Evaluating Parameter-Based Training Performance of Neural Networks and Variational Quantum Circuits

Michael Kölle, Alexander Feist, Jonas Stein, Sebastian Wölckert, and Claudia Linnhoff-Popien

LMU Munich, Oettingenstraße 67, Germany
michael.koelle@ifi.lmu.de

Abstract. In recent years, NNs have driven significant advances in machine learning. However, as tasks grow more complex, NNs often require large numbers of trainable parameters, which increases computational and energy demands. VQCs offer a promising alternative: they leverage quantum mechanics to capture intricate relationships and typically need fewer parameters. In this work, we evaluate NNs and VQCs on simple supervised and reinforcement learning tasks, examining models with different parameter sizes. We simulate VQCs and execute selected parts of the training process on real quantum hardware to approximate actual training times. Our results show that VQCs can match NNs in performance while using significantly fewer parameters, despite longer training durations. As quantum technology and algorithms advance, and VQC architectures improve, we posit that VQCs could become advantageous for certain machine learning tasks.

Keywords: Variational Quantum Circuits · Parameter Efficiency · Quantum Supervised Learning · Quantum Reinforcement Learning

1 Introduction

Machine learning has advanced rapidly in recent years, with neural networks (NNs) playing a pivotal role in this progress [1]. As tasks become more complex, NNs often require a large number of trainable parameters, increasing computational and energy demands [25, 4]. Variational quantum circuits (VQCs) are a promising alternative to classical NNs [9, 5]. They harness quantum mechanics to model intricate relationships and usually need fewer parameters [22, 17].

Quantum computing shows considerable promise in supervised learning (SL) and reinforcement learning (RL). Schuld et al. [22] introduced a scalable VQC architecture and showed that it achieves strong SL performance with fewer parameters than NNs. Their design inspires the architecture used in our work. Chen et al. [6] illustrated that VQCs can perform well in RL by approximating the action-value function through Q-learning in simple environments. Inspired by this work, we use their custom Frozen Lake environment and Q-learning approach to evaluate NNs and VQCs with varying parameter counts. Kruse et al.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_33

[15] explored architectural factors in VQCs for the Pendulum and LunarLander tasks [3], revealing that design choices like input encoding, layering, and qubit count strongly affect outcomes. Despite using about 96% fewer parameters than NNs, VQCs earned lower rewards and struggled with scalability and robustness.

In this work, we evaluate the potential of VQCs relative to NNs on simple SL and RL tasks. To our knowledge, no existing work thoroughly compares NNs and VQCs for machine learning tasks with a detailed focus on model architectures, parameter counts, and training times. We carry out most VQC experiments on a simulator but approximate real hardware training times by running selected circuits on an actual quantum device. Our findings align with prior work, showing that VQCs can achieve performance similar to NNs while using fewer parameters. Although training VQCs takes longer, we suggest that continued advances in quantum technology, improvements in VQC architectures, and algorithmic optimizations may make VQCs appealing for certain applications. All code for the experiments is available here[1].

2 Approach

For each machine learning task, we evaluate NNs and VQCs with varying parameter counts to identify one of each with comparable performance. To ensure fairness, both models function as black-box components in the same classical learning algorithm. Although we conduct the VQC experiments primarily with a quantum simulator, we estimate real-hardware training times by running selected circuits—collected from simulator-based training—on an actual device.

2.1 Classical Neural Network Architecture

We employ a fully connected feedforward NN. The input layer has as many nodes as the input size, followed by one or more hidden layers whose quantities and sizes are hyperparameters. Each hidden layer uses element-wise ReLU activation [18, 14]. The output layer has as many nodes as the number of possible outputs and uses a softmax activation to produce a probability distribution [14].

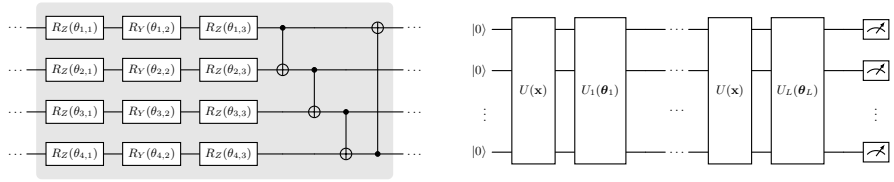
2.2 Variational Quantum Circuit Architecture

The proposed VQC follows a circuit centric design [22] and involves three stages: state preparation, variational layers, and measurement.

State Preparation We use angle embedding or amplitude embedding, chosen as a hyperparameter. Angle embedding maps each input feature to a rotation angle, using at least as many qubits as the input dimension. Amplitude embedding directly maps input values to the amplitudes of an n-qubit state, which requires $\lceil \log_2(D) \rceil$ qubits to represent D-dimensional data [23, 22].

[1] https://github.com/alexander-feist/nn-vqc-params

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_33



(a) Variational Layer Architecture (b) VQC With Data Re-Uploading

Fig. 1: (a) The l-th variational layer with 4 qubits, where $\boldsymbol{\theta}_l = [\theta_{1,1}, \theta_{1,2}, \ldots, \theta_{4,3}]$ are the trainable parameters for layer l. (b) A VQC with L layers; $U(\mathbf{x})$ encodes the input $\mathbf{x}$, and $U_l(\boldsymbol{\theta}_l)$ represents the trainable operations in layer l [5, 22].

Variational Layers Variational layers are composed of single-qubit rotations, followed by CNOT gates for entanglement (Fig. 1a). The trainable parameters $\boldsymbol{\theta}$, initialized randomly in $[-1, 1]$, are passed through $\varphi(\mathbf{z}) = \pi \cdot \tanh(\mathbf{z})$ to constrain angles to $(-\pi, \pi)$ [12]. We also use data re-uploading [20, 24], which embeds the classical input values before every variational layer (Fig. 1b).

Measurement We measure the expectation value of the Pauli-Z operator on the first K qubits, where K matches the output dimension, and add trainable biases (initialized in $[-0.001, 0.001]$). A softmax function is applied to derive output probabilities. To expand the $[-1, 1]$ Pauli-Z range, we include a trainable scaling parameter (initialized at 1) to enhance the effective output range [24].

3 Experimental Setup

We primarily use PyTorch [19] and PennyLane [2] with the `default.qubit` device for statevector simulation. All experiments run on a Linux cluster with Intel® Core™ i9-9900 processors. To ensure reproducibility, we fix seeds and repeat each experiment ten times (seeds 0–9). For each task, we conduct an exhaustive grid search over model-based hyperparameters. This yields equally sized sets of NNs and VQCs spanning a range of parameter counts, allowing us to identify pairs of models with comparable performance but different complexities.

3.1 Supervised Learning Experiments

We conduct SL classification tasks on the Iris, Wine, and WDBC (Wisconsin Diagnostic Breast Cancer) datasets. We use accuracy as the main performance metric. Features are scaled to $[0, 1]$ and data is split into 75% training and 25% testing, with the test portion evenly divided into validation and test sets.

We train for 50 epochs with cross-entropy loss and Adam [11] at a learning rate of 0.01, using a batch size of 8. Validation performance is checked after each epoch. The checkpoint with the highest validation accuracy is evaluated

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_33

on the test set. For Iris and Wine, the NN grid search covers {1,2,3} hidden layers ×{3,6,9,12} nodes per layer. For the VQC, we use angle or amplitude embedding and 1 to 6 variational layers. For WDBC (30 features), we limit VQCs to amplitude embedding to avoid 30-qubit circuits and vary layers from 1 to 6. To match, we search for NNs with {1,2} hidden layers ×{3,6,9} nodes.

3.2 Reinforcement Learning Experiments

Our RL experiments use Q-learning with the models (NN or VQC) approximating the action-value function Q. The test reward serves as the main performance metric. Following Chen et al. [6], we use the deterministic (non-slippery) Frozen Lake environment [3] with custom rewards. The 4×4 grid contains safe (frozen) tiles and holes. The agent starts in the top-left and must reach the bottom-right goal. Steps yield −0.01, reaching the goal +1.0, and falling into a hole −0.2.

We train for 500 episodes, each limited to 100 steps. The model observes a 4-dimensional binary-encoded state (one of 16 tiles) and outputs four action values. We use identical policy and target models, updating the target every 20 steps. Actions follow an ϵ-greedy strategy: ϵ starts at 1.0 and decays by 0.99 after each episode until reaching 0.01. We use experience replay [16] with memory size 1000, sampling a batch of 16 transitions per step. The policy model is trained via Adam [11] at a learning rate of 0.01, using MSE loss and a discount factor of 0.95. After training, we evaluate over 50 test episodes without exploration. The grid search for NNs spans {1,2,3} hidden layers ×{3,6,9,12} nodes. For VQCs, it varies embedding (amplitude or angle) and the number of layers from 1 to 6.

3.3 Executing Quantum Circuits on Real Quantum Hardware

Running full training on real hardware is costly, so we log certain circuits (inputs and parameters) during simulator-based training and re-run those circuits on actual quantum processors through IBM's cloud-based Qiskit Runtime. Specifically, we pick circuits from five epochs/episodes under seed 0, unparameterize them with the logged values, and import them into Qiskit [10]. The circuits are executed on `ibm_fez` (version 2) backed by IBM's Heron R2 processor. By comparing Qiskit Runtime's usage metric to the simulator time for the same circuits, we compute an average ratio and apply it to circuit execution times of simulator-based training to estimate real-hardware training times.

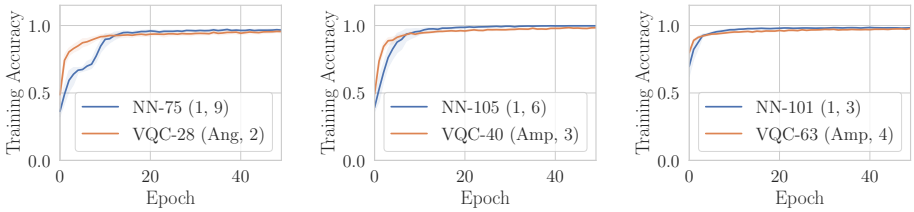
4 Results

All metrics are averaged over ten runs (seeds 0–9). We first examine grid-search outcomes to select NNs and VQCs with comparable performance for each task.

4.1 Supervised Learning Results

We focus on models that perform well and choose an NN-VQC pair whose test accuracies and training curves suggest similar performance. Across all datasets,

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_33



(a) Iris Dataset (b) Wine Dataset (c) WDBC Dataset

Fig. 2: Accuracy curves for each chosen NN and VQC. Averaged across ten runs (seeds 0–9); shaded areas are 95% confidence intervals.

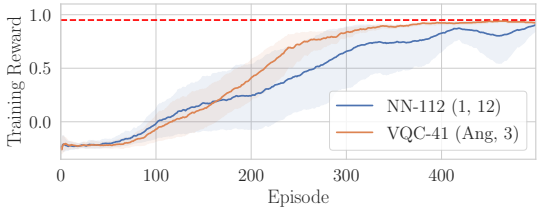


Fig. 3: Training reward (moving average over up to last 50 episodes) on Frozen Lake for the comparable NN and VQC. Mean across ten runs (seeds 0–9); shaded areas are 95% confidence intervals. Dashed red line (0.95) marks solved.

the NNs performed overall better compared to the VQCs. However, for each dataset, we could find an NN-VQC pair where both models are well-performing (more than 96% test accuracy) and comparable in performance. The VQCs have fewer parameters but consistently require substantially longer training times. Figure 2 illustrates training accuracy for our NN–VQC pairs. Although the VQCs converge faster initially, the NNs ultimately achieve slightly higher accuracy.

4.2 Reinforcement Learning Results

We select models that achieve the maximum test reward of 0.95 and choose an NN–VQC pair with similar learning dynamics but relatively low training times. Training-time variance is high because episode length depends on agent behavior. Over all models, VQCs generally outperform NNs here, which may be due to the grid search only considering low-parameter models, favoring VQCs. However, the VQCs take much longer to train. Figure 3 shows the training reward of our most comparable NN and VQC. The VQC converges faster and is more stable.

4.3 Training Times Using Real Quantum Hardware

Estimated real-hardware training times for VQCs are significantly higher than simulator times, as shown in Table 1, based on per-circuit execution durations.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_33

Table 1: Mean execution time per circuit on the simulator vs. real hardware, as well as the hardware-to-simulator time ratio.

Task	VQC	Qubits	Circuit Depth	Simulator (s)	Real Hardware (s)	Ratio
SL: Iris	VQC-28 (Ang, 2)	4	17	0.011	0.314	28.995
SL: Wine	VQC-40 (Amp, 3)	4	100	0.034	0.349	10.295
SL: WDBC	VQC-63 (Amp, 4)	5	261	0.084	0.329	3.932
RL	VQC-41 (Ang, 3)	4	25	0.016	0.322	20.406

Table 2: Mean training times (with 95% confidence intervals) for comparable NNs and VQCs.

	Model		Training Time (s)		
Task	NN	VQC	NN	VQC Simulator	VQC Real Hardware
SL: Iris	NN-75	VQC-28	1.8 ± 0.0	92.6 ± 0.2	1806.1 ± 4.1
SL: Wine	NN-105	VQC-40	1.7 ± 0.0	313.5 ± 1.3	2437.4 ± 11.6
SL: WDBC	NN-101	VQC-63	5.4 ± 0.0	2482.9 ± 12.7	7732.5 ± 37.2
RL	NN-112	VQC-41	95.0 ± 30.7	2511.4 ± 219.3	39330.9 ± 3458.8

Using angle embedding leads to lower circuit depth compared to amplitude embedding, resulting in smaller simulator runtimes but higher hardware-to-simulator time ratios. For 5-qubit circuits, the hardware ratio decreases, indicating that as qubit counts increase, the simulator grows slower relative to hardware, consistent with existing literature [8, 7]. Since overhead and noise are excluded, these estimates likely represent ideal scenarios, but further optimizations (e.g., fewer shots, specialized training environments) could significantly reduce real-hardware training times.

4.4 Evaluating Training Performance

Table 2 shows training times of the selected NNs and VQCs. For similar performance, VQCs require 62.7% (Iris SL), 61.9% (Wine SL), 37.6% (WDBC SL), and 63.4% (Frozen Lake RL) fewer parameters, but take much longer to train. For the RL task, equalizing the two training times would require the VQC to be about 414 times faster, which may sound large but could become feasible as quantum hardware matures much faster than classical systems. Architectural and algorithmic improvements—such as specialized VQC optimizers—may also reduce this ratio. Furthermore, although VQCs often converge faster in accuracy or reward, our fixed training schedule does not exploit early convergence.

Even in our small-scale tasks, we see a trend of longer training times for larger models, especially for VQCs. This trend may be more pronounced in complex tasks where standard NNs can have millions of parameters, potentially offering

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_33

a more substantial advantage to VQCs that require fewer parameters [13, 17]. However, it remains unclear whether VQCs can scale effectively to complex tasks and still match NNs [21, 15]. Rather than replacing NNs outright, VQCs may find value in scenarios where they offer distinct benefits—especially if quantum hardware, training algorithms, and circuit designs continue to improve.

5 Conclusion

We created a unified environment to compare classical NNs and VQCs as interchangeable models for multiple machine learning tasks. VQCs achieved comparable performance with fewer parameters—especially in RL—but required much longer training times. Because our tasks used at most five qubits, real-hardware execution was slower than simulation. Our findings underscore the simplicity of the tasks, yet suggest that as quantum technology matures, VQC-friendly algorithms improve, and circuit architectures evolve, VQCs may offer advantages for specific applications. Future work could explore more complex tasks and assess quantum device fidelity to better evaluate VQC performance in realistic settings.

Acknowledgements

This work is part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. This paper was partly funded by the German Federal Ministry of Education and Research through the funding program "quantum technologies — from basic research to market" (contract number: 13N16196) and the BWMK (01MQ22008A).

References

1. Alom, M.Z., Taha, T.M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M.S., Hasan, M., Van Essen, B.C., Awwal, A.A., Asari, V.K.: A state-of-the-art survey on deep learning theory and architectures. Electronics **8**(3), 292 (2019)
2. Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Ahmed, S., Ajith, V., Alam, M.S., Alonso-Linaje, G., AkashNarayanan, B., Asadi, A., et al.: Pennylane: Automatic differentiation of hybrid quantum-classical computations. arXiv preprint arXiv:1811.04968 (2018)
3. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym. arXiv preprint arXiv:1606.01540 (2016)
4. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
5. Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S.C., Endo, S., Fujii, K., McClean, J.R., Mitarai, K., Yuan, X., Cincio, L., et al.: Variational quantum algorithms. Nature Reviews Physics **3**(9), 625–644 (2021)
6. Chen, S.Y.C., Yang, C.H.H., Qi, J., Chen, P.Y., Ma, X., Goan, H.S.: Variational quantum circuits for deep reinforcement learning. IEEE access **8**, 141007–141024 (2020)

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_33

7. Chen, Z.Y., Zhou, Q., Xue, C., Yang, X., Guo, G.C., Guo, G.P.: 64-qubit quantum circuit simulation. Science Bulletin **63**(15), 964–971 (2018)
8. Cicero, A., Maleki, M.A., Azhar, M.W., Kockum, A.F., Trancoso, P.: Simulation of quantum computers: Review and acceleration opportunities. arXiv preprint arXiv:2410.12660 (2024)
9. Du, Y., Hsieh, M.H., Liu, T., Tao, D.: Expressive power of parametrized quantum circuits. Physical Review Research **2**(3), 033125 (2020)
10. Javadi-Abhari, A., Treinish, M., Krsulich, K., Wood, C.J., Lishman, J., Gacon, J., Martiel, S., Nation, P.D., Bishop, L.S., Cross, A.W., et al.: Quantum computing with qiskit. arXiv preprint arXiv:2405.08810 (2024)
11. Kingma, D.P.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Kölle, M., Giovagnoli, A., Stein, J., Mansky, M.B., Hager, J., Rohe, T., Müller, R., Linnhoff-Popien, C.: Weight re-mapping for variational quantum algorithms. In: International Conference on Agents and Artificial Intelligence. pp. 286–309. Springer (2023)
13. Kölle, M., Topp, F., Phan, T., Altmann, P., Nüßlein, J., Linnhoff-Popien, C.: Multi-agent quantum reinforcement learning using evolutionary optimization. arXiv preprint arXiv:2311.05546 (2023)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)
15. Kruse, G., Dragan, T.A., Wille, R., Lorenz, J.M.: Variational quantum circuit design for quantum reinforcement learning on continuous environments. arXiv preprint arXiv:2312.13798 (2023)
16. Lin, L.J.: Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine Learning **8**, 293–321 (1992)
17. Lockwood, O., Si, M.: Reinforcement learning with quantum variational circuit. In: Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment. vol. 16, pp. 245–251 (2020)
18. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). pp. 807–814 (2010)
19. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
20. Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E., Latorre, J.I.: Data re-uploading for a universal quantum classifier. Quantum **4**, 226 (2020)
21. Qian, Y., Wang, X., Du, Y., Wu, X., Tao, D.: The dilemma of quantum neural networks. IEEE Transactions on Neural Networks and Learning Systems (2022)
22. Schuld, M., Bocharov, A., Svore, K.M., Wiebe, N.: Circuit-centric quantum classifiers. Physical Review A **101**(3), 032308 (2020)
23. Schuld, M., Fingerhuth, M., Petruccione, F.: Implementing a distance-based classifier with a quantum interference circuit. Europhysics Letters **119**(6), 60002 (2017)
24. Skolik, A., Jerbi, S., Dunjko, V.: Quantum agents in the gym: a variational quantum algorithm for deep q-learning. Quantum **6**, 720 (2022)
25. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for modern deep learning research. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 13693–13696 (2020)

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_33