ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

A Brief Review of Quantum Circuits and Tensor Network in Machine Learning

Zijian He[1], Wenyu Li[2], Jiangyi Yao[3], Yangfenghui Huang[4]

[1]School of Computer Science, University of Nottingham, Nottingham NG8 1BB, UK

[2]School of Computing and Information, Dietrich School of Arts & Science, University of Pittsburgh, Pittsburgh, PA 15213, USA

[3]Henry Samueli School of Engineering, University of California Los Angeles, Los Angeles, CA, 90024, USA

[4]School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei 430072, China

Abstract

This paper is an introductory review on the selected topics on quantum machine learning and tensor network, based on educational lectures and resources. Quantum computing has shown its ability to surpass classical computers which consists of conventional digital gates on many tasks. However, the major restriction of this technique lies in limited number of physical qubits available in a near-term device and its overwhelming noise when the size and complexity of a quantum circuit increases. Therefore, a hybrid quantum-classical system is proposed to take full advantage of quantum computing while leaving some of the computational burden to classical computers. Additionally, stemmed from the motivation of using tensor network in machine learning, the matrix product state (MPS) may be applied when designing the structure of model circuit part in the hybrid quantum-classical system to further reduce the number of elements required in quantum computing. A classical supervised learning algorithm based on tensor network is implemented, as well as a normal hybrid quantum-classical system with parameterized quantum circuit to examine their effectiveness independently. This work proposes several ideas on designing tensor network based hybrid quantum-classical model which contains strengths from both techniques to achieve quantum resource-efficient algorithm with promising noise resilience.

Keywords

Tensor Network; Matrix Product State; Parameterized Quantum Circuits; Convolutional Neural Network; Machine Learning.

1. INTRODUCTION

During the last few years, the integration of quantum computing, tensor network and machine learning has revealed great potential in promoting AI technologies. Many researches have been done in the pursuit of amplifying computation and innovating machine learning [1-3], not just as numerical methods, but also harnessing the equivalence of tensor networks to various quantum circuits. As both quantum computing and tensor network are ever-expanding fields, this work elaborates on the integration of Tensor Network and Quantum Circuits with machine learning separately, followed by a discussion of tensor network and quantum circuits together, in particular the tensor network MPS.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

2. TENSOR NETWORK IN MACHINE LEARNING

This section explains how to harness the advantage of tensor network, specifically MPS, in a classical supervised learning task - classifying written digits using MNIST dataset.

2.1. Tensor Network Background & Its Advantages

Tensor networks are approximation of high-dimensional tensors which makes problem solving in physics and now machine learning [4] more efficient, in other words, it's a numerical simulation method. As a scheme for assembling units (neurons in machine learning or tensors in quantum mechanics) into functions, tensor Network first became popular in dealing with quantum many-body systems on condensed matter physics, which was proved to be viable for simulation in 1984 [5]. Other consequential application like lattice models was also deeply explored [6-8]. Tensors act as interconnected building blocks for the wave function; thus, Tensor Network (TN) works as the decomposition where entanglement glues all pieces together. Intuitively, one may be inspired of dealing with neural network in a similar matter. TN has become a significantly popular choice for simulating systems with strong correlations, in the realm of numerical simulation algorithms; mathematically, the number of parameters in the tensor and the network pattern together determines the structure of entanglement.

With tensor network only having linear elements, neural networks that machine learning requires specifically uses non-linear elements for its success, so one may question the motivation behind using tensor network. As one encodes inputs in a quantum state as a nonlinear feature map that maps data to quantum Hilbert space, classical data is lifted to higher dimension, which is beneficent for more complicated problems. One of the major motivations behind tensor network is that the Hilbert space grows exponentially in a quantum many-body system, as is it in other applied fields. A more detailed explanation follows.

2.2. Encoding Data

The motivation of using tensor network has similarity with the motivation of employing kernel function to bypass the dimension curse when we are trying to lift the original data to an extremely high-dimension and apply linear operation in that feature space. The first step here would be to define such feature mapping function.

For a greyscale picture with N pixels, $N = 28^2 = 784$ in MNIST dataset, the traditional approaching would be to flatten the whole pixel "grid" to a list of pixels. The process is intuitive and simple to implement, but it eliminates the hidden feature in the context of two-dimensional nature of an image, where pixels may have intrinsic correlation with their neighbourhoods. Besides, change on a small amount of pixels would have negligible effect on the representation of an image input which is not desirable in a machine learning task. Alternatively, N pixels of an image is encoded into a 2^N −dimensional space as follows, similar approach is also proposed in [9].

First, each pixel will be encoded into a two-dimensional space using a local feature map.

$$\phi(p) = \begin{pmatrix} 1-p \\ p \end{pmatrix}$$

The notation $\Phi(pi)^j$ is used to indicate components in this two-dimensional space where i stands for ith pixel and j{0, 1}. For the feature mapping described above, $\Phi(pi)^0 = 1 - p_i$ and $\Phi(pi)^1 = p_i$. To include the hidden connection between each pixel to the feature, we further lift the feature space by applying tensor product over all the local pixel spaces which generates all possible combinations of pixels from an image. A key property of this feature space is that changes on one pixel, say flipping it from white to black, would result in a totally different feature. We denote the process as follows:

$$\kappa(p_1, p_2, ..., p_N) = \Phi(p_1) \otimes \Phi(p_2) \otimes ... \otimes \Phi(p_N)$$

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

We call the resulting object a "data tensor" which will be 2^N –dimensional in our model.

2.3. Bypass Dimension Curse

After encoding our desired feature space, the classical linear algorithm is utilized to tackle the classification problem.

$$f^l(p_1, p_2, \dots, p_N) = W^l \kappa(\mathrm{p}_1, \mathrm{p}_2, \dots, \mathrm{p}_\mathrm{N}) = \sum_J W^l_{j_1, j_2, \dots j_N} \phi(p_1)^{j_1} \quad \phi(p_2)^{j_2} \dots \phi(\mathrm{p}_\mathrm{N})^{\mathrm{j}_\mathrm{N}}$$

Looking into the equation above, one can make analogy to the energy representation of a wavefunction in physics many-body problem. The algorithm requires summation over 2^N elements which can be highly demanding for the computational capacity. However, borrowed from some clever approaches in quantum physics to bypass this dimension catastrophe, tensor network is applied to the "wave function" described above. As shown in Fig 1, each small box indicates a tensor contraction between $\Phi(p_i)$ and an associated weight tensor w_i. Contraction over all boxes are conducted to obtain the output label. In concrete, the result associates each local feature space $\Phi(p)$ with a weight matrix/tensor and contract over these small boxes instead.

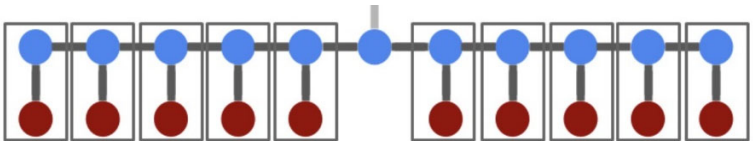


Figure 1. Decision function with tensor contraction [10]

The resulting weight can be viewed as a tensor network (MPS), we denote the new weight as:

$$\mathrm{W}_{\mathrm{i}_1, i_2, \dots, i_N} = \sum_{\alpha_1, \alpha_2, \dots, \alpha_N} A^{(1)}_{i_1 \alpha_1} A^{(2)}_{i_2 \alpha_1 \alpha_2} A^{(3)}_{i_3 \alpha_2 \alpha_3} \dots A^{(N)}_{i_N \alpha_N}$$

The range of each of the αk indices is called the bond dimensions which connected two adjacent weight tensors. The bond dimensions determine the element size of A and thus overall $W_{i1i2\dots iN}$. The bigger the bond dimension, the more accuracy for a MPS to represent the high dimensional tensor, along with the side effect as more computational capacity required to contract over all the weights. Thus, the bond dimension can also be a hyper-parameter which is to be trained and optimized in the learning process. That is to say, a unique model which not only reduce the element size from exponential scale to linear scale, but also can learn to control the number of parameters in the training process is created.

2.4. Matrix Product State

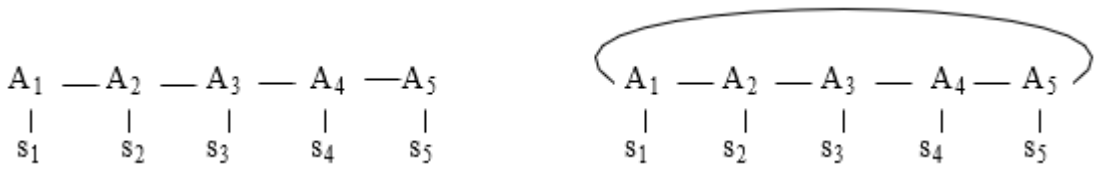


Figure 2. The graphical notation of MPS tensor network with open (left) and periodic (right) boundary conditions

MPS was the oldest Tensor Network model developed in the 80's. Given a large enough bond dimension or rank, an MPS/TT can represent an arbitrary tensor. Consider a tensor $T^{s1s2\dots sN}$ having N indices all of dimension d. Then this tensor can always be represented exactly as an

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

MPS/TT with bond dimension $m = d^{N/2}$. Generically, such a tensor must be specified by d^N parameters. In contrast, representing such a tensor by an MPS/TT network of bond dimension m requires Ndm^2 parameters, and this number can be reduced even further by imposing or exploiting certain constraints on the factor tensors.

As for S=1/2 spins on a lattice, there are two matrices associated to each lattice site, which matrix goes into the product depends on whether the spin is up or down. If the matrix sizes required to capture the true wavefunction do not grow with system size, then the MPS format compresses exponentially many parameters down to only a linear number of parameters $f(x) = W \cdot \Phi(x)$

Figure 3. Approximation of the weight tensor W by a matrix product state, also known as the kernel trick [10]

As shown in fig 3, the optimal weight vector W can be approximated as a tensor network, that is, as a contracted sequence of low-order tensors. [10]

Other algorithms based on TN are TEBD, Folding Algorithm, PEPS (for two-dimensional quantum systems), Tensor Renormalization Group, Tensor-Entanglement Renormalization Group (TERG), Entanglement Renormalization (ER), MERA (for critical system), ... just to name a few. [11]

2.5. Result

Layer (type)	Output Shape	Param #
mps_layer_4 (MPSLayer)	(None, 10)	157800
softmax_4 (Softmax)	(None, 10)	0

Total params: 157,800
Trainable params: 157,800
Non-trainable params: 0

(a) Number of parameters required by a MPS model with bond dimension 10

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 26, 26, 32)	608
conv2d_8 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 64)	0
dropout_4 (Dropout)	(None, 12, 12, 64)	0
flatten_2 (Flatten)	(None, 9216)	0
dense_4 (Dense)	(None, 128)	1179776
dropout_5 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 10)	1290

Total params: 1,200,170
Trainable params: 1,200,170
Non-trainable params: 0

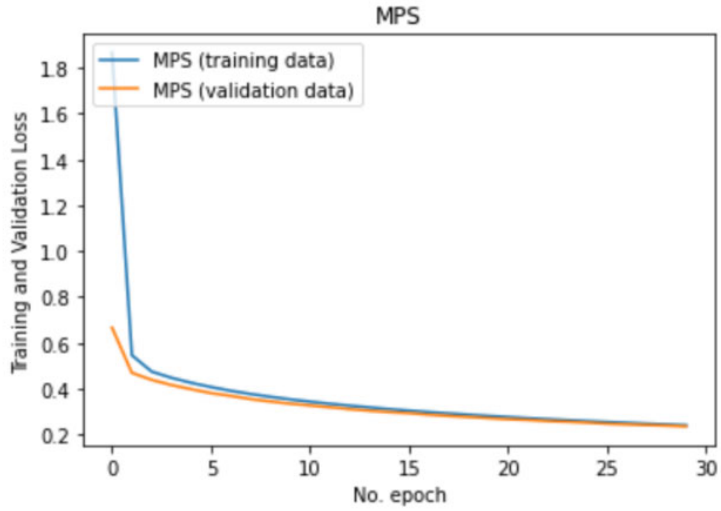
(b) Number of parameters required by a traditional CNN model

Figure 4. Configurations on the MPS model and a 4-layer traditional CNN with ReLU and softmax activations, followed by max pooling

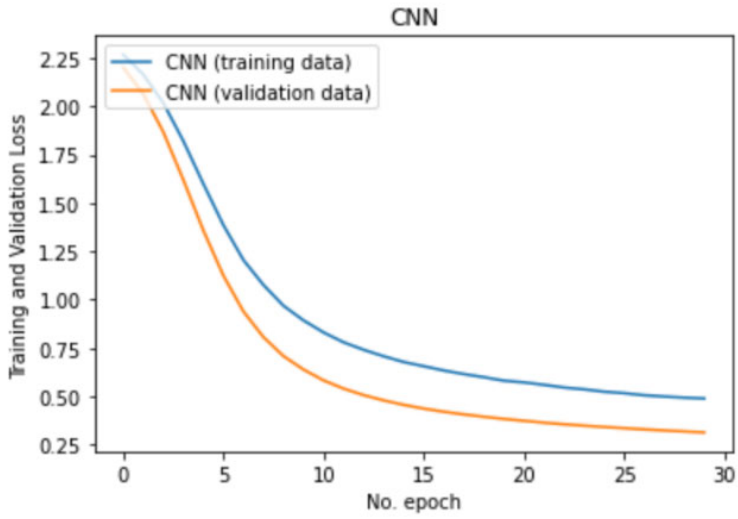
ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

A simple experiment is conducted by comparing the model with one MPS layer with a traditional CNN model on MNIST dataset. The batch size is set to 64 and they were run on 30 epochs. See Fig 4 for detailed information:

As the result denotes, the number of parameters required by a simple CNN is approximately ten times of the number of elements in a MPS model. It is worth noting that the majority of the parameter is concentrated on the fully connected layer named dense 4 in the model. A novel approach to individually tensorize this layer is proposed in [12]. Also, and ultimate version of this approach in which all the conventional layers are replaced by tensor network is proposed in [13].



(a) Loss function result for MPS



(b) Loss for CNN

Figure 5. Results from training two models

As Fig 5 shows, the training accuracy for the model with MPS layer is 92.9% with testing accuracy 92.8%; while the training accuracy for CNN is 85.1% with testing accuracy 91%. Though with a considerably sharp drop down in the number of parameters in MPS model, the accuracy reaches the same level as a normal CNN with ten times the scale of elements. Besides, it's noticeable that MPS model has a relatively faster convergent rate while CNN tends to follow a smoother pattern.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

3. PQC SYSTEM

3.1. Hybrid Quantum-Classical Circuits

An extensive believe called "quantum supremacy" that a quantum processor may execute certain computation-wise demanding tasks exponentially faster than a classical computer [14] is well supported by latest development in material science, hardware manufacturing and quantum algorithm such as error-correction. Google claimed to achieve random sampling calculation using only 53 qubits in about 200 seconds. Classical computing, on the other hand, is estimated to cost 10,000 years even on the world's most powerful supercomputer. However, the system noise is known to be the major drawback when lifting the depth and complexity of quantum circuits. Widespread investigations have been conducted to whether noisy intermediate-scale quantum (NISQ) [15] may find viable applications which can still hold quantum supremacy for specific tasks.

One popular approach towards noise involves formalizing problems as variational optimization models and then use hybrid – quantum and classical – systems to find the best solutions. The motivation is that by leaving some sub-tasks of a model to classical computers, the requirement of quantum computational resources, such as the number of qubits, circuit complexity and depth, can be considerably moderated. Parameterized quantum circuit (PQC) / Variational circuit has been proposed as a widely used hybrid quantum-classical system which, with simple structure, is demonstrated to outperform classical neural network for generative tasks [16]. The general hybrid system contains three components: the environment, the classical computer and the quantum computer [17]. The environment provides tasks and data to train the system. The classical computer initializes the random parameters and settings for the PQC. The quantum model prepares qubits' initial states corresponding to the input data and performs unitary operations as well as measurements on single or multiple qubits to obtain outcomes which are further processed by classical computer to generate forecast. Learning strategies is implemented in classical processors to optimize the forecasts by adjusting and updating parameters for PQC. This process is performed in loops, in which quantum and classical system are constantly interacting with each other.

The hybrid approach based on PQC does not gain its popularity in machine learning field until the similarity between PQC and celebrated classical machine learning models, e.g. kernel method [18], is realized more recently. It has now been shown to perform well on extensive tasks in classification, regression and generative modeling etc. In the rest of the report, this paper illustrates PQC based hybrid approach in machine learning context and introduce tensor network application when designing the structure of quantum circuit.

3.2. Quantum Circuits

Similar to classical computers, a quantum processor achieve computation by chaining a sequence of quantum gates applied to qubits. Analogous to classical logic gates, any computation in a quantum circuits can be achieved by a series of basic gates which consists of rotation gates and controlled-Not (CNOT) gates. Four general gates are defined below:

x-axis rotation gate R_X (λ)

$$R_x(\lambda) = \begin{pmatrix} \cos\left(\frac{\lambda}{2}\right) & i * \sin\left(\frac{\lambda}{2}\right) \\ i * \sin\left(\frac{\lambda}{2}\right) & \cos\left(\frac{\lambda}{2}\right) \end{pmatrix}$$

y-axis rotation gate R_Y (ε)

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

$$R_Y(\varepsilon) = \begin{pmatrix} \cos\left(\frac{\varepsilon}{2}\right) & \sin\left(\frac{\varepsilon}{2}\right) \\ -\sin\left(\frac{\varepsilon}{2}\right) & \cos\left(\frac{\varepsilon}{2}\right) \end{pmatrix}$$

z-axis rotation gate $R_Z(\theta)$

$$R_x(\lambda) = \begin{pmatrix} e^{\frac{i\theta}{2}} & 0 \\ 0 & e^{-\frac{i\theta}{2}} \end{pmatrix}$$

NOT gate conditionally flips the target qubit if the control bit is $|1\rangle$

$$CNOT) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Other quantum gate is derivative from the above gates. E.g. The Pauli-Z gate, defined as $P_Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ can be represented by R_Z ($\theta = \pi$).

3.3. Parameterized Quantum Circuits

Variational or parameterized quantum circuits are quantum special quantum circuits that based on external parameters which can be controlled and fed by classical computers. A standard PQC consists of three components.

1. Preparation of initial state for each qubit according to the given data. (non-trainable)
2. A unitary quantum circuit U(Θ), parameterized by external arguments. (trainable)
3. Measurement of an observable $\hat{B}$ at the output. This result can be made up from observable from each wire or just a subset of wires.

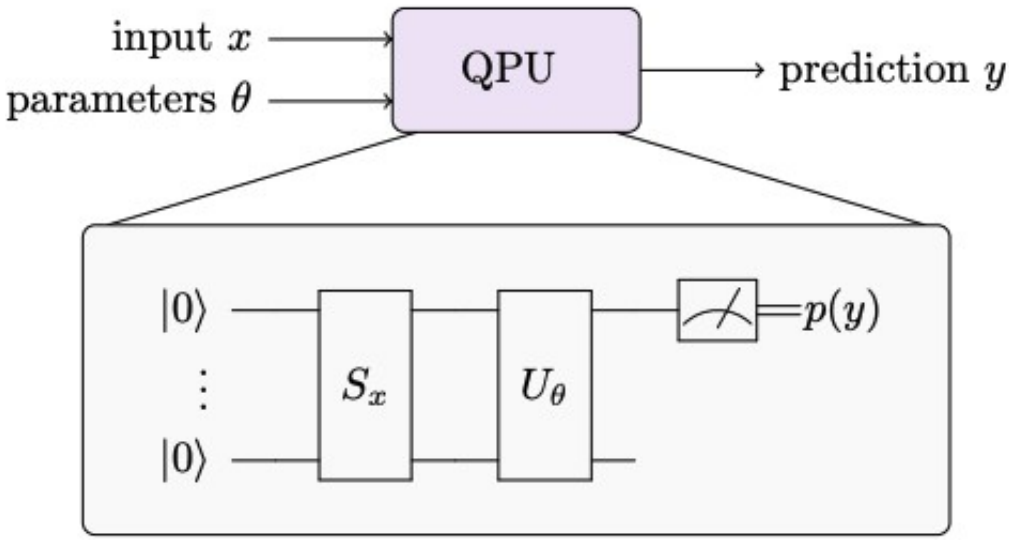


Figure 6. Variational circuit of a hybrid quantum-classical system [19]

The structure of PQC (see Fig. 6) can be viewed as the celebrated machine learning model — kernel methods — except that quantum feature mapping and direct interaction with data in Hilbert space is implemented in quantum processor whereas classification and parameter learning is conducted in classical computers. In the following section each one of the three components in variational circuit will be explained in detail. Prediction with the linear model f(x, θ) = y is calculated by a quantum "box" which consists of a state preparation circuit Sx

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

encoding the input x into quantum state information, a model circuit $U\theta$ and a measurement of qubit on a single wire in this case. The measurement extracts the probability of the prediction of "True" or "False", which can be post-processed classical computer to infer the binary prediction on that task. The free parameter controlling unitary gate Uθ can be trained and updated in classical processor according to the prediction using gradient descent.

3.3.1 Feature-embedding circuits

From the perspective of quantum computing, a circuit preparing initial state for each qubit is equivalent to a quantum feature mapping which encodes a classical input into quantum Hilbert space $x \rightarrow |\varphi(x)\rangle$. The circuit $U_\varphi(x)$ is called the feature-embedding circuit which is applied to a ground/vacuum state, where all qubit can be represent as state $|0\rangle$, in Hilbert space as $U_\varphi(x)|0...0\rangle = |\varphi(x)\rangle$. Let's consider classical input data containing M samples each with N features.

$$D = \{x^{(1)}, x^{(2)}..., x^{(M)}\}$$

Where $x^{(i)}$ is a N−dimensional data tensor for i = 1, ..., M. To encode this data tensor into quantum information with n qubits available, one can utilize various data embedding schemes, two of which are widely applied and will be explained below.

Basis Embedding Basis embedding associates each input with a computational basis state of a qubit system. Therefore, target data has to be in the form of binary strings. The encoded quantum state is actually the bit-wise version of the binary string in quantum state context. For example, x = 1001 is encoded into a 4-qubit quantum system $|1001\rangle$.

In terms of classical dataset D above. $x^{(i)}$ has to be a N-bit binary logits $\rightarrow x^{(i)} \in \{0,1\}^N$. The number of qubits required n, is at least equal to N, the dimension of feature or the length of its binary representation.

$$U_\phi : x \in \{0,1\}^N \rightarrow |i\rangle$$

with $|i\rangle$ being the corresponding representation of bit string in computational basis state.

Amplitude Embedding In this encoding strategy, data is embedded into the coefficient of each quantum state. A normalized target N-dimensional data x can be represented by the amplitudes of a n-qubits quantum state $|\psi_x\rangle$ as:

$$U_\phi : x \in^N \rightarrow |\psi_x\rangle = \sum_{i=1}^{N} x_i |i\rangle$$

Where $N = 2^n$ and $|i\rangle$ denotes the i'th computational basis state in this n-qubits quantum system. To take a concrete example to demonstrate this embedding scheme, say, a four-dimensional data array x = {1., 0., −5., 0.} is to be encoded. First step is to normalize it: $x_{norm} = \frac{1}{\sqrt{26}}\{1., 0., -5., 0.\}$. Then the corresponding amplitude encoding can use the above unitary operator to construct quantum state $|\psi_x\rangle$ as:

$$|\psi_x\rangle = \frac{1}{\sqrt{26}}\{1.|00\rangle + 0.|01\rangle + 5.|10\rangle + 0.|11\rangle\}$$

In this case, the input data is embedded into the coefficient σ assigned to each superposition instead of quantum state itself. The normalization ensures that summation of probabilities σ^2 over all superpositions is equal to one.

3.3.2 Model Circuit

Having embedded the input data into a quantum state, certain processing can now be performed. As for an empty wire with no operator on it, our circuit would represent the model f(x) = x with no free trainable parameters in it. However, by chaining additional parameterized gates on wires, one can achieve more complex manipulation on input data with adaptive

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

arguments which is trained to minimize the dissimilarity between result of processing and the true label.

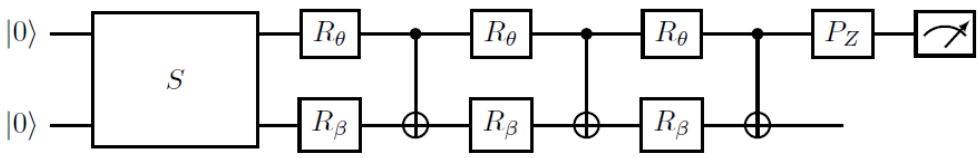


Figure 7. Ansatz for quantum processing

In the context of variational circuits, an Ansatz is usually denoted as a subroutine consisting of a series of operators on specific wires. Similar to the idea of neural network, ansatz only defines the basic structure of a model circuit, while the types of gates and ratio of trainable/non-trainable parameters when defining the operator is contingent to the task. One can distinguish three major different base structures, layered gate ansatz, alternating operator ansatz and a tensor network ansatz, which will be discussed in the Tensor Network Section. Many variations and adaptions of these basic circuit ansatz have been proposed [19, 20]. The advantage of a specific ansatz depends on the given task, use-case, data structure and is not always clear. There is even an interesting "structure learning" strategy [21] in which the choice of gates on fixed position is constantly updated in each run, though the implementation can be considered as brute force which basically examine all the possible option and chose the one with minimal loss.

Fig 7 is an illustration of the circuit model this paper utilized. To prepare the initial quantum state, a state operator S is applied to both vacuum state, then three repeated circuit layer is chained to achieve model circuit which processes the embedded data. Finally, the expected value of Pauli Z operator of the first qubit is measured.

3.3.3 Measurement and Post-processing

Remember that unitary operators are just linear transformations that preserve the length of a vector. And we can view the overall model circuit as a single unitary operation based on some trainable parameters set θ. After executing the model circuit Uθ in the above step, on the measurement site, the expected value of Paulic operator on the qubit at the first wire is obtained. The Pauli operator would generate either +1 or −1, however, the estimated outcome before measure is not certain - a state of a qubit fits certain probabilistic distribution, that's to say, it's uncertain - our expected value of a measurement can be obtained by running the entire circuit several times and take the average over all the observed outcomes. Consequently, our predicted label value is a real number πx;θ between -1 and 1 which indicated the possibility of corresponding label. To post-process the outcome in classical processor, one can simply apply a threshold mapping after adding some learnable bias to the measurement from quantum circuit [19], which acts like activation layer ReLU in classical neuron network.

$$y_{(x;\theta;b)} = \pi_{(x;\theta)} + b$$

$$f(\theta; b) = \begin{cases} 1, if\ y_{(x;\theta;b)} > 0.5 \\ 0, else \end{cases}$$

3.4. Implementation of PQC

This work implemented a basic Variational Circuit which is used to tackle Multi-class margin classifier. The Iris dataset which has 4 features and 3 classes is utilized to demonstrate the effectiveness of our implementation. It's based on an open-source library Pennylane [22] which is a Python based framework for optimization and machine learning of quantum and hybrid quantum-classical computations. The library provides built-in support for qubit and simulate gate operations on classical computer. Its main feature is the ability to compute gradients of

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

PQC compatible with backpropagation supported in other conventional machine learning library such as TensorFlow or PyTorch.

3.4.1 Variational Circuit Implementation

Our model is explained here with respect to three components listed above. For embedding circuit, amplitude embedding scheme which maps each feature to the coefficient assigned to corresponding computational basis state is employed. Since N, the dimension of feature is 4, we can represent the data array with only O(log(N)) = 2 number of qubits. Multi-layered ansatz is applied here as the basis structure of our modeling circuit. In fact, the number of repetitions of a layer can become a hyper-parameter of the variational circuit. But here the repeated layer is fixed to depth-3, each consists of rotation gates for each qubit, followed by entangling/CNOT gates. (See Fig 7 for an illustration)

In what follows, this paper defines the quantum circuit that can be used to indicate the possibility of an input belonging to one specific class. To implement our multi-class classifier as one-vs-all classifiers, three identical quantum circuits are included in the model, each represents one classifier on one specific class. The variational model circuit here is parameterized by the weights α, β controlling the extent of rotation on each qubit. A trainable bias b is also appended to the result in post-processing step. Both variational circuit weights and classical bias should be optimized with respect to designed loss function.

3.4.2 Loss Function

It's intuitive to use margin loss when implementing multi-class classifier by one-vs-all classifiers. The score

s_i of ith classifier c_i on input x is between [−1, 1] as mentioned above. More specifically, we denote s_i as:

$$s_i = c_i(x;\theta) + b$$

As we should expect that the score from the classifier of true label should be higher than that from others, we can define the loss function as:

$$L(x,y) = \sum_{i \neq y} \max(0, s_i - s_y + \Delta)$$

Where Δ denotes the magnitude of margin which can also be a hyper-parameter in Multi-class linear SVM [23].

3.4.3 Training Loop

In the training process, first the free parameters and classical bias are randomly initialized. Thanks to the power-autograd functionality provided by the Pennylane library, we can set the requires grad tag to True. The result also employs mini-batch in the training process in order to moderate computational complexity of updating parameters in each iteration and to contribute to noise tolerance and stable convergence. In each run, the data would be pre-processed in classical computer, such as batching, and then feed to variational quantum circuit. The outcome would be sent back to classical computer to facilitate gradient descent to alleviate the burden on quantum computational resource. Finally, the parameters and bias would be updated and enter the next loop.

3.4.4 Result

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

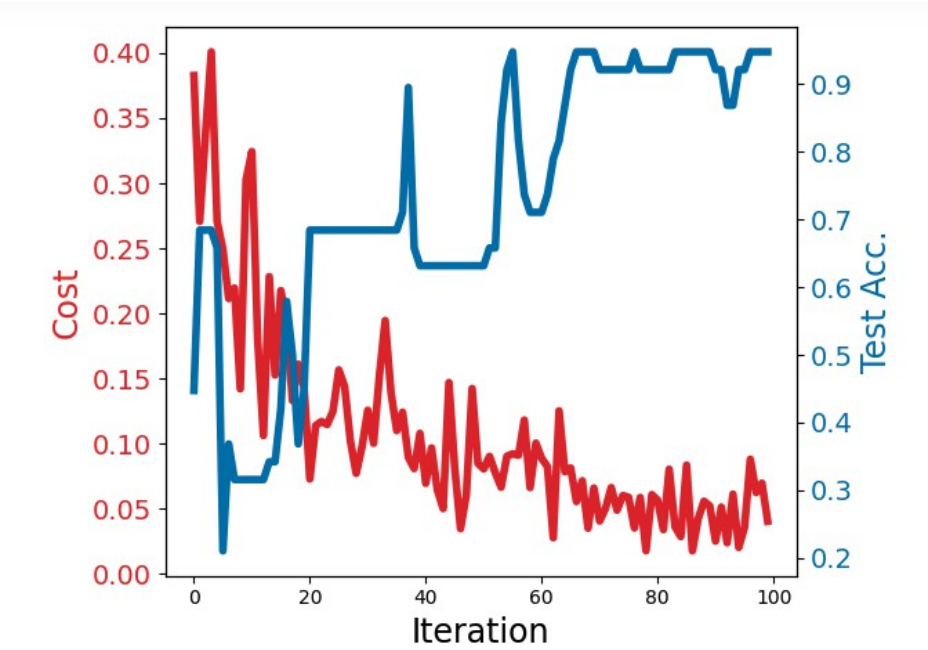


Figure 8. Training and test results for the PQC model

As shown in Fig 8, the huge fluctuate in cost value along the training process reviews a significant drawback of our hybrid quantum-classical model. One possible explanation could be insufficiency of depth and complexity of model circuit which may dramatically drag down the expressive power of the overall model. Besides, naivete amplitude embedding scheme may oversee hidden correlation between features. Therefore, the model is not promising when it is feed with a batch of input with certain hidden patterns.

4. QUANTUM MACHINE LEARNING WITH TENSOR NETWORKS

Tensor networks is a promising architecture in classical machine learning models. These approaches borrowed from physics has shown its potential to give good performance in both supervised [12, 24], unsupervised [25] and hybrid [26] tasks with largely reduced number of features. It's intuitive to combine tensor network as a basis structure when designing the model circuit since the major obstacle of near term quantum machine involves limited number of qubits and excessive noise when dealing with real data which typically contains hundreds or thousands of components.

The algorithm introduced in this paper is aimed at discriminative tasks using circuits equivalent to tensor networks, specifically matrix product states. As already mentioned in the PQC section, the TN circuit will contain free parameters to be optimized, in contrast to some well-known quantum algorithms such as Shor's algorithm for factorization. Only the geometry identity/Ansatz is determined, while the variational parameters controlling the unitary gates would be trained and updated with the help of auxiliary optimizer running on classical computers. To motivate the use of tensor network as an advantageous quantum circuit for machine learning tasks, three main reasons [20] have been proposed:

Quantum resource efficient circuit The number of physical qubits required can be largely scale down to either logarithmic or even independent with the size of the input data if certain tensor network models can be implemented on a neat-term quantum machine. The structure of tensor network also indicates the extent of intrinsic entanglement in the model circuit, which potentially permits the measurement of a subset of qubits on several wires without extensive interfere with the others. This may contribute to a promising resilience to noise.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

Compatibility A quantum circuit to be evaluated can be analogous to the variational tensor network to be trained in classical machine learning field. With the gradual crossover from quantum circuit to tensor network, the same models trained in conventional machine learning context and be easily migrate to quantum processor which grants more expressive power and faster computational speed.

Matured and fruitful outcome in tensor network area The benefits and properties of tensor networks have been thoroughly investigated and are likely to be developed further in the machine learning context. Some of these properties can be utilized in quantum circuit, such as using less qubits to achieve robustness or noise-resilience.

4.1. MPS Circuit Model

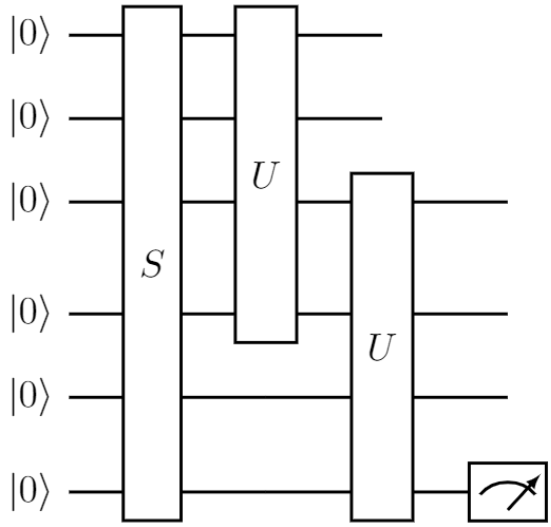


Figure 9. Discriminative circuit based on matrix product state

Fig 9, in which V = 2 qubits are passed to the next subtree (MPS can be treated as special unbalanced trees), gives an intuition about the design of our MPS circuit. The first gates applying to multiple qubits is the state preparation operator as mentioned before, including basis embedding, amplitude embedding and other Hilbert space feature mapping. The following circuit model is in the form of a matrix product state, with V qubits connecting to the next "layer" of unitary operator. We can make analogy of this V to the bond-dimension notation in the classical MPS structure. A large V can represent more complex gate operators and intensive entanglement between linked qubits, which is basically equivalent to more expressive power of a MPS network with higher bond-dimension.

At each "layer", V qubits obtained from unitary operators in the previous step combined with V "intact" qubits form a set of qubits input for the current parameterized gates. Then V qubits are abandoned or even recycled, while the others proceed to the next step. Once all subtrees are carried out, we measure the expected value of last qubit, whose estimated probability of certain state can serve as the prediction.

5. CONCLUSION

Although promoted by Physics, tensor network and quantum circuits align with machine learning in a propitious way. By applying feature mapping and tensor contraction, out results with MPS layer show an obvious refinement on CPU time with comparative accuracy. The quantum-classical hybrid system allows us to better realize the power of quantum computer to its full potential. With PQC, as this work shows, quantum supremacy can be explored in a

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

machine learning like manner. Furthermore, employing tensor network such as matrix product state in quantum machine learning tasks yields great benefit: the model only requires small number of physical qubits. The major property of achieving this goal is to allow an isolated measurement of a subset of qubits which may not disturb the probable state of other qubits. Another key benefit of embedding tensor network into circuit model can be their robustness to noise which can be a major problem in near-term quantum machines.

In the future, based on the proposed model, more work could be done to implement the MPS-quantum system. Additionally, since tensor network parameterizing reduces the number of parameters, it would be beneficial to explore the impact of that for some particular tasks. For areas like spectral image, one may want to Apply the MPS algorithm as explained in this paper to multi-spectral cameras, where datasets such as [27, 28] could be trained and the stochastic gradient descent could be replaced with ADMM method.

REFERENCES

[1] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. Nature, 549(7671):195–202, 2017.

[2] Rose Yu, Stephan Zheng, Anima Anandkumar, and Yisong Yue. Long-term forecasting using higher order tensor rnns. arXiv preprint arXiv:1711.00073, 2017.

[3] Jinhui Wang, Chase Roberts, Guifre Vidal, and Stefan Leichenauer. Anomaly detection with tensor networks. arXiv preprint arXiv:2006.02516, 2020.

[4] Florian AYN Schroder, David HP Turban, Andrew J Musser, Nicholas DM Hine, and Alex W Chin. Tensor network simulation of multi-environmental open quantum dynamics via machine learning and entanglement renormalisation. Nature communications, 10(1):1–10, 2019.

[5] EL Pollock and David M Ceperley. Simulation of quantum many-body systems by path-integral methods. Physical Review B, 30(5):2555, 1984.

[6] Michael Levin and Cody P Nave. Tensor renormalization group approach to two-dimensional classical lattice models. Physical review letters, 99(12):120601, 2007.

[7] Ashley Milsted, Martin Ganahl, Stefan Leichenauer, Jack Hidary, and Guifre Vidal. Tensornetwork on tensorflow: A spin chain application using tree tensor networks. arXiv preprint arXiv:1905.01331, 2019.

[8] Pietro Silvi, Ferdinand Tschirsich, Matthias Gerster, Johannes Ju¨nemann, Daniel Jaschke, Matteo Rizzi, and Simone Montangero. The tensor networks anthology: Simulation techniques for many-body quantum lattice systems.

[9] Alexander Novikov, Mikhail Trofimov, and Ivan Oseledets. Exponential machines, 2016.

[10] Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. In Advances in Neural Information Processing Systems, pages 4799–4807, 2016.

[11] Roman Oru´s. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. Annals of Physics, 349:117–158, 2014.

[12] Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry Vetrov. Tensorizing neural networks, 2015.

[13] Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, and Dmitry Vetrov. Ultimate tensorization: compressing convolutional and fc layers alike, 2016.

[14] Aram W. Harrow and Ashley Montanaro. Quantum computational supremacy. Nature, 549(7671):203–209, Sep 2017.

[15] John Preskill. Quantum computing in the nisq era and beyond. Quantum, 2:79, Aug 2018.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202101_7(1).0037

[16] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. Expressive power of parametrized quantum circuits. Physical Review Research, 2(3), Jul 2020.

[17] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. Quantum Science and Technology, 4(4):043001, nov 2019.

[18] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. Physical Review Letters, 122(4), Feb 2019.

[19] Maria Schuld, Alex Bocharov, Krysta M. Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. Physical Review A, 101(3), Mar 2020.

[20] William Huggins, Piyush Patil, Bradley Mitchell, K Birgitta Whaley, and E Miles Stoudenmire. Towards quantum machine learning with tensor networks. Quantum Science and Technology, 4(2):024001, Jan 2019.

[21] Mateusz Ostaszewski, Edward Grant, and Marcello Benedetti. Quantum circuit structure learning, 2019.

[22] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M. Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, Keri McKiernan, Johannes Jakob Meyer, Zeyue Niu, Antal Szava, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations, 2018.

[23] Cs231n convolutional neural networks for visual recognition, 2020.

[24] Ivan Glasser, Nicola Pancotti, and J. Ignacio Cirac. From probabilistic graphical models to generalized tensor networks for supervised learning, 2018.

[25] Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang. Unsupervised generative modeling using matrix product states. Physical Review X, 8(3), Jul 2018.

[26] E Miles Stoudenmire. Learning relevant features of data with multi-scale tensor networks. Quantum Science and Technology, 3(3):034003, Apr 2018.

[27] Inchang Choi, Daniel S. Jeon, Giljoo Nam, Diego Gutierrez, and Min H. Kim. High-quality hyperspectral reconstruction using a spectral prior. ACM Trans. Graph., 36(6), November 2017.

[28] F. Yasuma, T. Mitsunaga, D. Iso, and S.K. Nayar. Generalized Assorted Pixel Camera: Post-Capture Control of Resolution, Dynamic Range and Spectrum. Technical report, Nov 2008.