Toward Quantum Reinforcement Learning for Automated Neural Architecture Optimization in Vision-Based Structural Health Monitoring

Álvaro Romero Mato[1, 2], Boyang Chen[2], Nathan D. Eskue[1, 2], Vahid Yaghoubi[1, 2]
[1]Q-VAIbe group, Aerospace Structures and Materials, Faculty of Aerospace Engineering, Delft University of Technology, Delft, Netherland
[2]Aerospace Structures and Materials, Faculty of Aerospace Engineering, Delft University of Technology, Delft, Netherlands

Abstract Neural Architecture Search (NAS) involves exploring large and combinatorial search spaces, for which numerous approaches have been proposed, including reinforcement learning, evolutionary algorithms, Bayesian optimization, and gradient-based methods among others. Despite these advances, efficient exploration remains a key challenge, particularly as the search space scales combinatorially. Motivated by prior work based on Q-learning strategies, which are limited in large spaces, this work investigates a Quantum Reinforcement Learning (QRL) framework based on probabilistic policy optimization.

We propose a QRL approach where a parameterized quantum circuit represents a policy over candidate architectures, and is trained to maximize the expected reward defined by validation accuracy. This method is compared against two baselines: Random Search and a Classical Reinforcement Learning approach using policy gradient with a Bernoulli distribution.

All methods are evaluated under identical budgets using the NATS-Bench benchmark, enabling efficient and controlled comparison without full model training. The results demonstrate that both QRL and Classical RL learn meaningful search policies, consistently outperforming Random Search. QRL achieves competitive performance with respect to Classical RL, confirming its ability to effectively explore the architecture space.

We further analyze the impact of key hyperparameters to identify practical trade-offs between accuracy, model complexity, and computational cost.

Overall, the results demonstrate that QRL is a viable approach for discrete architecture search, capable of learning effective policies in large combinatorial spaces, while indicating clear directions for improving its efficiency and scalability.

Keywords: Quantum Reinforcement Learning; Neural Architecture Search; Variational Quantum Circuits, Reinforcement Learning

https://doi.org/10.58286/33912

1. Introduction

Structural Health Monitoring (SHM) increasingly relies on deep learning techniques for tasks such as damage detection, defect classification, and condition assessment from images, vibration measurements, and sensor data [1]. However, the performance of these systems depends strongly on the underlying neural network architecture. Identifying suitable architectures remains a challenging and computationally expensive task, particularly in SHM, where publicly available datasets are often limited in size and diversity. This makes large-scale Neural Architecture Search (NAS) studies difficult, as evaluating thousands of candidate architectures directly on SHM datasets is often impractical. To address this challenge, benchmark search spaces provide a controlled and reproducible environment in which optimization algorithms can be studied independently of application-specific datasets. This allows the search strategy itself to be evaluated before deployment on real-world SHM problems.

Neural Architecture Search (NAS) aims to automate the design of neural network architectures by exploring a predefined search space of candidate models. Despite its success, NAS remains challenging due to the combinatorial growth of the search space.

A wide range of strategies have been proposed to address this challenge, including reinforcement learning, evolutionary algorithms, Bayesian optimization, and gradient-based methods [2]. While effective in many settings, these approaches face scalability issues as the size of the search space increases. In particular, reinforcement learning methods based on Q-learning require explicit representations of state–action values, which become impractical in large combinatorial spaces [1].

An alternative perspective is to directly learn how to sample architectures. Instead of evaluating all possible configurations, a policy defines a probability distribution over candidate solutions and is trained to favor high-performing regions of the search space. In practice, however, such approaches often rely on simplifying assumptions to remain tractable. For example, classical policies commonly assume independence between decision variables, modeling the distribution as a product of simple factors. While this makes learning feasible, it limits the ability to capture dependencies between architectural components.

Motivated by these limitations, this work investigates a Quantum Reinforcement Learning (QRL) framework for discrete architecture search. In this approach, a parameterized quantum circuit defines a policy over candidate architectures. Unlike classical factorized models, the quantum representation encodes a joint probability distribution, allowing correlations between variables to emerge naturally through the structure of the circuit.

We evaluate the proposed approach against two baselines: Random Search and a Classical Reinforcement Learning method based on policy gradients. Experiments are conducted using the NATS-Bench benchmark [3], which provides a controlled and computationally efficient setting for comparing NAS algorithms without the need for full model training.

Overall, this work provides an initial step toward quantum-enhanced search strategies for large combinatorial problems, demonstrating that QRL can learn effective sampling policies while highlighting the trade-offs inherent to current implementations.

2. Methodology

2.1 Problem Formulation

NAS is formulated as the problem of learning a probability distribution over a discrete set of candidate architectures. Let $z \in \{0, 1\}^n$ denote a binary encoding of an architecture.
The objective is to learn a policy $\pi_\theta(z)$, parameterized by θ, that maximizes the expected reward:

$$J(\theta) = \mathbb{E}_{z\sim\pi_\theta}[R(z)]$$

where $R(z)$ corresponds to the validation accuracy of the sampled architecture.
The use of validation accuracy as reward follows standard model selection practice: it provides an estimate of generalization performance during optimization, while avoiding overfitting to the training data and preventing information leakage from the test set.
The goal is therefore to iteratively update π_θ such that high-performing architectures are sampled with increasing probability.

2.2 Search Space and Encoding

We consider the NATS-Bench search space [3], which defines neural architectures using a cell-based directed acyclic graph (DAG). Each cell consists of a fixed topology, where edges represent operations selected from a predefined set. In this work, each cell contains 6 edges and 5 candidate operations per edge, resulting in:

$$|\mathcal{Z}| = 5^6 = 15625$$

possible architectures.
To enable optimization, architectures are encoded as binary vectors. Since each edge requires 5 possible values, at least 3 bits are needed (as $2^2 < 5 < 2^3$). Therefore, the full encoding uses:

$$n = 6 \times 3 = 18 \text{ bits}$$

Each group of 3 bits is mapped to an operation index using:

$$\text{op} = (\text{binary value}) \mod 5$$

Since $2^3 = 8 > 5$, this mapping introduces redundancy, leading to a non-uniform representation where some operations are sampled more frequently. This bias is accepted in exchange for a compact and differentiable encoding.

All experiments are conducted on the CIFAR-100 dataset provided within NATS-Bench. CIFAR-100 is selected because it presents a broader distribution of validation accuracies than CIFAR-10, making the search problem more discriminative and reducing the likelihood of obtaining strong results through random sampling alone.

Although the long-term objective of this research is the application of QRL to NAS for SHM, the present study focuses on evaluating the optimization behavior of the proposed framework under controlled conditions. For this reason, experiments are conducted using the NATS-Bench benchmark [3] and its associated CIFAR-100 search space. The use of a benchmark repository eliminates the need to repeatedly train neural networks, enabling reproducible comparisons between optimization strategies while isolating the performance of the search algorithm itself. Once the behavior of the proposed QRL framework is understood in this controlled setting, the methodology can be extended to SHM-specific datasets and applications.

2.3 Classical Policy-Based Reinforcement Learning

A direct modeling of the full joint distribution over all 2^n architectures would require an exponential number of parameters:

$$|\mathcal{Z}| = 2^n = 2^{18}$$

which is computationally intractable.

To address this, we adopt a factorized policy model assuming independence between decision variables:

$$\pi_\theta(z) = \prod_{j=1}^{n} p_j^{z_j}(1-p_j)^{1-z_j}$$

where $p_j \in [0,1]$ denotes the probability of the j-th bit being equal to 1. This reduces the number of parameters from exponential to linear:

$$2^n \;\rightarrow\; n$$

making optimization feasible.

The probabilities are parameterized via continuous variables θ_j using a sigmoid transformation:

$$p_j = \sigma(\theta_j)$$

and optimized using policy gradient methods to maximize $J(\theta)$.

While efficient, this formulation assumes independence between variables, preventing the model from capturing interactions between architectural decisions.

2.4 Quantum Policy via Variational Quantum Circuits

To overcome the limitations of factorized models, we employ a Quantum Reinforcement Learning (QRL) policy based on a parameterized quantum circuit.

The policy is defined by the quantum state

$$|\psi(\theta)\rangle = U(\theta)\,|0\rangle^{\otimes n}$$

which admits the expansion

$$|\psi(\theta)\rangle = \sum_{z\in\{0,1\}^n} \alpha_z(\theta)\,|z\rangle, \quad P(z) = |\alpha_z(\theta)|^2.$$

Sampling is performed via measurement in the computational basis, yielding bitstrings $z \sim P(z)$. Unlike classical factorized policies, this formulation represents a full joint distribution, where correlations are naturally captured through entanglement. The distribution is not stored explicitly but generated implicitly by the circuit, allowing the number of parameters to scale with the circuit size rather than the search space.

The parameters θ are optimized to maximize

$$J(\theta) = \mathbb{E}_{z\sim P(z)}[R(z)],$$

using gradient-based methods. Gradients are computed via the parameter-shift rule [4], which for gates of the form $U(\theta) = e^{i\theta G}$ with eigenvalues ± 1 reduces to

$$\frac{d}{d\theta}E(\theta) = \frac{1}{2}\left[E\left(\theta + \frac{\pi}{2}\right) - E\left(\theta - \frac{\pi}{2}\right)\right].$$

Parameters are updated via gradient ascent:

$$\theta \leftarrow \theta + \eta \nabla_\theta J(\theta).$$

We use the EfficientSU2 ansatz with alternating single-qubit rotations and linear entanglement. The circuit depth controls expressivity.

This approach enables compact policies capable of modeling dependencies between architectural components.

3. Experimental Setup

All methods are evaluated under identical computational budgets on the same hardware platform to ensure fair comparison. Experiments are conducted on a Dell laptop equipped with a 13th Gen Intel Core i7-1365U processor and 16 GB of RAM. All computations are performed on CPU, without GPU acceleration. Experiments are carried out using the NATS-Bench benchmark [3], which enables efficient evaluation of architectures without full model training.

The implementation is developed in Python using a Qiskit-based framework. All quantum circuits are executed via classical simulation, with the option to include noise models to approximate real hardware behavior. At each optimization step, architectures are sampled from the policy and evaluated using their validation accuracy, which serves as the reward signal. The key hyperparameters are defined as follows:

- *Number of training steps:* number of policy update iterations. At each step, a new batch of architectures is sampled and used to update the policy.
- *Number of samples (shots) per step:* number of architectures sampled from the policy at each step. In the quantum setting, each circuit execution produces a single bitstring, so multiple shots are required to approximate the underlying distribution. The same sampling procedure is used for the classical policy to ensure consistency.
- *Circuit depth (repetitions):* number of repeated layers in the variational quantum circuit. The circuit is constructed in a modular (cell-based) manner, where a block of parameterized gates is repeated multiple times to increase expressivity.

The objective of the experiments is to maximize the expected reward $J(\theta)$, corresponding to the validation accuracy of sampled architectures, while identifying configurations that achieve a balanced trade-off between performance, model complexity, and runtime.

4. Results

We evaluate the impact of key hyperparameters—training steps, number of shots, and circuit depth—by varying one parameter at a time while keeping the others fixed. This enables a controlled comparison of their influence on performance and computational cost.

All results are reported as aggregated statistics over multiple random seeds. Each experiment is repeated across different seeds, and the reported metrics correspond to mean values and standard deviations, providing a robust estimate of performance under stochastic variation.

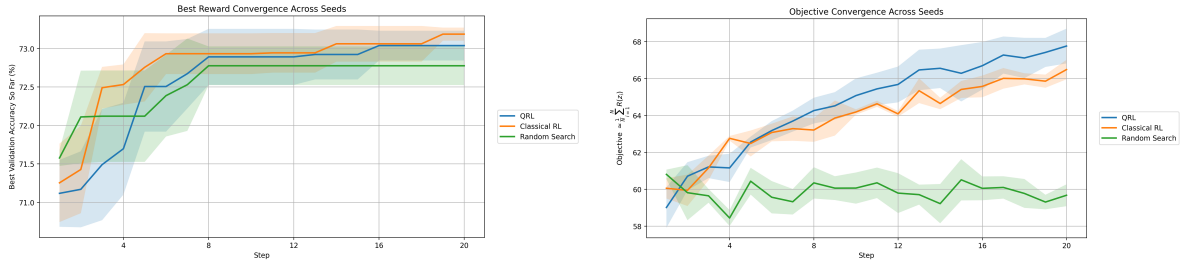
4.1 Effect of Number of Shots

We analyze the effect of the number of measurement shots by comparing configurations with 100, 200, and 300 shots, while keeping the number of training steps and circuit depth fixed at

20 and 1 respectively.

Increasing the number of shots improves the stability of the optimization process, as more samples are available to estimate the policy distribution at each update step. This results in smoother convergence behavior and more reliable estimates of the expected reward.

For the selected configuration with 200 shots, the convergence of the best validation reward and the objective function across seeds is shown in Fig. 1. Both QRL and classical RL outperform Random Search, indicating that the learning-based methods successfully capture meaningful sampling strategies. The corresponding performance and runtime statistics are summarized in Table 1. In this setting, classical RL achieves the highest mean final reward, while QRL remains competitive and consistently outperforms Random Search.



(a) *Best validation reward* **(b)** *Objective convergence*

Fig. 1. *Performance of QRL, Classical RL, and Random Search for 200 shots (20 steps, depth = 1). Shaded regions indicate variability across seeds.*

Table 1. Performance and runtime for different numbers of shots (20 steps, depth = 1)

Shots	Method	Parameters	Acc. Mean $\pm$ σ (%)	Runtime Mean $\pm$ σ (s)
100	QRL	72	72.964 ± 0.093	681 ± 79.33
	RL	18	72.858 ± 0.331	77 ± 3.88
	Random	0	72.600 ± 0.230	30 ± 4.69
200	QRL	72	73.038 ± 0.193	1302 ± 180.33
	RL	18	73.187 ± 0.085	176 ± 1.23
	Random	0	72.776 ± 0.251	64 ± 1.11
300	QRL	72	73.316 ± 0.251	1355 ± 144.27
	RL	18	73.180 ± 0.228	258 ± 7.88
	Random	0	72.799 ± 0.250	102 ± 6.66

When comparing different shot configurations, the improvement from 100 to 200 shots is more pronounced than the improvement from 200 to 300 shots. While 300 shots provide slightly smoother convergence, the gain in final performance is marginal. In contrast, runtime increases significantly with the number of shots, reflecting the higher number of circuit evaluations required at each optimization step.

Overall, 200 shots provide the best trade-off between optimization quality and computational cost. Therefore, this configuration is used in all subsequent experiments.

4.2 Effect of Circuit Depth

We next analyze the effect of circuit depth by varying the number of repetitions in the variational quantum circuit, while keeping the number of training steps and the already analyzed shots fixed at 20 and 200 respectively.

Increasing circuit depth enhances the expressive power of the quantum policy by introducing additional trainable layers and entangling operations. However, within the explored range, deeper circuits do not yield a substantial improvement in optimization performance.

Since the configuration with depth $=$ 1 corresponds to the same setting shown in

Fig. 1, no additional convergence plot is included. The results indicate that shallow circuits are already sufficient to learn effective sampling strategies, as both QRL and classical RL consistently outperform Random Search.

The performance and runtime statistics for different circuit depths are summarized in Table 2. While the final reward remains relatively stable across depths, the computational cost increases significantly due to the larger number of parameters and circuit evaluations. In particular, QRL exhibits a strong increase in runtime as depth grows, whereas classical RL and Random Search remain comparatively lightweight.

Table 2. Performance and runtime for different circuit depths (20 steps, 200 shots)

Depth	Method	Parameters	Acc. Mean $\pm$ σ (%)	Runtime Mean $\pm$ σ (s)
1	QRL	72	73.038 $\pm$ 0.193	1118 $\pm$ 294.588
	RL	18	73.187 $\pm$ 0.085	192 $\pm$ 34.649
	Random	0	72.776 $\pm$ 0.251	75 $\pm$ 19.957
2	QRL	108	73.004 $\pm$ 0.346	1842 $\pm$ 69.244
	RL	18	73.138 $\pm$ 0.251	157 $\pm$ 4.411
	Random	0	72.776 $\pm$ 0.251	61 $\pm$ 4.783
3	QRL	144	72.932 $\pm$ 0.139	3529 $\pm$ 293.938
	RL	18	73.253 $\pm$ 0.221	236 $\pm$ 5.472
	Random	0	72.776 $\pm$ 0.251	89 $\pm$ 1.864

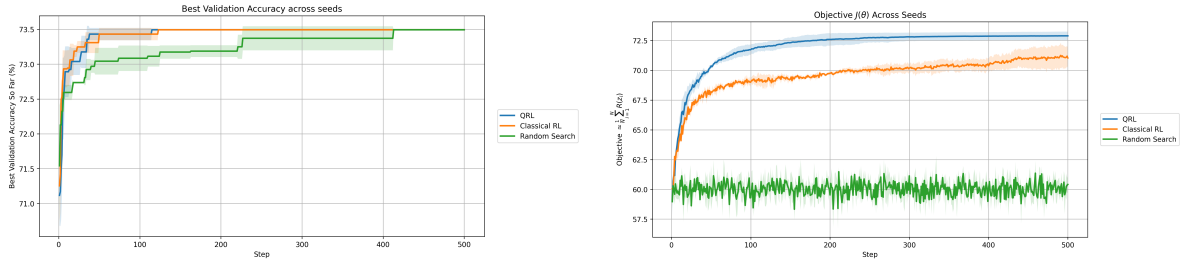
Overall, increasing circuit depth does not provide a meaningful improvement in performance, while significantly increasing computational cost. Therefore, a circuit depth of one repetition offers the best trade-off between efficiency and performance, and is used in the subsequent experiments.

4.3 Effect of Training Steps

We finally analyze the effect of the number of training steps by comparing configurations with 100, 250, and 500 steps, while keeping the number of shots and circuit depth fixed at 200 and 1 respectively.

Increasing the number of training steps allows the policy to refine its sampling distribution over a longer optimization horizon. This leads to improved convergence of the objective function and higher final validation performance, particularly for QRL and classical RL.

The convergence behavior for the configuration with 500 training steps is shown in Fig. 2. Both QRL and classical RL exhibit steady improvements over time, with QRL achieving a higher objective value and faster convergence. In contrast, Random Search remains largely flat, indicating the absence of learning.



(a) *Best validation reward* **(b)** *Objective convergence*

Fig. 2. *Performance of QRL, Classical RL, and Random Search for 500 training steps (200 shots, depth = 1). Shaded regions indicate variability across seeds.*

The final performance and runtime statistics across different training step configurations are summarized in Table 3. Increasing the number of steps leads to consistent improvements in final accuracy for both QRL and classical RL. In particular, QRL benefits significantly from longer training, achieving the highest objective values.

However, this improvement comes at the cost of increased computational time. The runtime grows approximately linearly with the number of training steps, with QRL exhibiting the highest computational overhead due to repeated circuit evaluations. Classical RL remains more efficient, while Random Search shows increased runtime without corresponding performance gains.

Table 3. Performance and runtime for different training steps (200 shots, depth = 1)

Steps	Method	Parameters	Acc. Mean $\pm$ σ (%)	Runtime Mean $\pm$ σ (s)
100	QRL	72	73.431 $\pm$ 0.088	1273 $\pm$ 25.590
	RL	18	73.431 $\pm$ 0.088	358 $\pm$ 12.153
	Random	0	73.118 $\pm$ 0.143	300 $\pm$ 5.519
250	QRL	72	73.493 $\pm$ 0.000	1596 $\pm$ 55.404
	RL	18	73.493 $\pm$ 0.000	409 $\pm$ 16.936
	Random	0	73.307 $\pm$ 0.000	702 $\pm$ 2.425
500	QRL	72	73.493 $\pm$ 0.000	2059 $\pm$ 50.567
	RL	18	73.493 $\pm$ 0.000	457 $\pm$ 22.803
	Random	0	73.307 $\pm$ 0.000	1331 $\pm$ 14.537

Overall, increasing the number of training steps improves optimization performance up to a point, after which the gains saturate. While 250 and 500 steps achieve similar final performance, the additional computational cost of longer training provides limited benefit.

Therefore, a moderate number of training steps (e.g., 250) offers a good balance between convergence quality and computational efficiency.

5. Conclusion and Future Work

We investigated the use of a simple variational quantum policy for neural architecture search and compared it with classical reinforcement learning and random search.

The results show that even a shallow quantum circuit with a limited number of parameters can achieve performance comparable to classical reinforcement learning. While QRL does not outperform the classical baseline, it consistently learns effective sampling strategies and clearly surpasses random search, demonstrating that meaningful learning behavior can emerge from simple quantum models.

Notably, under the best configuration (250 steps, 200 shots, depth = 1), all methods converge to the same final architecture. This suggests that all approaches are able to identify the same optimal region of the search space despite their different learning mechanisms.

However, QRL incurs a higher computational cost due to repeated circuit evaluations. Among the tested configurations, 250 training steps, 200 shots, and a single circuit repetition provide the best trade-off between performance and efficiency, as further increases yield only marginal gains.

A key direction for future work is to better understand the learning dynamics of quantum policies. In particular, analyzing the evolution of the Shannon entropy of the policy distribution could reveal whether learning occurs locally at the level of individual decisions or globally over sampled architectures.

Another promising direction is the design of more expressive quantum ansätze. Instead of manually defined circuits, approaches such as quantum architecture search could be

used to discover better initial circuit structures, potentially improving convergence speed and efficiency.

Overall, these results indicate that QRL can match classical approaches in this benchmark setting, while highlighting the need for improved efficiency and a deeper understanding of the underlying learning process. From the perspective of SHM, the present work should be viewed as a first step toward automated neural architecture optimization for SHM applications. Since large-scale SHM datasets suitable for extensive architecture search remain relatively limited, benchmark search spaces provide a controlled environment in which optimization algorithms can be developed and evaluated independently of application-specific constraints. Future work will investigate the scalability of the proposed framework on larger search spaces and its application to SHM-related datasets, including sensing technologies currently under development within the group, such as quantum vibrometer measurements and induction welding monitoring systems. Such studies will allow the assessment of whether quantum-enhanced search strategies can accelerate the discovery of accurate and computationally efficient models for practical SHM applications.

References

[1] Armin Dadras Eslamlou and Shiping Huang. Reinforcement learning for multi-objective automl in vision-based structural health monitoring. *Automation in Construction*, 166:105593, 2024.

[2] Sasan Salmani Pour Avval, Nathan D. Eskue, Roger M. Groves, and Vahid Yaghoubi. Systematic review on neural architecture search. *Artificial Intelligence Review*, 58(3), Jan 2025.

[3] Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. Nats-bench: Benchmarking NAS algorithms for architecture topology and size. *CoRR*, abs/2009.00437, 2020.

[4] David Wierichs, Josh Izaac, Cody Wang, and Cedric Yen-Yu Lin. General parameter-shift rules for quantum gradients. *Quantum*, 6:677, Mar 2022.