Playing Atari with Hybrid Quantum-Classical Reinforcement Learning

Owen Lockwood
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180
lockwo@rpi.edu

Mei Si
Department of Cognitive Science
Rensselaer Polytechnic Institute
Troy, NY 12180
sim@rpi.edu

Abstract

In order to address the challenge of classical to quantum data conversion in reinforcement learning, we propose to use a neural network as a data encoder, and apply this technique to Atari games. Specifically, the neural network converts the pixel input from the games to quantum data for a Quantum Variational Circuit (QVC); this hybrid model is then used as a function approximator for the Q value in Double Q Learning. Through several ablation studies we investigate the effect of encoding architectures, number of quantum bits, and output techniques on the performance. In addition, we compare the hybrid quantum-classical approach to the purely classical neural network approach and evaluate empirically the computational advantage.

1 Introduction

Reinforcement Learning (RL) has advanced dramatically in the last decade. Superhuman performance has been achieved in a variety of very complex tasks such as Go [31], Dota 2 [4] and StarCraft II [36], via deep RL. Common RL benchmarks include games and robotic manipulation as both have defined rules and reward signals. Algorithmic improvements in the field of deep RL are usually driven by a desire to achieve superior performance, achieve this performance faster, and/or reduce the training time or model size.

Parallel to the acceleration of deep RL achievements, significant advancements have been made in quantum computing. Early work in quantum computing was catalyzed by Shor's algorithm, a polynomial time algorithm for integer factorization with significant cryptographic implications [30]. However, it has only been in recent years that quantum computing became a realizable technology, with recent claims of 'quantum supremacy', i.e. running an algorithm on a quantum computer that would be intractable on a classical computer [2]. Quantum algorithms are able to offer unique speedups because of their exploitation of quantum mechanical properties, such as superposition and entanglement.

Quantum machine learning (QML) has attracted an increasing amount of attention in recent years. There is significant potential for theoretical quantum speedups on machine learning tasks, e.g. quantum perceptrons and quantum RL have the potential for $O(\sqrt{N})$ speedups [5]. Already work has been done to develop quantum generative adversarial networks [11], [17], quantum Hopfield networks [27] and quantum support vector machines [26]. Recently, the quantum RL field [12] has been expanding with a variety of approaches such as using Q Learning and SARSA [14] and quantum energy methods [15]. These works differ from this proposal as they utilize different techniques (such as Grover iterations and Boltzmann machines), environments, encodings, usually being restricted to Gridworld based environments.

We investigate the potential that quantum computing has to aid with reinforcement learning. We expand upon our previous work Lockwood and Si [18], which was in turn inspired by Chen et al. [8] to use Quantum Variational Circuits (QVC) – quantum circuits with gates parameterized by learnable values – in reinforcement learning. In [8], QVCs were used with Double DQN for a deterministic 4x4 gridworld. Chen et al. [8] reported that the parameter space complexity scales linearly with the input space in QVCs, i.e. $O(N)$, which is a significant improvement over the traditional neural network DQN which has parameter space complexity $O(N^2)$. They utilized computational basis encoding, which involves converting an input into binary and flipping the qubits to that binary state. However, this technique is unsuitable when the size of inputs is large or when floating point inputs are involved. Lockwood and Si [18] demonstrated the applicability of using QVCs to environments with larger input spaces by utilizing a more efficient encoding scheme. Specifically, the encoding scheme transformed each value in the input into rotations for a single qubit. This means that the number of qubits required is equal to the length of the input. This is feasible when the input is of size 4 (like in CartPole), but not possible for larger input spaces. For Atari, this would require 7,056 qubits for a single frame. This is infeasible, meaning that traditional benchmarks (like Atari) remain inaccessible. Although algorithms exist for optimal (amplitude) encoding schemes, i.e. encoding 2^N numbers in N qubits, they require an exponential number of gates (in relation to the input size) which is not only intractable but negates the exponential gains from the amplitude encoding [29] [23].

In order to solve the encoding problem, we employ a neural network to approximate encoding classical data into quantum circuits. The Atari environments were previously impossible due to the dimensionality and size of the required inputs, but are made available by using a neural network encoder. To investigate this encoder, and the potential for QML to assist with RL tasks, we propose a large empirical study to be conducted on the Noisy Scale Intermediate Quantum (NISQ) [25] simulator developed by Google as an extension to Tensorflow: Tensorflow-Quantum [7]. Combining recent improvements in the field of quantum RL and expanding from previous work, we investigate and compare hybrid quantum classical approach with purely classical approaches. We apply our techniques to two pixel based Atari OpenAI Gym environments, Breakout and Pong [6]. The input into our proposed model is more than 7,000 times larger than CartPole used in [18], one of the most complex previous environments. We hypothesize that this technique of using a neural network encoder will solve the previous problems of encoding and enable us to unlock quantum advantages even for large, high dimensional input spaces.

2 Background

2.1 Reinforcement Learning

Reinforcement learning is conceptualized as at least one agent interacting with an environment with the goal of maximizing a numerical reward signal [33]. A common formalization of RL are Markov Decision Processes (MDPs). The MDP tuple, $\langle S, A, P, R, \gamma \rangle$, consists of a set of states S, actions A, the probability of transition from one state to the next $P = P[s_{t+1} = s' | s_t = s, a_t = a]$, the reward, R, and the future reward discount γ. The goal is to design an agent that with policy π, $\pi(s_t) = a_t$, such that it maximizes the expected reward, $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | \pi\right]$.

Deep Q Networks (DQN) are an off policy and model free algorithm that uses a function approximator to estimate the Q function [21]. The Q function approximator, parameterized by θ, is defined as the expected future reward $Q_\theta(s, a) = \mathbb{E}_\theta[R | s_0 = s, a_0 = a]$. This can also be defined recursively for easier updates: $Q(s_t, a_t) = r_t + max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$, where $Q(s_t, a_t)$ is the Q value of a certain action in a given state at time t. The original Q learning policy is defined for discretized action spaces and is defined as $\pi(s; \theta) = max_a Q(s; \theta)$, i.e. the policy is to choose the action with the largest Q value as approximated by the parameters θ [37]. Updates to this policy are made via the mean squared Bellman loss, $L_t(\theta) = \mathbb{E}[(r_t + \gamma * maxQ(s', a'; \theta) - Q(s, a; \theta))^2]$ from which gradients can be calculated [21]. In Q learning with function approximation, failure to converge is a common problem. One source of this problem is the max operation, which leads to over-estimations of the Q value [34]. One solution to this is Double Q learning, used in this work, in which a separate target network is used exclusively for predicting the future Q value inside the max operation [35]. In this work we use neural networks and QVCs as function approximators that estimate the Q values.

2.2 Quantum Machine Learning

Quantum machine learning (QML) is the intersection of machine learning and quantum computing. It seeks to use quantum computing to obtain quantum advantage on machine learning tasks. Quantum advantages usually stem from the abilities of quantum computers to represent and operate on information that scales exponentially with the number of qubits.

Two of the most important features of quantum mechanics that quantum computing exploits are superposition and entanglement. Unlike in classical computers, where bits are limited to be 0 or 1, quantum bits (qubits) are capable of representing both 0 and 1 simultaneously. This is because the qubit is a quantum mechanical wavefunction Ψ that can be a linear combination of terms, e.g. $\Psi = \alpha|0\rangle + \beta|1\rangle$. This enables information represented to scale $O(2^N)$ for N qubits, giving an exponential advantage over linearly scaling classical bits. However, it is important to note that only a single value can be obtained from the wavefunction as it 'collapses' once a Hermitian operator (i.e. a measurement) is applied.

Entanglement is a more complex phenomenon results from the inseparability of combined wavefunctions. When two qubits are separate (i.e. not entangled) their wavefunction can be mathematically divided into individual wavefunctions. Consider one qubit in a superposition and another in the pure state $|0\rangle$, the two qubit wavefunction would be: $\Psi = \alpha|00\rangle + \beta|10\rangle$. This can easily be separated into $\Psi = (\alpha|0\rangle + \beta|1\rangle)(|0\rangle)$. However if the two qubits are entangled this is not possible. Consider the two qubit wavefunction, called the Bell State or EPR state, $\Psi = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. If we were to attempt to separate this wavefunction and write it as two distinct wavefunctions that are simply multiplied together, we would see get $\Psi = (a|0\rangle + b|1\rangle)(c|0\rangle + d|1\rangle)$. However, this would require $a * c = \frac{1}{\sqrt{2}}$, $b * d = \frac{1}{\sqrt{2}}$, $a * d = 0$ and $b * c = 0$ which is clearly not possible. This is what is meant by "inseparable". This has important implications because it means that a single operation on one qubit influences all qubits because their wavefunctions are mathematically inseparable.

2.2.1 Quantum Variational Circuits

Special operations, called quantum gates, are required in order to manipulate qubits. There are a number of quantum gates, but the ones relevant to this work are the Pauli rotation gates and the controlled NOT (CNOT) gate. The Pauli rotations gates, $R_x(\theta), R_y(\theta), R_z(\theta)$, rotate around the specified axis θ radians. Mathematically: $R_\alpha(\theta) = e^{-i\frac{\theta}{2}\sigma_\alpha}$, where $\alpha = X, Y, Z$. The controlled NOT (CNOT) gate is a two qubit gate that induces entanglement in qubits. The CNOT is not parameterized and is used for entanglement purposes only. The aforementioned θ are the learnable parameters that are updated via gradient descent.

Quantum Variational Circuits (QVCs) are a collection of qubits and the set of gates that operate on them [19]. There are three main components of a QVC: an encoding circuit, a parameterized circuit, and a readout circuit [3]. The encoder converts classical data into quantum data (i.e. parameter free quantum circuits). The parameterized circuit operates on the on the quantum data to produce an approximation of the desired state. Finally, a readout measurement is taken, usually one of the Pauli operators (X, Y, Z) is used to extract information from the circuit. In this work we use the Z operator, or the 'computational basis state'. External to the quantum circuit (on a classical computer) a loss function and associated gradients are calculated then the parameters are updated.

The gradients for QVCs cannot be calculated using the same differentiation techniques as traditional neural networks. The parameter shift differentiator is used for this work. This differentiator is implemented as part of TensorFlow-Quantum package. The rotation that a gate enacts on a qubits can be represented in the form: $U_i^\ell(\theta_i^\ell) = e^{-iaG\theta_i^\ell}$ (1), where ℓ is the layer index, a is a constant and G a linear combination of Pauli gates, called a generator [10]. A QVC is a function of θ, and is equivalent to the expectation value of the readout operator ($\hat{Z}$ in this work). This is written as: $f(\theta) = \langle\Psi_0|U^\dagger(\theta)\hat{Z}U(\theta)|\Psi_0\rangle$, where Ψ_0 represents the initial wavefunction [7]. The parameter shift rule states that $\frac{\partial}{\partial\theta}f(\theta) = \langle\Psi_0|(\frac{\partial}{\partial\theta}U^\dagger(\theta))\hat{Z}U(\theta)|\Psi_0\rangle + \langle\Psi_0|U^\dagger(\theta)\hat{Z}(\frac{\partial}{\partial\theta}U(\theta))|\Psi_0\rangle$ (2) [28]. Equations (1) and (2) can be combined to yield a differentiation rule: $\frac{\partial}{\partial\theta}f(\theta) = r[f(\theta + \frac{\pi}{4r}) - f(\theta - \frac{\pi}{4r})]$ [10]. In this formula r is a value that can vary between implementations but is often set in relation to the eigenvalues of the operations e_0, e_1 where $r = \frac{a}{2}(e_1 - e_0)$. Thus in the case of Pauli gates, $r = \frac{1}{2}$ because the eigenvalues of all Pauli matrices are $-1, 1$. Equation (2) is the parameter shift

technique for how to differentiate through a quantum circuit enabling both gradients for the circuit and backpropagation through the circuit.

The idea behind this work is to replace the traditionally hard problem of encoding classical states into quantum circuits with a neural network. Because of the differentiability of the quantum circuit, gradients can be carried through and applied to the neural network encoder. The neural network will output the rotations of gates that transform pure states. This is a significant departure from previous approaches to quantum RL and should enable significantly larger state encoding.

3 Approach

3.1 Methodology

The algorithm used in this work is Double Deep Q Learning (DDQN) [35]. As in other works, the only algorithmic differences are the function approximators, the fundamentals of the algorithm remain the same [18] [8]. The difference in this work are internal to the hybrid model, not the QVCs role in the algorithm. The simplicity of just replacing the neural network with a QVC or hybrid model has been shown to work in simple applications like CartPole [18] and Gridworld [8] environments. The setup of the Atari benchmark also remains unmodified, in that the goal is to maximize the reward achieved and the input is the 4 framestacked 84X84 images which has been cropped and grey scaled.

For the quantum architecture, we use the quantum convolution: a parameterized two qubit unitary, i.e. arbitrary entangled rotation, on every set of adjacent qubits [9]. A single two bit unitary operation is shown in Figure 1. After the quantum convolutional layers, there are 3 layers of the circuit with the same architecture shown in Figure 2, generated via QuanTikz. More qubits can be added to this circuit (which only displays 4) by expanding either the inner set or outer set and using a CNOT gate to induce entanglement with the rest of the circuit. Note that $R_\alpha(\theta)$ rotates about α by θ, but α^θ is that gate raised to the power of θ, where $\alpha = X, Y, Z$.

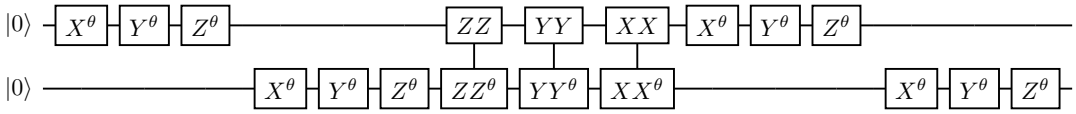

Figure 1: Two Qubit Unitary

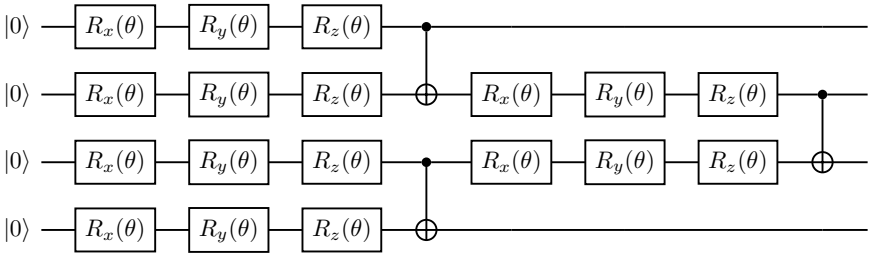

Figure 2: Single Layer of Parameterized Circuit

It is important to note that the concept of a 'layer' is mainly aesthetic in QVCs, the layer operations are not the same as in classical neural networks. In neural networks a layer indicates that there is matrix multiplication of the inputs and weights, a layer in a QVC merely indicates a group of operations, i.e. after you make the circuit you could change all the layer 'cutoffs' and that would not change the mathematical operations of the circuit. This layer architecture is an expansion upon an design that has been empirically shown to be one of the most powerful QVC architectures [32].

Significant improvements have been made in encoding classical data to quantum circuits. However, it falls far short of the necessary efficiency for the large inputs Atari requires, as previously indicated. In this work we explore an approximate solution by using a neural network. We use a classical neural network to convert the classical pixel data into quantum data. Specifically, this network takes

as input the classical data and outputs rotations for the gates in order to establish an approximate encoding. We use the same 3 layers as before to encode the QVC due to their expressibility, i.e. the function space they can learn. Similar to traditional neural networks, for QVCs with large numbers of qubits and many layers, one challenge is the quantum barren plateaus problem [20]. We combat this problem in two ways. First by utilizing QCNN layers which are able to more effectively sidestep barren plateaus in gradients [24]. Second we use an initialization technique, called identity block initialization, specifically designed to combat this problem [13]. In order to match the action space of the environment we use two techniques: quantum pooling and classical dense. The quantum pooling operation consists of Pauli X, Y, Z gates to a parameterized power, as shown in Figure 3. This operation reduces two qubits into just one qubit; the qubit that is pooled out is called the source and the qubit that remains in operation (and is pooled 'to') is called the sink.

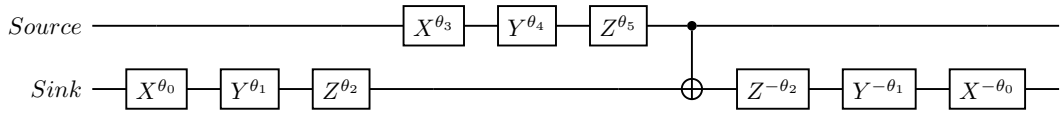

Figure 3: Parameterized Quantum Pooling Operation

The two output techniques are quantum pooling or classical dense neural networks. For quantum pooling we can apply this operation the desired number of times to reduce the number of qubits to the action space, then apply the readout operator and extract the estimated Q value for each action. Or we can apply readout operators to all the qubits without pooling them then feed these results in the a classical dense neural network layer that matches the action space. We investigate and compare both of these options.

As there are many moving parts in this hybrid model, we present a brief overview of what a single forward pass of the model would look like. A 84x84x4 input is fed into a neural network. That neural network outputs rotation parameters for 3 layers of the circuit shown in Figure 2. The goal being that the quantum circuit is now encoded with the pixel information. A quantum convolution is then applied (i.e. the unitary circuit shown in Figure 1 is applied to all neighboring qubit pairs). This serves the same purpose as the traditional convolution operation, specifically for image and spatial analysis. Attached to this is 3 more layers of Figure 2. Finally, either measurements are made for every qubit and fed into a neural network, or quantum pooling operations are applied until the number of active qubits is equal to the action space and measurements are made for the Q values.

The classical architecture we compare the hybrid to is well established and we use a similar architecture as in Mnih et al. [22]. Specifically our architecture has 4 convolution operations with 32, 64, 128, 128 filters followed by a 1024 unit and 512 unit dense layer followed by an output layer the size of the action space. Note that this network has on the order of 10^6 trainable parameters.

3.2 Experimental Protocol

The purpose of this study is to empirically evaluate performance of classical and hybrid quantum-classical DDQN on 2 Atari environments: Pong and Breakout. We chose these environments as they represent two distinct types of game play, with Pong being a multiplayer game in which the agent must learn to play against the default player and Breakout is a single player game, in which the only challenge is the environment (not other players). Quantum techniques have the potential to use less parameters and achieve better policies at a faster rate, as previously stated [18] [8]. We hypothesize that using a neural network to encode classical information into quantum circuits will enable the successful use of QVC's leading to better and faster rewards on these Atari environments.

To this end, we propose a total of 130 experiments, 120 hybrid experiments and 10 classical for comparison. For both environments, we repeat each experiment 5 times for both hybrid and classical networks. This results in 10 experiments using the classical neural network as described in Section 3.1. For the hybrid model, there are 12 different configurations, resulting in 12*10 = 120 experiments. See Table 1 for an outline of all the experiments.

Hybrid Variations	Encoder	Qubits	Output
D5D	Dense	5	Dense
D5Q	Dense	5	Quantum
D10D	Dense	10	Dense
D10Q	Dense	10	Quantum
D15D	Dense	15	Dense
D15Q	Dense	15	Quantum
C5D	Convolution	5	Dense
C5Q	Convolution	5	Quantum
C10D	Convolution	10	Dense
C10Q	Convolution	10	Quantum
C15D	Convolution	15	Dense
C15Q	Convolution	15	Quantum

Table 1: The 12 Hybrid Variations

The goal of proposing these experiments is to thoroughly evaluate the potential of the hybrid quantum classical approach and to eliminate statistical outliers, and alleviate the problems of brittleness common in RL. There are three different aspects of the hybrid model that we experiment with. The first is the encoding scheme. There are two different approaches, classical densely connected neural network layers or classical convolutional layers. Each of these networks will have on the order of 10^4 trainable variables. This results in a network with two orders of magnitude fewer trainable variables than our traditional approach (as the QVC has on the order 10^2 parameters). The input into the classical dense layers will necessarily be flattened. The classical convolutional layers are not intended to do the pixel analysis for the QVC. Because in an entangled system, single rotations can change the overall wavefunction, nearby inputs and spatial relations are important considerations for encoding. It is important to reiterate that this is not a dimensionality reduction strategy. Our goal is not simply to reduce the information to fewer numbers, but rather to have the neural network learn to convert the information into rotations that can represent the information. Specifically, the input to the parameterized circuit will be the circuit shown in Figure 2 with the rotations being the outputs of the neural network, i.e. if there are 4 qubits and 3 layers, then each layer has 18 parameters and there are $3 * 18 = 54$ numbers the neural encoder outputs each one corresponding to one rotation gate. To help elucidate the goal of the neural network encoder, consider a single qubit with a single rotation gate. The one parameter of this gate, θ, can create a state which represents two numbers, e.g. by rotating $\pi/5$ it creates the superposition $\Psi = cos(\frac{\pi}{10})|0\rangle + sin(\frac{\pi}{10})|1\rangle$. The second aspect we vary is the number of qubits, specifically we evaluate using 5, 10 and 15 qubits. The reasoning behind this is straightforward: not all the information present in the pixels is relevant for making informed actions, thus the amount of encoded information may not have to be the full 84 by 84 by 4 array. The computational expense of simulating quantum circuits also exponentially increases with the number of qubits. 5, 10 and 15 qubits have representational power of $2^5, 2^{10}, 2^{15}$ or 32, 1024, and 32,768 respectively. Thus, the 15 qubits are capable of representing the 84*84*4 = 28,224 numbers from the pixels. The third and final aspect is the output of the model. The output can be directly evaluated from the quantum readout operators (after pooling) or a classical dense layer can be attached at the end of the model. This idea has been presented before, but there was limited empirical analysis and little theoretical explanation for the relative performances of the two [18].

Finally, the hyperparameters will be held constant across experiments in order to ensure an accurate comparison. Our current set of hyperparameters are outlined here and although they are subject to small changes and optimizations (as is important in machine learning), whatever is done to optimize hyperparameters will be shared across models. These initial hyperparameters are inspired by those used in [22] combined with modern knowledge of RL hyperparameters [1]. The replay buffer is size 1,000,000 with a minibatch size of 32. For ϵ greedy exploration the initial $\epsilon = 1.0$ with a decay of $\epsilon_{decay} = 0.99$, $\epsilon_{min} = 0.01$ and a reward discount factor of $\gamma = 0.99$. In addition, both models use the Adam optimizer [16] with the same learning rate schedule, starting at 0.001 decaying linearly to 0.0001 over 10,000,000 frames.

With all 130 experiments, this work should provide substantial empirical insight into the use of hybrid quantum classical models for complex reinforcement learning tasks. We hypothesize that the convolutional encoders will perform superior to the dense encoders due to their ability to work with spatial relations, and that all qubit numbers will be able to learn but the best performing will be the 15 qubits because of the ability to represent all the input data (with fewer than 15 qubits, some pixel information is inherently lost), and finally that the quantum output will perform better than the dense. If the qubit encoding performs better with fewer qubits, that demonstrates there is substantial unnecessary information in the input as the fewer qubits can only represent a small fraction of the total input. Therefore, we predict C15Q to perform the best.

References

[1] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters in on-policy reinforcement learning? a large-scale empirical study. *arXiv preprint arXiv:2006.05990*, 2020.

[2] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

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

[4] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

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

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[7] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J Martinez, Jae Hyeon Yoo, Sergei V Isakov, Philip Massey, Murphy Yuezhen Niu, Ramin Halavati, Evan Peters, et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

[8] Samuel Yen-Chi Chen, Chao-Han Huck Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Hsi-Sheng Goan. Variational quantum circuits for deep reinforcement learning. *IEEE Access*, 8:141007–141024, 2020.

[9] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[10] Gavin E Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv preprint arXiv:1905.13311*, 2019.

[11] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1):012324, 2018.

[12] Daoyi Dong, Chunlin Chen, Hanxiong Li, and Tzyh-Jong Tarn. Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5):1207–1220, 2008.

[13] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[14] Wei Hu et al. Empirical analysis of decision making of an ai agent on ibm's 5q quantum computer. *Natural Science*, 10(01):45, 2018.

[15] Sofiene Jerbi, Hendrik Poulsen Nautrup, Lea M Trenkwalder, Hans J Briegel, and Vedran Dunjko. A framework for deep energy-based reinforcement learning with quantum speed-up. *arXiv preprint arXiv:1910.12760*, 2019.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *Physical review letters*, 121(4):040502, 2018.

[18] Owen Lockwood and Mei Si. Reinforcement learning with quantum variational circuit. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 245–251, 2020.

[19] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[20] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):1–6, 2018.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[23] Mikko Mottonen, Juha J Vartiainen, Ville Bergholm, and Martti M Salomaa. Transformation of quantum states using uniformly controlled rotations. *arXiv preprint quant-ph/0407010*, 2004.

[24] Arthur Pesah, M Cerezo, Samson Wang, Tyler Volkoff, Andrew T Sornborger, and Patrick J Coles. Absence of barren plateaus in quantum convolutional neural networks. *arXiv preprint arXiv:2011.02966*, 2020.

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

[26] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

[27] Patrick Rebentrost, Thomas R Bromley, Christian Weedbrook, and Seth Lloyd. Quantum hopfield neural network. *Physical Review A*, 98(4):042308, 2018.

[28] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[29] Vivek V Shende, Stephen S Bullock, and Igor L Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006.

[30] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[31] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[32] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12): 1900070, 2019.

[33] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[34] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[35] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*, 2015.

[36] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[37] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.