Quantum Boltzmann Machines

Yurun Tian, Prabh Simran Singh Baweja, Jaron Powser

May 2021

1 Motivation

The rapid advancement of practical quantum supremacy offers to fundamentally redefine computational problems and their available solutions in countless fields. Problems which are classically intractable may become approachable, the cost-prohibitive may become affordable, and issues of scalability may be mitigated.

1.1 Quantum Computing and Machine Learning

Machine learning is a domain which has been formally described for decades, but has dramatically accelerated in popularity as classical hardware has advanced and high-performance computing resources have become more widely available (1). Classical machine learning models require computation in high-dimensional space, while deep learning models such as the Boltzmann machine require repetitive training with large data sets to be effective.

Quantum computation offers another solution to improving the tractability of machine learning problems like these. With respect to the Boltzmann machine, there exist two methods for improvement on classical approaches. Quantum annealers offer a specialized path to modeling deep learning networks, offering improved performance over classical methods, but at the cost of reduced tunability. The second method offers speedup through improved sampling (1).

As general purpose quantum computers using gates and circuits have become more accessible than those using quantum annealing, techniques for translating or simulating the performance of quantum annealers have been developed. The Quantum Approximate Optimization Algorithm (QAOA) is one technique for simulating the analog processes used in annealing (2).

1.2 Objective

In this paper, we examine the use of QAOA to heuristically model a Boltzmann machine and evaluate its performance (2).

2 Literature Review

As a result of its ability to be implemented with quantum annealing processors, the Boltzmann machine has been widely studied since the earliest usable quantum computers were made available. Contemporary interest is concerned with techniques for implementing the model using circuit-based quantum computers. QAOA offers a bridge between the two architectures.

2.1 Quantum Annealing and the Boltzmann Machine

The Boltzmann machine is a model which stood as an early quantum computing application because it does not require a quantum computer with universal logic; quantum annealing is capable of effectively implementing the neural network in analog (2, 3). As these types of quantum computers were more quickly developed and scaled up than universal logic gate quantum computers, the quantum Boltzmann machine was especially well-suited to early explorations of quantum advantage and supremacy (1, 3). The quantum annealers offered the first opportunity to practically experiment with quantum machine learning using models suited to their working within their constraints (3).

2.2 From Annealing to Circuits

Since the advent of quantum annealers, general-purpose quantum computers using gate and circuit logic have advanced and become far more widely available. QAOA emerged as a circuit-based hybrid classical-quantum algorithm which could be used to solve heuristically solve combinatorial optimization problems in 2014 (4). Originally designed to address the Max-Cut graphing problem, QAOA is now leveraged in a variety of domains and applications.

QAOA effectively serves the role of the quantum annealers in implementing the quantum Boltzmann machine by "approximately simulating the adiabatic evolution" which is performed in analog by quantum annealing (2). This allows the analog representation to be heuristically rendered in circuit logic and run on a general-purpose quantum computer.

2.3 Machine Learning and Quantum Supremacy

Whether or not quantum machine learning offers a practically applicable speedup remains a debated research question alongside the broader debate over quantum supremacy. There are contemporary research papers which attest to improvement over classical approaches to training and using machine learning models, and multiple avenues for gaining quantum advantage have been identified (5).

3 Classical Boltzmann Machines

A Boltzmann machine is a type of a recurrent neural network, where the nodes receive and send signals in sequence along the edges of the graph. These neural networks are capable of self-modification according to some objective, causing them to gradually approach modeling the solution to a learning problem. Figure 1 describes an example of a classical boltzmann machine.

3.1 Boltzmann Machine Structure and Function

The nodes in a Boltzmann machine send binary signals; the state of each node can be said to be on or off within some global state. A node's individual state is determined probabilistically based on input from other nodes or the environment. Inputs to a node are scaled by the "weight" parameter of edges and the "bias" parameter of nodes (6).

When evaluated repeatedly in sequence, the network converges to a state expressible as a Boltzmann distribution (6), from which the model takes its name. The Boltzmann distribution is a probability distribution developed to describe thermal equilibrium. It expresses the probability of a state as a function of "energy" or "temperature". This physics-derived nomenclature also leads to a converged Boltzmann machine to be declared "at thermal equilibrium," and the process of reaching this state is called "thermalization" (2).

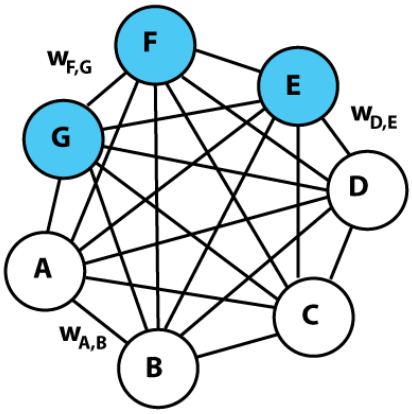

Figure 1: An example of Boltzmann Machine

3.2 Learning with a Boltzmann Machine

The Boltzmann machine can be applied to a learning problem by dynamically updating weights to the edges and bias to the nodes to train the network. These

values are updated little by little. Each update seeks either to maximize the probability value of a given training input or to approach thermalization. (6)

If a node receives information from the environment, it is said to be a "visible" node. If a node only receives information from other nodes, it is said to be a "hidden" node. When adjusting the model parameters according to training inputs, visible nodes are said to be "clamped" to a training vector; that is, their values represent the input. This approach is alternated with allowing the model to approach convergence freely until the parameters of the model adequately model the objective function of the learning problem (6, 7).

The input state vectors express the energy parameter of the Boltzmann distribution, and sampling the converged network yields a probability value associated with a given input. This probability expresses the suitability of a given input vector as a solution to the function expressed by the weights of the model which have been updated according to the training data (6).

3.3 Restricted Boltzmann Machines

Instead of studying the general form of Boltzmann Machine, we focus on the restricted version of it, which is called Restricted Boltzmann Machine (RBM). A basic form of a RBM is a two layer neural network where one layer contains only visible nodes (v_1, v_2, v_3) and the other contains only the hidden nodes (h_1, h_2). Each visible node is connected to each hidden node with a weight w_{ij}. Hidden nodes and visible nodes have their own biases a_j and b_i. There are edges between these two layers but within the layers. It's also known as a bipartite graph mathematically as shown in the figure 2.

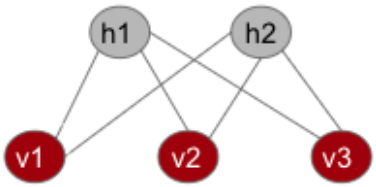

Figure 2: An example of Restricted Boltzmann Machine. A bipartite graph composed of visible nodes(red) and hidden nodes(grey).

After training, conditional probabilities $P(h_j = 1|v)$, $P(v_i = 1|h)$ and joint probability $P(v, h)$ will be obtained. Equations 1 to 2 are the expressions for conditional probabilities and equation 3 describes the joint distribution. Energy function which is the objective function of RBM is shown in equation 4. The goal of training is to minimize the overall energy of the system.

$$P(h_j|v) = \sigma(a_j + \sum_i w_{i,j} v_i) \tag{1}$$

$$P(v_i|h_j) = \sigma(b_i + \sum_j w_{i,j} h_j) \tag{2}$$

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{3}$$

$$E(v,h) = -\sum_i v_i a_i - \sum_i h_i b_i - \sum_{i,j} v_i b_j w_{i,j} \tag{4}$$

3.4 Bottleneck

The process of thermalization is the computational bottleneck which dominates a Boltzmann machine learning model. The neural network must advance from its initial state to its converged state before it can be used to evaluate a data vector. It is this thermalization process that quantum Boltzmann machines seek to improve upon (1, 2).

4 Quantum Boltzmann Machines

Quantum Boltzmann machines were among the earliest applications of quantum computing due to the ability of quantum annealing to implement neural networks in analog and the potential of these types of quantum computers to rapidly develop (1). However, contemporary efforts are typically focused on circuit-based implementations on general-purpose quantum computers (2).

Quantum algorithms depend on the use of an Ising Hamiltonian, which is a quantum function modeling the energy function used in the Boltzmann machine (2, 3). The use of this function arises from usage in physics with which the Boltzmann machine and quantum computing share lineage.

4.1 Quantizing the RBM

Figure 3 illustrates the analogy in terms of three key elements of a classical machine learning method: training dataset, objective function (energy function) and evaluation metric (KL-divergence). The traning data is conceptually equivalent to the quantum distribution of traning data. The energy function is fitted into the quantum energy operator which is expressed by the Hamiltonian matrices of Ising model. The KL-divergence is mapped to the relative entropy between the output distribution and training distribution.

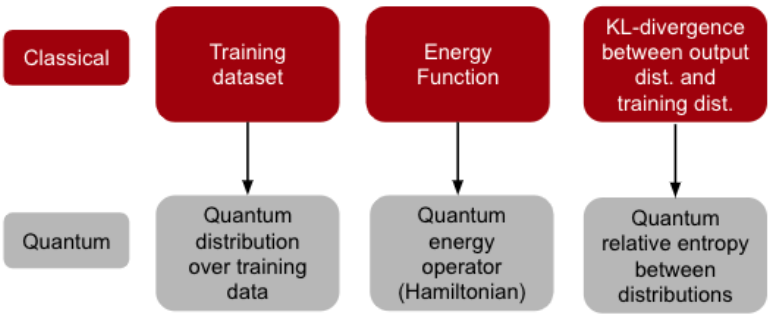

Figure 3: Quantizing the RBM. The analogy from classical RBM to quantum RBM

The energy of the system, i.e. the Hamiltonian function is defined as a linear matrix representation in the quantum case. It's represented via the Ising model to be a symmetric Hamiltonian diagonal in the standard basis given by equation 5:

$$\hat{H} \equiv -\sum_{j,k \in u} J_{jk} \hat{Z}_j \hat{Z}_k - \sum_{j \in u} B_j \hat{Z}_j \tag{5}$$

where u is the set of indices for the nodes in the neutal network graph G, and $\hat{Z}$ is the Pauli-Z operator. Formally, the aim of QRBM is to reduce the thermal state on the visible nodes $\rho|_v = tr_h\left(e^{-\beta H}\right)/tr\left(e^{-\beta H}\right)$ to approximate the normalized sum over all the data $\rho_{data} = |D|^{-1} \sum_{d_j \in D} |d_j\rangle \langle d_j|$ (2). Providing the weights and biases of the network, sampling the Gibbs states of the Ising Hamiltonians for these parameters compares the statistics of the reduced state on the visible unites to the ground truth data through relative entropy given by equation 6. It suggests the weight updates to reduce the relative entropy, which indicates training of the network.

$$S\left(\rho \mid Tr_h\left(e^{-H}/Z\right)\right) = Tr\left(\rho \log \rho - \rho \log\left(Tr_h\left(e^{-H}/Z\right)\right)\right) \tag{6}$$

4.2 Implementation

4.2.1 Modeling Thermalization with Annealing

Quantum annealing allows sampling to be performed in analog at the cost of requiring a pre-programmed Hamiltonian function to model cost and allowing it evolve in accordance with the constraints of a Boltzmann machine (2).

While the use of quantum annealing plays an important role in the history

of quantum machine learning and the importance of the Boltzmann machine itself, this paper is principally concerned with circuit-based implementations of the quantum Boltzmann machine.

4.2.2 Simulating Thermalization with QAOA

QAOA allows the analog processes which model thermalization to be simulated in a circuit-based quantum algorithm. This uses stochastic techniques to simulate the same system evolution which is replicated in analog by annealing-based approaches to constructing a quantum Boltzmann machine (2).

It is notable that this method is heuristic and does not produce exact representations of the thermal state of quantum Boltzmann machine. The fidelity of the heuristic is sufficient for modeling the adiabatic evolution sufficiently for use in a functioning Boltzmann machine (2).

4.2.3 Overall Workflow

The main goal of QAOA is to optimize a set of parameters, which are denoted as $< \gamma, \beta >$. The quantum circuit evaluates the cost function to produce a cost value. The cost function is parameterized by $< \gamma, \beta >$. It combines the cost and partial Hamiltonians (2) defined for the RBM problem. The cost and partial Hamiltonians evolve alternatively by P pulses. The cost value is also known as an expectation value, represented by $\langle \psi | Cost | \psi \rangle$, which is sometimes called the wave function in literature. Here our cost function is the overall energy of the system which is encoded by the Hamiltonians. The classical Optimizer takes the expectation value as input, and updates the parameters for the next cycle of evaluation-update. Figure 4 depicts the workflow of the optimization process.

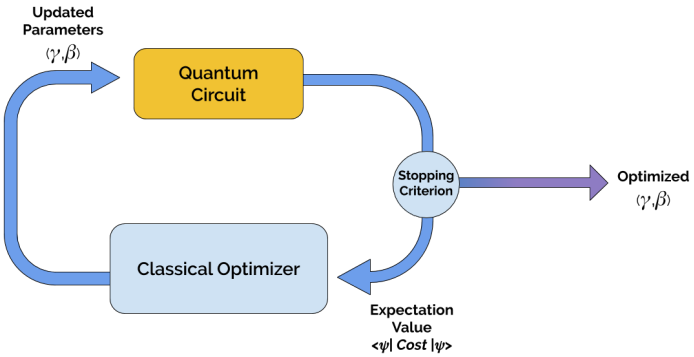

Figure 4: Workflow of QRBM implementation (Image from (16)).

5 Advantages and Disadvantages of Quantum Boltzmann Machines

The quantum Boltzmann machine offers a theoretical computational advantage over the classical Boltzmann machine and may even offer a basis for new machine learning techniques and applications (1, 5).

5.1 Advantages

Quantum Boltzmann machines offer a theoretical quadratic speedup over a classical implementation with respect to the rate at which a given network achieves thermalization. The training itself can also be quadratically improved through the sampling mechanism. The randomized, probabilistic nature of the classical approach to sampling means that many iterations are required to train. The quantum approach allows fewer samples to achieve comparable performance and may also allow for fewer queries to training data (1).

The quantum Boltzmann machine may also provide a foundation for novel machine learning techniques which are fundamentally quantum in nature. Modification of the underlying structures with quantum operations impart unique properties such as entanglement, which could find application to machine learning or other problems (1).

An intermediate output of a quantum Boltzmann machine is itself a quantum state before being evaluated. This means that quantum Boltzmann machines can represent other quantum systems in a way that classical implementations are not able to, and that these states can be used in subsequent quantum computations. This may offer practical advantages in future quantum applications of which a quantum Boltzmann machine or derived quantum machine learning model might be but one component (1).

5.2 Disadvantages

While theory supporting a quantum advantage is rigorous, skepticism remains with respect to the practicality of using quantum machine learning models to solve real-world problems (8).

The promise of fundamental quantum structures and learning techniques is enticing, but a survey of contemporary literature does not yield any concrete examples.

6 Experiments

6.1 Coin Flip Experiment

The aim of the coin flip experiment is to analyze if the hidden states in the QRBM are able to learn the representation of the training data. We generate a sequence of random bits (0, 1) by flipping an unbiased coin. For each bit, we encode the bit into a 4-bit representation (0 is encoded as [1, 1, -1, -1] and 1 is encoded as [-1, -1, 1, 1]). As this is a coin flip experiment, only 1 hidden state should be able to learn the representation of the data. Therefore, in order to evaluate the QRBM model, we train the model with 1 hidden state using the encoded bits as the training data. Figure 5 demonstrates the behaviour of the model after 8 epochs. We can notice that the QRBM model is able to distinguish the two states of the coin flip. After 20 epochs, the model is able to predict the original coin value 1 with probability 1. The code for the experiment can be found here.

RBM Pr.	Original Coin Value
0.599	1.000
0.599	1.000
0.401	0.000
0.599	1.000
0.401	0.000
0.401	0.000
0.401	0.000
0.599	1.000
0.599	1.000
0.599	1.000
0.599	1.000
0.599	1.000
0.401	0.000
0.401	0.000
0.599	1.000
0.401	0.000
0.401	0.000
0.401	0.000
0.401	0.000
0.401	0.000

Figure 5: Coin flip experiment results after 8 epochs. RBM Pr. represents the probability of measuring the original coin value 1 in the training data.

6.2 Comparison of generative models

In the following experiment, we compare the performance of various generative models (10) on the Language Modeling task (11). In the Language Modeling

task, we are given a bitstring, and we predict the next bit, given the observed bits. The generative models compared for the task are: Long Short Term Memory Networks (LSTM) (12), Generative Adversarial Networks (GANs) (13), Deep Boltzmann Machines (14), and Autoregressive GANs (15). The experiments demonstrate that LSTM shows the best performance among all of the generative models.

7 Analysis

We evaluate the QRBM model on three parameters:

1. **Noisy Quantum Computers**: This is one of the main issues that the quantum community is dealing with at the moment. It is crucial to understand the impact of noise on the performance of the models.
2. **Number of Hidden States**: With the success of Deep Learning, we have noticed the importance of hidden states in Classical Computers. We are curious to find out the behaviour of hidden states in quantum computers.
3. **Time complexity**: Theoretically, QRBM should provide faster computations as compared to classical RBMs. We want to find out if that really happens.

7.1 Gate Noise Model

We analysed the behaviour of QRBM by instantiating the Quantum Virtual Machine (QVM) with Pauli gate noise probabilities $[P_x, P_y, P_z]$. Pauli-X, Pauli-Y, Pauli-Z are applied to each qubit after every gate application with respective gate noise probabilities. We compare the performance of various noisy gate models on the coin-flip experiment: how well can the models predict the state of the coin with more epochs. Figure 6 demonstrates the impact of gate noise with the increasing number of epochs. As the number of epochs increase, the circuit starts to become different from the original circuit with the introduction of Pauli-X, Pauli-Y, Pauli-Z gates. Due to this, the performance of the noisy circuits decrease as compared to the noiseless circuit. This shows the importance of noiseless gates in the quantum computers and how only running the model for 10 epochs can have a large impact on the performance. The code for the analysis can be found here.

7.2 Measurement Noise Model

We further analyse the behaviour of QRBM by instantiating the QVM with Pauli measurement noise probabilities $[P_x, P_y, P_z]$. Pauli-X, Pauli-Y, Pauli-Z are applied to each qubit being measured it is being measured with respective measurement noise probabilities. Figure 7 shows the results of the evaluation

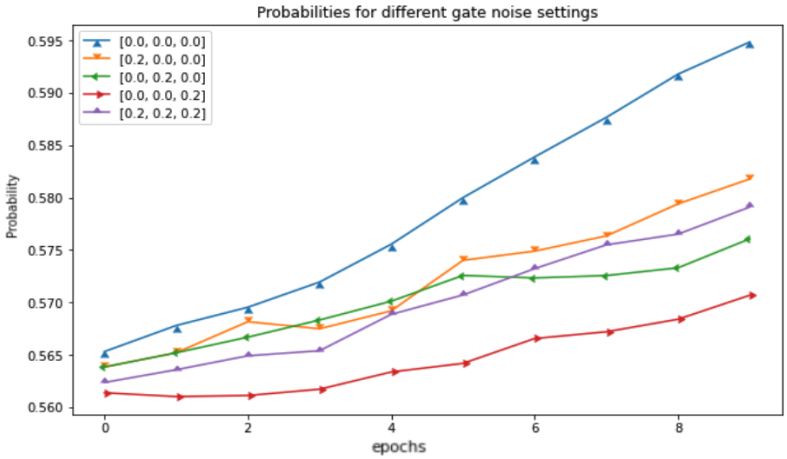

Figure 6: Impact of gate noise with increasing epochs. The difference between the noiseless model and noisy models increases with the number of epochs.

of the QRBM models on the coin flip experiment. In contrast to the gate noise models, we notice that the first few epochs do not have any impact on the measurement noise. Although, as the number of epochs increase, the impact of the noise starts to increase as well. Interestingly, we also notice that in both of the noise models, the QVM with only Pauli-Z has the worst performance. The code for the analysis can be found here.

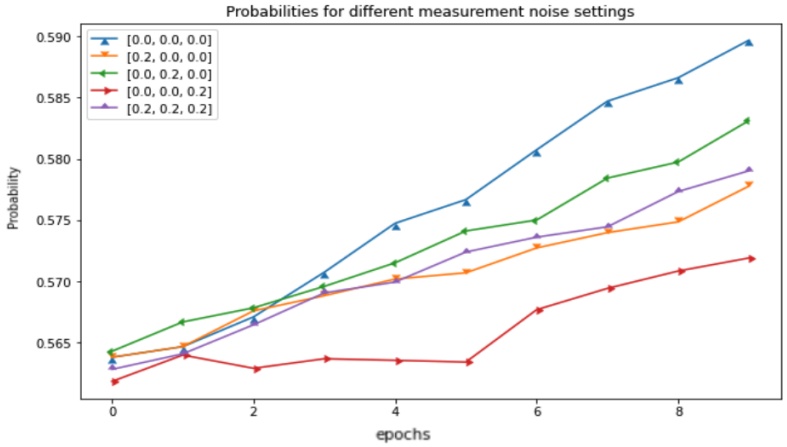

Figure 7: Impact of gate noise with increasing epochs. The difference between the noiseless model and noisy models increases with the number of epochs.

7.3 Impact of hidden states

To analyse the impact of hidden states, we trained both classical and quantum RBMs. We generate a sequence of 8 randomly sampled numbers (0, 8) and encode them in 4-bit representations. To evaluate the performance of the RBMs, we use a discriminative model, Support Vector Machine (SVM) (17) to classify the 4-bit representations using the hidden states learned by the RBMs. We notice that in the case of classical computers, the performance of the SVM increases with the number of hidden states in the RBMs. However, in the case of QRBMs, this behaviour is not reflected. Figure 8 shows the results for both of the cases. The code for the analysis can be found here.

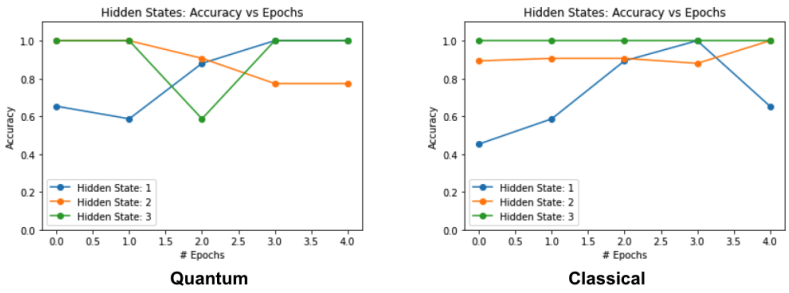

Figure 8: Comparison of number of hidden states in QRBMs and classical RBMs.

To learn more about why this is happening, we found an interesting behaviour happening in QRBMs (18). In classical deep learning, all the information required to make a decision is local, i.e it is stored in the hidden states. However, in quantum computers, we have entanglement, therefore information is no longer localised, instead it's stored in the correlation between the bits. In QRBMs, performing the partial trace while calculating the quantum relative entropy (equation 6) throws away that correlation that holds the entirety of the data.

7.4 Time complexity Analysis

We analyse the time taken to train a QRBM with the increasing number of hidden states, and compare that with the classical RBM time complexity. As demonstrated in figure 9, the time complexity to train a QRBM increases exponentially as compared to classical RBMs. It took us 12 hours to train a QRBM with 6, 7 and 8 states. It is important to mention that these results are calculated on quantum virtual machines, and therefore, they might be exactly representative of the real quantum computers scenario. The code for the analysis can be found here.

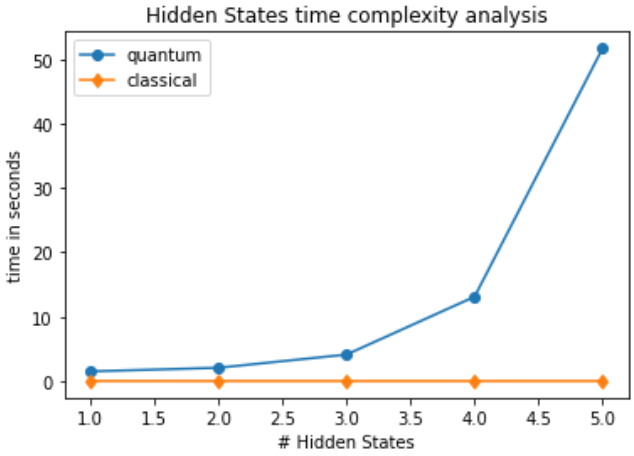

Figure 9: Analysis of time complexity of QRBMs and classical RBMs.

7.5 Other attempts

1. As Rigetti does not provide open source real quantum computers for experimentation anymore, all of the above analysis and experiments use Rigetti QVMs.

2. *pyquil* does not support CNOT gates while adding decoherence noise (19). Therefore, we analysis of decoherence noise on QRBM was not feasible.

3. With the current state of the quantum computers, we are unable to run medium-sized experiments with QRBMs. We tried to train an MNIST (20) model with image size 28x28, but the model failed to compile.

8 References

1. https://arxiv.org/pdf/1611.09347.pdf

2. https://arxiv.org/pdf/1712.05304.pdf

3. https://arxiv.org/pdf/1601.02036.pdf

4. https://arxiv.org/pdf/1411.4028.pdf

5. https://journals.aps.org/pra/abstract/10.1103/PhysRevA.101.032304

6. https://www.cs.toronto.edu/ hinton/csc321/readings/boltz321.pdf

7. https://web.archive.org/web/20110718022336/http://learning.cs.toronto.edu/ hinton/absps/cogscibm.pdf

8. https://www.protocol.com/manuals/quantum-computing/machine-learning-ai-quantum-computing-move-beyond-hype

9. https://commons.wikimedia.org/w/index.php?curid=22915782
10. https://arxiv.org/pdf/2010.11983.pdf
11. https://en.wikipedia.org/wiki/Language_model
12. https://www.bioinf.jku.at/publications/older/2604.pdf
13. https://arxiv.org/abs/1406.2661
14. http://www.cs.toronto.edu/ fritz/absps/dbm.pdf
15. https://openreview.net/forum?id=rJWrK9lAb
16. https://docs.entropicalabs.io/qaoa/notebooks/6_clusteringwithqaoa
17. http://web.cs.iastate.edu/ honavar/hearst-svm.pdf
18. https://www.youtube.com/watch?v=B1NU9DBh2t8
19. https://pyquil-docs.rigetti.com/en/latest/noise.html
20. https://en.wikipedia.org/wiki/MNIST_database