http://www.scirp.org/journal/ns Natural Science, 2018, Vol. 10, (No. 3), pp: 99-109

Towards a Real Quantum Neuron

Wei Hu

Department of Computer Science, Houghton College, Houghton, NY, USA

Correspondence to: Wei Hu, Wei.hu@houghton.edu

Keywords: Quantum Computation, Quantum Machine Learning, Quantum Neural Network, Quantum Neuron
Received: January 22, 2018 **Accepted:** March 13, 2018 **Published:** March 16, 2018

ABSTRACT

Google's AlphaGo represents the impressive performance of deep learning and the backbone of deep learning is the workhorse of highly versatile neural networks. Each network is made up of layers of interconnected neurons and the nonlinear activation function inside each neuron is one of the key factors that account for the unprecedented achievement of deep learning. Learning how to create quantum neural networks has been a long time pursuit since 1990's from many researchers, unfortunately without much success. The main challenge is to know how to design a nonlinear activation function inside the quantum neuron, because the laws in quantum mechanics require the operations on quantum neurons be unitary and linear. A recent discovery uses a special quantum circuit technique called repeat-until-success to make a nonlinear activation function inside a quantum neuron, which is the hard part of creating this neuron. However, the activation function used in that work is based on the periodic tangent function. Because of this periodicity, the input to this function has to be restricted to the range of [0, π/2), which is a serious constraint for its applications in real world problems. The function's periodicity also makes its neurons not suited for being trained with gradient descent as its derivatives oscillate. The purpose of our study is to propose a new nonlinear activation function that is not periodic so it can take any real numbers and its neurons can be trained with efficient gradient descent. Our quantum neuron offers the full benefit as a quantum entity to support superposition, entanglement, interference, while also enjoys the full benefit as a classical entity to take any real numbers as its input and can be trained with gradient descent. The performance of the quantum neurons with our new activation function is analyzed on IBM's 5Q quantum computer and IBM's quantum simulator.

1. INTRODUCTION

History was made in 2016 when Google's AlphaGo computer program defeated a Go world champion.

One of the key components of this program is the deep neural network and the amazing performance of AlphaGo provides motivation and excitement for extensive research in the area of deep learning. One technique accountable for the success of deep learning is the use of many layers of neural networks in which the output of one layer can be the input of the other and each layer is made up of a number of interconnected neurons. More importantly, there is a nonlinear activation functions inside each neuron, otherwise all these deep neural networks are essentially a single layer network. This is due to the fact that composition of many linear transformations is again a linear transformation which can serve only as a linear regression technique. It is the nonlinearity of neural networks that gives their ability to approximate any continuous functions to solve complicated tasks like language translations, image classifications, or even playing the Go game. Furthermore, the main source of this nonlinearity comes from the nonlinear activation function inside each neuron of the networks.

Mathematically neural networks can be viewed as weighted directed graphs in which neurons are nodes and connections among the neurons are directed edges with weights. The weights represent the strength of the interconnection between neurons. Neural networks learn by adjusting its weights and bias iteratively during a training session to produce the desired output. So each neuron makes decisions by summing the weighted evidence (input). The rules to change these weights during training are called the learning algorithm. Neural networks have found a wide array of applications in supervised learning, unsupervised learning, and reinforcement learning.

Quantum computers are rapidly developing and along with it, the availability of public accessible quantum computers is more of a reality today. This tend makes the study of quantum machine learning algorithms on a real quantum computer possible. With the unique quantum mechanical features such as superposition, entanglement, interference, quantum computing offers a new paradigm for computing. Research has shown that artificial intelligence and in particular machine learning can benefit from quantum computing. It is reasonable to hope that the next historical breakthrough in artificial intelligence like AlphaGo may well be realized on a quantum computer.

We have finished two reports recently using IBM's 5Q quantum computer [1, 2]. One is analyzing a distance-based quantum classifier where we show the prediction probability distributions of this classifier on several well-designed datasets to reveal the inner working of this classifier and extend the original binary classifier quantum circuit to a multi-class classifier. The second work is to compare quantum hardware performance on the decision making of an AI agent between an ion trap quantum system and a superconducting quantum system. Our investigation suggests that the superconducting quantum system tends to be more accurate and underestimate the values which are used by the learning agent while a previous research shows their system tends to overestimate [3].

Since the early days of quantum computing, how to use it in the areas of machine learning to gain the quantum speed up has been a long time endeavor [4-7]. Neural networks can perform versatile learning tasks like clustering, classification, regression, pattern recognition, and more. As such the classical neural networks are among the top targets for researchers to find their quantum counterpart. Numerous efforts have been made and claimed but unfortunately without much success [8]. One of the major difficulties is how to create a quantum nonlinear activation function like the classical sigmoid function inside a neuron, the fundamental building unit of each network layer, since the operations on quantum states are required to be linear and unitary under the laws of quantum mechanics.

Using a new technique called repeat-until-success, the work in [9] shows that it is possible to design a quantum circuit to create a nonlinear activation function inside a quantum neuron. So it can function as a classical neuron to be in the state $|0\rangle$ or $|1\rangle$, but can also be in the superposition of both states $|0\rangle$ and $|1\rangle$, a feat classical neuron cannot accomplish. The key to their design of the nonlinear activation function is using a periodic tangent function. Therefore, it requires the input of this function to be restricted in the range of $[0, \pi/2)$, which is a serious constraint for its uses in real world problems. The periodicity of the function also makes it not suited for being trained with the efficient gradient descent method since its derivatives oscillate. When data is big, an efficient training algorithm is much desired.

One work after [9] is reported in [10], where nonlinearity is realized in a quantum perceptron using

the evolution of a Hamiltonian. Inspired by the work in [9, 10], we set our goal to remove the restriction in [9] by creating a new nonlinear activation function that can take any real numbers as its input and its neuron be trained with gradient descent just like a classical neuron, making this neuron to enjoy features of a fully quantum neuron as well as a fully classical neuron.

2. NEURAL NETWORKS

Neural networks are typically organized in layers, each of which is made of neurons. One kind of commonly used neurons have binary states which are proposed by McCulloch-Pitts [11]. A neuron can be in an active state or a resting state, represented by 1 or 0 in mathematical notation.

Figure 1 shows the structure of a neuron: input $(x_1, x_2, \cdots, x_n)$ and their corresponding weights $(w_1, w, \cdots, w_n)$ and bias b and the activation function f that combines all the components of the input to generate an output. Note that without f, the output is a straightforward linear transformation of the input. Also the value of $\sum_{i=1}^{n} x_i w_i + b$ can be any real number. A neuron is a simple commuting unit with many inputs and one output.

There are several inputs for one neuron with one weight for each input, the weight of that specific connection. When the neuron receives inputs, it sums all the inputs multiplied by its corresponding connection weight plus a bias. The purpose to have a bias is to make sure that even when all the inputs are 0 there can be an output from the neuron. For mathematical convenience, we usually treat the bias as a normal weight that corresponding to an input of constant 1. After computing the weighted sum of its inputs, the neuron passes it through its activation function, which normalizes the result to get the desired output, depending on the purpose of the learning task such as classification or regression. So the key feature of neurons is that they can learn. The behavior of a neural network depends on both the weights and the activation function. Some simple examples of activation functions are step function that returns 1 if the input is positive or 0 otherwise and sigmoid function which is a smooth version of the step function.

The output of one layer becomes the input for the next layer. The first layer (input layer) receives its inputs and its output serves as an input for the next layer. This relay of information is repeated until reaching the final layer (output layer). The networks with this kind of layout are called feedforward neural networks. Other layouts of neural networks are also possible. A neural network can learn from data and store its learned knowledge inside the weights for the connections among the neurons.

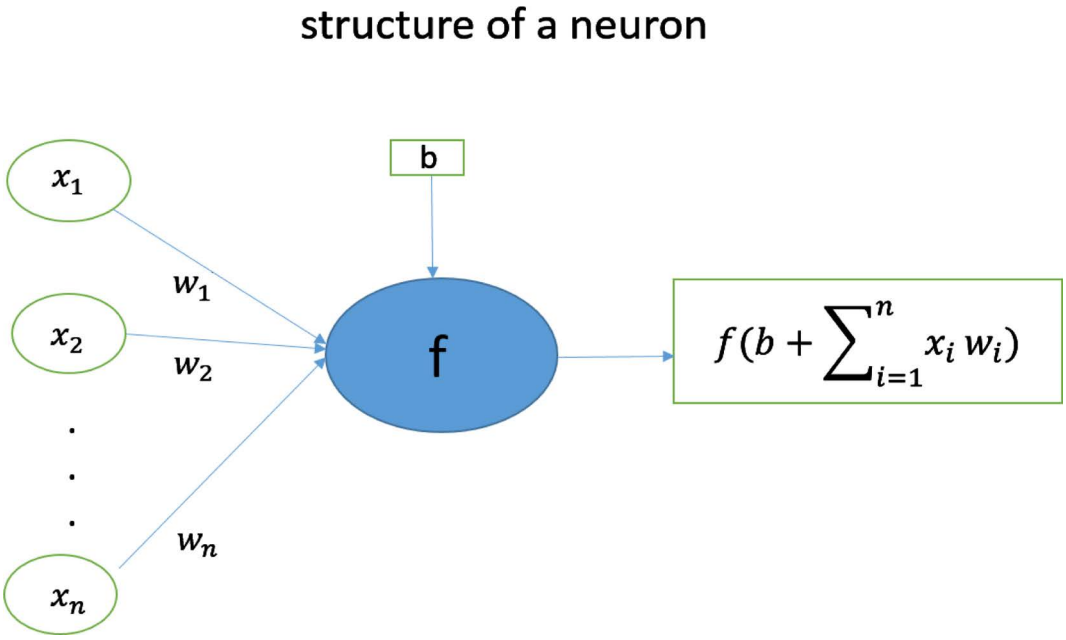

Figure 1. A diagram to show the work of a neuron: input *x*, weights *w*, bias b, activation function f.

3. DESIGN OF A QUANTUM NEURON

Biologically inspired, classical artificial neurons is a mathematical function serving as a model of biological neurons. The hard part of creating a quantum neuron is the design of a nonlinear activation function as the laws of the quantum mechanics require the operations on quantum states be linear.

3.1. Repeat-Until-Success (RUS) Circuit

To implement a quantum algorithm on a quantum computer needs to translate the high level description of the algorithm into a low level physical quantum circuit representation. This task is usually accomplished by two steps: the first is to select a universal gate set, and the second is a decomposition algorithm that can create a quantum circuit with a sequence of the gates from this set. The Solovay-Kitaev theorem [12, 13] is the first result that guarantees a single qubit unitary operation can be efficiently approximated by a sequence of gates from a universal gate set. Since then, many advances have been made but the circuits designed so far are all deterministic. In [14], a new approach is discovered, *i.e.*, using non-deterministic quantum circuits. In this kind of circuits, a unitary operation is applied to a quantum state only if a certain expected measurement outcome is observed. Otherwise a cheap unitary operation can be utilized to reverse it. This process can then be repeated until the desired unitary operation is performed and therefore these circuits are called "Repeat-Until-Success" (RUS) circuits. A clear advantage of RUS circuits is their extremely low resource cost.

3.2. Yet, Another Quantum Neuron

Different versions of neurons have been proposed, but all of them fall short as true quantum neurons [8]. So why is this neuron promising? The short answer is that this time its nonlinear activation function can be executed on a quantum computer.

The idea to create a nonlinear activation function inside a quantum neuron [9] is using a qubit to represent a superstition of two states $|0\rangle$ and $|1\rangle$ with this formula:

$R_y\left(a\frac{\pi}{2}+\frac{\pi}{2}\right)|0\rangle=\cos\left(a\frac{\pi}{4}+\frac{\pi}{4}\right)|0\rangle+\sin\left(a\frac{\pi}{4}+\frac{\pi}{4}\right)|1\rangle$, where $a\in[-1,1]$ and $R_y(t)=\exp\left(-\frac{itY}{2}\right)$ is a quantum operator defined by the Pauli Y operator. When $a=-1$ or = 1, this qubit can be either $|0\rangle$ or $|1\rangle$. In this case, it works like a classical neuron, but when $a\in(-1,1)$, this qubit is in the superposition of $|0\rangle$ and $|1\rangle$ and no classical neurons can do that. The novelty of the work in [9] is using the circuit in Figure 2 to implement this idea on a quantum computer. Further, this circuit can be repeated, say *k* times, to move any point $\theta\in[-1,1]$ closer to one of the ends of the interval $\left[0,\frac{\pi}{2}\right]$. So when $\theta>\frac{\pi}{4}$ this can

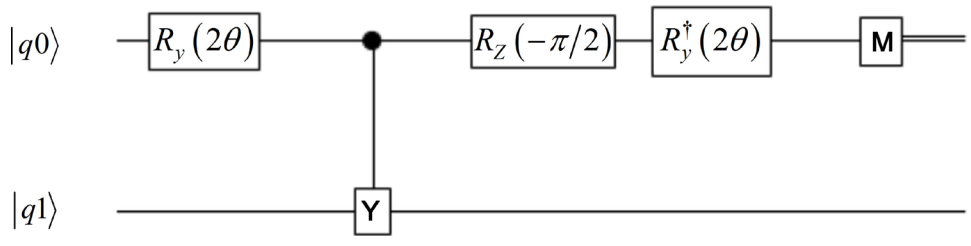

Figure 2. The Repeat-until-success (RUS) circuit created in [9] to generate their nonlinear activation function (Figure 1(c) in [9]). It realizes a rotation with an angle $f(\theta)=\arctan\left(\tan^2\theta\right)$ when the measurement on *q*[0] is 0, otherwise the circuit can be repeated until a 0 is measured. $R_p(\theta)=\exp\left(-\frac{iP\theta}{2}\right)$ where $p\in\{X,Y,Z\}$, the three Pauli operators and M represents a measurement.

move the output of the circuit closer to $|1\rangle$ and otherwise move it closer to $|0\rangle$. It takes k extra ancilla qubits to carry out the k repetitions.

The activation function used in [9] (**Figure 3**) is $f(x)=\arctan\left(\tan^2 x\right)$ that is based on periodic tangent function. As a result, the input to the activation function of their quantum neuron is limited to the range of [0, π/2), which is a serious constraint on any real applications. As **Figure 1** shows, the input to the activation function can be any real numbers. The approach to solve this problem taken by [9] is to use a scaling factor to confine the input within [0, π/2), which has to be tuned for each different application of their neuron. Our approach to resolve this difficulty is to use this function, $f(x)=\arcsin\left(\sqrt{\text{sigmoid}(x)}\right)$, based on sigmoid function that is not periodic (**Figure 3**). Another advantage of our activation function is that our threshold point is 0 that is usually better than the one π/4 used in [9]. As we know that the domain for arcsine is [−1, 1] and its range is [−π/2, π/2], the domain for arctangent is all real numbers and its range is (−π/2, π/2). Essentially, the two activation functions are based on the function's half range [0, π/2) which look like a typical sigmoid function. Finally, how to train these neurons is critical for their practical use in real world problems. As the right plot in **Figure 3** shows the activation in [9] is not suited for gradient descent as its derivatives oscillate over a large domain.

Note that sigmoid function $f(x)=\dfrac{1}{1+\mathrm{e}^{-x}}$ is a widely used activation function in classical neural networks. With its range to be between 0 and 1, it is a good choice if the network needs to produce a probability at the end. It has a very nice mathematics property, $f'(x)=f(x)\left(1-f(x)\right)$, so its derivative is easy to compute and the function itself can be considered as a smoothed out version of a step function. But the maximum value of its derivative is 0.25, during the backpropagation training process the errors are being squeezed by a quarter at each layer, to say the least. Near the two ends of the sigmoid function, its values tend to be flat, implying the gradient is almost zero. As a result, it gives rise to a problem of vanishing gradients. Therefore it may not be a good choice for deep neural networks.

4. RESULTS

In order to show the ability of the quantum neurons, Nelder-Mead algorithm is employed to train them to solve the XOR problem in [9]. To demonstrate the advantage of our activation function, here we use gradient descent to train the quantum neurons with our nonlinear activation function to solve the OR problem and another simpler binary classification problem. To this end, two binary classification datasets are described in **Table 1**.

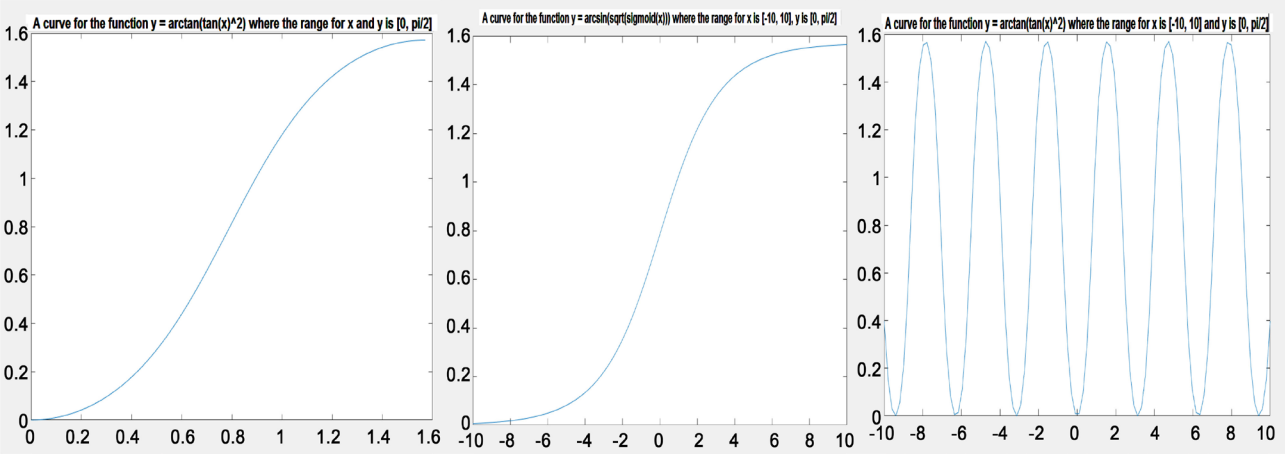

Figure 3. The left function is used in [9] and the middle one is our own invention, and the right one is the left one with a larger domain to show the periodic nature of this function.

4.1. Test Our Neurons on IBM's 5Q Processor and IBM's Quantum Simulator

We use the neuron training circuit in Figure 4 to train our quantum neurons to solve the two binary classification problems as displayed in Table 1.

Figure 5 and Figure 6 illustrate that our quantum neurons can differentiate the data points between classes 0 and 1 within 4 steps of training regardless it is on simulator or on real device. Here we calculate the probability for each data point belonging to class 1, which means that when value of the probability gets closer to 1, it is more likely the point is in class 1, otherwise, it is more likely to be in class 0. One trend in these figures is that the gaps between probabilities for being in class 1 and 0 in the simulator are bigger than those in the real device. And it is easier to distinguish the data points if the gaps are bigger, thus our experiments imply that the performance of our quantum neurons is better on simulator than that on the real device.

4.2. The Advantage of Our Activation Function in Generating a ReLU Activation Function

Another common activation function is Rectified Linear Units (ReLU) $f(x)=\max(0,x)$. This is a simple function, nonetheless is nonlinear. It has become popular in recent years because it is efficient to calculate and can speed up the training process for large networks, compared with the more complicated functions like sigmoid function. Its derivative takes a value of 1 so it rectifies the gradient vanishing problem introduced by sigmoid function. The gradient of the ReLU function does not vanish as x increases, a sharp contrast with sigmoid function. However, its derivative is zero when x is negative, which can result

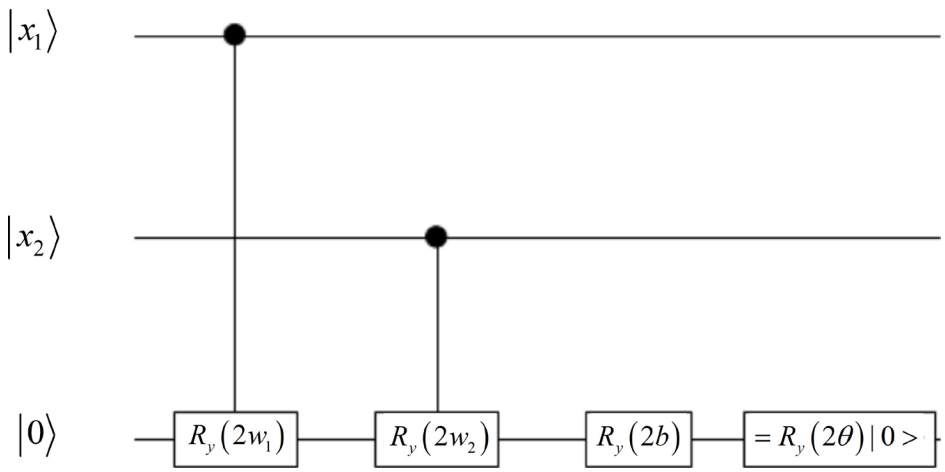

Figure 4. A circuit created in [9] to train the quantum neurons (Figure 4(c) in [9]) using equation $\theta = w_1 * x_1 + w_2 * x_2 + b$.

Table 1. Two datasets created for binary classification to test the quantum neurons with our nonlinear activation function.

Dataset 1		Dataset 2	
Data point	Class label	Data point	Class label
$x_0=(0,0)$	0	$x_{00}=(0,0)$	0
$x_1=(1,1)$	1	$x_{01}=(0,1)$	1
		$x_{10}=(1,0)$	1
		$x_{11}=(1,1)$	1

in "dead" neurons and they can never be activated during the whole training period. Nonetheless, it is used in most convolutional neural networks or deep learning.

The work in [9] proposed a circuit (**Figure 7**) to use their nonlinear activation function to generate the ReLU function. Here we use our sigmoid based nonlinear function to implement this circuit. To show the major difference between their function and ours, we intentionally use a larger domain for their function, *i.e.*, $(-\pi/2, \pi/2)$ to highlight the issue that may cause by the periodicity of their function. We can see from **Figure 8**, their ReLu is only valid if the domain is $[0, \pi/2)$, and beyond that it does not look like a normal ReLu. In contrast, our ReLU function behaves like a normal ReLU function regardless of the size of domain.

Figure 7 is like **Figure 9** in [9], here $q[0] = q[1] = |0\rangle$, so the success is based getting 0 from measuring

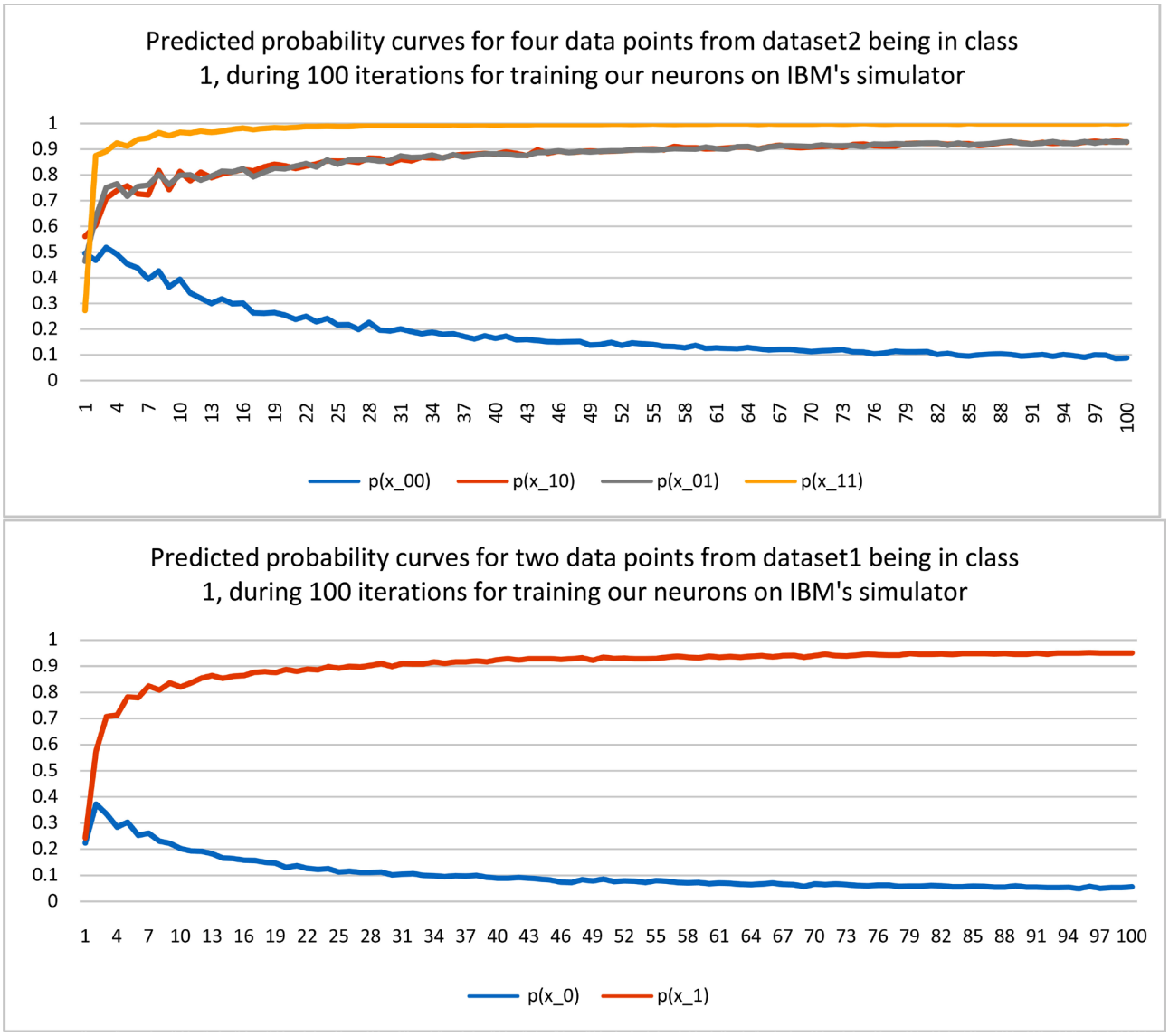

Figure 5. To test the performance of our nonlinear activation function, we train the quantum neurons with our function on the dataset 1 and dataset 2. There are two data points in this dataset where *x*_0 is in class 0 and *x*_1 is in class 1. Similarly there are four data points with *x*_00 in class 0 and *x*_10, *x*_01, and *x*_11 in class 1. We calculate the values of the probability for each data point belonging to class 1 using IBM's simulator with 8192 shots.

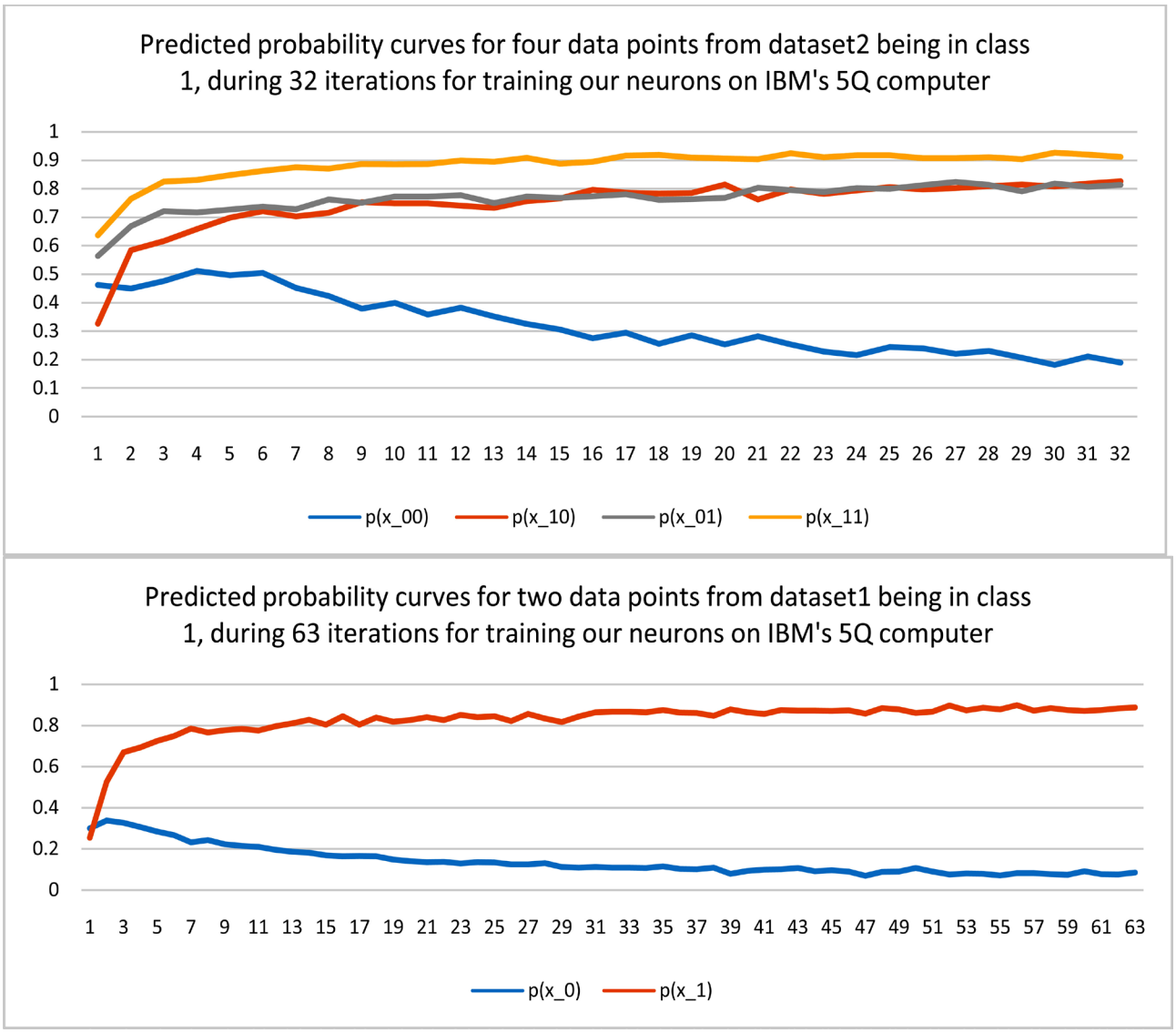

Figure 6. The same as in Figure 5, but this time we use 1000 shots on the IBM's 5Q computer to calculate the values.

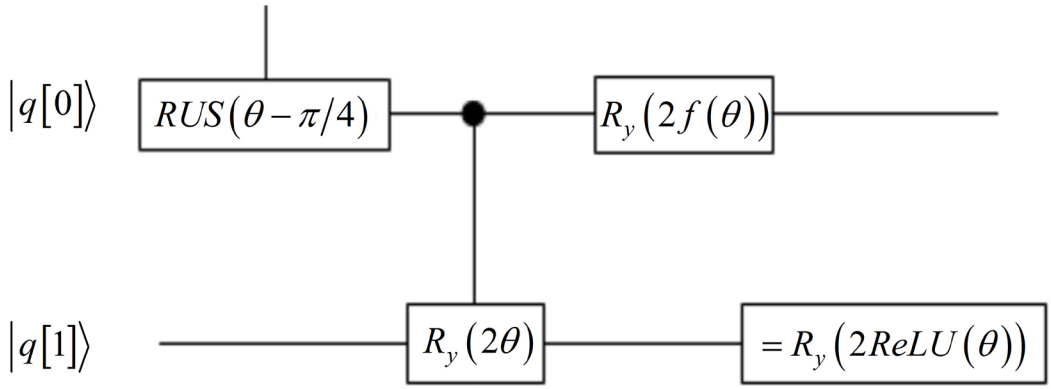

Figure 7. A circuit created in [9] to use their nonlinear activation function to generate the ReLU function (Figure 9 in [9]).

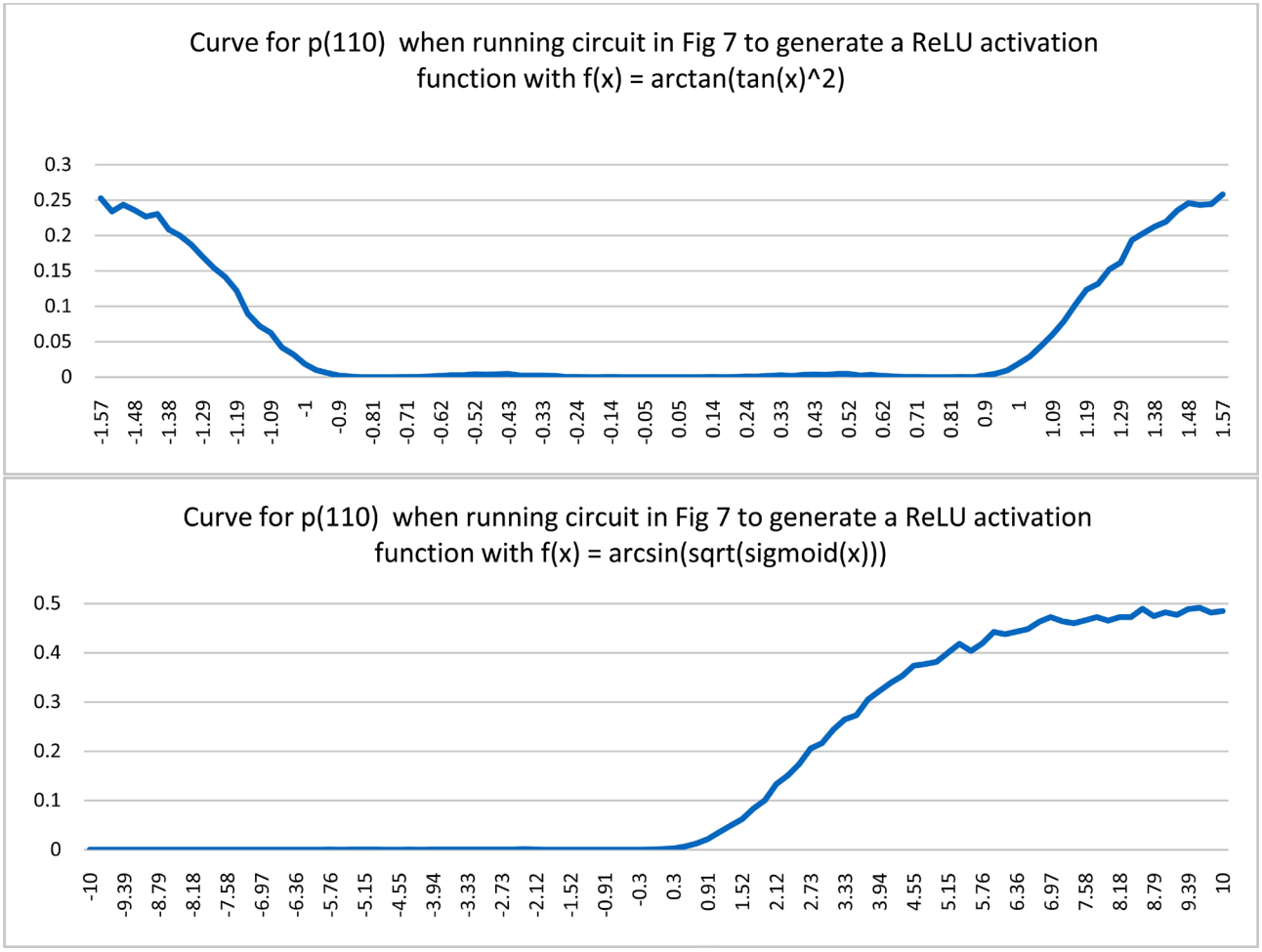

Figure 8. ReLU functions generated by the activation function in [9] (top plot) and ours (bottom plot) using the circuit in Figure 8. P(110) means the probability of seeing 110 in the three measurements: 0 from $RUS\left(\theta-\frac{\pi}{4}\right)$, 1 from $R_y\left(2f\left(\theta\right)\right)$, and 1 from $R_y\left(2ReLU\left(\theta\right)\right)$.

$RUS\left(\theta-\frac{\pi}{4}\right)$, 1 from $R_y\left(2f\left(\theta\right)\right)$, and 1 from $R_y\left(2ReLU\left(\theta\right)\right)$. These three measurements are taken in sequence to generate the ReLU curves in Figure 8.

To summarize our findings, our nonlinear activation function offers the following advantages that the original one in [9] cannot offer: 1) It is not periodic so it can take any real numbers. 2) It can be trained with efficient gradient decent. 3) It can generate a ReLU function that looks more like a classical ReLU function.

5. CONCLUSIONS

As demonstrated by Google's AlphaGo, deep learning has gained its reputation from its unprecedented success in so many areas including computer vision, speech recognition, natural language processing, and many more. The backbone of this great achievement is the workhorse of neuron networks, which is a computational model inspired from the biological neural networks. Mathematically neural networks are nonlinear statistical techniques to estimate unknown functions based on the training of the input data. The most important element in a neural network is neuron and key feature of neuron is that it

can learn.

As quantum computing gets more into machine learning, how to create quantum neural networks has been an attractive topic [8]. One of the challenges in the design of quantum neurons is how to create a nonlinear activation function as quantum operations are required to be unitary and linear. There have been a number of publications that focus on creating quantum neural network [4-7]. One research has found that none of the proposals for quantum neural networks fully exploits both the advantages of quantum physics and computing in neural networks [8].

One recent breakthrough [9] shows that it is possible to construct a simple quantum circuit to generate a nonlinear activation function. However, this function is based on a periodic function making it can only take input in the range of $[0, \pi/2)$, a serious restriction for its uses in the real world problems. One report after [9] uses the evolution of a Hamiltonian to produce nonlinearity in a quantum perceptron [10].

Inspired by the work in [9, 10], we think the activation function in [9] could be further investigated and clarified. For this goal, our study proposes a new nonlinear activation function to remove this limit, thus making it possible to create a real quantum neuron. Another advantage of our neurons is that they can be trained with efficient gradient descent while the original one cannot. Our neurons are tested on IBM's quantum simulator and IBM's 5Q quantum computer on two datasets for binary classification and show their benefit in generating a ReLU activation function that is closer to the classical counterpart than that from [9].

After finishing our study reported here, we find one recent paper [15], which creates a quantum neural network with classical learning parameters. As a result, their network can be trained with gradient descent as well.

ACKNOWLEDGEMENTS

We thank the IBM Quantum Experience for the use of their quantum computers and the IBM researchers, Dr. Andrew W. Cross, Dr. Douglas T. McClure, and Dr. Ali Javadi, who help me learn how to use their quantum computers.

REFERENCES

1. Hu, W. (2018) Empirical Analysis of a Quantum Classifier Implemented on IBM's 5Q Quantum Computer. *Journal of Quantum Information Science*, **8**, 1-11.
2. Hu, W. (2018) Empirical Analysis of Decision Making of an AI Agent on IBM's 5Q Quantum Computer. *Natural Science*, **10**, 45-58. https://doi.org/10.4236/ns.2018.101004
3. Sriarunothai, T., W¨olk, S., Giri, G.S., Friis, N., Dunjko, V., Briegel, H.J. and Wunderlich, C. (2019) Speeding-Up the Decision Making of a Learning Agent Using an Ion Trap Quantum Processor. *Quantum Science and Technology*, **4**, Article No. 015014. https://doi.org/10.1088/2058-9565/aaef5e
4. Andrecut, M. and Ali, M. (2002) A Quantum Neural Network Model. *International Journal of Modern Physics C*, **13**, 75. https://doi.org/10.1142/S0129183102002948
5. Altaisky, M. (2001) Quantum Neural Network. arXiv preprint quant-ph/0107012
6. Gupta, S. and Zia, R. (2001) Quantum Neural Networks. *Journal of Computer and System Sciences*, **63**, 355-383. https://doi.org/10.1006/jcss.2001.1769
7. Zhou, R., Wang, H., Wu, Q. and Shi, Y. (2012) Quantum Associative Neural Network with Nonlinear Search Algorithm. *International Journal of Theoretical Physics*, **51**, 705-723. https://doi.org/10.1007/s10773-011-0950-4
8. Schuld, M., Sinayskiy, I. and Petruccione, F. (2014) The Quest for a Quantum Neural Network. *Quantum Information Processing*, **13**, 2567-2586. https://doi.org/10.1007/s11128-014-0809-8
9. Cao, Y., Guerreschi, G.G. and Aspuru-Guzik, A. (2017) Quantum Neuron: An Elementary Building Block for Machine Learning on Quantum Computers. arXiv:1711.11240

10. Torrontegui, E. and Garc´ıa-Ripoll, J.J. (2018) Universal Quantum Perceptron as Efficient Unitary Approximators. arXiv:1801.00934
11. McCulloch, W.S. and Pitts, W. (1943) A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biology*, **5**, 115-133. https://doi.org/10.1007/BF02478259
12. Kitaev, A.Y. (1997) Quantum Computations: Algorithms and Error Correction. *Russian Mathematical Surveys*, **52**, 1191–1249. https://doi.org/10.1070/RM1997v052n06ABEH002155
13. Kitaev, A.Y., Shen, A.H. and Vyalyi, M.N. (2002) Classical and Quantum Computation. *Graduate Studies in Mathematics*, **47**. https://doi.org/10.1090/gsm/047
14. Paetznick, A. and Svore, K.M. (2014) Repeat-until-Success: Non-Deterministic Decomposition of Single-Qubit Unitaries. *Quantum Information & Computation Archive*, **14**, 1277-1301.
15. Wan, K.H., Dahlsten, O., Kristjánsson, H., Gardner, R. and Kim, M.S. (2017) Quantum Generalisation of Feedforward Neural Networks. *Quantum Information*, **3**, 36. https://doi.org/10.1038/s41534-017-0032-4