Embedding Fock States of the XXZ Heisenberg Model in a Quantum Neural Network with IBM-Q

Daniel Pompa
Lyle School of Engineering
Southern Methodist University
University Park, TX
dpompa@mail.smu.edu

***Abstract*—Integrating so-called hybrid machine learning models on classical and quantum machines is a relatively new venture and growing prospect for identifying optimal computational models. With the creation of the open-access platform, IBM-Q, adaptations to classical machine learning models can be constructed to work, at least partially, in a quantum environment. The focus of this project will be to build a hybrid machine learning model that utilizes a mixture of classical python libraries and intermediary executions of QASM code on the IBM-Q.**

I. Introduction

Since the advent of the first machine learning algorithms, many areas in both industry and academics have been impacted by the computational benefits of their numerous applications. The versatility and effectiveness of these algorithms is a product of the dynamic nature of their design. Machine learning (ML) algorithms are constructed with the intent and purpose to automate processes, that often involve complex calculations or simulations, in a computationally inexpensive and efficient manner. In this way, the improved process performance gained through the use of these algorithms has allowed for significant advancements in a multitude of areas spanning from industry to academics that allows its users to push the boundaries of classical computational methods for use in data analysis. A notable limitation, however, in the use of ML is not necessarily with the algorithms themselves, as they are currently both numerous and ubiquitously applicable across fields, but with the limited nature of the machines through which they are executed.

The primary limitation of the current, deemed "classical", machines is experienced in the way in which they process information. For exponentially large and equally more complex systems of data, the physical restraints of classical processing hardware is often untenable to the size and complexity of the system. This "real-time" processing ineptitude is a known impediment to a number of potential advancements in computer-aided research today. Arguably most notable of these research impediments are in the fields of medicine and chemistry. Current research in these fields often involves highly complex and dynamic systems, that aided by machine learning, have seen promising results, but require multiple classical machines to run in tandem, through the use of parallel processing, or worse, require expensive super computers to run the calculations and simulations that may or may not yield time-sensitive and useful results.

As a possible alternative hardware platform, quantum computers, or machines that are designed to take advantage of certain quantum mechanical properties such as superposition and entanglement by processing information through subatomic particles, Josephson-junctions, or photons, offer an avenue through which systems with greater magnitude and complexity can be configured and managed in a state that is measurable in less time and with less processing demand than that of their classical analog. A natural evolution of machine learning may then be in algorithms designed and developed on quantum machines with the intent to utilize the computational benefits of ML in a quantum mechanical environment that is less bounded in information processing power.

II. Developing ML Algorithms on a Quantum Computer

A. Expectation vs Reality

Though both burgeoning areas of research, the idea of marrying ML with quantum computing has already found a fair amount of attention by researchers in either field. This inevitable wedlock comes not only from the computational allotments both theorized and realized in either developing area, but also with the complimentary nature of their mathematical frameworks. Within the scope of theory, significant research is underway to develop quantum machine learning algorithms, but with quantum computing hardware development still in its infancy, executions of machine learning on actual quantum computers has only more recently been explored. With the creation of IBM's Quantum Experience platform, an open-access environment which allows users to design and test quantum mechanically inspired algorithms on a small-scale quantum computer, a number of research team's have begun to make use of this new technology and have already developed ways to integrate commonly used machine learning libraries into the IBM software development kit known as Qiskit.

B. A Path Forward

One approach to constructing these hybrid quantum-classical machine learning algorithms is through the use of classical optimization techniques that are informed by parameter shifts computed on a quantum device. A popular algorithm used for these purposes is the Variational Quantum Eigensolver or VQE. This eigensolver is an application of the Ritz variational principle where a quantum computer is used to prepare a wave function ansatz of the molecule and estimate the expectation value of its electronic Hamiltonian while a classical optimizer is used to adjust the quantum circuit parameters in order to find the molecule's ground state energy [1]. In other words, VQE is a minimum eigensolver algorithm that can find the minimum eigenvalue of an operator, which when implemented on a quantum system or quantum-like representation of a classical system, yields an approximation of the eigenvalues corresponding to the lowest energy state of the system. VQE was designed to mitigate concerns with decoherence when ran for extended times on actual quantum machines [1].

C. *Using VQE in quantum chemistry*

A useful application of VQE, for instance, could be to solve for the ground state of a 1-dimensional lattice model used to describe polyacetylene $(C_2H_2)_n$ molecules known as the Su-Schrieffer-Heeger (SSH) Model [2]. The SSH model is used in condensed matter and quantum field theory to study the effects of topological features on properties of the molecule. The interactions of electrons and phonons within the model can be measured through the coupling constant g by observing changes to the ground state of the system. These measurements can be done by first constructing the Hamiltonian representative of the SSH model and decomposing it into a series of quantum operators or quantum gates in the case of the IBM-Q circuit. In [2], Samaroo and McGuigan use VQE as an optimization algorithm on the IBM-Q to study and visualize the ground state of the SSH model in order to determine the accuracy of the algorithm when compared to classical approaches.

III. Heisenberg XXZ model

In [5], we built a similar 1-dimensional model, the Heisenberg XXZ model, and constructed its Hamiltonian using the Python library Quspin. The objective of this study was to explore the possibility of encoding a quantum spin ½ system in a binary form in order to train a quantum neural network or QNN.

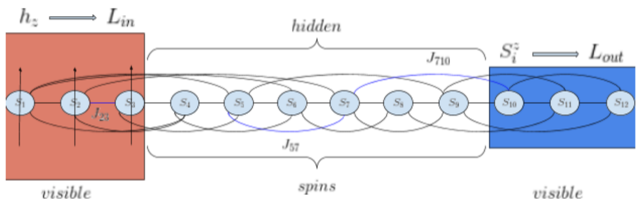

Fig. 1. 1D Spin Model with Coupling

Once the spins are transformed into binary form, a bit count is calculated for each state of the system. An exact diagonalization of the XXZ Heisenberg Hamiltonian is then used to compute the energy eigenvectors and eigenvalues. The model is trained to identify the bit counts of the lowest energy state of the system which is found through a stochastic search algorithm known as simulated annealing. This method is similar to VQE in that it seeks to identify the eigenvalues corresponding to the ground state, with the exception that VQE is an approximation method while exact diagonalization yields the exact eigenvalues.

IV. Building a Hybrid ML-Q Model

In this experiment, an eigensolver is built to mimic the functionality of a VQE, like the one used in [2], but designed to identify the ground state of the XXZ Heisenberg Hamiltonian and with an interface to an IBM-Q backend. The eigensolver is based off the simulated annealing algorithm used in [5] with a bit string encoded into the initial qubit states of an 8-qubit circuit. The initialization of an 8-qubit circuit with this bit-string encoding is then analogous to the 1D spin 1/2 chain [5] with the binary basis states, or Fock states, representative of any given integer value.

V. Exact Diagonalization of the XXZ Heisenberg Model

The system we examine is a relatively small 1D spin chain, simulated by our 8-qubit system. These small spin systems have been an area of interest for nearly a century now since Hans Bethe first developed his famous Bethe ansatz approach to solving the Heisenberg model in 1931 [6]. The more general Heisenberg model has also been used to characterize the interactions between the magnetic moments of 3D crystal lattices and has been shown to be an accurate description of ferromagnetism and antiferromagnetism [6]. The origin of these exchange interactions is found in the antisymmetry of the wave functions and the constraints imposed on the electronic structure [7].

The Heisenberg model consists of a spin Hamiltonian that relates neighboring spins through their exchange interactions, or spin-spin couplings which we denote as J, and the spin operators for the corresponding dimensions of each spin [6]. The general Heisenberg XYZ model is given by,

$$H = -\frac{1}{2}\sum_{j=1}^{N}\left(J_x\sigma_j^x\sigma_{j+1}^x + J_y\sigma_j^y\sigma_{j+1}^y + J_z\sigma_j^z\sigma_{j+1}^z\right). \quad (1)$$

If we single out one direction, the z-direction, and we can assume $J_x = J_y$, then we can simplify this to,

$$H = -\frac{1}{2}\sum_{j=1}^{N}\left(2J_x\left(\sigma_j^+\sigma_{j+1}^- + J_y\sigma_j^-\sigma_{j+1}^+\right) + J_z\sigma_j^z\sigma_{j+1}^z\right), \quad (2)$$

which is known as the Heisenberg XXZ model. Due to the duality transition of the Pauli spin matrices [3] we can simplify this even further to the form,

$$H = \frac{1}{2}\sum_{j=1}^{N} \frac{J_{xy}}{2}\left(S^{+}_{j+1}S^{-}_{j} + h.c.\right) + J_z S^z_{j+1}S^z_j, \quad (3)$$

and if we introduce an external magnetic field we get,

$$H = \sum_{j=0}^{L-2} \frac{J_{xy}}{2}\left(S^{+}_{j+1}S^{-}_{j} + h.c.\right) + J_z S^z_{j+1}S^z_j + h_z \sum_{j=0}^{L-1} S^z_j. \quad (4)$$

The addition of this external magnetic field is necessary for our particular model as we will use it later to fix the orientation of certain spin sites in order to make predictions about their states. Also, notice here how we change our notation of the upper bound from N to L as L (representing the length of the chain) will be used to represent the number of spins in our system, and our model will then be the sum of the interactions between spins in the 1D spin chain.

The $S^z_{j+1}S^z_j$ terms in the Hamiltonian (4) measure the mutual alignment or misalignment of spins at sites j and $j+1$; they lie on the diagonal of the Hamiltonian matrix. On the other hand, the ladder terms generate a new state (quantum fluctuations), and these will enter as off-diagonal terms [8]. The diagonalization of the Hamiltonian matrix is a strategically advantageous alternative to the root-polynomial method of finding the eigenvalues of a system [11]. We use a method known as exact diagonalization to compute the eigenstates of a 1D system.

In theory, all the eigenstates of the Hamiltonian can be computed exactly for a finite system, provided that the system being evaluated is relatively small, by diagonalizing the Hamiltonian numerically. The reason the system must be small is due to the unavoidable exponential increase in the basis size that grows as 2^L for some L positive integer number of L with $s = \frac{1}{2}$ spins. Nevertheless, exact diagonalization solutions are valuable for testing the correctness of quantum Monte Carlo programs and for examining certain symmetry properties of many-body states. The exact diagonalization method itself relies on taking advantage of symmetries, e.g. the conservation of the magnetization, to block diagonalize the matrix as shown in Figure 2.

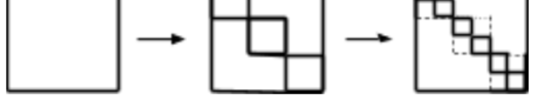

Fig. 2. Block Diagonalization

In order to compute the energy spectrum of the 1D spin system through an exact diagonalization of the Hamiltonian we use an open-source python package called Quspin. The Quspin library is designed to help perform computational methods like exact diagonalization and quantum dynamics of spin(-photon) chains [9].

VI. Quantum Neural Network of Spins

We observe that there exists certain distinct similarities between artificial neural networks (ANN) and quantum spin systems in terms of their overall structure. Like ANN, quantum spin systems consist of some number of "nodes"—sites in the lattice—which are interconnected with various weights—the exchange interactions J_{ij}. If each site is represented in a basis of up and down spin configurations then when the Hamiltonian acts on the system, $\widehat{H}\,|\Psi\rangle$, the spin operators will flip the spins of their corresponding sites. For example, the $j = 1$ term in the XXZ model (4) induces spin arrangements in the following manner:

$$S^+_1 S^-_2 \,|{\downarrow_1}{\uparrow_2}{\uparrow_3}\rangle = \left(S^+_1\,|{\downarrow_1}\rangle\right)\left(S^-_2\,|{\uparrow_1}\rangle\right)\left(|{\uparrow_3}\rangle\right) = |{\uparrow}{\downarrow}{\uparrow}\rangle \quad (5)$$

For spin ½ systems there are 2^L possible configurations for any given system. These possible configurations represent the basis states of that system. For example, a system consisting of 8 sites would have 2^8 or 256 possible configurations or basis states.

Constructing these binary representations lends to the conversion of each binary basis state, or Fock state, to just a single integer value, the bit-count, which is used in the optimization model as target values for predictions. An initial conversion of the Fock states is computed by iterating through each binary vector and adding up all of the spin values, the 0's and 1's, to produce a bit-count. Each vector's bit-count is then stored in an array. The elements of this bit-count array will function as the target values that the model will try to predict. An example of what this process would yield for a system of 3 spin sites with 2^3 or 8 basis states is as follows:

$$|{\downarrow\downarrow\downarrow}\rangle = |000\rangle = 0 \quad (6)$$
$$|{\downarrow\downarrow\uparrow}\rangle = |001\rangle = 1 \quad (7)$$
$$|{\downarrow\uparrow\downarrow}\rangle = |010\rangle = 1 \quad (8)$$
$$|{\downarrow\uparrow\uparrow}\rangle = |011\rangle = 2 \quad (9)$$
$$|{\uparrow\downarrow\downarrow}\rangle = |100\rangle = 1 \quad (10)$$
$$|{\uparrow\downarrow\uparrow}\rangle = |101\rangle = 2 \quad (11)$$
$$|{\uparrow\uparrow\downarrow}\rangle = |110\rangle = 2 \quad (12)$$
$$|{\uparrow\uparrow\uparrow}\rangle = |111\rangle = 3 \quad (13)$$
$$\Rightarrow target = [0,1,1,2,1,2,2,3]\,. \quad (14)$$

The model is then constructed by separating spin sites into three categories; L_{in} spins which act as the input sites, L_{out} spins which output the predictions for the target values to match the L_{in} spins, and all the remaining spins act as memory neurons in the hidden layer of the neural network. For the

IBMQ implementation of our model, the L$_{\text{in}}$ spins are set and fixed by the application of measurement gates in the circuit.

The L$_{\text{out}}$ spins are then reconfigured to replicate the target bit-counts of the L$_{\text{in}}$ spins by encoding the $\widehat{S_z}$ measure of the 2$^{\text{L}_{\text{out}}}$ possible outputs. Here is where the XXZ Heisenberg Hamiltonian (4) is called and the eigenvalues and eigenvectors are computed. In order to determine the state of each L$_{\text{out}}$ spin we calculate the expectation of $\widehat{S_z}$ acting on the 2$^{\text{L}}$ spins that correspond to the ground state eigenvector. This way the spin ↑ and spin ↓ given by the ± 1/2 at each site corresponding to the lowest energy state can be compared to the bit value at that site, with the expectation value of $\widehat{S_z}$ given by,

$$\left\langle \psi \left| \widehat{S_i^z} \right| \psi \right\rangle = \sum_{n=0}^{2^L} |V_{n,0}|^2 \left(\pm \frac{1}{2} \right). \tag{15}$$

Here $V_{n,0}$ is the eignevector corresponding to the ground state of our Hamiltonian. With the expectation value calculated, we then iterate through each configuration of the subsystem and make this comparison in order to set the $\widehat{S_z}$ values in L$_{\text{out}}$. Finally, we make a short loop over the range of L$_{\text{out}}$ to accumulate our predicted bit-counts. Once the bit-counts are accumulated, a chi square test measures the error between our predictions ant the initial target values,

$$accumulated\ error = (prediction - target)^2. \tag{16}$$

VII. Optimization with Simulated Annealing

Next, we choose an optimization model that is both effective and appropriate for the dynamics of our system. In the case of simple, low dimensional models, analytic methods like computing derivatives and solving equations can be used to find optimal model parameters. The leading machine learning methods that incorporate near-term quantum computers primarily make use of some variation of gradient descent.[2] However, as the dimensionality and complexity of a model increase, a alternative methods, like the Nelder-Mead method, are often necessary to find an acceptable local maximum or minimum [1]. In our case, we use a search method based on an optimization problem addressed in physics that attempts to find the lowest energy configuration known as simulated annealing [10].

Given a number of sites $s_i = 1, 2, ..., L$, each with two possible values ↑ or ↓, the optimization problem is to find the configuration of all the sites that minimizes the cost or energy,

$$E = -\frac{1}{2} \sum_{i,j=1}^{L} w_{i,j} s_i s_j, \tag{17}$$

with weighted coefficients $w_{i,j}$ with values dependent on the interactions between neighboring sites.

In physics, a process known as annealing is used to try to find the lowest energy of a system. This process of annealing involves the slow and gradual cooling of a material from a very high temperature to a very low temperature. The motivation behind this process is that at very high temperatures, the relative orientation of the magnetic dipoles in a material is highly randomized. As the material is cooled these tiny magnets will begin to either align or anti-align with one another in such a way that minimizes the energy distributed among them and thus create a material that is more stable in its configuration. The higher the temperature T and/or the slower the cooling schedule the more likely the system will find its global minimum. The configurations of these systems can be thought of in terms of their corresponding probabilities. If each configuration is indexed by γ, then its probability is given by,

$$P(\gamma) = \frac{e^{\frac{-E_\gamma}{T}}}{Z(T)}, \tag{18}$$

with the exponential term in the numerator being the Boltzmann factor, and in the denominator is known as the partition function.

The partition function is given by the sum over all the possible configurations [13],

$$Z(T) = \sum_{\gamma'} e^{\frac{-E_\gamma}{T}}, \tag{19}$$

and the probability of finding the state in energy E_γ decays exponentially. The sum in the denominator is due to the exponential decrease in the number of states with increasing energy. This equation also shows the dependence of the probability on T. When T is high, the probability is evenly distributed among all of the possible configurations and when T is low the probability is concentrated at the lowest-energy configuration [10]. Though this interpretation works well to describe systems with sites that are independent of one another, we wish to evaluate systems with a changing interdependence on site-site connections.

Simulated annealing therefore utilizes a method of finding the minimum energy of a system through a random probabilistic analysis of the change in energy with respect to the system's pseudo-cooling schedule. The pseudo-cooling schedule is determined by the temperature T and is initially set to be high e.g. $T \geq 200$. We then assign random values to all the J couplings in order to randomize the states. Next, we calculate the energy E_a in the current state and compare it to the energy E_b of a new state that is found after the J couplings have been assigned new values. If the energy E_b is less than the previous energy E_a, we accept the change in state. If the energy is not a less but greater, then we accept the change with a probability that is equal to,

$$e^{\frac{-E_{ab}}{T}} \tag{20}$$

,where $\Delta E_{ab} = E_b - E_a$. This acceptance of unfavorable energy has an advantage as it allows the system to jump out of undesired local minima and continue searching for the desired global minimum [10]. Following this step, the temperature is lowered incrementally according to the pseudo-cooling rate and the process is repeated. As the temperature decreases so does the probability that a new energy will be accepted and so the simulation will eventually terminate when T is either very "low" or the algorithm reaches its max number of steps.

The effects of T with various pseudo-cooling rates and system size L can be seen in figure 3. In this experiment [5], the system size was scaled between $L = 6 - 10$ with $L_{in} = 3,\ L_{out} = 2\ and\ T = 800$.

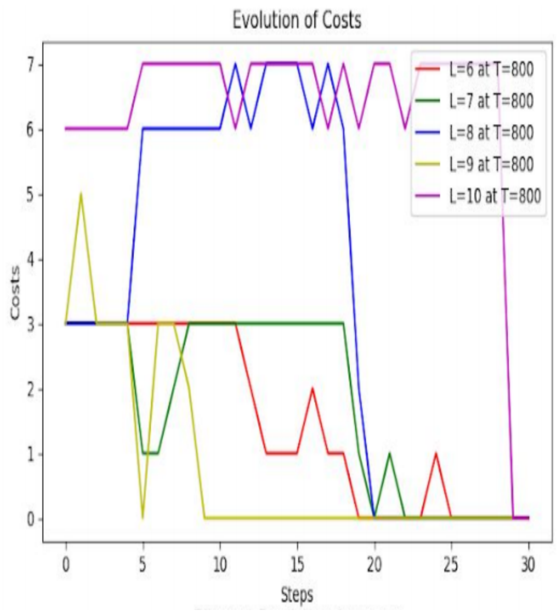

Fig. 3. Minimum Cost Convergence with Classical QNN

Each system was shown to converge to the minimum costs within the 30 steps of the algorithm with a prediction accuracy of 100

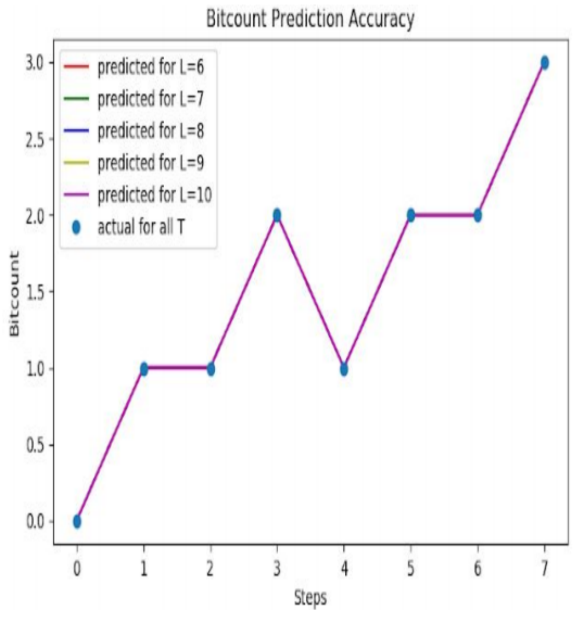

Fig. 4. Bit-count Predictions with Classical QNN

however, with the implementation of this QNN computed strictly on a classical machine, the computational complexity grows as a 2^3 [5],

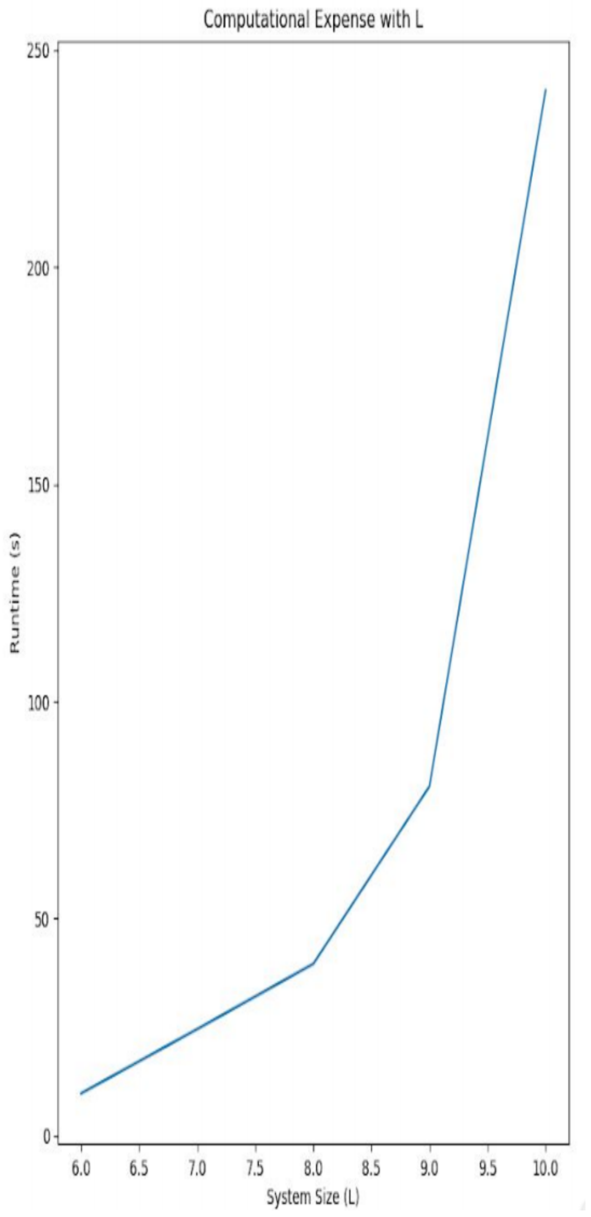

Fig. 5. Computational Complexity with Classical QNN

VIII. Integrating ML model with the IBM-Q

One of the goals of this experiment is to execute the parameter shifts made by the cost function of our ML model on a near-term quantum device. To simulate this implementation, we use Qiskit Aer's $'qasm_simulator'$ backend. By embedding each binary string into an 8-qubit circuit on a quantum machine and calculating the $J - coupling$ weights as parameter shifts in the circuit, the cost function can now be set to update the application of the Hamiltonian (4) on each qubit acting as a spin site. A method for parameter shifts within the circuit, similar to the trial circuits used in [2] is created within the annealing function of the ML model. Two single-qubit gates, $R_Y(\theta)$ and $R_Z(\theta)$, with arbitrary initial values are applied to the qubits representing the L_{out} and hidden layer sites. The rotational parameter change θ for each $R_Y(\theta)$ and $R_Z(\theta)$ gate is calculated as a result of the cost function that returns the values of the J weights to the annealing function given the measure of the probability of acceptance. If the probability of acceptance P based on the accumulated error of the cost function is $P \geq randomnumber \leq 1$, then the parameter θ shifts in unity with the weight J corresponding to that qubit.

The annealing function then acts to minimize the cost of predictions found through the repeated application of the Hamiltonian (4) by identifying the configuration corresponding to the ground state. This then leaves the final configuration of the 8-qubit circuit representative of the ground state of the XXZ Heisenberg Hamiltonian (4) with Fock states depicting the L_{in} and L_{out} bit configuration,

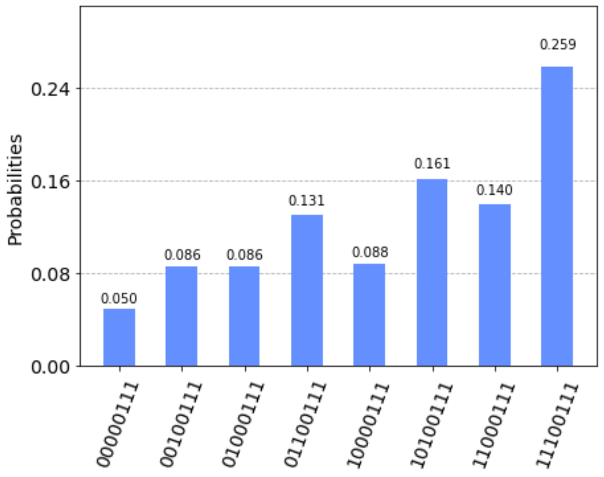

Fig. 6. IBMQ count results of QNN

Note the 3 leading bits of each set of states in Figure 6 are the L_{out} Fock states. Prediction results of the target bit-count values are consistent with [5] as shown in Figure 7.

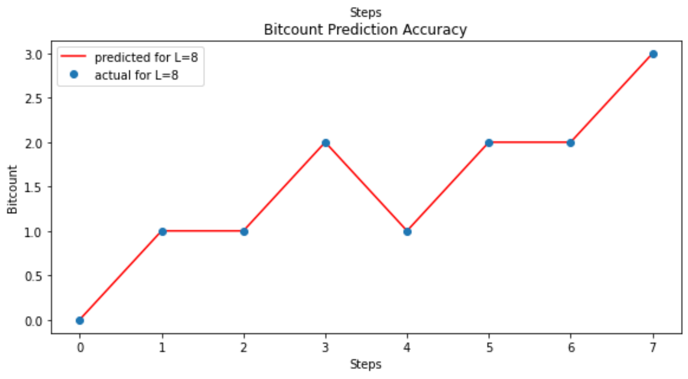

Fig. 7. Prediction results of QNN

IX. Conclusion

A successful implementation of a hybrid QNN was simulated with the IBM-Q.Aer $'qasm - simulator'$ backend. A simulated annealing algorithm incorporating a cost function that minimizes the error in predictions of an 8-qubit circuits bit-counts by calculating the expectation values of the $\widehat{S_z}$ of the XXZ Heisenberg Hamiltonian (4) was run and the bit-count values of each Fock state of the of the L_{out} qubits used to predict the state of the L_{in} qubits.

As of this point, the method used in this experiment is predominantly classical in terms of device execution. A more complete implementation of this method may be applied on an IBM-Q machine once the machines are scaled-up and the number of jobs per active client increases. This would allow for multiple iterations of the cost function with outputs informed by the quantum circuit.

References

[1] A. Peruzzo, et al., “A variational eigenvalue solver on a quantum processor,” arXiv:1304.3061v1 [quant-ph]
[2] A. Samaroo, M. McGuigan, ”Using IBM-Q to study and visualize the ground state properties of the Su-Schrie.er-Heeger model, New York Scientific Data Summit (NYSDS), 2018
[3] Fisher, Matthew P.A. . Duality in Low Dimensional Quantum Field Theories [PDFfile]. Retrieved from https://www.kitp.ucsb.edu/sites/default/files/users/mpaf/publications/p109.pdf
[4] J. P. Hague and C. Maccormick, New Journal of Physics, vol. 14, pp. 033019, 2012.
[5] D. Pompa, K. Beach “Encoding a 1-D Heisenberg Spin 1/2 Chain in a Simulated Annealing Algorithm for Machine Learning,” unpublished.
[6] Lenhard L. Ng. (1996) Heisenberg Model, Bethe Ansatz, and Random Walks [PDF file]. Retrieved from https://services.math.duke.edu/ ng/math/papers/senior-thesis.pdf
[7] Giannozzi, Paulo . Exact Diagonalization of Quantum Spin Models [PDF file]. Retrieved from http://www.fisica.uniud.it/ giannozz/Corsi/MQ/LectureNotes/mq-cap11.pdf
[8] Lee, Christina C. . Quantum Spin Chain Prerequisites [Blog post]. Retrieved from http://albi3ro.github.io/M4/graduate/Spin-Chain-Prerequisites.html
[9] Weinberg, Phillip and Bukov, Marin. arXiv:1804.06782
[10] Duda, Richard O. , Hart, Peter E. , and Stork, David G.. Pattern Classification. New York, NY. John Wiley Sons, Inc, 2001. Print
[11] Weimer, Hendrik. arXiv:1704.07260