tag{6}$$

Here, k number of qubits which represents quantum bit (Q). A single qubit (q_{ji}) where i=1,2.....k, is the smallest unit of representing information. A qubit is fundamentally different from the binary bit used in traditional digital computers in the sense of representing data. A single binary bit can represent only two states, "0" and "1", whereas the qubit (q_{ji}) has the capability to represent the linear superposition of two states simultaneously, which is determined by probability model [13]. Thus, qubit q_{ji} can be represented as

$$q_{ji} = \alpha_{ji} \mid 0\rangle + \beta_{ji} \mid 1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{7}$$

Where, α and β is a complex number representing the probability of qubit in "0" state and in "1" state. A probability model is applied here, which represent "0" state by α^2 and in "1" state by β^2, represented as:

$$\alpha_{ji}^2 + \beta_{ji}^2 = 1; 0 \leq \alpha \leq 1, 0 \leq \beta \leq 1 \tag{8}$$

As discussed above, a quantum bit (Q_j) formed using qubits $(q_{j1}, (k$=1)) which represent two states e.g. "0" state or "1" state. An individual quantum bit (Q_j) having two qubits $(q_{j1}|q_{j2}, (k$=1, 2)) in it represents 4 states, e.g. "00", "01", "10" and "11". In the same way, three-qubits ($(q_{j1}|q_{j2}|q_{j2})$, (k=1, 2, 3)) system represent eight states, thus n qubits (q_{jn}, (k=1,..,n)) will have 2^n states. For example, an individual quantum bit Q_j having two qubits can be represented as follows:

$$Q_j = \left\langle \begin{array}{c} \alpha_{j1}|\alpha_{j2} \\ \beta_{j1}|\beta_{j2} \end{array} \right\rangle \tag{9}$$

The below example shows representation of the 4 states of quantum bit (Q_j) having two qubits (q_{j2}).

$$Q_j = (\alpha_{j1} \times \alpha_{j2})\langle 00\rangle + (\alpha_{j1} \times \beta_{j2})\langle 01\rangle \\ +(\beta_{j1} \times \alpha_{j2})\langle 10\rangle + (\beta_{j1} \times \beta_{j2})\langle 11\rangle \quad (10)$$

It is noted that qubit (q_{ji}) is made of two components α_{ji} and β_{ji}, where each component value lies between "0" and "1". Let us assume that a quantum bit (Q_j) having 2 qubits (q_{j2}) and any random value can be initialized for parameter mentioned in Eq (9) with constrains discussed in Eq (8). Here α_{j1}, α_{j2}, β_{j1} and β_{j2} initialized as follows.

$$Q_j = \left\langle \begin{array}{c} 1/\sqrt{2}|1/\sqrt{2} \\ 1/\sqrt{2}|1/\sqrt{2} \end{array} \right\rangle \quad (11)$$

With respect to Eq (10) and Eq (11), the state representation of quantum bits (Q_j).

$$Q_j = (1/\sqrt{2} \times 1/\sqrt{2})\langle 00\rangle + (1/\sqrt{2} \times 1/\sqrt{2})\langle 01\rangle + \\ (1/\sqrt{2} \times 1/\sqrt{2})\langle 10\rangle + (1/\sqrt{2} \times 1/\sqrt{2})\langle 11\rangle \quad (12)$$

In case of quantum threshold, there is requirement of only single quantum bit (Q_j), because the threshold value of a neuron is always a single value in each iteration.

1) Conversion from quantum bits to real value: The algorithm which is proposed here works on classical computers, therefore conversion from quantum bits to real value is required . The weight matrix in terms of quantum bits $W_j^{'}$ is converted into a real value weight matrix W_j^{real}. Similarly the threshold value in terms of quantum bits $\lambda_j^{'}$ is converted into real value λ_j^{real}. This conversion process starts by taking random number matrices R, where R_j= $[r_{j1}r_{j2}....r_{jk}]$, corresponding to Q_j= $(q_{j1}|q_{j2}|......|q_{jk})$. Then, further mapping is done by using binary matrix S_j where $S_j = [s_{j1}s_{j2}....s_{jk}]$ and Gaussian random number generator with mean value μ and variance σ, which can be represented as $N(\mu, \sigma)$.The mapping between binary value to Gaussian number generator is done with the help of formula binary to decimal conversion. The value of matrix S_j is passed into $bin2dec(S_j)$ formula to select value from Gaussian random generator [2]. The value of matrix S_j is generated as follows:

$$if(r_{ji} \leq (\alpha_{ji})^2) \;\; then \;\; s_{ji} = 1 \;\; else \;\; s_{ji} = 0.$$

Thus, using this algorithm, all the components of Q_j= $(q_{j1}|q_{j2}|......|q_{jk})$ is converted into real values. In this way, all Q_j components of quantum weight ($W_j^{'}$) are evaluated similarly, Q_j^{Th} is evaluated for quantum threshold ($\lambda_j^{'}$). The quantum weight ($W_j^{'}$) and quantum threshold ($\lambda_j^{'}$) are further evolved to finalize an optimal value by applying quantum updation which is discussed next.

2) Qubit Updation: Evolutionary algorithms are applied to optimize the solution of varying parameters and its find out in several iteration by observing their fitness [13]. Therefore, to evolve the new value of weight matrix W_j^{real} and real threshold λ_j^{real}, the quantum weights $W_j^{'}$ and quantum threshold $\lambda_j^{'}$ are updated using quantum update function which utilizes the fitness value, let us denote fitness by F and F^*. The fitness value is decided based on the number of possible class samples learnt by using parameter $count1$ and $count2$, which is described in detail in Section III. The quantum weights are evolved in g number of iterations and quantum threshold is evolved in t number of iterations. To update quantum weight $(W_j^{'})_{g+1}$ from $(W_j^{'})_g$ and quantum threshold $(\lambda_j^{'})_{t+1}$ from $(\lambda_j^{'})_t$, quantum rotation gates are required. These quantum rotation gate have variable $\Delta\theta$, which is decided by comparison of current iteration fitness value and best fitness value available in last iterations [13]. However, in the proposed algorithm two different parameter weights W_j^{real} and threshold λ_j^{real} are evaluated. Hence, fitness function for weight W_j^{real} are taken as F_g, F^* and fitness function for threshold λ_j^{real} are F_t, F_λ^* corresponding to F and F^* . For the easiness of understanding of updation process of qubit (q_{ji}), the current fitness F_g and best fitness value F^* is taken which represent qubit (q_{ji}^g) updation for quantum weight $(W_j^{'})_g$. The same process is applied to update qubits (q_{ji}^{Th}) of quantum threshold $(\lambda_j^{'})_t$ with fitness function F_t and F_λ^*. To update qubit, the required quantum gate is as follows:

$$U(\Delta\theta) = \begin{vmatrix} \cos\Delta\theta & -\sin\Delta\theta \\ \sin\Delta\theta & \cos\Delta\theta \end{vmatrix} \quad (13)$$

Where, $\Delta\theta$ is a rotation angle which is used to generate Q_j^{g+1} from Q_j^g. In the proposed algorithm, length of a quantum bit Q_j^g= $(q_{j1}^g|q_{j2}^g|......|q_{jk}^g)$. is considered as 2 (k=2). It means that $(W_j^{real})_g$ will be selected from 4 subspaces.

$$q_{ji} = \begin{vmatrix} \alpha_{ji}^{g+1} \\ \\ \beta_{ji}^{g+1} \end{vmatrix} = \begin{vmatrix} \cos\Delta\theta & -\sin\Delta\theta \\ \sin\Delta\theta & \cos\Delta\theta \end{vmatrix} * \begin{vmatrix} \alpha_{ji}^{g} \\ \\ \beta_{ji}^{g} \end{vmatrix} \quad (14)$$

To update quantum bit (Q_j^g), each single qubit (q_{ji}^g) is required to update. To update each qubit (q_{ji}^g), the current fitness value F_g, best fitness value F^*, current binary bit s_{ji}^g and binary bit corresponding to best fitness s_{ji}^* are considered [2]. The value of angular displacement must be selected in such a way so that it can cover the maximum value of α_{ji}^g in the range of (0 1) and also should not take much iterations to cover these values. Therefore $\Delta\theta$ must be initialized between $(0.01 \times \pi, 0.05 \times \pi)$ [2]. The evaluation of fitness function is done in the proposed algorithm that is discussed in the next Section III. Table I shows the tabular form of qubit updation process

Note: In case of threshold updation the binary bit value will be s_{ji}^t in comparison with $(s_{ji}^*)_\lambda$ and fitness value will be F_t and F_λ^*.

TABLE I. QUBITS UPDATION

s_{ji}^g	s_{ji}^*	$F_g < F^*$	$\Delta\theta$
0	*0*	*false*	*0*
0	*0*	*true*	*0*
0	*1*	*false*	$-0.03 * \Pi$
0	*1*	*true*	*0*
1	*0*	*false*	$0.03 * \Pi$
1	*0*	*true*	*0*
1	*1*	*false*	*0*
1	*1*	*true*	*0*

For preventing the component of qubit, α^g_{ji} from acquiring values 0 or 1, following constraint is applied:

$$\alpha^g_{ji} = \begin{cases} \sqrt{\epsilon}, & if \quad \alpha^g_{ji} < \sqrt{\epsilon} \\ \alpha^g_i & if \quad \sqrt{\epsilon} \le \alpha^g_i \le \sqrt{1-\epsilon} \\ \sqrt{1-\epsilon} & if \quad \alpha^g_i > \sqrt{1-\epsilon} \end{cases} \tag{15}$$

Where, the value of ϵ is assigned a very small (approximately approaching to zero), so that it can cover maximum value in the range of (0, 1). In this section, the superscript has been used for all variable as g just for notation. The updation of quantum threshold $\lambda^{'}_j$, will be accordingly t iteration.

B. Preliminaries for Firewall

The traditional firewall works in two modes, on one hand it works on a stand alone system and another is centralized firewall system dedicated to a group of computer systems. In the proposed work, our focus is to develop a centralized firewall system which works for particular groups. To develop such system, a proxy server has been used as an mediator between user Web request and proposed system.

To know whether the accessed URL is objectionable or not, the HTTP response packet of the requested URL is proposed to analyze by quantum neural network. Simultaneously to make a more efficient system, a database is prepared for objectionable URLs, with the help of Web crawler. This Web crawler searches list of URL's on the basis of objectionable key words and update that database with new objectionable URLs.

III. Proposed Approach

In this section, a system named as Quantum based neural network classifier and its application for a firewall to detect malicious Web request (QNN-F) is proposed. Firstly, the quantum neural network classifier is trained by preparing the offline database, which contain objectionable and non objectionable URLs. Total 1000 objectionable URLs and 1000 non objectionable URLs have been visited and data set has been prepared on the basis of the features described in Table II. The dataset has been divided into two categories. The dataset contain all the features of objectionable URLs called class "A" dataset, while collection of features of all non objectionable URL is called class "B" dataset. $x(i)$ where i=$1, 2, 3.....c_1$; number of sample of class "A" (which gives output "1")
$y(j)$ where $j = 1, 2, 3, ..., c_2$; number of sample of class "B" (which gives output "0").

Thus, QNN-F detects objectionable Web request by using neural network architecture, which decides weights and threshold using the quantum computing concept by adding neurons at hidden layer constructively one by one. First of all, the basic components of system are discussed, then quantum based learning algorithm is explained.

A. Basic Principle

QNN-F has five basic components which are Web crawler, requested Web page analyzer, hyperlink analyzer, feature selection and quantum neural network as shown in Fig. 1.

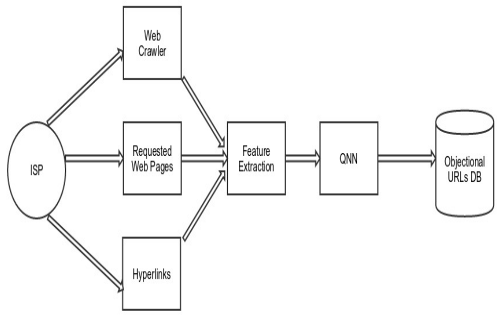

Fig. 1. Outline of QNN-F

TABLE II. Number of Features

Features	Description
nw	Number of words in Web page
nw_o	Number of Objectionable words in Web page
r_nw	Ration of total number of words to the objectionable word
sen	Number of sentences in the web page
sen_o	Number of sentences having objectionable words
img	Number of images in the Web page
img_o	Number of images having objectionable words as title
url_o	number of objectionable word in URL

The first three mentioned components are used for fetching source contents of the Web page. The fourth component is feature selection, it contains all the required features that are used for analysis of Web pages to identify whether it is objectionable or not as shown in Table II. These features act as input for quantum based neural network. Here the quantum concept has been used to decide weights for connection and threshold of neurons. To illustrate QNN-F, each component has been explained in detail. As discussed in section II-B, the proposed system monitors each user's request moving towards ISP (Internet Service Provider) server. The process starts from proxy server, a web crawler, which is attached to proxy server starts accessing objectionable Websites on the basis of objectionable keywords.

B. Web Crawler

To speed up the process of finding objectionable URLs, a web crawler is proposed here. The main objective of this Web crawler is to generate lists of objectionable URLs with the help of objectionable keywords. The content of URLs extracted by Web crawler is fetched for feature selection and analysis. After selecting all the features of particular URLs, then quantum neural network classify these URLs into objectionable or non objectionable URLs. If it is found objectionable then it is appended into the database of objectionable URLs. The processing of Web crawler is an independent activity from the user, which only makes database of objectionable URLs. Hence, if the user try to access any objectionable URLs, then there is a great chance of URL being present in the objectionable URLs database.

C. Requested Web Page Analysis Module

This module analyses all URLs which are permitted by a proxy server to be processed by the ISP. This module captures

HTTP Response packet of requested URLs and extracts all the features described in Table II. These all features are provided as input to quantum neural network to deduce whether the requested URL is objectionable. If requested URLs found to be objectionable then it is added in the database.

D. hyperlink Web Page Analysis Module

As requested Web page may contain some hyperlinks, which redirects to other Websites. Therefore, checking of all hyperlinked URLs is also an important factor to make the system more efficient. This module forms a queue of all hyperlinks, subsequently fetching their content. The features of hyperlinked URLs are extracted and provided as input to the quantum neural network. If URL found to be an objectionable then objectionable URLs database is updated according to it.

E. Detection of Malicious Web Request (QNN)

In this section, the proposed quantum based neural network learning algorithm is presented. The algorithm uses quantum computing concept to determine weights W_j^{real} and threshold λ_j^{real} value. It uses the input as feature of described database. In the proposed algorithm, the fitness value of weight $W^{'}$ updation are F_g, F^* and threshold $\lambda^{'}$ are F_t, F_λ^*. To calculate fitness here two parameters $count1$ and $count2$ have been used which determines number of samples of class A and B that are learnt. Here, S^* and S_λ^* vector is used which stores best value from S^g and S^t corresponding to best result or value of the fitness denoted by F^* and F_λ^* .

1) Learning of QNN-F: Learning of QNN-F start by taking one neuron first at hidden layer and initializing quantum weight for connection of input nodes to hidden layer neuron $(W_1^{'})_1$. Along with initialization of quantum weights $(W_1^{'})_1$, some more parameters, like F^* and S^* are initialized. For learning of QNN-F algorithm the dataset is divided into different ratio. Here, 60% and 70% data is taken for learning of the system and 40% and 30% for testing the system.

After initialization of all above parameters, the conversion process is called to convert quantum weights $(W_1^{'})_1$ into the real value weight matrix $(W_1^{real})_1$. Now this weight is passed into the quantum threshold function. Before starting execution the parameters like quantum threshold $(\lambda_1^{'})_1$ (t=1), S_λ^*, count1=0, count2=0, and F_λ^* are also initialized. In quantum threshold function, quantum threshold $(\lambda_1^{'})_1$ (t=1) converted into real value threshold $(\lambda_1^{real})_1$ using conversion process. After getting real coded value of the weight and threshold, the input is applied to the real value the weight matrix $(W_1^{real})_1$g= 1 and compared with real value threshold $(\lambda_1^{real})_1$. For $(W_1^{real})_1$,g= 1, the threshold will update from $(\lambda_1^{real})_1$, (t=1) to $(\lambda_1^{real})_z$ (t=z) using quantum threshold $(\lambda_1^{'})_t$. The best fitness value F_λ^*, F^* is evaluated from iteration g=1 and t=1 to z. Now this value is compared with stopping criteria if it is satisfied, then learning will stop else the same learning will continue with new weight value $(W_1^{real})_2$, (g=2) for first neuron. After completion of all iterations t=1 to z for each g=1 to m, the best weight $(W_1^{real})_g$ and threshold $(\lambda_1^{real})_t$ is selected corresponding to fitness values. Now, for the unlearnt sample a new neuron is selected at hidden layer and the same process of learning is applied here. After deciding the hidden layer neuron and their connection weight from input layer, the weight and threshold of output layer are decided. The learning of the output layer neuron is also done in same way as done for hidden layer neurons. As there is only one neuron at output layer as discussed previously, thus determining the weights and threshold of output layer neurons requires comparatively less time as required for the hidden layer. The Quantum based neural network learning algorithm in the form of pseudocode is presented next:

Algorithm: *QNN-F Algorithm*

```
Step-1: Take first neuron with the weights $W_g^{'}$
            $W_g^{'} = (Q_{w1}, Q_{w2}, Q_{w3}, ......, Q_{wn})$
        where $g = 1, ..., m$; $m$ is the number of
        iterations to update weights
        $i = 1......n$; $n$ is the number of features
        sample.
        Initialization of more parameters
        $F^* = 0$
        $S^* = 0$;
Step-2: for g=1 to m
          Call conversion process($W_g^{'}$)
          Call Quantum threshold function($W_g$)
          if($F_g \geq (c_1 + c_2)$ )
            Stop learning
          else
            $F^*$=max($F^*$, $F_g$)
            Evaluate $F_g$, $F^*$, $s_i^g$, $s_i^*$
            and update quantum bits by using
            quantum update($F_g$, $F^*$),
            Eq (8), (9)
            Here $i$ in binary bits denotes the
            index of conversion process.
          endif
          if (($g == m) \wedge (F^* \leq (c_1 + c_2)$ ))
            Add new neuron for unlearnt sample
            (($c_1 + c_2$)-$F^*$) and
            finalize its weight by using
            Step-1 and Step-2
          endif
        endfor
```

Quantum threshold function()

```
Step-1: Initialization of different parameters
      for t=1 to z
          z is user defined variable to update $\lambda^{'}$
        $\lambda_t^{'}$ = ($\alpha_i^{Th} \mid \alpha_{i+1}^{Th}$);
        count1=0;
        count2=0;
        $F_\lambda^* = 0$;
        $S_\lambda^* = 0$;
        Call conversion process($\lambda^{'}$)
        to generate real value $\lambda_t$
        from quantum value $\lambda_t^{'}$
      for i=1 to $c_1$
        $net_A(i)$=$\sum W_g \times x_i$
```

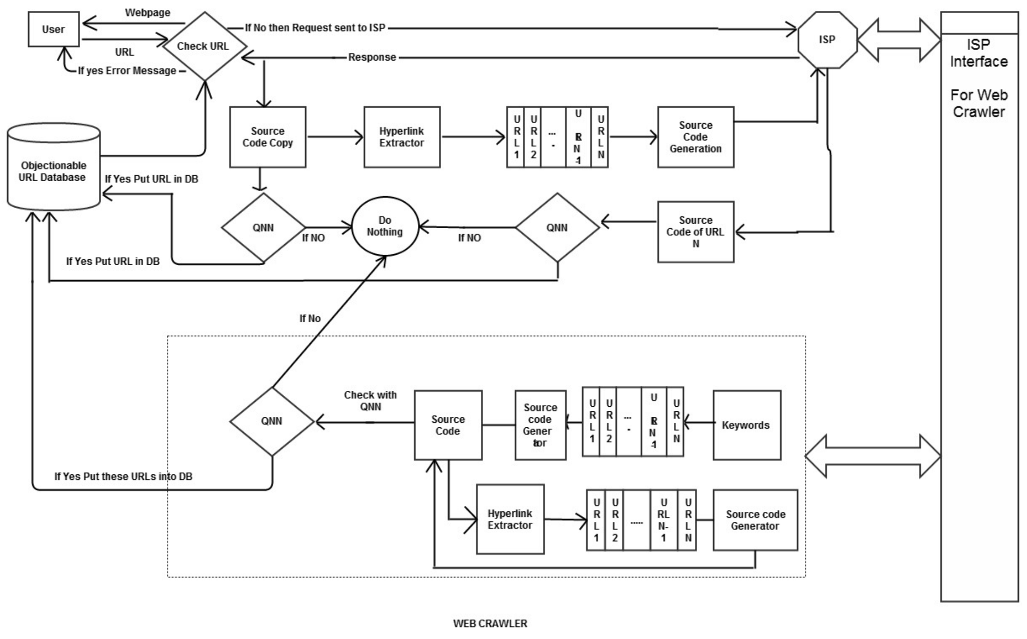

Fig. 2. Architecture of QNN-F

TABLE III. TRAINING AND TESTING SET PARTITION

Training-Testing Partition	Total Training URLs	Objectional URLs in Training Set	Non-objectional URLs in Training Set	Total Testing URLs	Objectional URLs in Testing Set	Non-objectional URLs in Testing Set
60-40	1200	600	600	800	400	400
70-30	1400	700	700	600	300	300

```
            if($net_A(i) > \lambda_t$)
                increase count1 by 1;
            endif
        endfor
        for j=1 to $c_2$
            $net_B(j)$=$\sum W_g \times y_j$
            if($net_B(j) \le \lambda_t$)
                increase count2 by 1;
            endif
        endfor
        ($F_t$ = $count1 + count2$);
        $F^*_\lambda$=max($F^*_\lambda$, $F_t$)
        update quantum bits for $(\lambda')_{t+1}$ by using
        Table I, Eq (13) and Eq (14)
        generate updated real coded value of $(\lambda)_{t+1}$
        corresponding to $(\lambda')_{t+1}$ by
        using conversion process
        $F_g$=$F^*_\lambda$
    endfor
    return $F_g$;
```

F. Detection of Malicious Web Request

Testing of proposed system starts, after the system learns for whole sample or after completion of the user defined number of iterations. As discussed in the above section, the real weight (W_j^{real}) and the threshold (λ_j^{real}) is stored regrading best fitness value F^*, F^*_λ corresponding to particular hidden layer neuron. Once the real weight and real threshold is decided of the hidden layer and output layer, then testing starts. Testing starts with 40% and 30% of dataset by providing them as input to the hidden layer. If the output layer gives output as "1", it means that it belongs to objectionable class "A" otherwise input belongs to non objectionable class "B". The next section illustrates the algorithm with a simple example.

TABLE IV. PARAMETERS USED IN EXPERIMENT

Parameters	Values
Angular displacement ($\Delta\theta$)	$0.03 * \Pi$
Quantum bit length (k)	2
ϵ	0.005

IV. ILLUSTRATION OF QNN-F

The overall process can be easily understood with the help of Fig. 2. The process starts from the Web crawler site. Web crawler fetches list of top objectionable URLs on the basis of objectionable keywords. The source code generator program generates source code or content of all URLs. After generating source code of URLs, all hyperlinks are extracted into a separate file for further analysis. The remaining part of the source code is analyzed and all the features are extracted. Now these features are provided as input to quantum neural network. On the basis of analysis if content is found objectionable then the objectionable URL database is updated by adding accessed URLs otherwise no operation is performed. In the same way, all hyperlinked URLs are also analyzed. Thus, Web crawler regularly updates the objectionable database with number of objectionable URLs.

Now if a user try to access any URL then it is first matched with URLs available in objectionable database. If accessed URL is matched with the objectionable URLs present in database, then a warning message is sent to user regarding not permitting user to access such objectionable URL. If accessed URL does not match with available objectionable URLs then user request is forwarded to an ISP in the form of HTTP Request packet. The ISP replies to the user with HTTP response packet. Now this response is forwarded to the user and a copy of it is kept for analysis. After extracting all the hyperlinked URLs from content, the source content is analyzed for features. All features are provided as input to quantum neural network. On the basis of quantum neural response, if accessed URL is found objectionable then database of URLs is updated by adding this otherwise no operation is performed. Now analysis of all hyperlinked URLs is done. The source code generator program, generates the source code of all hyperlinked URLs one by one. After generating source code features are extracted from the content of hyperlinked URLs. These features are provided to quantum neural network and on the basis of decision, it is kept in objectionable URLs database else no operation is performed on it. The same process is applied here for all hyperlinks one by one.

V. Experimental Work

The Proposed QNN-F as a firewall is implemented in Java(Version7) and on a computer with i7 processor, 2.5GHz and 8GB RAM. For the testing and training purpose, total 1000 objectional and 1000 non-objectional URLs have been randomly selected. The dataset has been prepared in two classes, one is objectionable and another is non objectionable, both are contains eight number of features as described in Table II. These features are collected on the basis of number of objectionable and non objectionable keywords, number of objectionable and non objectionable sentences and their ratio. In the proposed algorithm, objectionable content have been considered as pornographic, religious/racial aggression, proxy bypass and online gaming Websites.

In machine learning field, it is common to partition the dataset into two separate sets: a training set and a testing set. To evaluate the efficiency of our approach, the training and testing data have been divided into two different partitions. Here first partition is done of 60-40, where 60 % of the samples are used for training the classifier and the rest for testing. The second partition is 70-30, 70 % of the samples are used for training and remaining are used for testing. The details of the training-testing partitions are given in Table III. The all required parameters with its values are defined in Table IV

In order to evaluate the performance of our proposed QNN-F architecture, classification accuracy, sensitivity, specificity, confusion matrix are calculated using parameters as true positive, true negative, false positive and false negative [14] .

As shown in Table V, classification accuracy achieved by QNN-F model for 60-40 trainingtesting samples is 96.37% which is very remarkable, it shows that even for smaller training samples, QNN-F model is having good adaptation and generation capability and is able to provide better solutions. The classification accuracy of QNN-F model for 70-30 is 97.83 which confirms the importance of proper training data. Proposed method achieves maximum accuracy of 97.53% and 99.2% for 60-40 and 70-30 trainingtesting partition respectively. The mean and Sttd. Dev. values of sensitivity and specificity of all trainingtesting partitions for our model is presented in Table VI. Table VII represents the classification accuracies in the form of confusion matrix. It is clear from Table VII, the sum of true positive and true negative increases with the increase in training set size. Especially, for 70-30 trainingtesting data the true positive rate and true negative rate is around 100 which again shows that, if proper training data is provided then QNN-F model will work with great efficiency and validity.

In order to compare the proposed approach two well known machine learning algorithms Back propagation neural learning algorithm, SVM (support vector machine) and classifier Q-BNN is considered [3]. The Here Q-BNN algorithm has been modified according to the input data of proposed algorithm. Table V shows the classification accuracy in terms of parameters like mean, standard deviation (Std. Dev.) and Max. In the dataset ratio of 60-40, the mean, Std. Dev. and max value corresponding to QNN-F are 96.375, 0.86 and 97.53 respectively. Meanwhile mean, Std. Dev. and max value corresponding to Back Propagation, SVM and Q-BNN are 92.625, 0.46, 94.12, 95, 0.4, 95.63 and 95.625, 0.32, 96.4 respectively. In the dataset ratio of 70-30, the mean, Std. Dev. and max value are 97.833, 0.36 and 99.2 respectively of QNN-F. Meanwhile mean, Std. Dev. and max value corresponding to Back Propagation, SVM and Q-BNN are 93, 0.83, 94.8, 96.17, 0.58, 97.84 and 96.67, 0.51, 97.1 respectively. Table VI shows the comparison between sensitivity and specificity. With the dataset partition of 60-40, the best value achieved by QNN-F for sensitivity and specificity in terms of mean and Std. Dev. which are 98, 0.53 and 92.75, 0.26 respectively. In the dataset partition of 70-30, the best value achieved by QNN-F for sensitivity and specificity in terms of mean and Std. Dev. are 98.758, 0.62 and 95.67, 0.74 respectively. Meanwhile with the dataset partition of 60-40, the best value achieved by Back

TABLE V. Classification Accuracy of QNN-F

Comparison of classification accuracy				
Classifiers	**Training-Testing Partition** (%)	**Mean**	**Std Dev**	**Max**
QNN-F	60-40	96.375	0.86	97.53
	70-30	97.833	0.36	99.2
Back Propagation	60-40	92.625	0.46	94.12
	70-30	93	0.83	94.8
SVM	60-40	95	0.4	95.63
	70-30	96.17	0.58	97.84
Q-BNN	60-40	95.625	0.32	96.4
	70-30	96.67	0.51	97.1

TABLE VI. Sensitivity and Specificity

Comparison of sensitivity and specificity					
	Training-Testing Partition (%)	**Sensitivity**		**Specificity**	
		Mean	**Std Dev**	**Mean**	**St Dev**
QNN-F	**60-40**	98	0.53	92.75	0.26
	70-30	98.758	0.62	95.67	0.74
Back Propagation	**60-40**	95.254	0.36	85.25	0.5
	70-30	94.654	0.2	86	0.23
SVM	**60-40**	94.584	0.57	90	0.43
	70-30	95.894	0.42	92.33	0.77
Q-BNN	**60-40**	95.86	0.74	91.25	0.61
	70-30	96.35	0.29	93.34	0.41

TABLE VII. Confusion matrix

Confusion matrix					
		60-40		70-30	
Classifiers		Objectionable	Non-objectionable	Objectionable	Non-objectionable
QNN-F	Objectionable	400	0	300	0
	Non-objectionable	29	371	13	287
Back Propagation	Objectionable	400	0	300	0
	Non-objectionable	59	341	42	258
SVM	Objectionable	400	0	300	0
	Non-objectionable	40	360	23	277
Q-BNN	Objectionable	400	0	300	0
	Non-objectionable	35	365	20	280

Propagation, SVM and Q-BNN for sensitivity and specificity in terms of mean and Std. Dev. are 95.254, 0.36, 85, 0.5, 94.584, 0.57, 90, 0.43 and 95.86, 0.74, 91.25, 0.61 respectively.. In the dataset ratio of 70-30, the mean, Std. Dev. value of sensitivity and specificity are 94.654, 0.2, 86, 0.23, 95.894, 0.42, 92.33, 0.77 and 96.35, 0.29, 93.3, 0.41 respectively. Table VII shows the confusion matrix corresponding to all three methods. The results prove that, with respect to objectionable dataset, each algorithm performs well but corresponding to non objectionable dataset, quantum neural network perform better than Back propagation, SVM and Q-BNN in both data partitions.

VI. Conclusion

In this paper, quantum based neural network learning algorithm for detection of malicious Web request is presented. The algorithm forms neural network architecture constructively by adding neurons at hidden layer, where connection weights and threshold of neurons is decided using quantum computing concept. Finding weight and threshold in quantum computing concept provides large subspace to find out optimal value. The proposed algorithm is tested on objectionable or non-objectionable Web dataset. The idea is not only to block content on the basis of objectionable keywords, but the dynamic filtering system is proposed with semantic analysis of contents by enhancing an additional module to increase performance. To speed up the process of blocking objectionable content, Web crawler is presented, which updates the objectionable URL database prior to the request of user. The requested web contents are analyzed at run time with the help of quantum neural network. The hyperlink analysis module analyses web in depth by traversing each hyperlink presented on requested web page and Web pages extracted by Web crawler. The efficiency of proposed classifier for detection of malicious Web request, is also measured on the various parameters such as accuracy, specificity, sensitivity with well known algorithms that is support vector machine (SVM) back propagation neural learning algorithm. This proposed method is also compared with a recent approach [3] that is Q-BNN and the results show that proposed classifier performs better than these methods.

References

[1] S. Kumar, *Neural networks: a classroom approach*. Tata McGraw-Hill Education, 2004.

[2] T. C. Lu, G.-R. Yu, and J.-C. Juang, "Quantum-based algorithm for optimizing artificial neural networks," *IEEE Transactions on Neural Networks and Learning Systems* , vol. 24, no. 8, pp. 1266–1278, August 2013.

[3] O. P. Patel, A. Tiwari,"Quantum inspired binary neural network algorithm," in *Proceeding of 2014 International Conference on Information Technology (ICIT)*. IEEE, pp. 270–274, 2014.

[4] J. H. Kim and S.-K. Park, "The geometrical learning of binary neural networks," *Neural Networks, IEEE Transactions on*, vol. 6, no. 1, pp. 237–247, 1995.

[5] L. Zhang, L. Zhang, and H. Peng, "Quantum clone genetic algorithm based multi-user detection," in *2011 The 2nd International Conference on Next Generation Information Technology (ICNIT)*. IEEE, pp. 115–119, 2011.

[6] M. S. Sharbaf, "Quantum cryptography: An emerging technology in network security," in *2011 IEEE International Conference on Technologies for Homeland Security (HST)*. IEEE, pp. 13–19 , 2011.

[7] S. Feng, J. Zhang, and B. Zeng, "Design of the visualized assistant for the management of proxy server," in *2010 Third International Symposium on Electronic Commerce and Security (ISECS),* . IEEE, pp. 204–208, 2010.

[8] H. Zheng, H. Liu, and M. Daoudi, "Blocking objectionable images: adult images and harmful symbols," in *2004 IEEE International Conference on Multimedia and Expo, 2004. ICME'04.* vol. 2. IEEE, pp. 1223–1226, 2004.

[9] W. Zeng, W. Gao, T. Zhang, and Y. Liu, "Image guarder: An intelligent detector for adult images," in *Asian Conference on Computer Vision*, pp. 1080–1084, 2004.

[10] Z. Chen, O. Wu, M. Zhu, and W. Hu, "A novel web page filtering system by combining texts and images," in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE Computer Society, pp. 732–735, 2006.

[11] U. Thakar, O. Patel, and L. Purohit, "Web shield: A modified firewall to detect malicious request," in *Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam, India*. Springer, pp. 497–506, 2012.

[12] M. Hammami, Y. Chahir, and L. Chen, "Webguard: A web filtering engine combining textual, structural, and visual content-based analysis," *IEEE Transactions on Knowledge and Data Engineering*. vol. 18, no. 2, pp. 272–284, February 2006.

[13] K. H. Han and J. H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 580–593, December 2002.

[14] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.