Quantum Neural Network Design via Quantum Deep Reinforcement Learning

Anca-Ioana Muscalagiu[a]
Babeș-Bolyai University, Cluj-Napoca, Romania

Keywords: Quantum Deep Reinforcement Learning, Neural Architecture Search, Quantum Neural Network, Parameterized Quantum Circuit, Quantum Machine Learning, Variational Quantum Algorithm.

Abstract: Quantum neural networks (QNNs) are a significant advancement at the intersection of quantum computing and artificial intelligence, potentially offering superior performance over classical models. However, designing optimal QNN architectures is challenging due to the necessity for deep quantum mechanics knowledge and the complexity of manual design. To address these challenges, this paper introduces a novel approach to automated QNN design using quantum deep reinforcement learning. Our method extends beyond simple quantum circuits by applying quantum reinforcement learning to design parameterized quantum circuits, integrating them into trainable QNNs. As one of the first methods to autonomously generate optimal QNN architectures using quantum reinforcement learning, we aim to evaluate these architectures on various machine learning datasets to determine their accuracy and effectiveness, moving towards more efficient quantum computing solutions.

1 INTRODUCTION

Neural Architecture Search (NAS) is a key paradigm in deep learning, playing a significant role within the field of Automated Machine Learning (AutoML) (Elsken et al., 2019). The objective of AutoML is to apply effective machine-learning techniques to a variety of problems automatically, reducing reliance on manual engineering and leveraging domain-specific expertise. NAS achieves this by automating the design of optimal neural network architectures for various tasks. This approach not only enables the discovery of new architectures for unexplored problems but also has the potential to optimize existing benchmarks across different domains. By using search algorithms, NAS explores a vast space of potential architectures, aiming to minimize an objective function that measures performance, such as various loss functions. The search iteratively refines itself, improving upon the most promising architectures to discover superior solutions.

Extending this concept to quantum computing, Quantum Neural Network Architecture Search (QNNAS) automates the design of Quantum Neural Networks (QNNs). QNNs integrate classical neural layers with parametrized quantum circuits (PQCs), harnessing quantum mechanics for enhanced data processing capabilities. In QNNAS, the first and the last layers are fixed classical layers tailored to the investigated problem. The input layer transforms the classical data and loads it into a quantum register, while the output layer extracts the expectation values from the qubit register back to their classical encoding. The focus of QNNAS is on optimizing the hidden layers, which are generated through the architecture search algorithm. This involves exploring configurations such as the number of qubits per layer, types of quantum operations, and inter-layer connectivity. By fixing the input and output layers, the algorithm can concentrate on fine-tuning the hidden layers for maximum efficiency and effectiveness.

Automating the design of these networks is crucial due to the complexities involved in designing PQCs, which require extensive quantum mechanics knowledge. Automation simplifies this process, making QNNs more accessible and practical for various applications. The development of optimal QNNs through QNNAS holds promise for significant advancements in quantum machine learning, potentially revolutionizing the field by solving problems previously deemed too complex.

In the following sections, this paper explores Quantum Neural Network Architecture Search (QNNAS), focusing on automating the design of Quantum Neural Networks (QNNs). Firstly, the paper outlines

[a] https://orcid.org/0009-0000-3139-4311

Muscalagiu, A.
Quantum Neural Network Design via Quantum Deep Reinforcement Learning.
DOI: 10.5220/0012997500003837
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Joint Conference on Computational Intelligence (IJCCI 2024)*, pages 560-567
ISBN: 978-989-758-721-4; ISSN: 2184-3236
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

our original contributions and reviews the state of the art, comparing classical computing approaches with the developments brought by quantum algorithms applied in QNNAS across three primary research directions. Afterwards, we provide a section about theoretical foundations, which covers the basic concepts behind our methodology: reinforcement learning, quantum computing, and quantum reinforcement learning (QRL), highlighting differences from classical RL and potential quantum speedups. The investigated approach discusses our modelling of the quantum reinforcement task and the training process of the agent. Furthermore, we present some initial experiments, including the experiemental settings and results, which validate the feasibility of the approach. Finally, the paper concludes with insights and future development directions for our proposed technique.

1.1 Original Contributions

This paper introduces a novel approach to Quantum Neural Architecture Search (QNNAS), a field that has been scarcely explored and predominantly addressed through classical methodologies. Our research distinguishes itself in several ways:

- Novel Approach. To the best of our knowledge, this work is one of the first to apply QRL specifically to optimize Parameterized Quantum Circuits (PQCs). Prior studies have focused on constructing general quantum circuits, using quantum state fidelity as the primary metric. While general circuits can be part of a learning process and are often less costly to implement, they lack the flexibility that PQCs offer. General circuits are typically fixed in structure and do not allow for easy modification of parameters to adapt to new data or learning objectives, making them less suitable for tasks like machine learning. Therefore, in contrast to previous work, we design PQCs through QRL and integrate them into Quantum Neural Networks (QNNs), using the accuracy of these QNNs as a metric. The performance is measured after training the generated architectures on specific datasets in the context of the machine learning problem they were designed for. This represents a significant departure from existing literature of QRL applied in QNNAS.
- Theoretical and Practical Contributions. We not only provide a comprehensive theoretical model of the QRL algorithm tailored for this application but also present a practical implementation. Our proof of concept, which includes a publicly available QRL implementation for QNN circuit construction, successfully generates an architecture for classifying the Iris dataset (Fisher, 1988). This demonstrates the practical viability of our approach.

Our research combines theoretical innovation with practical application, laying a foundation for future investigations and implementations of QRL in optimizing QNN architectures.

2 CURRENT STATE OF THE ART

The current literature on Quantum Neural Architecture Search (QNNAS) is significantly influenced by advancements in Classical Neural Architecture Search (NAS). While NAS has a substantial and well-established corpus of research with over 1000 papers published, QNNAS is still in its early stages, with most studies emerging in the last two years. This slower growth is due to the relatively recent development of Quantum Neural Networks (QNNs). The approaches in QNNAS generally adapt classical NAS techniques to the specific structure and constraints of QNNs, utilizing both discrete and continuous representation methods.

Classical approaches to QNNAS often utilize reinforcement learning (RL) or evolutionary algorithms (EAs) for discrete architecture optimization. Examples include the benchmark RL method (Kuo et al., 2021), which trains an agent to place gates in quantum circuits to minimize circuit length while maintaining fidelity, and EQAS-PQC (Ding and Spector, 2022), an evolutionary algorithm that evolves QNN architectures through genetic operations, evaluating fitness based on QNN fidelity. Continuous representation approaches, such as differentiable architecture search (DARTS), model circuits as directed acyclic graphs (DAGs) and use a weighted sum of primitive operations for differentiable search, achieving high fidelities in QNN architectures for tasks like 2-qubit Bell states and 4-qubit circuits (Zhang et al., 2022).

While classical approaches have made significant contributions to QNNAS, quantum approaches offer the potential for remarkable improvements in performance and efficiency. Among these quantum approaches, QDARTS (Wu et al., 2023), EQNAS (Li et al., 2023) and QRL (Chen, 2023) have emerged as the most recent research directions with promising outcomes in comparison to their classical counterparts.

EQNAS combines the classical EAs applied in QNNAS with quantum computing. It represents quantum neural architectures as a chromosome encoded in a quantum register and employs quantum operations to evolve the population of candidate architectures. As far as we know, this is the only study in the current literature that applies the quantum circuits to machine learning tasks, achieving notable results. Specifically, their algorithm improves the accuracy of QNNs by up to 5% and significantly reduces the number of parameters compared to the original QNN. Our study similarly utilizes these evaluation metrics. In contrast to their method, which solely optimizes known architectures, our approach extends to both optimizing existing architectures and generating entirely new ones for previously unexplored problems in quantum machine learning.

Another research direction is QDARTS, a quantum version of the classical DARTS algorithm, represents QNNs as directed acyclic graphs (DAGs) and uses quantum operations to optimize the computations. One of the tasks on which the authors applied QNNAS was image classification. In this experiment, the optimized QuantumDARTS model was compared to benchmark QCNN and CNN models from the literature, on the MNIST dataset. QuantumDARTS consistently outperformed both, demonstrating superior accuracy and cost-efficiency with fewer parameters, its performance reminaing robust under circuit noise conditions.

The last of the novel methods applied in QNNAS that has shown potential is quantum reinforcement learning (QRL) (Chen, 2023). This approach is based on the modeling of the Classical Reinforcement Learning Framework for QNNAS (Kuo et al., 2021), aiming to further optimize it through the introduction of quantum operations. In this approach, the agent's role is to design a quantum circuit that achieves the desired outcome. The environment represents the quantum circuit itself, and the agent interacts with it by selecting quantum gates and placing them at specific locations within the circuit. Our method aims to continue to pave the path for this particular research direction, which is very scarcely and least explored among all three. We focus on studying the automated optimization and design of actual parameterized quantum circuits (PQCs) trained on machine learning tasks, not only of general circuits. Unlike previous research, which primarily used fidelity as a metric, we use accuracy as our metric to evaluate their performance on the learning task. Furthermore, we integrate much more complex circuits and adopt a different modeling of the reinforcement learning problem. While previous approaches generated circuits with two or at most three qubits, our method aims to significantly enhance the efficiency and effectiveness of QRL in optimizing quantum neural network architectures by using larger and more complex circuits.

3 THEORETICAL BASIS

In this chapter, we present the theoretical foundations necessary for defining our methodology and modeling of the problem, covering essential concepts in both reinforcement learning and quantum computing.

3.1 Deep Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm where an agent interacts with an environment $\mathcal{E}$ to maximize cumulative rewards. The agent perceives the state $s \in \mathcal{S}$ of the environment and takes actions $a \in \mathcal{A}$, leading to state transitions and receiving rewards r. The agent's behavior is defined by a policy $\pi : \mathcal{S} \to \mathcal{A}$, aiming to find an optimal policy π^* that maximizes the expected return:

$$\pi^* = \arg\max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid \pi\right], \tag{1}$$

where $\gamma \in [0,1]$ is the discount factor. Q-Learning is an off-policy RL algorithm that learns the optimal action-value function $Q^*(s,a)$, representing the expected utility of taking action a in state s, which is initially set to arbitrary values, such as zero or small random values, to indicate that the utility of these actions is unknown. The Q-value update rule is:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\Big[r_{t+1} + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\Big], \tag{2}$$

where α is the learning rate. Deep Q-Learning uses a neural network, the Deep Q-Network (DQN), to approximate the Q-value function. The network parameters θ are optimized to minimize the loss:

$$L(\theta) = E_{(s,a,r,s')\sim\mathcal{D}}\Big[\Big(r + \gamma \max_{a'} Q(s',a';\theta) - Q(s,a;\theta)\Big)^2\Big], \tag{3}$$

where $\mathcal{D}$ is the experience replay buffer storing tuples (s,a,r,s').

3.2 Quantum Computing

In this section, the fundamental concepts of quantum computing essential for understanding Quantum Neural Networks (QNNs) are introduced. Qubits, analogous to classical bits, are the basic units of quantum information. Unlike classical bits, a qubit can exist in a superposition state, expressed in bra-ket notation as:

$$|s\rangle = \alpha|0\rangle + \beta|1\rangle, \text{ with } \alpha,\beta \in C, \text{ and } |\alpha|^2 + |\beta|^2 = 1 \tag{4}$$

Here, $|0\rangle$ and $|1\rangle$ are the basis states. During measurement, a qubit collapses to one of these states with probabilities $|\alpha|^2$ and $|\beta|^2$, respectively. Quantum gates, represented by unitary operators U, modify qubit states. This concept extends to multi-qubit systems, typically initialized in the computational basis state $|0\rangle^{\otimes n}$.

3.3 Parameterized Quantum Circuits. Quantum Neural Networks

Parameterized Quantum Circuits (PQCs) are quantum circuits with gates that depend on a set of trainable parameters, similar to the weights in an Artificial Neural Network. These circuits are used in variational algorithms where the parameters are optimized to minimize a cost function. Quantum Neural Networks (QNNs) use PQCs to model and learn from data, consisting of an input layer, a series of parameterized quantum gates (similar to hidden layers in classical neural networks), and an output measurement. During training, the parameters of the quantum gates are adjusted to minimize a loss function, akin to weight updates in classical neural networks. An example of a PQC is displayed in Figure 1:

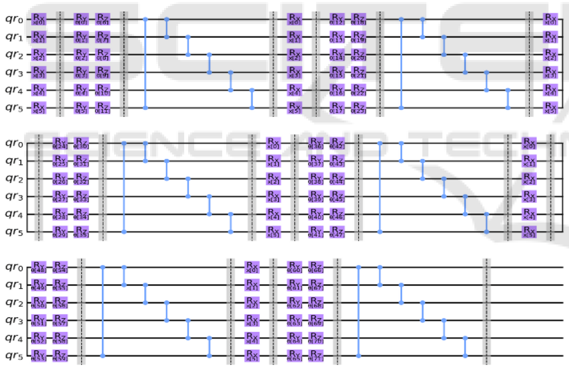

Figure 1: Example of a PQC architecture.

A Quantum Neural Network is composed of several essential components, which include:

- Encoding Layer. Prepares input data of any data type for quantum processing by encoding or transforming it into a suitable numeric representation through classical operations. This layer is crucial for data preprocessing, enhancing expressibility, and preserving information.
- Parameterized Quantum Circuit. Consists of variable quantum gates with trainable parameters. It includes the encoding circuit, which encodes classical data into a quantum register using R_x rotations based on input features, and the parameterized circuit, comprising repeated $R_x(\theta)$, $R_y(\theta)$ or $R_z(\theta)$ rotations and an entangling scheme with CX, CY or CZ gates. The parameters θ represent the trainable weights which get updated during training.
- Post Processing Layer. Receives the probability vector from the quantum circuit, containing the probabilities of measuring states $|i_1..i_n\rangle$, and computes expectation values for possible outputs.

The learning process in a Quantum Neural Network (QNN) involves adjusting the parameters θ of the quantum gates to minimize a loss function $L(\theta)$. The parameter update rule can be expressed as:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t) \tag{5}$$

where θ_t represents the parameters at iteration t, η is the learning rate, and $\nabla_\theta L(\theta_t)$ is the gradient of the loss function with respect to the parameters.

3.4 Quantum Reinforcement Learning

Quantum Reinforcement Learning (QRL) leverages quantum computing to enhance Reinforcement Learning (RL) algorithms by exploiting superposition, entanglement, and quantum parallelism. In QRL, the classical neural network in the Deep Q-Network (DQN) algorithm is replaced with a Quantum Neural Network to approximate Q-values (Du et al., 2020).

The QRL framework retains essential components of classical Q-learning: a target network, epsilon-greedy policy, and experience replay. In this paper, we employ the Double Q-Learning Algorithm (Van Hasselt et al., 2016), which addresses overestimation bias by using two sets of Q-values: the policy network for action selection and the target network for value estimation. The Q-network PQC, $U_\theta(s)$, is parameterized by θ, with the target network $U_{\theta_\delta}(s)$ being an episodic snapshot of $U_\theta(s)$. This method reduces bias, leading to more accurate action value estimations and improved performance in reinforcement learning tasks.

4 INVESTIGATED APPROACH

In this section, we outline the methodology adopted in our study, encompassing the detailed modeling of the reinforcement learning task and the subsequent training process of the quantum agent.

4.1 Modelling of the Reinforcement Learning Task

In this paper, we propose the following modelling for the reinforcement task: A state in the environment is defined as a parameterized quantum circuit (PQC), reflecting the current configuration of quantum gates. Subsequently, an action involves the placement of a quantum gate within the circuit architecture. The agent selects actions that determine the type of the gate to be added to the PQC. Afterwards, the reward is measured through the accuracy of the generated QNN when assessed on a specific dataset, which acts as the primary performance metric for the optimization process.

Our method allows the agent to incrementally build a PQC, adding one layer at a time. The agent can position the following types of quantum gates on these layers:

- Parameterizable Gates. These gates are essential in parameterized quantum circuits, functioning similarly to weights in artificial neural networks. The specific gates used include parameterizable rotation gates such as $R_X(\theta)$, $R_Y(\theta)$, and $R_Z(\theta)$. Each gate performs a rotation around a specific axis on the Bloch sphere with angle θ.
- Entanglement Gates. A few of the gates in this category are CX, CY, and CZ, which are responsible for creating quantum entanglement between qubits.
- Standard Gates. Additional standard gates like RX, RY, and RZ may be used to perform basic quantum operations.

Figure 2 illustrates an example of a state in the environment, depicting the architecture of a parameterized quantum circuit (PQC) after the agent has placed the gates on the layers.

The central part of the figure represents the architecture of the PQC, which is integrated with the input and output layers to form the actual QNN. The input layer can handle various types of data, such as text for supervised learning tasks or states for reinforcement learning problems. Depending on the specific machine learning task, the output layer can generate different metrics, including probabilities for each class in classification problems, Q-values for reinforcement learning, or other relevant outputs.

At each step of the process, the current state, which represents the architecture of a parameterized quantum circuit (PQC), is incorporated into the quantum neural network (QNN) to generate a new child network. This child network is then trained on a custom dataset tailored to the specific task. For supervised learning problems, the performance of the child network is evaluated by assessing its accuracy on the dataset. This accuracy metric serves as the reward, indicating how well the generated architecture handles the given problem.

On the other hand, in reinforcement learning tasks, the child network acts as an estimator for the agent within the environment. The agent's actions are based on the current PQC architecture and involve estimating the values of these actions using the generated PQC. The training involves playing the actions within the environment, which provides feedback in the form of rewards. Here, the performance consists of the best reward obtained by the agent in the subproblem, guiding the agent's learning process. This iterative cycle of generating, training, and evaluating the child network continues until the end of the episode, with the goal of refining the PQC architecture to optimize performance for the given task, whether it is a supervised learning or reinforcement learning problem.

At the end of an episode, which consists of a series of steps, the environment is reset with an empty architecture, prompting the agent to place gates anew from the beginning. An episode concludes after a fixed number of steps, ideally set to less than 10. This limi-

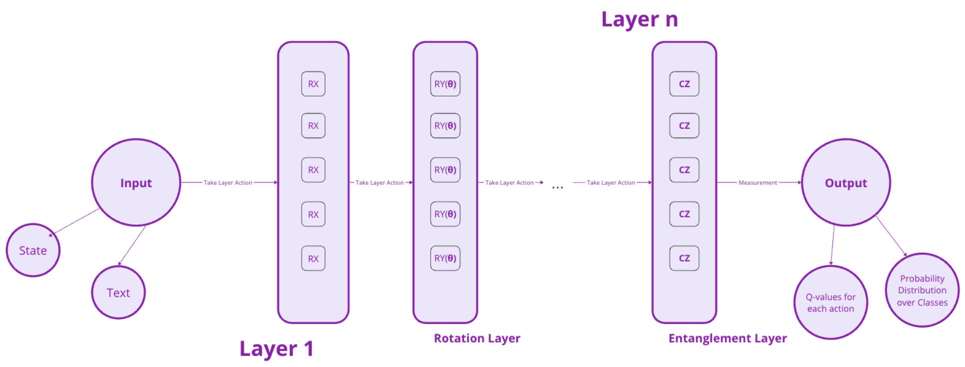

Figure 2: Overview of the Generated Circuit from the Environment State.

tation is intentional, aiming to create simple architectures that are less prone to overfitting. The primary goal of Quantum Neural Architecture Search (QNNAS) is not only to achieve high accuracy but also to minimize the number of layers and parameters requiring training. Despite the fixed number of steps per episode, the agent has the flexibility to skip placing a gate on a layer, which leads to generating architectures with a variable length.

4.2 Agent Training Process

In order to reach an optimal policy that generates high performance QNNs, the agent must undergo training for multiple episodes within the environment. An overview of the entire training process is displayed in Figure 3. During each episode, the agent operates within an environment where it performs actions, specifically choosing and placing quantum gates in the PQC. Each of these actions, along with the resulting rewards, are systematically stored in the Experience Replay Buffer. This buffer plays a crucial role by maintaining a diverse set of experiences that are utilized as input during training, thereby preventing the agent from overfitting to recent experiences and enhancing its ability to generalize across different states.

The actual quantum agent is represented through the Controller Network Trainer, which utilizes the experiences stored in the Experience Replay Buffer to compute optimal action values. It achieves this by estimating the current Q-values, which quantify the expected cumulative reward of executing specific actions in given states. These Q-values are critical as they guide the agent in understanding the long-term benefits of its actions. The Policy Network, which is the core decision-making component of the agent, selects actions based on its current Q-value estimates. This network's parameters are updated by minimizing the loss function derived from the computed Q-values, thereby improving its policy and enabling the agent to make more informed decisions about which quantum gates to place in the PQC.

To ensure the stability and consistency of the training process, the Target Network provides stable Q-value targets. This network is periodically updated to match the Policy Network's parameters. This periodic update is essential for mitigating oscillations and preventing divergence that can occur during training. The Target Network offers a stable reference for the Policy Network, facilitating steady learning progression. The interaction between the Policy Network and the Target Network ensures that the agent's learning process remains robust and reliable over time.

Overall, this training process equips the quantum agent with the capability to effectively learn and refine the architecture of PQCs. By optimizing the placement of quantum gates, the agent aims to enhance the performance of quantum neural networks (QNNs), achieving a balance between high accuracy and minimal complexity. The ultimate objective is not only to improve the accuracy of QNNs but also to minimize the number of layers and parameters, thereby reducing the risk of overfitting and enhancing the generalizability of the quantum models.

5 INITIAL EXPERIMENTS

To evaluate the effectiveness of our proposed Quantum Neural Architecture Search (QNNAS) approach, we conducted some initial experiments to create a proof of concept for the proposed method. The experiment focused on generating architectures to classify irises using the well-known Iris dataset. These experiments aimed to assess the capability of our reinforcement learning-based framework in generating efficient parameterized quantum circuits (PQCs) tailored for specific machine learning tasks.

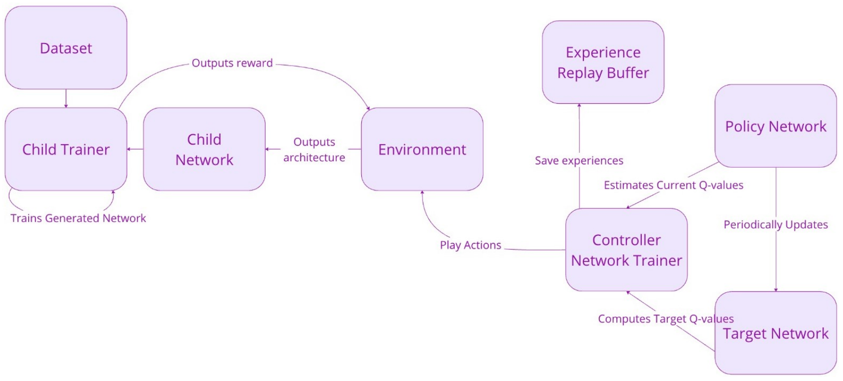

Figure 3: Overview of the Training Process in our Quantum Reinforcement Learning Approach.

5.1 Experimental Setup

The initial experiments were conducted under specific reinforcement learning hyperparameters and an optimized architecture for the controller network. The maximum length of the architecture, which represents the state, was set to 4, based on the difficulty of the problem. Given that the Iris classification task should not require overly complex architectures, limiting the length to 4 layers ensures that the generated parameterized quantum circuits (PQCs) remain manageable in both size and complexity. While playing the environment, the agent had a set of possible gates to choose from, including rotation gates $R_X(\theta)$, $R_Y(\theta)$, $R_Z(\theta)$, and entanglement gates CX, CY, CZ. These gates provide the necessary operations to create and manipulate quantum states effectively.

The reinforcement learning framework was configured with a discount rate of 0.99, which ensures that the agent values future rewards almost as much as immediate rewards, promoting long-term strategy over short-term gains. The learning rate was set to 1 $\times$ 10^{-4}, allowing the agent to update its policy gradually and avoid drastic changes that could destabilize the learning process.

The architecture of the controller network was carefully designed to facilitate effective decision-making. It included an initial linear layer that mapped the input features (4 dimensions) to a higher dimensional space (16 dimensions), followed by a ReLU activation function to introduce non-linearity. The next layer was a quantum layer incorporating $R_X(\theta)$, $R_Y(\theta)$ and $R_Z(\theta)$ rotations operating on a 16-qubit register, which processed the quantum states and produced the observation values, from which the Q-values for each of the seven possible actions were extracted. This design ensures that the controller network can effectively process input data and determine the optimal quantum gates to place within the PQC.

5.2 Experimental Results

As previously mentioned, the initial experiments focused on the classification problem of Iris flowers, categorized into three different classes. The dataset for this experiment was the Iris dataset, comprising 150 samples. The input features, which included the length and width of the sepals and petals measured in centimeters, determined the number of qubits in the generated parameterized quantum circuit.

The agent was tasked with generating a PQC architecture from scratch to achieve a high classification accuracy. The resulting architecture, as shown in Figure 4, consisted of a combination of rotation and controlled gates. Specifically, it included layers of $R_X(\theta)$ and $R_Y(\theta)$ gates applied to individual qubits, followed by entanglement created through CY gates.

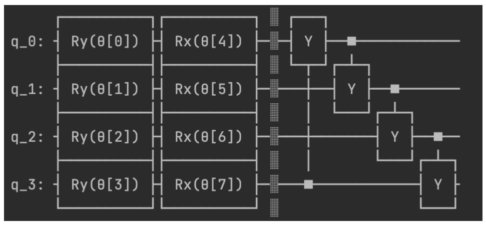

Figure 4: Generated Architecture.

5.2.1 Results Analysis

The generated PQC was trained using the custom dataset, and its performance was evaluated based on the classification accuracy. The best accuracy attained by the generated architecture was 70%. This result serves as a proof of concept, demonstrating the feasibility of our reinforcement learning-based approach to effectively design quantum circuits for machine learning tasks. However, it is important to note that further analysis and optimization of the best architectures for the controller network are necessary to enhance performance.

The experimental setup and results indicate that the agent was able to learn and generate PQC architectures that are capable of performing the given task with a reasonable degree of accuracy. The architecture generated was relatively simple, with a limited number of layers and parameters, aligning with our objective of minimizing complexity to avoid overfitting.

These initial experiments demonstrate the feasibility of our approach and set the stage for further optimization of the QNNAS framework. While they highlight the potential of QRL in generating effective PQC architectures, it is important to note that comparisons were not possible for the Iris classification task, as prior work focused mainly on generating Quantum Convolutional Neural Networks (QCNNs). These initial experiments serve as a proof of concept, and in future work, we aim to generate QCNNs using the proposed method in order to compare with quantum evolutionary approaches.

6 FURTHER ADVANCEMENTS

As previously mentioned, this paper introduced a novel approach to Quantum Neural Architecture Search using Quantum Reinforcement Learning. In future work, we will conduct extensive experiments across various machine learning tasks and architectures to better understand and optimize our approach.

The set of problems for which we aim to automatically design QNN architectures encompasses both reinforcement learning and supervised learning tasks. In the context of reinforcement learning, we plan to focus on well-known environments such as the CartPole environment and Frozen Lake. These environments will allow us to test the capabilities of our QNNAS framework in generating effective QNNs that can learn optimal policies and adapt to their conditions. For supervised learning tasks, our experiments will focus on classification problems using established datasets, specifically the Iris and MNIST datasets. We will continue our work with the Iris dataset, aiming to improve the performance of our generated architectures. On the other hand, the MNIST dataset will help us evaluate the robustness and generalization capability of our approach in generating QCNNs and compare our results with other quantum approaches.

By conducting these experiments across both reinforcement learning and supervised learning domains, we intend to compare the performance of our generated QNNs against benchmark architectures in the state of the art of Quantum Machine Learning.

7 CONCLUSION

In this paper, we have presented a novel approach to Quantum Neural Architecture Search (QNNAS) using Quantum Reinforcement Learning (QRL). Our proposed method automates the design of Quantum Neural Networks (QNNs) by generating parameterized quantum circuits (PQCs) tailored to specific machine learning tasks. This proof of concept has demonstrated the feasibility and potential of our approach through initial experiments, particularly in the context of classifying the Iris dataset. The experimental results underscore the capability of our reinforcement learning-based framework to generate efficient PQCs, paving the way for future work that will involve a broader range of machine learning tasks and a variety of controller network architectures.

8 CODE AVAILABILITY

The implementation of our approach is publicly available at https://github.com/915-Muscalagiu-AncaIoana/QNNAS-Implementation.

ACKNOWLEDGEMENTS

The work presented became real with the help of our supportive professors. Therefore we would like to express our gratitude to Associate Professor Maria Iuliana Bocicor.

REFERENCES

Chen, S. Y.-C. (2023). Quantum reinforcement learning for quantum architecture search. In *Proceedings of the 2023 International Workshop on Quantum Classical Cooperative*, pages 17–20.

Ding, L. and Spector, L. (2022). Evolutionary quantum architecture search for parametrized quantum circuits. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 2190–2195.

Du, Y., Hsieh, M.-H., Liu, T., and Tao, D. (2020). Expressive power of parametrized quantum circuits. *Physical Review Research*, 2(3):033125.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017.

Fisher, R. A. (1988). Iris. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56C76.

Kuo, E.-J., Fang, Y.-L. L., and Chen, S. Y.-C. (2021). Quantum architecture search via deep reinforcement learning. *arXiv preprint arXiv:2104.07715*.

Li, Y., Liu, R., Hao, X., Shang, R., Zhao, P., and Jiao, L. (2023). Eqnas: Evolutionary quantum neural architecture search for image classification. *Neural Networks*, 168:471–483.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Wu, W., Yan, G., Lu, X., Pan, K., and Yan, J. (2023). QuantumDARTS: Differentiable quantum architecture search for variational quantum algorithms. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 37745–37764. PMLR.

Zhang, S.-X., Hsieh, C.-Y., Zhang, S., and Yao, H. (2022). Differentiable quantum architecture search. *Quantum Science and Technology*, 7(4):045023.