Quantum Machine Learning using Quantum Simulation

Glen S. Uehara
School of Electrical, Computer,and Energy Engineering
Arizona State University
Tempe, AZ U.S.A.

***Abstract*— It is generally recognized that quantum inspired algorithms achieve exponential speed over classical algorithms. This advantage is seen when these algorithms run on quantum systems. The near-term quantum processors are still unpredictable and in its infancy. They are cost-prohibitive, and reliability is an issue. In the short term, to gain insight in the effectiveness of these algorithms, quantum system simulation is available to model and develop these algorithms. The advantage of this is that the model can be simulated on a classical computing system or directed to run on a quantum system. The simulation allows for individuals to learn about the quantum algorithms and its approach. This paper looks at using the quantum algorithm approach on pattern recognition problems. The goal is to look at the built-in algorithms and model an equivalent quantum-circuit using the simulation tools.**

I. Introduction

In recent years, more research and modeling are performed to gain deeper understanding of quantum algorithms and quantum computing. Quantum inspired algorithms comes in many forms, a pure quantum algorithm that solves quantum mechanics problems or modifying classical algorithms to run on quantum systems. Taking advantage of classical algorithms is one way to demonstrate the strength of quantum systems. One aspect of this is to demonstrate the potential of quantum machine learning (QML) algorithms [4]. Machine learning (ML) algorithms have shown the potential of improvements by quantum inspired algorithms and quantum computing. Machine Learning is an established field that mathematically extract generalizable information from data. These algorithms are shown to be transformable between the two types of systems. This method of transformation is one way to show the strength of new quantum and quantum-hybrid algorithms.

The quantum computing community has built a system to help develop and model quantum algorithms. These are open-source quantum simulation tools [4]. The advantage of using these systems are that they can run on a classical computer system. When designed and developed correctly, these algorithms can then run on a quantum computer. This hybrid development allows the designer to optimize the algorithm on classical systems and demonstrate the capability on real quantum processing units (QPUs).

The designer starts the development by design new quantum circuits to model a system. The new circuit is run by using algorithms based on quantum bits (qubits). This research looks into applying basic qubits and quantum circuits to build a system to solve the classifier problem. This research is based on building a hybrid quantum-classical system which applies only portions of the classical algorithm and applies quantum algorithms. In this paper, we also look at understanding a quantum system and go through the steps to build the hybrid quantum-classical system. We begin by using the built in QSVM in Qiskit [4], determine how to build a quantum circuits by designing and developing a hybrid system based on [1] and finally implementing a hybrid quantum-classical system based on the research in [5]. The final research [5], takes the classical gradient descent algorithm and builds a quantum circuit to represent the algorithm.

II. Background

There are several QML algorithm development techniques and methodologies. We will look at the Kernel methods for machine learning in pattern recognition, with support vector machines (SVMs) and neural networks. As found in [3], SVMs are mathematically similar to what goes on inside a quantum computer. This allows for using the SVM algorithms and applying them on a QPU.

To build the pattern recognition system, we start with a supervised learning approach. This is learning a function that maps an input to an output based on example input-output pairs. We start with labeled training data consisting of a training example. This algorithm is supervised in its learning, allowing an input-output pair of objects. These are typically in the form of vectors. This is referred to as the supervisory signal.

In this sense, supervised learning can be categorized as classification and regression when data-mining the input. Classification attempts to assign test data into specific categories. This allows for data to be recognized within the dataset. The algorithm then attempts to draw conclusions and places a “label”. Regression is used to understand the relationship between dependent and independent variables. This research looks at applying quantum machine learning on classification of data.

In this paper, we look at a simple classifier problem and investigate the different quantum machine learning techniques used in various research papers. The goal is to replicate and find an effective means to implement these algorithms on quantum simulators.

We begin by looking at learning about qubits and quantum entanglement. A classical bit can have a value of two states 0 or 1. This can be represented with a transistor switch set to "off" or "on". Another way to see this is an arrow being "up" or "down". When looking at a qubit, we see this having more possibilities than the two states. The state is represented by arrow point to a location on a sphere (Fig. 1). The north pole is equivalent to 0 and the south pole is equivanet to a 1. Another position in the sphere is a superposition of 0 and 1.

For this research, we will simplify the quantum mechanics of qubits. In the case of quantm computers, a simplified vision is that entanglement is used as a computational multiplier for qubits. As more qubits are entaganled together, ability of the system to make calculations grows. This is not in a linear fashion, but exponentially.

To begin the translation from classical computing to quantum computing, some basic concept of quantum systems is needed. For simplicity, we will stay with the mathematical concept of quantum circuit and qubits. These are the base components for our quantum system.

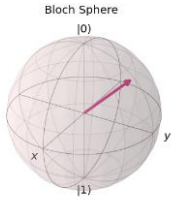

Fig. 1. Bloch Sphere represetnation of qubit. Generated using Qiskit.

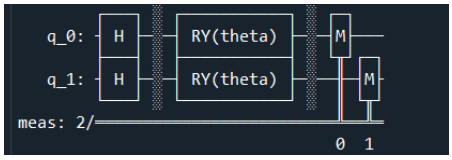

Fig. 2. Two-qubit quantum circuit. Generated using Qiskit, a simple H-gate and R_ϕ. M in the diagram represents where the measurement on the quantum gate takes place.

The quantum circuit is the model for quantum computation. Quantum (logic) gates are the main building blocks for the quantum circuit (Fig. 2). The computation of the quantum circuit is a sequence of quantum gates that uses qubits for its register. Looking at this is like an n-bit register, where the structures instead uses n-qubits. In classical computer, the *NOT* gate, are not reversible and an *AND* gate cannot always recover the two input bits from the output bit. Looking at this mathematically, the qubits can be reversible by unitary transformation under the quantum gate. For a single qubit, unitary transformations correspond to rotations of the qubit vector on the Bloch sphere (Fig. 1) to specific superpositions.

The vector representation of qubits is

$$|a\rangle = v_0|0\rangle + v_1|1\rangle \rightarrow \begin{bmatrix} v_0 \\ v_1 \end{bmatrix}$$

where v_0 and v_1 are the complex probability amplitudes of the qubit. The values determine the probability of measuring a 0 or a 1, when measuring the state of the qubit. This is typically written as

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

We also define tensor product, denoted as $\otimes$, to combine the quantum states. The combined state of two qubits is the tensor product of the two qubits.

$$|\text{ab}\rangle = |a\rangle \otimes |b\rangle$$

$$= v_{00}|00\rangle + v_{01}|01\rangle + v_{10}|10\rangle + v_{11}|11\rangle \rightarrow \begin{bmatrix} v_{00} \\ v_{01} \\ v_{10} \\ v_{11} \end{bmatrix}$$

Finally, gate on a specific quantum state is found by multiplying the vector $|\psi_1\rangle$ and the matrix $\mathfrak{U}$. The matrix $\mathfrak{U}$ represents the gate and vector $|\psi_1\rangle$ represents the state. This results in a new quantum state $|\psi_2\rangle$.

$$\mathfrak{U}|\psi_1\rangle = |\psi_2\rangle$$

These are the very base components in understanding how to build a quantum system. The quantum gates represent the unitary matrices. We can use different gates to manipulate the qubits in order to build the quantum circuit.

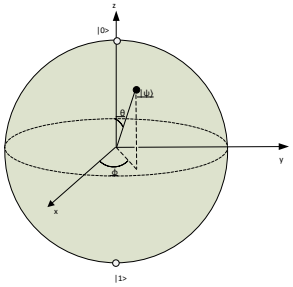

Fig. 3. Bloch sphere representation. We see the represetnation of the vector, θ, ϕ on the sphere.

Using these representations, we can express the quantum representation of qubits. Using the Bloch sphere (Fig. 3), a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)e^{i\phi}|1\rangle$ can be represented as a unit vector from the origin to the point on the unit sphere with spherical coordinates (θ, ϕ). From above, we now see that a single-qubit quantum gate $\mathfrak{U}$ operating on $|\psi_1\rangle$ produces a rotated qubit $\mathfrak{U}|\psi_1\rangle = |\psi_2\rangle$. Some of the basic gates operating on the qubits are represented by H, X, Y, Z, S and T.

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\cos(\theta/2)\\ \sin(\theta/2)\, e^{i\,\phi}\end{pmatrix}$$

Hadamard gate: H = $\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$

Pauli X gate: X = $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$

Pauli Y gate: Y = $\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$

Pauli Z gate: Z = $\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$

$\pi/8$ gate: T = $\begin{pmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{pmatrix}$

R_ϕ gate: $R_\phi = \begin{pmatrix}1 & 0\\ 0 & e^{\mathrm{i}\phi}\end{pmatrix}$

U gate: $U_3(\theta,\phi,\lambda) = \begin{pmatrix}\cos(\frac{\theta}{2}) & e^{-\mathrm{i}\lambda}\sin(\frac{\theta}{2})\\ e^{\mathrm{i}\phi}\sin(\frac{\theta}{2}) & e^{\mathrm{i}\lambda+\mathrm{i}\phi}\cos(\frac{\theta}{2})\end{pmatrix}$

The representation on qubit by quantum gate can be defined as a matrix. This may be represented by starting with a 2x2 unitary matrix that is acting on the qubit.

The quantum gates affect the qubit by rotating the position or changing the amplitude of the vector. An example of this is seen in Fig. 4. This is a derivation of a Pauli X gate using Pauli Z gate and Hadamard gate.

$$\mathrm{HZH} = X$$

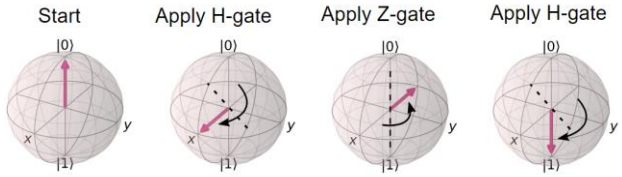

Fig. 4. Qubits affected by quantum gates. The qubit transformation to an X-gate by applying the H-gate, Z-gate and H-gate

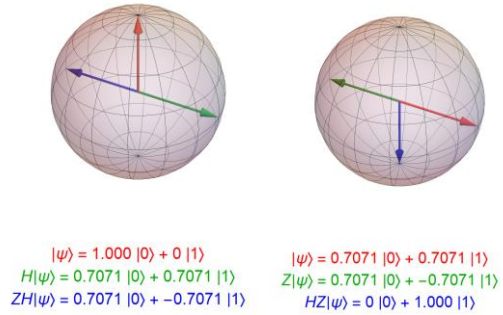

Fig. 5. Qubits value representation on the Bloch Sphere. The digram shows the affect of the H and Z gate on the qubit. The effect is shows as the rotation of the vector and its ampitude.

In Fig. 5, we see the mathematical representation of the quantum bit affected in rotation and amplitude change by the different quantum gates.

This background provides the basic understanding of how to build a quantum system. The mathematical represenation is a simplified and generalized way to look at the quantum system. When building a hybrdi quantum-classical system, the end goal is to use to apply various quantum techniques and tie in with current classical systems.

III. Literature Research

To begin our research, we first look at building the knowledge of building a quantum circuit for machine learning applications. This research uses Qiskit [4] as the quantum programming toolkit. This toolkit provides the base quantum toolbox and allows developing quantum circuits at a higher order language. To being, we start with a Quantum SVM (QSVM), based on [3] by using the built-in tools. In this case a system is defined aa feature map on n-qubits generated by the unitary $\mathfrak{U}_{\phi\,(x)} = U_{\phi\,(x)}H^{\otimes n}U_{\phi\,(x)}H^{\otimes n}$ where,

$$\mathfrak{U}_{\phi\,(x)} = \exp\left[i\sum_{S\subseteq[n]}\phi_S(x)\prod_{i\in S}Z_i\right]$$

This is a diagonal gate in the Pauli-Z basis. This circuit acts on the initial state $|0\rangle$. The coefficients for this are $\phi_S(x) \in \mathbb{R}$ to encode the data x $\in\Omega$. This research claims any unitary gate may be used if it can be implemented efficiently. This feature map used the Hadamard gate and Unitary gates to build the quantum circuit. The generalized model for the mapping is seen in Fig. 6.

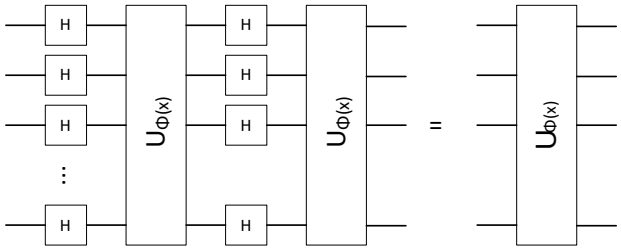

Fig. 6. For the general circuit $U_{\Phi(x)}$ is formed by products of single- and two-qubit unitaries that are diagonal in the computational basis. The circuit family depends non-linearly on the data through the coefficients $\Phi(x)$ with $|S| \leq 2$

The research experiment designs a quantum circuit (Fig. 7) that shows a larger 5 qubit system that adds 3 sets of Hadamard and Unitary combination. This set is used to estimate the fidelity between a pair of feature vectors. The circuit is comprised of the Hadamard gates interleaved with the Unitary gates that are parameterized by $\Phi(x)$.

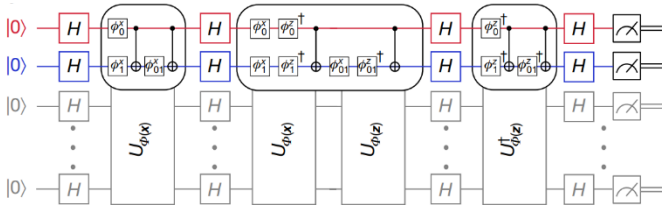

Fig. 7. Quantum ciruit based on [3]. This figure was taken from [3] in order to show the experimental quantum circuit used in the research paper

We build a QSVM circuit and verify the feature map according to the [3]. The purpose of this research is to help verify the quantum simulator that is used in this paper.

The next research [1] looks at a hybrid quantum-classical system. This takes a classical system and applies quantum algorithms to the model. In this research, the quantum system is used in the hidden layer of the neural network.

In this research, we look at a quantum neuron model where the weight is represented with qubits

$$|\phi_i\rangle = \alpha_i|0\rangle + \beta_i|1\rangle, i = 1,2,..,n$$

Here, α_i and β_i are complex numbers, where the absolute square represents the probability of $|0\rangle$ or $|1\rangle$. The normalization conditions for α_i and β_i are represented as:

$$|\alpha_i|^2 + |\beta_i|^2 = 1, i = 1,2,\dots n$$

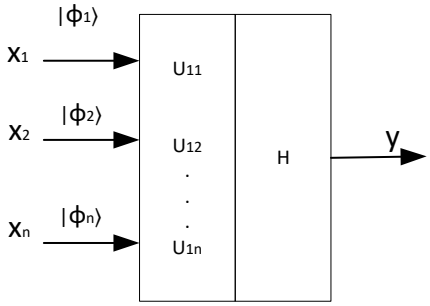

Fig. 8. Quantum neuron concept figure as proposed in [1]. The quantum neuron proposed in [1], the weight is represented by qubits, and the transform function is represented by inner-product operator.

We can start with a quantum neuron representation with Fig. 8. We can realize that

$$\mathrm{XW} = \sum_{i=1}^{n} x_i w_i$$

where we have that $X = (x_1, x_2, \dots, x_n)^T$ and $W = (w_1, w_2, \dots, w_n)^T$. We have w_i as m dimension real vector. We can now express y as

$$y = H(\mathrm{XW}) = C \cdot (\mathrm{XW}) = C \cdot \sum_{i=1}^{n} x_i|\phi_i\rangle$$

where H is the inner product operator $H(X) = H \cdot X$ and $C = (1,1)^T$. This operation transforms the input of the quantum neuron to a real number. U_i is the quantum rotation gate to modify the phase of the $|\phi_i\rangle$.

Using the quantum neuron, we now look at the Quantum Neural Network (QNN) and its definition. The QNN structure is similar to the Artificial Neural Network (ANN) with the hidden layer. A QNN with only quantum neurons is defined as normalization QNN. The QNN including both quantum neurons and general neurons is defined as hybrid QNN. The QNN transform function adopts a linear operator, therefore the non-linear mapping capability falls under restriction. In the research, [1], the hybrid QNN consideration includes the hidden layer of quantum neurons. This is the advantage of quantum computing and the nonlinear mapping capability. We can represent this as Fig. 9.

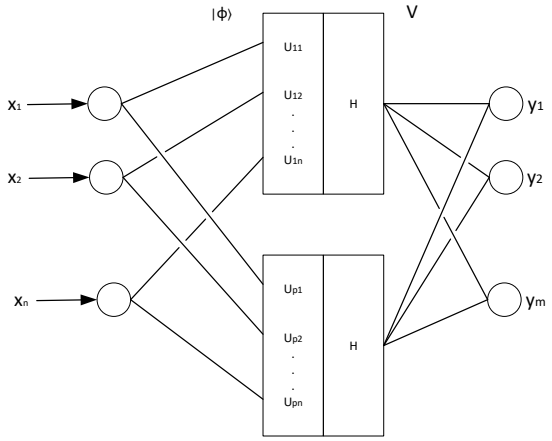

Fig. 9. Hybrid quantum-classical system based on [1]

The figure above represents a hybrid quantum-classical neural network system that is based on the three-layer structure. The model shows the input layer and the output layer contains n and m traditional neurons. The hidden layer contains p quantum neurons. This relationship can be described as

$$y_i = g(\sum_{j=1}^{p} v_{\mathrm{ij}}(C \cdot \sum_{k=1}^{n} (x_k|\phi_{\mathrm{jk}}\rangle)))$$

$$i = 1,2,\dots m: j = 1,2,\dots p: k = 1,2,\dots n$$

Also v_{ij} is the linked weight between the ith neuron in the output layer and the jth neuron in the hidden layer. We also have g in this case as a Sigmoid function or Gauss function.

This research is used to our understanding on how to build a quantum circuit. The model that is built is used to verify the quantum circuit and the quantum simulator.

Finally, research [5] looks at building a quantum neural network. This research develops a quantum circuit for a classifier of various dataset. Taking the classifier problem that was used in [3] with QSVM, the goal is to use hybrid techniques in [1] to implement the new model based on the research [5]. The system is built based hybrid quantum-classical gradient descent training algorithm. It uses the bias and gradient in neural network in an adaptive manner to strengthen the base during training. To verify the model, we look at implementing the quantum circuit and verify using the Iris dataset.

To begin, we break the system down into four steps.

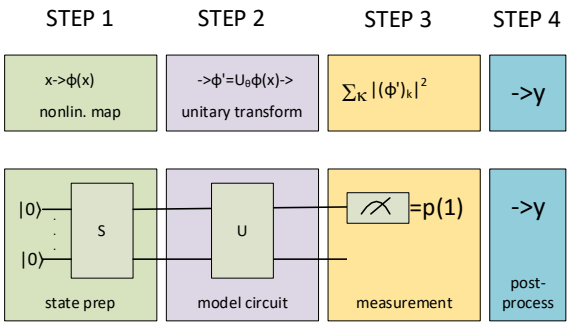

Fig. 10. Circuit-centric quantum classifier with four steps. The cirucit applies unitary transform to the feature vector as one linear layer of the neural network..

The State Preparation is used to apply various strategies to encode the input vectors into n-qubits. The goal in the research paper [5] is to use amplitude encoding to on the input vector. We see that the map of the input into higher dimensional space can apply a tensorial feature map by preparing d copies of the state. This can be described as

$$|\psi\rangle \rightarrow |\psi\rangle \underset{d\text{times}}{\otimes \ldots \otimes} |\psi\rangle$$

As an example, amplitude encoding with $N = 2$ and $d = 2$ without any preprocessing can be mapped with feature vector $(x_1, x_2)^T$. Where N is used to describe a binary classification task on a N-dimensional real input.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \otimes \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} {x_1}^2 \\ x_1 x_2 \\ x_2 x_1 \\ {x_2}^2 \end{bmatrix}$$

The Model Circuit maps the vector to another vector $\varphi' = U_\theta \varphi(x)$ by a unitary operation U_θ. In this, the unitary U can be decomposed into

$$U = U_L \ldots U_\ell \ldots U_1$$

where each U_ℓ is a single qubit or two-qubit quantum gate. The research in [5] also further restricts and simplifies the gate by the "elementary parametrized gate set". This 2-qubit unitary gate is called imprimitive if it maps a 2-qubit product state into a non-product state. The paper further creates a new qubit gate G as a 2x2 unitary gate. This advantage is to using angles instead of parameterization with Pauli matrix. This technique is involved and a simplified method was used for this paper.

The measurement and post-processing steps are ways to look and inspect the quantum bit and transforms. This is the method to observe and determine the statistics of the model's run.

IV. Modeling and Analysis

In order to have a foundation for the research, a basic understanding of the open-source quantum simulation is needed. To properly develop and understand the strength and weaknesses of QML algorithms, the three approaches are used to design, model and weigh the algorithms.

The first model uses the QSVM (Fig. 11) that is built in the open-source Qiskit simulation. Taking Fig. 6 as the base implementation, we used the built in QSVM and Feature mapping capability build the following quantum circuit representation of the feature map.

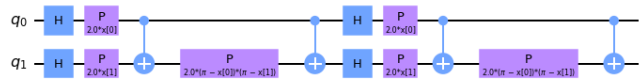

Fig. 11. Model generated from Qiskit. The built in QSVM quantum circuit, when running iris-set. H is defined as the Hadamard gate, P are the representation of the Unitary gate.

When looking at this as a simple circuit with two qubits, we see the circuit represented with several unitary gates. The representation of the QSVM from Qiskit (Fig. 9). In this case the P gate is representing the U-gate for the system. Using the system method described in [3], we now run a test to verify the results.

The dataset that is used for testing is the Iris dataset. This is used as the main baseline to validate the models. The system is run using the feature map in Fig. 11 and the results for the classification can be seen in Fig. 13.

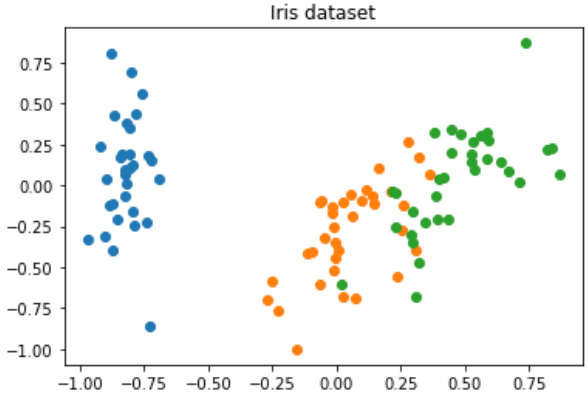

Fig. 12. Iris data set taken from sklearn. The experiment used only 2-dimentions when run. Training size of 100 and Test size of 50 was used.

With the simple model, the success rate is at 70%. Given classical systems, there are SVM models that achieve higher accuracies.

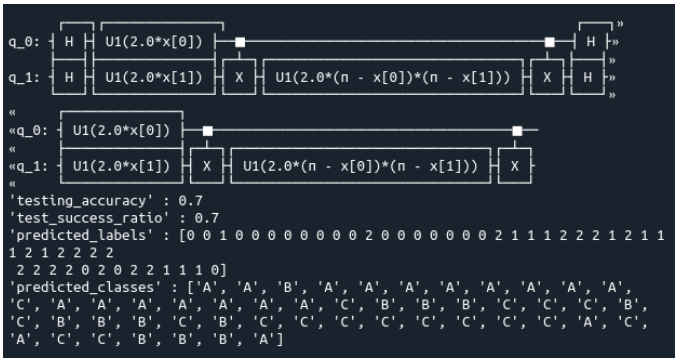

Fig. 13. Output generated from Qiskit. Current result from running QSVM built in with Iris dataset and its prediction result at 70%

Unlike the paper, the Qiskit simulation was built with only 2 qubits and two Hadamard/Unitary sets. This missing pair may have caused a loss in fidelity with trying to classify the system. Unlike manually building the circuit, the intent

of this experiement was to determine the strenght of the built in qiskit toolkits. This shows that the toolkit may need to be extended to allow for a more complex circuit.

Taking the next step in quantum cirucuit design is to build one that is based on an hybrid quantum-classical system. This allows to take the portion of classical system and add quantum circuits in order to expand the algorithm capability. First look at the the hybrid quantum-classical system in [1]. The model is to build a system where the quantum circuit in in the hidden layer. Taking some of the concepts of [1], the first step is to build a simplified quantum circuit. Fig. 14 shows the current quantum circuit that is created for this system.

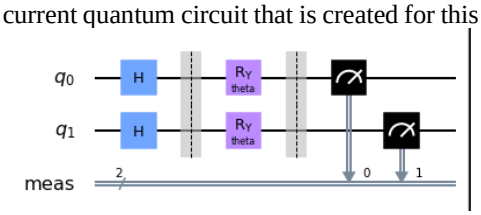

Fig. 14. Model generated from Qiskit. Model is based on a hybrid-classical design described in [1]. The quantum circuit is used for classification of a simplified MNIST Dataset, using only 2 classes (0, 1).

In the hybrid quantum-classical system use a simplified approach from [1] and used two qubit system. This system was built based on training demonstration provided by Qiskit [11]. The dataset was also simplified to a two set MNIST (0 & 1) dataset was used to verify the system. The convergence of the system is shown in Fig. 15.

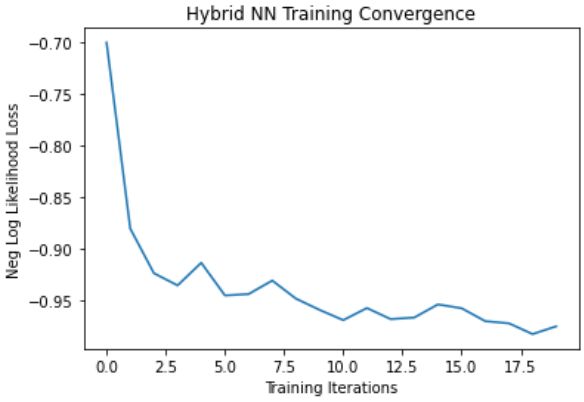

Fig. 15. . Analysis generated from Qiskit. This is the training convergence based on the hybrid quantum-classical model with training iteration.

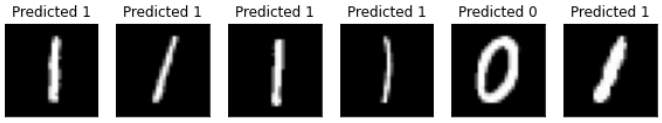

Fig. 16. . Output generated from Qiskit. This is the accuracy at 100%.

We see that the building a hybrid quantum-classical system is possible using the Qiskit simulation system. In order to determine the effectiveness of the approach, we return to build a hybrid system to determine if we can improve upon the built in QSVM classifier.

In order to research the hybrid quantum-classical classification system, we take the two methods that we coded above and apply them per research in [5]. We begin with built-in functionality and algorithms based on research of [10]. This provided a foundation for the software baseline. The input parameters, the iteration and the circuits were modified to run on the classical computer.

We first look at various state preparation of the data. We start with different data-set from [10] with a modified and normalized transform of the Iris data set.

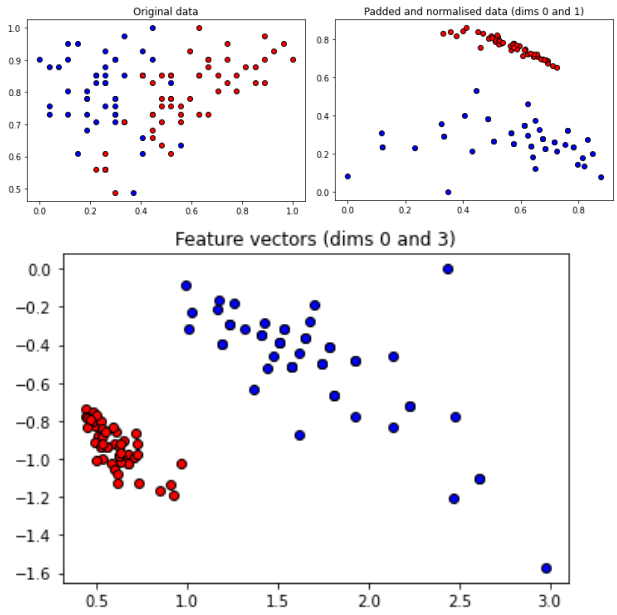

Fig. 17. Iris data set looking at only two classifier data (versicolor, virginica). Using the first 2 features for classification

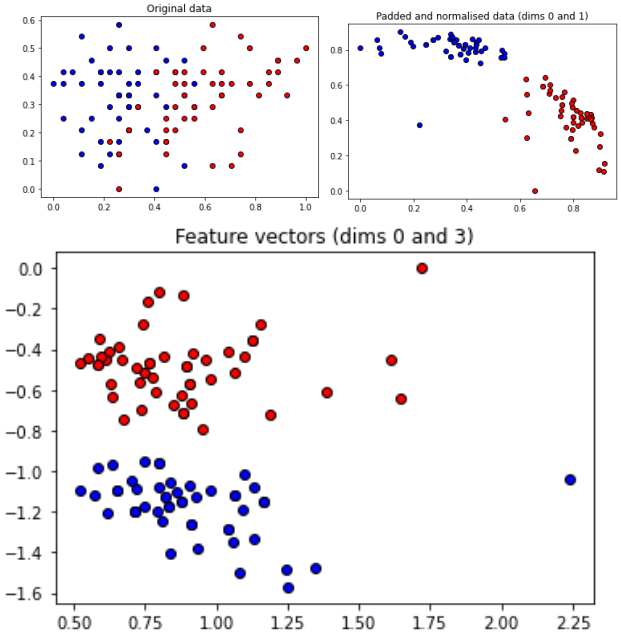

Fig. 18. Iris data set looking at only two classifier data (setosa, versicolor).

We begin by normalizing the data set and padding and transforming the data into feature vectors. This is to help with the state preparation of the data set. The quantum circuit that is used for this system is a 2-qubit system with the ability to look at only two feature sets. This allows for the quantum simulation to run in a more reasonable time. However, this means that 2 simulations were run to characterize the classification

The current quantum-circuit uses the qubits and attempts to run through the Hadamard gate (Fig. 19). The dataset currently used is a modified Iris (class 1 & class 2) dataset (Fig. 21). This allows for a reasonable simulation and computation time on classical computer.

Fig. 19. Model generated from Qiskit. Model is based on a simplified design described in [5]. Quantum circuit designed using only 2-qubits

Using the quantum circuit, we run the gradient descent algorithm to determine the weights needed to optimize the training. We start with a standard least-squares objective to evaluate the cost of a parameter configuration θ and the bias b. Being with a training set $D = \{(x^1, y^1), \ldots, (x^M, y^M)\}$. We can look at the cost as

$$C(\theta, b, D) = \frac{1}{2}\sum_{m=1}^{M} |\pi(x^m; \theta, b) - y^m|^2$$

where π is the continuous output of the model: $\pi(x; \theta, b) = p(q_0 = 1, x, \theta) + b$. We define

$$(q_0 = 1, x, \theta) = \sum_{k=2^{n-1}+1}^{2^n} |(U_\theta \varphi(x)))_k|^2$$

where this is the probability of state 1 after the execution of the quantum circuit $U_\theta \varphi(x)$.

We run through similar gradient descent updates with each step size μ. We can now define as

$$\mu^{(t)} = \mu^{(t-1)} - \eta \frac{\mu C(\theta, b, D)}{\partial \theta}$$

and the bias is

$$b^{(t)} = b^{(t-1)} - \eta \frac{\mu C(\theta, b, D)}{\partial b}$$

The learning rate η may be adapted during the training as needed to decrease the convergence time.

Running the model with the two different training sets, we see the following results

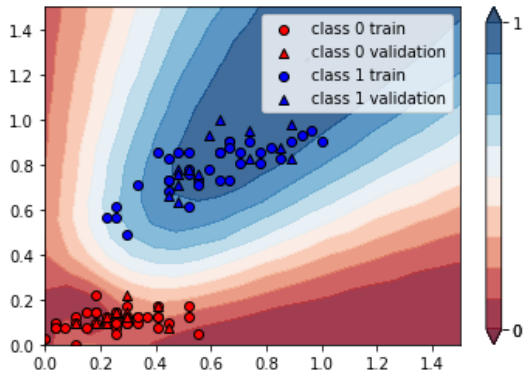

Fig. 20. Descision region plot generated from Qiskit Training and validation used for classification of a simplified Iris dataset, using only 2 classes (versicolor, virginica).

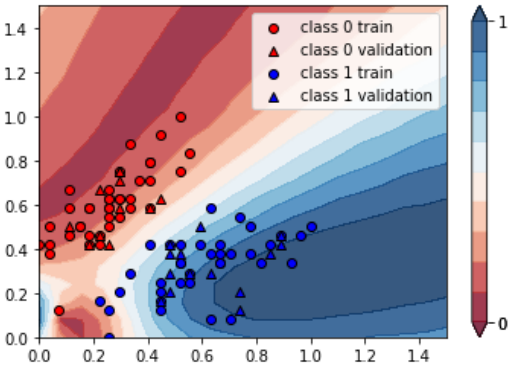

Fig. 21. Descision region plot generated from Qiskit Training and validation used for classification of a simplified Iris dataset, using only 2 classes (setosa, versicolor).

The training results are seen in the following figures

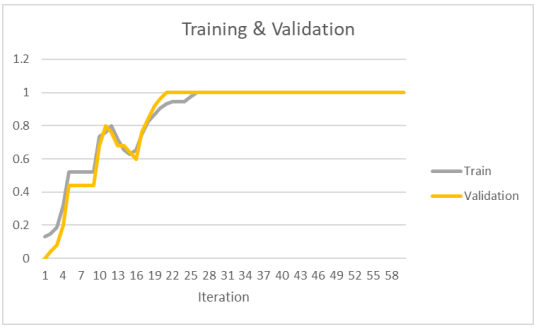

Fig. 22. Training and validation used for classification of a simplified Iris dataset, using only 2 classes (versicolor, virginica).

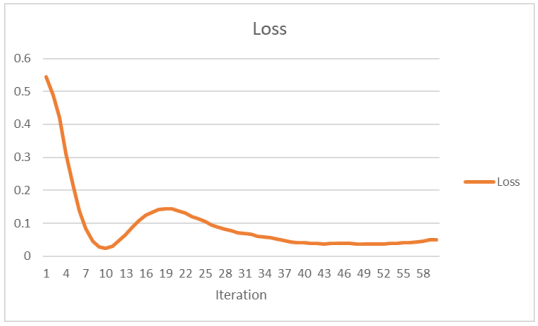

Fig. 23. Loss graph with respect to the itertation for classification of a simplified Iris dataset, using only 2 classes (versicolor, virginica).

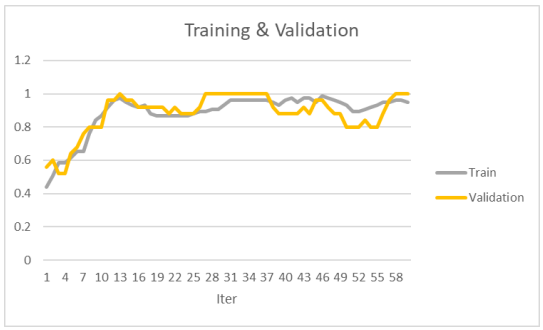

Fig. 24. Training and validation used for classification of a simplified Iris dataset, using only 2 classes (setosa, versicolor).

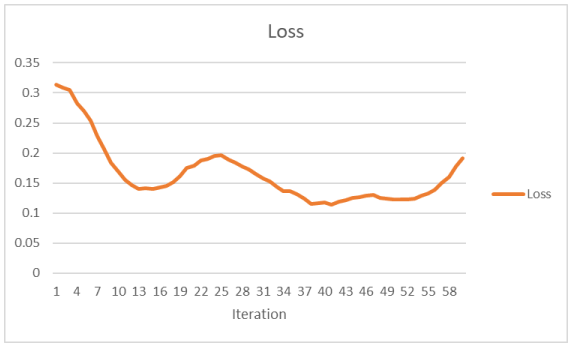

Fig. 25. Loss graph with respect to the itertation for classification of a simplified Iris dataset, using only 2 classes (setosa, versicolor).

The results currently show that simulating the hybrid quantum-classical approach can be run on a classical computer. As we seen in various research, similar Machine Learning algorithm techniques can be used in the Quantum Machine Learning system. The Quantum simulators are sufficient in modeling the Quantum algorithms. The QSVM and the Quantum Neural network approach has both advantages and disadvantages. In the current solution, the QNN used a smaller feature set. This can be improved to build the capability to handle all the features. This will allow this approach to be used in classifier problems. For a larger set of data, QSVM that is built in has its advantages. Though it may not produce the 100% training accuracy, it will provide a quick means to train data and observe the strength of quantum inspired algorithms.

V. Research Conclusion

We have developed a quantum machine learning design that are both Quantum inspired and implementable using quantum simulators. To build the QNN, the building block of this is the unitary model circuit with few trainable parameters that assumes amplitude encoding of the data vectors. This allows the use of systematically entangling properties of quantum circuits. Another key area is the state preparation, though not shown in the results, the preparation of the data took some significant time. After state preparation, the prediction of the model is computed by applying only a small number of one- and two-qubit gates quantum gates. This allows for a simpler testing and use on a quantum simulator.

What we have realized with quantum simulator is the CPU utilization on a classical system. The open-source Qiskit baseline used for the simulation was based on CPU, rather than GPU computation. The runtime on a I7 7th Generation Intel processor for the gradient descent quantum circuit was 10 minutes. The CPU was loaded at nearly 100% during this time. Running a similar gradient descent algorithm on MATLAB with GPU enhancement takes less than 30 seconds for the same type of simulation. Current open-source community are looking at optimizing some of the Qiskit toolkit to run on GPUs. There are examples of toolkits that updated this. An interesting approach may be to look at running and optimizing the simulators to run-on High-Performance Computing (HPC) systems. This would be another area to look into in the future.

This research looked at the different quantum simulation model for pattern recognition. This research is on-going and its goal is to continue to refine the current base model and simulation that were developed during the proposal research. The next steps are to run the different dataset as defined in [6] to verify the results from the research. This involves looking at classical systems and determining the strength and weaknesses of the quantum circuits that were built. Where applicable, a note may be provided to show theoretical concept vs applied circuit to show the possible full potential of the quantum circuit. The final summary the research intends to provide is the generalized approach of a hybrid quantum-circuit. This is to show the strengthen and weakness of the system when simulating on a classical computer.

Current research is running new quantum algorithms on quantum computers. However, the current issue is that the probability and prediction that comes out of one quantum computer does not necessarily match another. The quantum computers are built by hand, by excellent engineers and scientists. Unlike the low probability of errors on classical computers, the quantum computer is still in its infancy and may product unpredictable results. IBM and XANADU are allowing open-source developers to reserve quantum computers. These wait times are queued and sometimes takes 2-4 hours to run a simple algorithm. With the cost being a factor, a lot of initial development are being done using quantum simulators. These allow engineers and scientist to refine their algorithms before reserving a quantum time slice on the quantum computer.

In a final conclusion, we have seen classical personal computers' rapid emergence in the past few decades. It has grown in its capability with adaptation and creative development

in cybersecurity, social media and home streaming. The availability of the classical computer realizes that the next generation of innovators can look into up and coming technologies such as machine learning, computer vision and embedded processing, such as Internet of Things. It is safe to assume the next step in innovation is rolling up of quantum algorithms such as Quantum Machine Learning. As we seen there are still challenges to overcome but having the capability to simulate on classical computers will overcome the current availability issue and help leap the innovation forward.

REFERENCES

[1] H. Xiao and M. Cao, "Hybrid Quantum Neural Networks Model Algorithm and Simulation," 2009 Fifth International Conference on Natural Computation, Tianjin, 2009, pp. 164-168, doi: 10.1109/ICNC.2009.128.

[2] O. P. Patel and A. Tiwari, "Quantum Inspired Binary Neural Network Algorithm," 2014 International Conference on Information Technology, Bhubaneswar, 2014, pp. 270-274, doi: 10.1109/ICIT.2014.29..

[3] Havlíček, V., Córcoles, A.D., Temme, K. *et al.* Supervised learning with quantum-enhanced feature spaces. *Nature* **567,** 209–212 (2019). https://doi.org/10.1038/s41586-019-0980-2

[4] Abraham H, Akhalwaya IY, Aleksandrowicz G, Alexander T, Alexandrowics G, Arbel E, Asfaw A, Azaustre C, Barkoutsos P, Barron G. Qiskit: An Open-source Framework for Quantum Computing(2019)

[5] Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe, "Circuit-centric quantum classifiers" (2018)

[6] C. G. Almudever et al., "Towards a scalable quantum computer," 2018 13th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS), Taormina, 2018, pp. 1-1, doi: 10.1109/DTIS.2018.8368579.

[7] E. P. DeBenedictis, "A Future with Quantum Machine Learning," in Computer, vol. 51, no. 2, pp. 68-71, February 2018, doi: 10.1109/MC.2018.1451646.

[8] N. Toronto and D. Ventura, "Learning Quantum Operators From Quantum State Pairs," 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, 2006, pp. 2607-2612, doi: 10.1109/CEC.2006.1688634.

[9] S. Zhou, Q. Chen and X. Wang, "Deep Quantum Networks for Classification," 2010 20th International Conference on Pattern Recognition, Istanbul, 2010, pp. 2885-2888, doi: 10.1109/ICPR.2010.707.

[10] Thabet, Slimane. "Decomposing Step by Step a Quantum Machine Learning Algorithm." Medium, Towards Data Science, 24 Mar. 2020, towardsdatascience.com/decomposing-step-by-step-a-quantum-machine-learning-algorithm-8adfa66aed7e.

[11] Shafiq, Maham. "Quantum Machine Learning: Hybrid Quantum-Classical Machine Learning with PyTorch and Qiskit." Medium, Noteworthy - The Journal Blog, 11 May 2020, blog.usejournal.com/quantum-machine-learning-hybrid-quantum-classical-machine-learning-with-pytorch-and-qiskit-d03da758d58b.

[12] Agnihotri, Shubham. "Quantum Machine Learning 102 - QSVM Using Qiskit." Medium, QuantumComputingIndia, 23 Sept. 2020, medium.com/quantumcomputingindia/quantum-machine-learning-102-qsvm-using-qiskit-731956231a54.

[13] "Variational Classifier." PennyLane, pennylane.ai/qml/demos/tutorial_variational_classifier.html.

[14] Association, Quantum World. "Quantum Computation: a Journey on the Bloch Sphere." Medium, Medium, 22 May 2018, medium.com/@quantum_wa/quantum-computation-a-journey-on-the-bloch-sphere-50cc9d73530.