

# Differentiable Quantum Architecture Search for Quantum Reinforcement Learning

Yize Sun\*, Yunpu Ma<sup>†</sup>, Volker Tresp<sup>‡</sup>

Ludwig Maximilians University

Siemens AG

Munich, Germany

Email: \*yize.sun@campus.lmu.de, <sup>†</sup> cognitive.yunpu@gmail.com, <sup>‡</sup> volker.tresp@lmu.de

**Abstract**—Differentiable quantum architecture search (DQAS) is a gradient-based framework to design quantum circuits automatically in the NISQ era. It was motivated by such as low fidelity of quantum hardware, low flexibility of circuit architecture, high circuit design cost, barren plateau (BP) problem, and periodicity of weights. People used it to address error mitigation, unitary decomposition, and quantum approximation optimization problems based on fixed datasets. Quantum reinforcement learning (QRL) is a part of quantum machine learning and often has various data. QRL usually uses a manually designed circuit. However, the pre-defined circuit needs more flexibility for different tasks, and the circuit design based on various datasets could become intractable in the case of a large circuit. The problem of whether DQAS can be applied to quantum deep Q-learning with various datasets is still open. The main target of this work is to discover the capability of DQAS to solve quantum deep Q-learning problems. We apply a gradient-based framework DQAS on reinforcement learning tasks and evaluate it in two different environments - cart pole and frozen lake. It contains input- and output weights, progressive search, and other new features. The experiments conclude that DQAS can design quantum circuits automatically and efficiently. The evaluation results show significant outperformance compared to the manually designed circuit. Furthermore, the performance of the automatically created circuit depends on whether the super-circuit learned well during the training process. This work is the first to show that gradient-based quantum architecture search is applicable to QRL tasks.

**Index Terms**—DQAS, QRL, QAS

## I. INTRODUCTION

In recent years, quantum computing has developed rapidly and has achieved remarkable progress [1]. Diverse quantum algorithms were proposed in different fields [2]–[7]. However, current quantum hardware has limitations on the number of qubits and the quantum gates because of the relatively low quantum gate fidelity on the deep quantum circuit [8]. These noisy intermediate-scale quantum (NISQ) devices can not reach fault tolerance in the near future [8]. In the NISQ era, variational quantum algorithms (VQAs) are considered the leading strategy [9]. It has shown the potential to improve ML performance in quantum supervised learning (QSL) [2], [10]–[13], quantum unsupervised learning (QUL) [14]–[17] and quantum reinforcement learning (QRL) [3], [18]–[20] on NISQ devices.

Most studies in QRL focused on the framework with a predefined circuit architecture. These architectures were

inspired by mathematical models and constructed through manual experience. It would take much time to design an architecture from the experts, especially when the number of qubits is increasing and the circuit is getting deeper and deeper. Besides, the lack of flexibility of predefined architecture and the hardware constraints in the NISQ era limit us from solving a QRL task efficiently. Therefore, we aim to solve these issues by automating the architecture search for QRL. This work applies quantum deep Q-learning specifically, which is an off-policy QRL algorithm. It was proposed to solve RL environments (cart pole and frozen lake) with quantum computer [19]. The circuit architecture they used is viewed as the baseline in this work.

Differentiable quantum architecture search (DQAS) is a general gradient-based framework for automating quantum circuit design problems. It has shown its effects on error mitigation and rediscovery of quantum circuits in the NISQ [4], [21]. All DQAS previously solved problems have a specific condition with a fixed dataset, in which the losses are strongly correlated to the objective (e.g., fidelity or ground state).

However, this can not be satisfied in quantum deep Q-learning. The dataset in quantum deep Q-learning strongly depends on the current learned experiences and varies by increasing training epochs. There is no guarantee that a loss decline will lead to good rewards [22]. Here is a question: Would DQAS apply to quantum deep Q-learning?

In this work, we try to apply a gradient-based framework of DQAS to reinforcement learning tasks. It is motivated by the progress toward realizing quantum advantage by utilizing DQAS in QRL. We summarise our contributions as follows:

- 1) We conduct sufficient experiments to evaluate the proposed algorithm in two RL environments based on various super-circuits. We demonstrate the success of newly discovered architectures in both environments.
- 2) We also show that the discovered architecture will have a bad performance by evaluation if the corresponding super-circuit does not learn well during the training process.
- 3) We test the newly found architecture on a simulator of a noisy device.

This work includes five chapters. In chapter I, the motivation and overview of the study are introduced. Chapter II reviews two related works. Chapter III shows our method. The exper-

imental results are analyzed in chapter IV. The last chapter V contains conclusions and directions for future work.

## II. RELATED WORK

### A. DQAS

The quantum architecture search (QAS) constructs a quantum circuit automatically. Its goal can be described as

$$(\theta^*, a^*) = \arg \min_{\theta \in \mathcal{C}, a \in \mathcal{S}} \mathcal{L}(\theta, a, \mathcal{Z}, \epsilon_a) \quad , \quad (1)$$

where  $\mathcal{L}$  represents an objective function,  $\theta$  denotes the parameters of an ansatz  $a$ ,  $\mathcal{C}$  is a constraint set,  $\mathcal{Z}$  represents the input, and  $\epsilon_a$  denotes the quantum system noise.

As one of the QAS algorithms, DQAS is a hybrid quantum framework [4]. It has one super-circuit including an operation pool  $\mathcal{O}$ ,  $p$  placeholders, architecture parameters  $\alpha_p$ , and circuit weights  $\theta$  with dimension  $p \times |\mathcal{O}| \times l$ , where  $l$  is the largest number of gate parameters. It is inspired by differentiable architecture search (DARTS) and designs a circuit by placing multiple operation candidates in a placeholder. Given some placeholders and an operations pool, people usually test all combinations of placeholders and operation candidates individually to get the best-performing architecture. The search space is thus discrete. However, DQAS gives weight to each operation candidate within a placeholder, and people refer to the normalized weights as operation probabilities. The probability of a circuit with a specific architecture is then defined as the product of probabilities of its component's operations. DQAS makes the search space continuous through this probability model. It samples a batch of circuit architecture candidates to calculate the global loss, which is a sum of the loss of each batch member and represented as

$$\mathcal{L} = \sum_{k \sim P(k, \alpha)} \frac{P(k, \alpha)}{\sum_{k' \sim P(k, \alpha)} P(k', \alpha)} L_k(\theta) \quad , \quad (2)$$

where

$$P(k, \alpha) = \prod_i \frac{e^{\alpha_{ij}}}{\sum_k e^{\alpha_{ik}}} \quad , \quad (3)$$

$k$  denotes the sampled architecture from the probability model  $P(k, \alpha)$ . The architecture and circuit parameters are optimized using gradient descent [4] with a global loss function.

In practice, DQAS can compose a quantum circuit for error mitigation and optimization problems [4].

### B. QRL

Similarly to the classical deep Q-learning algorithm, quantum deep Q-learning uses a variational quantum circuit VQC as a counterpart of DQN. VQC here is called Quantum-DQN (QDQN). One observation state is input for the encoding block, and measurement provides output. Trainable parameters of QDQN correspond to the parameters of DQN. There is a replay memory to manage observations. The Q-values are calculated by

$$Q(s, a) = \frac{1}{2} (\langle 0^{\otimes n} | U_\theta^\dagger(s) O_a U_\theta(s) | 0^{\otimes n} \rangle + 1) \quad (4)$$

in range of  $[0, 1]$  for each action  $a$  by corresponding observables  $O_a$ . The local loss function is represented as

$$L_i(\theta_i) = \mathbb{E}_{(s, a, r, s') \sim ER} [ (r + \gamma \max_{a'} Q^{\text{target-QDQN}}(s', a'; \theta_i^-) - Q^{\text{QDQN}}(s, a; \theta_i))^2 ] \quad , \quad (5)$$

where  $\gamma$  denotes the discount factor. The vector  $(s, a, r, s')$  is one of the sampled observations including state  $s$ , action  $a$ , reward  $r$  and next state  $s'$ .  $\theta_i^-$  and  $\theta_i$  denote parameters of target-QDQN and QDQN, respectively. For every  $n$  epochs, the target-QDQN copies  $\theta_i$  from QDQN into itself.

With the help of quantum deep Q-learning, the cart-pole environment and the frozen lake environment were solved in [20]. They also used data re-uploading technique input and output weights to improve training performance. We will refer to these methods in our following work as new features for DQAS.

## III. METHOD

Algorithm 1 shows an overview for RL-DQAS. Firstly, a super-circuit is defined, including a circuit with placeholders, an operation pool  $\mathcal{O}$ , and trainable shared parameters for circuit gates  $\theta$  and architecture  $\alpha$ . Secondly, we start training with multiple agents. According to the current architecture distribution model  $P$ , a batch of architecture candidates is sampled from the super-circuit. Each candidate shares circuit parameters  $\theta$  and calculates individual loss value  $\mathcal{L}$  between the predicted and target Q-values. With a global loss, the shared architecture parameters  $\alpha$  and circuit parameters  $\theta$  can then be updated by gradient descent with optimizer Adam. Meanwhile, the corresponding probability is updated as well. For every  $n$  iterations, the progressive search checks for each placeholder if one operation candidate and its architecture parameter should be removed by comparing the corresponding architecture distribution. The target circuit will periodically copy the circuit values of the shared parameters. If the architecture is fixed, the tuning phase only updates the trained circuit parameters iteratively before the reward converges to a given value. After training, we rank training performances among agents and take the architectures of the first  $K$  top-performing candidates. Finally, we evaluate these architectures to find out the best-performing architecture.

### A. Super-circuit

A super-circuit builds a search space using a circuit with placeholders, architecture parameters  $\alpha$ , and operations from operation pool  $\mathcal{O}$ . The circuit consists of three blocks. The encoding block encodes the weighted input state into a quantum state. The parameterized block contains all placeholders and should be stacked to create a deep circuit. The measurement block is used to provide output value for a classical computer.

The super-circuit adopts the architecture distribution model  $P(U, \alpha)$ , where  $U$  denotes the sequence of placeholders or an architecture candidate. This work defines operation candidates and placeholders differently than the original DQAS. Each

---

**Algorithm 1** DQAS for RL
 

---

**Step 1: Super-circuit definition:**

Initialize op  $\mathcal{O}$  and super-circuit with  $\alpha$  and  $\theta$   
 Initialize two QDQNs and environment  $e$

**Step 2: Super-circuit training:**

**while** Architecture search **do**  
   Sample minibatch of architectures  
   Calculate global loss  $\mathcal{L}$  via Eq: 9  
   Update  $\alpha_t$  and  $\theta_t$  via gradients  $\nabla \mathcal{L}_\alpha$  and  $\nabla \mathcal{L}_\theta$   
   The tar-circuit copy  $\theta$  from pred-circuit periodically  
**end while**  
 Circuit parameter tuning  
 Early stop training if  $e \ni r^{avg} \geq r^{max} \in e$

**Step 3: Rank training performance among agents:**

Take top K architectures for evaluation

**Step 4: Evaluation of architectures:**

Take the best performing architecture or retrain

---

placeholder  $u_i$  covers all qubits instead of one qubit and accepts one element from the operation pool containing the working range. This way, the number of parameters for each placeholder depends on the operation type and working range, and the search space can thus be reduced by controlling the working range of operation candidates. One parameterized block can then be described as

$$U = \prod_{i=0}^p u_i(\theta_i) \quad , \quad (6)$$

where  $u_i$  stands for unitary placeholder and can be replaced by any  $o_i \in \mathcal{O}$ .  $\theta_i$  can vanish if  $o_i$  has no trainable parameter. Furthermore, the depth of the block increases with every filled placeholder.

**B. Operation pool**

Operation pool  $\mathcal{O}$  in size of  $s = |\mathcal{O}|$  is a set of quantum gate candidates. Each operation in  $\mathcal{O}$  contains the type of operation and its working range. If there is a circuit with four qubits, an operation pool can be defined with elements:

$$\mathcal{O} = \left\{ \underbrace{o_1}_{\text{Type}} : \underbrace{[1, 2, 3, 4]}_{\text{Working range}}, o_2 : [1, 2, 3, 4], \right. \quad (7)$$

$$\left. o_3 : [1, 2, 3], o_4 : [2, 3, 4], E : [1, 2, 3, 4] \right\} \quad .$$

The  $o_i$  denotes the type of operation, and it could be any 1-qubit gate or multi-qubits gate (e.g., RZ, U3, or CNOT). The corresponding array shows the range in which the operation works. For example, if operation  $o_1 = \text{CNOT}$  is selected for one placeholder, four CNOT gates will work on all qubits with ring connections. This work defines two operation pools *op3* and *op4*. *op3* contains candidates such as ry, rz, cz, cnot and identity on all qubits, and there are candidates ry and rz both further working on  $\{[1, 2, 3], [2, 3, 4], [1, 2], [2, 3], [3, 4]\}$ . *op4* contains almost the same candidates but has no cz, neither ry nor rz on  $\{[1, 2], [2, 3], [3, 4]\}$ .

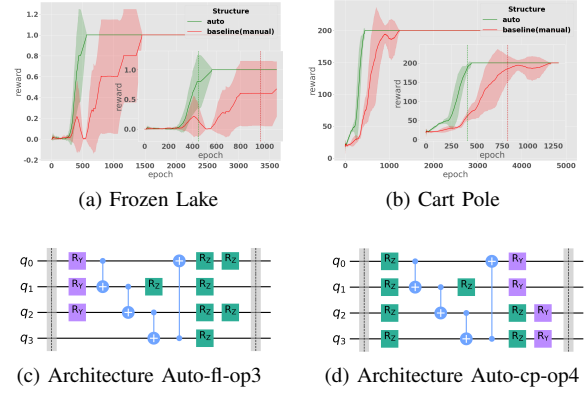


Fig. 1: Evaluation and information of newly found architectures on the simulator without noise. The results are averaged over five agents, where each agent has five parameterized blocks.

**C. Objectives and gradients**

In this work, the objective function takes two Q-values as inputs, created from two quantum circuits (predicting circuit and target circuit). The predicted Q-value should gradually converge to the target Q-value. The local objective  $L(U, \theta)$  is calculated as the mean squared error (MSE) between predicted Q-values and target Q-values

$$L(U, \theta) = (Q_{U, \theta}^{\text{predicted}} - Q_{U, \theta^*}^{\text{target}})^2 \quad , \quad (8)$$

where  $U$  stands for circuit architecture and  $\theta$  represents circuit parameter tensor. The global objective function is defined as the total sum of local objectives according to the architecture distribution model  $P(U, \alpha)$ :

$$\mathcal{L} = \sum_{U \sim P(U, \alpha)} L(U, \theta) \quad . \quad (9)$$

The Q-value for action  $a \in A$  is calculated by the circuit measurement with the corresponding observable  $O_a$ . In order to compare Q-values easily, we shift and scale the measurement result by 1 and  $\frac{1}{2}$  respectively.

We will iteratively update the parameters by gradient descent, including the circuit parameter tensor  $\theta$ , the architecture parameter matrix  $\alpha$ , and the input weights  $w^{in}$ , the output weights  $w^{out}$  if needed. Since the circuit tensor  $\theta$  is independent of the architecture distribution, its gradient takes the form

$$\nabla_{\theta} \mathcal{L} = \sum_{U \sim P(U, \alpha)} \nabla_{\theta} L(U, \theta) \quad . \quad (10)$$

In practice, this gradient can be calculated by the parameter shift rules [12], [23], [24]. The gradient of the architecture parameter matrix  $\alpha$  is related to the architecture distribution and calculated as described in DQAS [4].

## IV. EXPERIMENTS AND RESULTS

### A. Experiment setting

In this chapter, we conduct two benchmark RL experiments (e.g., Cart Pole v0 [25] and Frozen Lake v0 [26]) from OpenAI Gym. This work uses four qubits and repeats one parameterized block five times to create the super-circuit. Each block has four placeholders. We select the same encoding scheme and observables used in quantum deep Q-learning [20] for both experiments, and their manually designed circuit architecture is referred to as our baseline. The baseline circuit has five parameterized blocks, each composed of three operations columns. Each column covers all qubits, followed by  $r_y$ ,  $r_z$ ,  $c_z$ .

### B. Results and discussion

We first focus on the most surprising result shown in Fig. 1. The automatically-designed architecture (green lines) improves the performance by about 200% compared to the baseline, decreasing the average solving point from around 800 to 400 for the cart pole and 1000 to 400 for the frozen lake. Furthermore, the shaded standard deviation of this architecture is much smaller than the baseline, showing the homogeneous training performance of agents.

The best architecture Auto-op4 shown in Fig. 1d derives from the super-circuit `op4` during the training process. By manual design, CNOT gates are placed at a block's front or back end, and people rarely reuse gates. In contrast, the machine places the CNOT gates in the middle of the parameterized block and then chooses to add some additional RZ gates on part of qubits. This design could provide more control for the z-axis on these selected qubits and improve the mapping between agent state and agent action. Although the reason for this selection is unknown, in practice, this design improves the training performance significantly and is hard to mimic.

As shown in Fig 1c, the architecture has redundant gates - RZ gates at the rear, which is distinct from manually designed architectures. These redundant gates might improve the training performance through stacking parameterized blocks.

We now show the performance of discovered architectures and corresponding super-circuits in Fig. 2. While the super-circuit `op4` (dashed green line) learns well during the training process, the super-circuit `op3` (dashed blue line) has a bad performance, which might be caused by instability of RL or bad super-circuit definition. The difference in learning performance between these two super-circuits results in the evaluation difference between corresponding designed architectures. Additionally, the circuit architectures of the five agents do not show convergency, while the training performance converges to the top. Different architectures could perform similarly. The performance and architectural similarity could be discussed in the future. Consequently, the performance of the designed circuit depends on whether the corresponding super-circuit learns well during the training process. A badly performing architecture can be ascribed to the failure of the training process of the super-circuit.

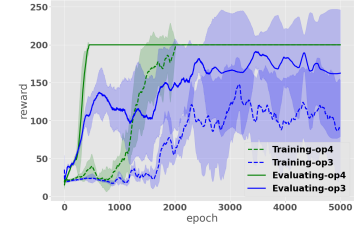


Fig. 2: Relationship between training and evaluation.

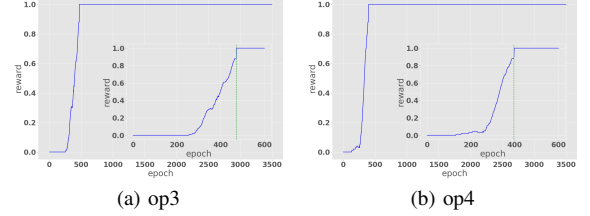


Fig. 3: Evaluation in the frozen lake environment. (IBM noisy simulator qiskit-aer "ibmq-quito").

In addition, as shown in Fig. 3, we investigate whether these two architectures work in the same environment on the noisy device. Due to unpredictable delays of real hardware, we utilize the noisy simulator to evaluate two architectures in the frozen lake environment. We are surprised that the newly discovered circuit architectures performed well on the noise model, although one is designed for another experiment. It will also encourage further investigation into a general architecture based on the gradient-based QAS for multiple QRL tasks. In the next chapter, we will conclude.

## V. CONCLUSION AND OUTLOOK

In this work, we apply DQAS for QRL. The results show that it can design quantum circuit architectures in different RL environments (cart pole and frozen lake) efficiently and automatically. We successfully searched with different super-circuits, and the newly discovered architectures differed from and outperformed the manually designed architecture. Furthermore, we show that the evaluation depends on the training performance. We find a general architecture in the noisy model for the frozen lake. This work is the first to show that gradient-based QAS applies to QRL tasks.

This work uses a shallow super-circuit and mini-learning-step for the training process. However, better architecture might hide in a deep super-circuit. Furthermore, we could find some common circuit architecture features for multiple RL tasks.

### ACKNOWLEDGMENT

The project of this workshop paper is based on was supported with funds from the German Federal Ministry of Education and Research in the funding program Quantum Reinforcement Learning for industrial Applications (QLindA) - under project number 13N15644. The sole responsibility for the paper's contents lies with the authors.

## REFERENCES

- [1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, “Quantum supremacy using a programmable superconducting processor,” *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.
- [2] M. Alam, S. Kundu, R. O. Topaloglu, and S. Ghosh, “Quantum-classical hybrid machine learning for image classification (iccad special session paper),” *arXiv preprint arXiv:2109.02862*, 2021.
- [3] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, “Variational quantum circuits for deep reinforcement learning,” *IEEE Access*, vol. 8, pp. 141007–141024, 2020.
- [4] S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao, “Differentiable quantum architecture search,” *arXiv preprint arXiv:2010.08561*, 2020.
- [5] Y. Ma, V. Tresp, L. Zhao, and Y. Wang, “Variational quantum circuit model for knowledge graph embedding,” *Advanced Quantum Technologies*, vol. 2, no. 7-8, p. 1800078, 2019.
- [6] E. Farhi, J. Goldstone, and S. Gutmann, “A quantum approximate optimization algorithm,” *arXiv preprint arXiv:1411.4028*, 2014.
- [7] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmänn, D. Scheiermann, and R. Wolf, “Training deep quantum neural networks,” *Nature communications*, vol. 11, no. 1, pp. 1–6, 2020.
- [8] J. Preskill, “Quantum computing in the nisq era and beyond,” *Quantum*, vol. 2, p. 79, 2018.
- [9] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, “Variational quantum algorithms,” *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.
- [10] E. Farhi and H. Neven, “Classification with quantum neural networks on near term processors,” *arXiv preprint arXiv:1802.06002*, 2018.
- [11] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, “Supervised learning with quantum-enhanced feature spaces,” *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.
- [12] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, “Quantum circuit learning,” *Physical Review A*, vol. 98, no. 3, p. 032309, 2018.
- [13] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, “Circuit-centric quantum classifiers,” *Physical Review A*, vol. 101, no. 3, p. 032308, 2020.
- [14] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchitsky, and R. Melko, “Quantum boltzmann machine,” *Physical Review X*, vol. 8, no. 2, p. 021050, 2018.
- [15] S. Chakrabarti, Y. Huang, T. Li, S. Feizi, and X. Wu, “Quantum wasserstein generative adversarial networks,” *arXiv preprint arXiv:1911.00111*, 2019.
- [16] B. Coyle, D. Mills, V. Danos, and E. Kashefi, “The born supremacy: quantum advantage and training of an ising born machine,” *npj Quantum Information*, vol. 6, no. 1, pp. 1–11, 2020.
- [17] C. Zoufal, A. Lucchi, and S. Woerner, “Variational quantum boltzmann machines,” *Quantum Machine Intelligence*, vol. 3, no. 1, pp. 1–15, 2021.
- [18] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, H.-S. Goan, and Y.-J. Kao, “Variational quantum reinforcement learning via evolutionary optimization,” *Machine Learning: Science and Technology*, vol. 3, no. 1, p. 015025, 2022.
- [19] S. Jerbi, L. M. Trenkwalder, H. P. Nautrup, H. J. Briegel, and V. Dunjko, “Quantum enhancements for deep reinforcement learning in large spaces,” *PRX Quantum*, vol. 2, no. 1, p. 010328, 2021.
- [20] A. Skolik, S. Jerbi, and V. Dunjko, “Quantum agents in the gym: a variational quantum algorithm for deep q-learning,” *Quantum*, vol. 6, p. 720, 2022.
- [21] S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao, “Neural predictor based quantum architecture search,” *Machine Learning: Science and Technology*, vol. 2, no. 4, p. 045027, 2021.
- [22] Y. Miao, X. Song, J. D. Co-Reyes, D. Peng, S. Yue, E. Brevdo, and A. Faust, “Differentiable architecture search for reinforcement learning,” in *First Conference on Automated Machine Learning (Main Track)*, 2022.
- [23] G. E. Crooks, “Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition,” *arXiv preprint arXiv:1905.13311*, 2019.
- [24] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, “Evaluating analytic gradients on quantum hardware,” *Physical Review A*, vol. 99, no. 3, p. 032331, 2019.
- [25] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym, wiki, cartpole v0.” <https://github.com/openai/gym/wiki/CartPole-v0>, 2016.
- [26] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym, wiki, frozenlake v0.” <https://github.com/openai/gym/wiki/FrozenLake-v0>, 2016.