Modeling Quantum Neural Network Gradient With Reinforcement Learning

Nhan Trong Luu[1,3]*, **Duong Trung Luu**[2], **Pham Ngoc Nam**[3] **and Truong Cong Thang**[4]
[1]College of Information and Communication Technology, Can Tho University, Can Tho, Vietnam
[2]Center for Digital Transformation and Communications, Can Tho University, Can Tho, Vietnam
[3]College of Engineering and Computer Science, VinUniversity, Hanoi, Vietnam
[4]Department of Computer Science and Engineering, The University of Aizu, Fukushima, Japan

Abstract

Training quantum neural networks (QNNs) on near-term hardware remains hampered by two compounding difficulties: the exponential vanishing of gradient variance known as the barren plateau, and the $\mathcal{O}(L \cdot 2^n)$ time and memory cost of differentiating through an n-qubit, L-layer circuit. We propose RLQ-Grad, a reinforcement-learning-based optimizer in which a classical policy π_ϕ (a spectrally-normalized PPO agent) learns to propose parameter updates directly, conditioned on the QNN's current parameters, loss, accuracy, and previous update. Because the surrogate gradient is emitted by a classical network rather than obtained by differentiating through the unitary $U(\theta)$, its variance is not constrained by the barren plateau concentration bound, and its cost scales with the number of trainable parameters rather than the Hilbert-space dimension. We prove these properties formally and verify them on a hardware-efficient ansatz across four supervised benchmarks with up to $n = 20$ qubits. RLQ-Grad preserves a near-flat gradient-variance curve where backpropagation, parameter-shift, and adjoint differentiation decay by 1 to 2 orders of magnitude. Accounting for the full training pipeline (PPO rollouts, actor-critic updates, and optimizer states), RLQ-Grad needs under 2 MB of memory and runs $2490\times$, $7876\times$, and $673\times$ faster per iteration than these three methods at $n = 20$. It improves top-1 accuracy by up to $+10\%$ over gradient-based baselines on circuits of up to 12 qubits, and matches dedicated barren plateau mitigation methods on CIFAR-10 at 14 to 20 qubits, where evolutionary and gradient-free optimizers collapse to chance.

1 Introduction

The emergence of Noisy Intermediate-Scale Quantum (NISQ) devices has established variational quantum algorithms (VQAs) as a leading paradigm for achieving practical quantum advantage [1, 2, 3]. These hybrid quantum-classical frameworks, which include Quantum Neural Networks (QNNs), rely on the iterative optimization of parameterized quantum circuits (PQCs) to minimize a problem-specific cost function [1, 4, 5]. However, the optimization landscape of QNNs is notoriously challenging due to non-convexity, the presence of low-quality local optima, the "barren plateau" phenomenon [6], and the high computational cost of circuit evaluations [7, 1, 8]. To address these hurdles, a burgeoning body of research has explored the application of reinforcement learning (RL) to automate and enhance the optimization of both the variational parameters and the underlying circuit architectures.

Initial efforts in this domain focused on utilizing RL to find optimal variational parameters for fixed-structure circuits, such as the Quantum Approximate Optimization Algorithm (QAOA) [8, 7] which formulated parameter optimization as a learning task, where objective is to train RL policy

*Corresponding author. Email: luutn@ctu.edu.vn.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

networks on small problem instances to exploit geometrical regularities that generalize to larger, unseen instances. Similarly, [1] introduced a "learning to learn" framework using classical recurrent neural networks (RNNs) as black-box controllers to propose parameter updates for QAOA and Variational Quantum Eigensolvers (VQE), eventually highlights the potential for meta-learning to identify problem-class-specific heuristics that significantly reduce the number of optimization iterations required. Further advancing this theme, [9] proposed an RL scheme for feedback quantum control, demonstrating that an agent can learn smooth, optimal adiabatic schedules for quantum Ising chains that are successfully transferable across system sizes.

Parallel to parameter optimization, some research has shifted toward Quantum Architecture Search (QAS) and circuit optimization, where RL agents autonomously design or refine the circuit structure itself. [10] demonstrated that deep RL could learn generic strategies to reduce circuit depth and gate count by selecting sequences of logically equivalent transformations. For building circuits from scratch, [11] proposed a DRL framework to search for gate sequences that synthesize specific quantum states without prior physical knowledge. [3] and [5] also further refined QAS for chemistry applications, employing feedback-driven curriculum learning to adapt problem complexity and discover gate-efficient ansatzes for molecules like LiH.

Recent advancements have focused on improving the scalability and robustness of these RL-based methods. [12] evaluated SAC-based optimizers capable of minimizing loss across random circuits of varying sizes and objectives. To tackle the prohibitive simulation costs of large systems, [5] introduced TensorRL-QAS, which warm-starts the RL search using tensor network approximations to narrow the search space and accelerate convergence. Additionally, specialized architectures for machine learning tasks have been explored, such as [13] (RL-QMLAS) aim to perform QAS for QAOA problems and [4] (RLVQC), which discovered novel "Ryz-connected" ansatz families for combinatorial optimization. Finally, [2] have systematically investigated the impact of specific techniques (such as data re-uploading [14], output scaling, and metaheuristic strategies [15]) to stabilize training and reduce the parameter count of VQCs within RL environments.

While RL has driven notable advances in QNN research, its integration into supervised learning for QNNs remains largely underexplored. Early efforts framing RL as an exhaustive parameter search over QNN configurations prove fundamentally ill-suited to larger-scale problems [12]. Motivated by these limitations, we make the following contributions:

- **A novel RL-based optimizer for QNNs:** We introduce **RLQ-Grad**, an RL-based optimizer in which an agent learns to accurately approximate QNN gradients during training, rather than performing exhaustive parameter search or act as a complete replacement of traditional optimizer. Across a range of benchmarks, RLQ-Grad consistently outperforms conventional gradient-based optimizers in classification accuracy.
- **Memory reduction and accelerated computation:** Beyond accuracy, RLQ-Grad circumvents the gradient variance decay caused by the barren plateau phenomenon, while replacing the 2^n dependence of gradient computation time and memory with a cost that grows only linearly in the number of trainable QNN parameters. RLQ-Grad maintains an approximately flat gradient-variance profile, whereas backpropagation, parameter-shift, and adjoint differentiation exhibit declines of 1 to 2 orders of magnitude. Under full-pipeline accounting (actor, critic, gradient buffers, and Adam states), RLQ-Grad requires 1.95 MB at $n = 20$ qubits (versus 6174 MB for backpropagation) and achieves per-iteration CPU speedups of $\sim 2490\times$, $\sim 7876\times$, and $\sim 673\times$ over backpropagation, parameter-shift, and adjoint differentiation, respectively, while also transferring directly to GPU execution.
- **Scaling to 20 qubits against stronger baselines:** RLQ-Grad improves top-1 accuracy by up to $+10\%$ over the strongest gradient-based baseline from 2 to 12 qubits. On CIFAR-10 extended to 20 qubits ($P_{\mathrm{QNN}} = 600$), it keeps improving where backpropagation stagnates, matches dedicated barren plateau mitigation (layerwise learning [16] and Gaussian initialization [17]), and clearly outperforms evolutionary and gradient-free optimizers (CMA-ES, sep-CMA-ES, SPSA), which collapse to chance.

To the best of our knowledge, this work is the first to successfully employ a reinforcement learning agent as a gradient-based optimizer for QNNs, reducing the memory complexity of gradient computation to linear scaling while remaining acceleratable using modern GPU.[2]

[2]The source code for our paper is available on Zenodo at https://doi.org/10.5281/zenodo.22956233.

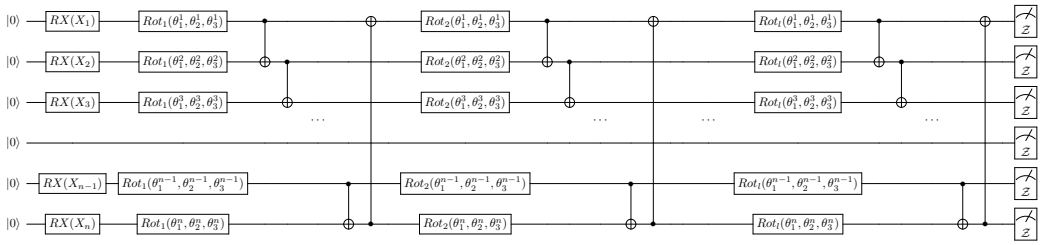


Figure 1: Illustration of a n-qubit l-depth HEA circuit utilized in the majority of our experiments.

2 Proposed method

2.1 Problem Formulation

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ denote a supervised learning dataset where $x_i \in \mathbb{R}^d$ represents the input features and $y_i \in \{0, 1, ...n\}$ denotes the n-class target label. We consider training a parameterized quantum neural network QNN defined by a set of parameters $\theta \in \mathbb{R}^P$. The goal of QNN supervised training is to minimize an empirical risk function of:

$$\min \mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i, \theta), y_i), \tag{1}$$

where $f_\theta(x)$ denotes the output of the quantum model and $\ell(\cdot)$ is a certain type of supervise learning loss function, and traditional QNN optimization relies on gradient estimation techniques such as backpropagation, adjoint differentiation [18] or parameter-shift rule [19]. However, these approaches often suffer from high computational cost and gradient vanishing phenomena. Instead of explicitly computing the gradient $\nabla_\theta \mathcal{L}$, we propose to learn a gradient generator using reinforcement learning.

2.2 QNN selection

In most of our problem, we consider a HEA-like [20] variational quantum circuit composed of n qubits and L layers. Illustration of HEA design used in our work can be found at Figure 1.

In our selected HEA, given classical inputs x, projected inputs over a fully connected layer FC_{in} are encoded through R_x rotation gates as:

$$U_{\text{enc}}(x) = \prod_{i=1}^{n} R_x(\mathrm{FC}_{in}(x, \mathbf{W}_{in}, \mathbf{b}_{in})_i) \text{ where } \mathrm{FC}(x) = x^\dagger \mathbf{W} + \mathbf{b}. \tag{2}$$

Each variational layer consists of parameterized rotations followed by entanglement operations. Let $\theta = \{\theta_{l,q}^{(1)}, \theta_{l,q}^{(2)}, \theta_{l,q}^{(3)}\}$ denote the parameters of a rotation gate on qubit q in layer l, $\mathbf{W} = \{\mathbf{W}_{in}, \mathbf{W}_{out}\}$ and $\mathbf{b} = \{\mathbf{b}_{in}, \mathbf{b}_{out}\}$ for parameter of data encode-decode FC layers. The variational unitary is therefore:

$$U(\theta) = \prod_{l=1}^{L} \left(U_{\text{ent}} \prod_{q=1}^{n} Rot(\theta_{l,q}) \right), \tag{3}$$

where $Rot(\theta_{l,q})$ represents a general single-qubit rotation and U_{ent} denotes the entangling operations. Given an input x, the circuit prepares the quantum state:

$$|\psi(x, \theta)\rangle = U(\theta) U_{\text{enc}}(x) |0\rangle^{\otimes n}. \tag{4}$$

The model output is obtained by measuring a Pauli observable Z, followed by a fully-connected layer FC that map output measurements to desired class counts:

$$f(x, \theta, \mathbf{W}, \mathbf{b}) = \mathrm{FC}_{out}\left(\langle \psi(x, \theta) | Z | \psi(x, \theta) \rangle, \mathbf{W}_{out}, \mathbf{b}_{out}\right), \tag{5}$$

and resulted output can be utilized as prediction in supervised learning problem[3]. To construct the HEA circuit compatible to our workflow, we utilized both Pennylane [21] and Torchquantum [22] at the base of our QNN framework.

[3] HEA can be considered a universal approximator, see Appendix A for more information.

Algorithm 1 Pseudocode for RLQ-Grad algorithm

Require: Train dataset $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N}$ of N samples with stochastic batch $\mathcal{B} \subset \mathcal{D}_{\text{train}}$, parameterized QNN model $f(x, \theta, \mathbf{W}, \mathbf{b})$, RL policy π_ϕ, total training steps T.

Initialize all QNN parameters θ_0 and previous gradient g_{-1} at zero.

Compute initial statistic on first stochastic batch $\mathcal{B}_0$, including loss $\mathbb{E}_{(x,y)\sim\mathcal{B}_0}\left[\mathcal{L}(f(x, \theta_0, \mathbf{W}, \mathbf{b}), y)\right]$ and accuracy $\mathbb{E}_{(x,y)\sim\mathcal{B}_0}\left[acc(f(x, \theta_0, \mathbf{W}_0, \mathbf{b}_0), y)\right]$.

for $t = 0$ to $T - 1$ **do**

  Construct environment state $s_t = \begin{bmatrix} \theta_t, \mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[\mathcal{L}(f(x, \theta_t, \mathbf{W}_t, \mathbf{b}_t), y)\right], \\ \mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[acc(f(x, \theta_t, \mathbf{W}_t, \mathbf{b}_t), y)\right], g_{t-1} \end{bmatrix}$

  Sample action (gradient proposal) from RL agent $g_t \sim \pi_\phi(\cdot|s_t)$

  Assign proposed gradient g_t to QNN parameter.

  Calculate the rest of the gradient graph components and update with backpropagation.

  Evaluate updated model training loss $\mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[\mathcal{L}(f(x, \theta_{t+1}, \mathbf{W}_{t+1}, \mathbf{b}_{t+1}), y)\right]$ and training accuracy $\mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[acc(f(x, \theta_{t+1}, \mathbf{W}_{t+1}, \mathbf{b}_{t+1}), y)\right]$.

  Compute reward $\mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[acc(f(x, \theta_t, \mathbf{W}_t, \mathbf{b}_t), y) + \frac{1}{\mathcal{L}(f(x,\theta_t,\mathbf{W}_t,\mathbf{b}_t),y)+\epsilon}\right]$.

  Store transition (s_t, g_t, r_t, s_{t+1}) and update RL policy π_ϕ using collected transitions.

end for

return Final trained QNN parameters θ

While showing promising potential as an alternative to construct predictor in supervise learning, HEA and many other quantum classifier tends to suffer from gradient variance decay, stem from quantum barren plateau [23] that prohibit them to converge towards an optimal approximation of given problems[4].

2.3 RLQ-Grad algorithm

Instead of computing analytical gradients, we formulate QNN training as a RL problem, where a RL agent interacts with the QNN training process and learns to generate parameter updates and termed our methodology RLQ-Grad. The intuition behind RLQ-Grad is to train a RL agent aim to learn a policy $\pi_\phi(a_t|s_t)$ parameterized by ϕ, which outputs gradient estimates conditioned on the QNN training state. Through interaction with the QNN training environment, the policy gradually learns to generate gradient directions that effectively optimize the quantum model. Pseudocode for RLQ-Grad can be found at Algorithm 1.

In most of our experiments, we use Proximal Policy Optimization (PPO) [24] as our RL agent in our RLQ-Grad algorithm, while also incorporating spectral normalization (SN) [25] improve performance and reducing variance during training. Details on training settings and justification for model selection can be found in Appendix D.

2.3.1 State representation

Given train dataset $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N}$ of N samples with stochastic batch $\mathcal{B}_t \subset \mathcal{D}_{\text{train}}$, at training step t the environment state is defined as:

$$
\begin{aligned}
s_t &= \left[\theta_t, \mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[\mathcal{L}(f(x, \theta_t, \mathbf{W}_t, \mathbf{b}_t), y)\right], \mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[acc(f(x, \theta_t, \mathbf{W}_t, \mathbf{b}_t), y)\right], g_{t-1}\right], \\
g_{t-1} &= \text{flatten}\left(\frac{\partial\, \mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[\mathcal{L}(f(x, \theta_t, \mathbf{W}_t, \mathbf{b}_t), y)\right]}{\partial\, \mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[f(x, \theta_t, \mathbf{W}_t, \mathbf{b}_t)\right]} \cdot \frac{\partial\, \mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[f(x, \theta_t, \mathbf{W}_t, \mathbf{b}_t)\right]}{\partial\, \theta_{t-1}}\right)
\end{aligned}
\tag{6}
$$

where θ_t are the current QNN parameters, $\mathcal{L}$ is the mean training loss, acc is the mean training accuracy, followed by all flattened trainable parameters' gradient g_{t-1} proposed by the agent at the previous step. Proposed gradient g_t is equivalent to RL agent outputs, interpreted as a gradient estimate $a_t = g_t \in \mathbb{R}^P$ of P optimizable parameter, and after the agent proposed the gradient g_t, it was assigned to QNN parameter gradient.

[4] We intentionally selected QNN that are subject to gradient decay to prove that our method is decay free. See Appendix B for more information regarding definition of plateau.

Finally, we calculate other components in the backpropagation graph (such as linear weights gradient $\frac{\partial \mathbb{E}_{(x,y)\sim\mathcal{B}_t}[\mathcal{L}(f(x,\theta_t,\mathbf{W}_t,\mathbf{b}_t),y)]}{\partial \mathbf{W}_t}$ and biases gradient $\frac{\partial \mathbb{E}_{(x,y)\sim\mathcal{B}_t}[\mathcal{L}(f(x,\theta_t,\mathbf{W}_t,\mathbf{b}_t),y)]}{\partial \mathbf{b}_t}$, etc.) and update $f(x,\theta,\mathbf{W},\mathbf{b})$ with Adam optimizer [26] (learning rate $lr = 1e-3$, weight decay of $1e-4$, momentum $\beta = (0.9, 0.999)$) along with cosine annealing scheduler [27].

2.3.2 Reward function

We formulate our reward function r_t at timestep t using training loss statistic $\mathcal{L}_t$ and accuracy statistic acc_t over stochastic batch of training set $\mathcal{B}_t \subset \mathcal{D}_{\text{train}}$ of train set $\mathcal{D}_{\text{train}}$ as:

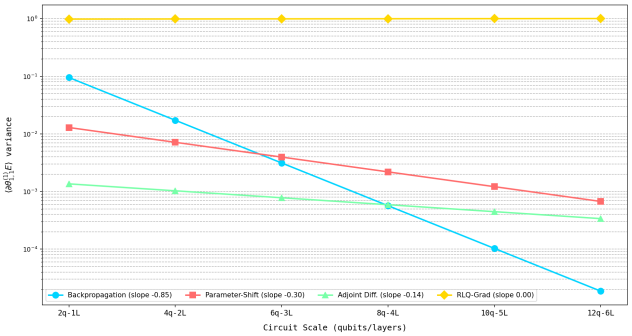
$$r_t = \mathbb{E}_{(x,y)\sim\mathcal{B}_t}\left[acc(f(x,\theta_t,\mathbf{W}_t,\mathbf{b}_t),y) + \frac{1}{\mathcal{L}(f(x,\theta_t,\mathbf{W}_t,\mathbf{b}_t),y)+\epsilon}\right], \tag{7}$$

where ϵ is a small constant to prevent zero division. The goal to encourages the agent to produce gradients that simultaneously reduce training loss and improve predictive performance.

3 Results

3.1 Surrogate gradient advantages of RLQ-Grad

Figure 2: Comparison of gradient variance of the first trainable parameter in the HEA $\text{Var}(\langle \partial\, \theta_{1,1}^{(1)}\, E\rangle)$ of gradient computation strategies for a depth-L, n-qubit QNN with P_{QNN} trainable parameters. Variance results for RLQ-Grad are obtained from post training agent on BC dataset on Section 3.2.



We assess the advantages of surrogate gradient produced by RLQ-Grad in light of the recently established unified QNN loss landscape theory of [28]. We claim that proposed RLQ-Grad algorithm can generate gradient that free from gradient decay caused by quantum barren plateau [23] over the following theorem:

Theorem 1 (Gradient variance decay free)**.** *We established some notations as follow:* $\mathcal{A} \cong \bigoplus_\alpha \mathcal{A}_\alpha$ *denotes the Jordan algebra generated by* $\{U_\theta^\dagger O U_\theta\}_\theta$, $\text{Aut}(\mathcal{A}_\alpha)$ *its automorphism group, and* $r_\alpha = [\text{Tr}_\alpha(O_\alpha)^2/\text{Tr}_\alpha(O_\alpha^2)]$ *the degrees of freedom of the* α*-th simple component. Let* $U(\theta)$ *be a QNN whose associated Jordan algebra* $\mathcal{A}$ *satisfies:*

$$\text{Var}_\theta[\ell(\rho;\theta)] = \sum_\alpha \frac{\text{Tr}(O_\alpha)^2\,\text{Tr}(\rho^\alpha)^2}{\dim_{\mathbb{R}}(\text{Aut}(\mathcal{A}_\alpha))} \in \mathcal{O}\big(\text{poly}(\log N)^{-1}\big), \tag{8}$$

i.e., the circuit is in a barren plateau[5]*. Under RLQ-Grad, the parameter update at step* t *is* $\theta_{t+1} = \theta_t - \eta\, g_t$ *where* $g_t \sim \pi_\phi(\cdot \mid s_t)$ *is the RL agent's output. Since* g_t *is not computed by differentiating through* $U(\theta)$*, the vanishing of Equation 8 places no constraint on* g_t*. The variance decay is therefore structurally bypassed.*

Proof. See Appendix C.1. □

To provide a more practical and comprehensive evaluation of the decay resistance of our method with respect to (w.r.t.) well-established gradient-based approaches, Figure 2 reports the gradient variance of the first trainable parameter in the HEA $\text{Var}(\langle \partial\, \theta_{1,1}^{(1)}\, E\rangle)$, as a function of both qubit count and circuit depth[6]. The comparison includes several gradient estimation techniques, namely traditional backpropagation, parameter-shift [19], adjoint differentiation [18], and RLQ-Grad (ours).

[5] Original derivation can be found in Definition 7 of [28].

[6] Denoted in the format $n\text{q}L\text{d}$ for an n-qubit, L-layer HEA. This notation is used universally in our paper.

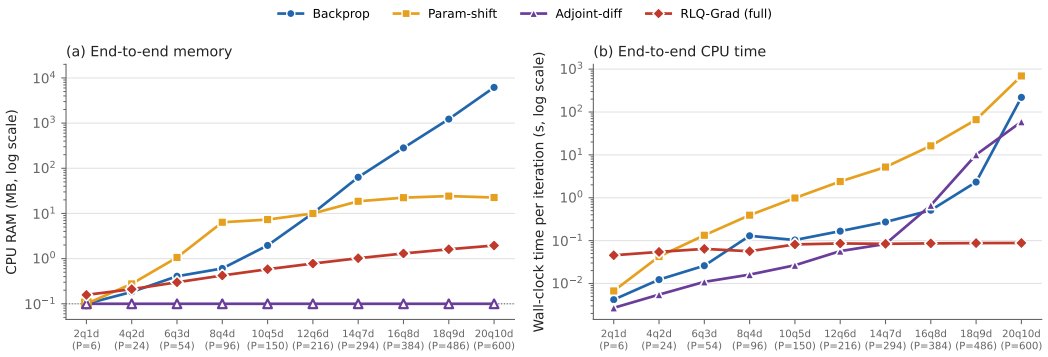


Figure 3: End-to-end memory (a, CPU RAM) and per-iteration CPU wall-clock time (b) of QNN gradient computation methods from 2 to 20 qubits (log scale, mean over 12 trials after 4 warm-up iterations). RLQ-Grad costs cover the full training pipeline: PPO rollout, actor and critic updates, gradient buffers, and Adam optimizer states. Hollow markers denote measurements below the 0.1 MB resolution. Exact values are listed in Appendix D.6.

As illustrated in Figure 2, our method exhibits robustness against decay, indicating a clear advantage over the baseline approaches.

Still, even when g_t is free of variance decay, does the agent's own training suffer from an analogous pathology? We can analyze this question by forming the follow theorem:

Theorem 2 (Plateau unrelated with agent optimization)**.** *The RL policy π_ϕ is a classical neural network (e.g. an MLP). Its parameter gradient $\nabla_\phi \mathcal{L}_{\mathrm{RL}}$ is computed entirely by classical backpropagation and is therefore not subject to the variance decay phenomenon.*

Proof for this theorem can be found at Appendix C.2. Taken together, Theorems 1 and 2 formalize a key advantage of RLQ-Grad: it decouples the optimization signal from the quantum circuit's intrinsic geometry while retaining a fully trainable classical learning mechanism. By construction, the surrogate gradient g_t is not constrained by the variance collapse characterized in Equation 8, allowing meaningful updates even in regimes where standard gradient-based methods fail. At the same time, the learning dynamics of the policy π_ϕ remain governed by well-behaved classical optimization, ensuring that no analogous decay emerges on the agent side. Other properties regarding training conditions and environment setting effects can be found at Appendix F.1 and F.2.

3.1.1 Resource advantages

Table 1: Theoretical complexity of gradient computation strategies for a depth-L, n-qubit QNN with P_{QNN} trainable parameters. Forward pass complexity is identical for all methods and therefore omitted. For an MLP agent of depth L_{MLP} and width d, $P_{\mathrm{RL}} = \mathcal{O}(L_{\mathrm{MLP}} d^2 + d\, P_{\mathrm{QNN}})$ counts actor and critic parameters.

Method	Time Complexity $\mathcal{O}_{\mathrm{time}}$	Memory Complexity $\mathcal{O}_{\mathrm{memory}}$
Backpropagation [21]	$L \cdot 2^n$	$L \cdot 2^n$
Parameter-shift [19]	$P_{\mathrm{QNN}} \cdot L \cdot 2^n$	$P_{\mathrm{QNN}} \cdot 2^n$
Adjoint differentiation [18]	$P_{\mathrm{QNN}} \cdot 2^n$	1
RLQ-Grad (actor inference)	$L_{\mathrm{MLP}} d^2 + d\, P_{\mathrm{QNN}}$	$d + P_{\mathrm{QNN}}$
RLQ-Grad (full pipeline)	P_{RL}	$4\, P_{\mathrm{RL}}$

To quantify how our method performs with respect to (w.r.t.) well-known gradient-based methods, Table 1 summarises the asymptotic cost of each gradient computation scheme of Figure 2. Backpropagation stores all L intermediate statevectors, incurring $\mathcal{O}(L \cdot 2^n)$ in both time and memory. Adjoint differentiation costs $\mathcal{O}_{\mathrm{time}}(P_{\mathrm{QNN}} \cdot 2^n)$ but reconstructs intermediate states on-the-fly via unitary inversion, so beyond the statevector already held by the forward pass it needs only a constant number of buffers, i.e. $\mathcal{O}_{\mathrm{memory}}(1)$ additional memory. The parameter-shift rule evaluates two full forward passes per parameter, giving $\mathcal{O}_{\mathrm{time}}(P_{\mathrm{QNN}} \cdot L \cdot 2^n)$ at $\mathcal{O}_{\mathrm{memory}}(P_{\mathrm{QNN}} \cdot 2^n)$, making it the least time/memory-efficient of the three for deep or wide circuits.

RLQ-Grad replaces explicit gradient propagation through the quantum circuit with a learned surrogate estimator. Since the agent operates entirely in the classical domain, its cost depends on its own parameter count P_{RL}, not on the Hilbert space dimension 2^n. Because both the state s_t and the action

Table 2: Comparison of top 1 validation accuracy (in percentage, averaged over 4 runs) of gradient computation strategies for a n-qubit, L-depth HEA. Best performance per variant is denoted in bold and second best is underlined.

Datasets	Algorithms	Top 1 validation accuracy per configuration					
		2q1d (Param = 6)	4q2d (Param = 24)	6q3d (Param = 54)	8q4d (Param = 96)	10q5d (Param = 150)	12q6d (Param = 216)
BC	Backpropagation [21]	80.62 ± 10.64	$\underline{90.86 \pm 1.998}$	$\underline{91.88 \pm 2.057}$	$\underline{91.88 \pm 1.126}$	91.96 ± 2.402	$\underline{93.15 \pm 2.942}$
	Parameter-shift [19]	$\underline{85.71 \pm 7.082}$	89.77 ± 1.316	89.60 ± 5.526	89.77 ± 2.010	91.99 ± 5.385	92.11 ± 5.526
	Adjoint-differentiation [18]	83.01 ± 11.64	84.93 ± 14.82	85.04 ± 3.689	87.32 ± 8.476	$\underline{92.75 \pm 10.28}$	93.02 ± 8.926
	RLQ-Grad (Ours)	$\mathbf{95.52 \pm 0.080}$	$\mathbf{95.20 \pm 2.379}$	$\mathbf{95.79 \pm 3.418}$	$\mathbf{95.96 \pm 0.380}$	$\mathbf{96.18 \pm 4.178}$	$\mathbf{96.58 \pm 5.612}$
MNIST	Backpropagation [21]	$\underline{59.24 \pm 3.171}$	78.52 ± 1.371	$\underline{88.81 \pm 0.434}$	$\underline{91.43 \pm 0.624}$	91.74 ± 0.305	$\underline{92.48 \pm 0.225}$
	Parameter-shift [19]	56.25 ± 0.419	$\underline{80.59 \pm 1.445}$	85.35 ± 0.447	87.14 ± 0.406	88.43 ± 0.181	88.52 ± 0.279
	Adjoint-differentiation [18]	58.16 ± 2.086	79.11 ± 1.902	87.42 ± 0.613	90.85 ± 0.771	$\underline{92.19 \pm 0.151}$	91.96 ± 0.412
	RLQ-Grad (Ours)	$\mathbf{61.36 \pm 2.216}$	$\mathbf{83.12 \pm 0.375}$	$\mathbf{90.76 \pm 0.462}$	$\mathbf{91.78 \pm 0.218}$	$\mathbf{93.63 \pm 0.114}$	$\mathbf{95.27 \pm 0.297}$
F-MNIST	Backpropagation [21]	$\underline{72.87 \pm 0.312}$	74.77 ± 0.573	$\underline{82.34 \pm 0.441}$	81.36 ± 0.407	84.62 ± 0.271	$\underline{85.72 \pm 0.198}$
	Parameter-shift [19]	68.84 ± 0.154	72.63 ± 0.601	79.75 ± 0.518	$\underline{84.12 \pm 0.336}$	82.41 ± 0.295	82.98 ± 0.266
	Adjoint-differentiation [18]	73.23 ± 0.576	$\underline{75.96 \pm 0.528}$	81.28 ± 0.462	83.54 ± 0.388	$\underline{85.08 \pm 0.221}$	85.13 ± 0.243
	RLQ-Grad (Ours)	$\mathbf{73.64 \pm 0.415}$	$\mathbf{76.43 \pm 0.442}$	$\mathbf{83.15 \pm 0.396}$	$\mathbf{85.87 \pm 0.285}$	$\mathbf{86.94 \pm 0.204}$	$\mathbf{87.62 \pm 0.181}$
CIFAR10	Backpropagation [21]	$\underline{24.69 \pm 1.429}$	$\underline{33.68 \pm 0.990}$	35.02 ± 0.884	$\underline{39.78 \pm 0.461}$	$\underline{40.36 \pm 0.375}$	38.05 ± 0.601
	Parameter-shift [19]	26.34 ± 1.585	32.11 ± 1.102	$\underline{37.34 \pm 0.475}$	37.45 ± 0.722	38.12 ± 0.653	$\underline{40.08 \pm 0.321}$
	Adjoint-differentiation [18]	25.06 ± 0.632	31.87 ± 0.955	34.76 ± 0.801	36.98 ± 0.690	37.65 ± 0.588	37.52 ± 0.544
	RLQ-Grad (Ours)	$\mathbf{27.17 \pm 0.782}$	$\mathbf{34.84 \pm 0.744}$	$\mathbf{37.91 \pm 0.533}$	$\mathbf{40.72 \pm 0.428}$	$\mathbf{41.63 \pm 0.367}$	$\mathbf{41.15 \pm 0.352}$

g_t have dimension $\mathcal{O}(P_{\text{QNN}})$, only the input and output layers of the MLP depend on the circuit size, while the hidden layers keep a fixed width d. Proposing a gradient (actor inference) therefore costs $\mathcal{O}(L_{\text{MLP}} d^2 + d\, P_{\text{QNN}})$ time. A complete training step must also hold the critic, the gradient buffers, and the two Adam moment estimates, so its memory is $\approx 4P_{\text{RL}}$. Both costs grow linearly in P_{QNN}, hence polynomially in n, and are independent of 2^n, shifting the computational burden from quantum simulation to classical function approximation. Full derivations are given in Appendix E.

Figure 3 reports the measured end-to-end cost for $n = 2$ to 20. Backpropagation grows from 0.0042s to 219.41s per iteration, following its $\mathcal{O}_{\text{time}}(L \cdot 2^n)$ complexity; adjoint differentiation grows from 0.0027s to 59.32s, and parameter-shift, which requires $2P_{\text{QNN}}$ circuit evaluations, from 0.0067s to 693.96s. The full RLQ-Grad pipeline, roughly $50\times$ the cost of an actor-only forward pass, instead stays between 0.0455s and 0.0881s across all configurations. Its fixed overhead makes it slower than analytic differentiation on the smallest circuits, but it overtakes parameter-shift from 6q3d, backpropagation from 8q4d, and adjoint differentiation from 16q8d (the two are tied at 14q7d). At $n = 20$, RLQ-Grad is $\sim 2490\times$, $\sim 7876\times$, and $\sim 673\times$ faster than backpropagation, parameter-shift, and adjoint differentiation, respectively. RLQ-Grad also runs natively on GPU, whereas parameter-shift and adjoint differentiation lack efficient GPU kernels in current QNN frameworks; on an RTX 3090 it is faster than GPU backpropagation at every scale ($1.03\times$ to $1.22\times$, Appendix D.6).

Memory consumption further delineates the scalability of these methods. Backpropagation grows exponentially from below 0.1 MB to 6174 MB at $n = 20$, while parameter-shift saturates around 20 MB. Adjoint differentiation stays below 0.1 MB at every scale, consistent with its $\mathcal{O}(1)$ additional memory, and remains the most memory-efficient method. The full RLQ-Grad pipeline grows from 161.22 KB to 1.95 MB, i.e. by ≈ 3 KB per QNN parameter, confirming the predicted linear dependence on P_{QNN}. This footprint decomposes into actor and critic parameters ($\approx 26\%$), gradient buffers ($\approx 26\%$), and Adam optimizer states ($\approx 52\%$); at 12q6d, for example, the total of 796.14 KB is roughly $4\times$ the actor-only 102.28 KB. Even under this full accounting, RLQ-Grad scales strictly better than backpropagation and parameter-shift, using $\sim 3166\times$ and $\sim 12\times$ less memory at $n = 20$.

3.2 Benchmarking against well-known quantum optimization algorithms

Table 2 and Figure 5 (Appendix D.4) present a comprehensive comparison of gradient computation strategies across multiple datasets for 200 epochs, including: BreastCancer (BC) [29], Fashion-MNIST (F-MNIST) [30], MNIST [31] and CIFAR10 [32], along with HEA configurations across gradient computation methods. Except for model trained on BC dataset with use a batch size of 128 due to small sample size, all dataset used a batch size of 256. We also use a fixed random seed sampler for dataset without predetermined split such as BC, and reduce input dimensionality with PCA (Appendix D.3).

Overall, RLQ-Grad consistently achieves the best performance across all datasets and configurations, demonstrating both strong accuracy and scalability as the number of qubits and circuit depth increase. Across all datasets, accuracy generally improves as the circuit size increases (from $2q1d$ to $12q6d$),

Table 3: CIFAR-10 top-1 validation accuracy (in percentage, mean $\pm$ std over 3 runs) at larger circuit scales. Best performance per configuration is denoted in bold and second best is underlined.

Type	Method	Top 1 validation accuracy per configuration			
		14q7d (Param = 294)	16q8d (Param = 384)	18q9d (Param = 486)	20q10d (Param = 600)
Gradient-based	Backpropagation [21]	38.72 ± 3.21	40.45 ± 2.67	39.86 ± 3.48	40.13 ± 2.91
Gradient-free	CMA-ES [33]	10.10 ± 0.14	10.12 ± 0.21	10.11 ± 0.20	10.13 ± 0.15
	sep-CMA-ES [34]	10.09 ± 0.23	10.14 ± 0.17	10.11 ± 0.26	10.15 ± 0.19
	SPSA [35]	10.08 ± 0.27	10.15 ± 0.18	10.11 ± 0.32	10.14 ± 0.24
BP mitigation	Layerwise [16]	44.73 ± 3.21	$\underline{48.58 \pm 2.74}$	$\underline{50.92 \pm 3.47}$	$\mathbf{53.16 \pm 3.65}$
	Gaussian Init [17]	43.92 ± 3.15	46.87 ± 2.38	49.76 ± 3.02	52.41 ± 2.71
Learnable (ours)	**RLQ-Grad**	$\mathbf{45.86 \pm 3.84}$	$\mathbf{48.64 \pm 2.46}$	$\mathbf{51.29 \pm 3.16}$	$\underline{53.02 \pm 2.83}$

indicating that higher parameter counts enhance representational capacity. Conventional methods such as backpropagation, parameter-shift, and adjoint differentiation exhibit diminishing returns or instability at higher configurations, whereas RLQ-Grad maintains steady improvements.

On BC, RLQ-Grad outperforms all baselines at every configuration, reaching up to 96.58% with a margin of $+2\%$ to $+10\%$ over the second-best method. On MNIST, the gap widens with circuit size, and RLQ-Grad surpasses the competitive backpropagation and adjoint differentiation by up to $\sim 3\%$ at 12q6d. On F-MNIST, RLQ-Grad again achieves the highest accuracy, reaching 87.62%. On CIFAR10, overall accuracies are lower due to dataset complexity, but RLQ-Grad still gives the best results at every configuration, with more modest ($\sim 2\%$ to 4%) but consistent gains.

Backpropagation generally performs well and often ranks second, especially in simpler datasets (BC, MNIST). Parameter-shift shows competitive performance at smaller scales but struggles to scale efficiently. Adjoint differentiation exhibits higher variance and inconsistent ranking, particularly in deeper circuits. In contrast, RLQ-Grad not only achieves the highest accuracy but also maintains relatively stable variance across runs. A key advantage of RLQ-Grad is its scalability, where performance continues to improve with increasing circuit size without the degradation or saturation observed in other methods. Additionally, its relatively low standard deviation (especially on MNIST, F-MNIST, and CIFAR10) indicates stable training dynamics.

3.3 Scaling to 20 qubits and comparison with stronger baselines

To probe the regime where barren plateaus become acute and where the action space of the agent is largest, we extend CIFAR-10 to $n = 14, 16, 18, 20$ qubits ($P_{\text{QNN}} = 294, 384, 486, 600$) and add two families of baselines: evolutionary and gradient-free optimizers (CMA-ES [33], sep-CMA-ES [34], and SPSA [35]), and QNN-specific barren plateau mitigation (layerwise learning [16] and Gaussian initialization [17]). Results are reported in Table 3.

3.3.1 Action-space scalability and non-stationarity

RLQ-Grad remains stable and improves monotonically up to $P_{\text{QNN}} = 600$. The action dimensionality enters the agent only through its input and output layers, while the hidden layers keep a fixed width, so the agent grows linearly and its training difficulty grows sub-linearly in P_{QNN}. The environment is non-stationary because the QNN changes under the agent's own actions; PPO with SN handles this through (i) on-policy rollouts, which avoid off-policy staleness, (ii) the clipped surrogate objective, which bounds the divergence between successive policies, and (iii) SN on the critic, which bounds the propagation of TD errors (Appendix D.2). The standard deviations of 2.46 to 3.84 in Table 3, comparable to those of backpropagation and the mitigation baselines, confirm that training remains stable at this scale.

3.3.2 Why reinforcement learning rather than evolutionary search

CMA-ES, sep-CMA-ES, and SPSA collapse to chance ($\sim 10\%$) at every scale. This is consistent with [36]: gradient-free optimizers cannot escape the exponential shot-budget scaling of high-dimensional barren plateaus, and the $\mathcal{O}(P_{\text{QNN}}^2)$ covariance estimation of CMA-ES becomes prohibitive for $P_{\text{QNN}} \geqslant 300$. RLQ-Grad avoids both issues because its amortized policy learns a state-conditional update rule instead of searching over a covariance model. Layerwise learning and Gaussian initializa-

tion reach accuracy comparable to RLQ-Grad, but they remain backpropagation variants that inherit its $\mathcal{O}(L \cdot 2^n)$ time and memory (Figure 3). RLQ-Grad matches their accuracy while retaining its computational advantages, which positions it as complementary to landscape-shaping approaches rather than as a replacement.

3.3.3 Are the benchmarks in the barren plateau limited regime?

The range $n = 2$ to 12 of Table 2 is best characterized as an *early-onset* rather than a *fully saturated* barren plateau regime. Figure 2 shows that gradient variance already decays with slopes of -0.85 (backpropagation), -0.30 (parameter-shift), and -0.14 (adjoint differentiation) over this range, versus ≈ 0 for RLQ-Grad, but the decay has not yet dominated final accuracy; this is expected, since barren plateaus become acute at $n \gtrsim 20$ for global observables in generic HEAs [23]. The extended experiments show the pattern predicted if barren plateaus are the operative bottleneck. Backpropagation gains only ~ 2 pp from 12q6d to 20q10d ($38.05\% \rightarrow 40.13\%$) although P_{QNN} nearly triples ($216 \rightarrow 600$), compared with ~ 13 pp from 2q1d to 12q6d ($24.69\% \rightarrow 38.05\%$); its rate of improvement collapses where the barren plateau is expected to activate. Over the same range, layerwise learning and Gaussian initialization gain ~ 8.5 pp, and RLQ-Grad follows the same trajectory ($45.86\% \rightarrow 53.02\%$).

These observations also separate learned-optimizer dynamics from barren plateau mitigation. If RLQ-Grad won through its optimizer dynamics alone, it should outperform mitigation methods that lack such dynamics; instead, the three methods stay within 2 pp of each other from 14q7d to 20q10d, which suggests that all three succeed through the same mechanism, namely avoiding the variance collapse. Conversely, if the advantage were purely a matter of avoiding circuit gradients, any gradient-free method would succeed, which CMA-ES and SPSA rule out. Gradient-freeness (Theorem 1) is therefore necessary but not sufficient, and the contribution of RLQ-Grad is to combine it with amortized policy learning that can handle a high-dimensional action space.

4 Limitations and discussions

Our analysis identifies three concrete regimes in which RLQ-Grad does not offer the advantage suggested by Theorems 1 and 2, and two practical caveats that temper the scope of the reported gains.

Reward concentration in deep barren-plateau regimes. Theorem 1 establishes that the surrogate gradient g_t is not constrained by the Jordan-algebraic variance bound because g_t is emitted by a classical network rather than computed by differentiating through $U(\theta)$. However, as we show in Proposition 1 (Appendix F.2), the reward $r_t = \mathrm{acc}_t + (\ell_t + \epsilon)^{-1}$ is itself a function of the loss ℓ_t, and Wishart-process description implies that ℓ_t concentrates exponentially whenever the circuit is on a barren plateau [28]. In that regime the reward differences used to drive policy gradients vanish at the same rate as conventional cost-function differences, echoing the result of [36] that no gradient-free optimizer (including surrogate-gradient models) can escape the exponential shot-budget scaling of a barren plateau. RLQ-Grad therefore delivers a genuine landscape advantage only when the Jordan algebra $\mathcal{A}$ leaves the cost non-flat (Condition 1 of Theorem 4); when the cost is truly exponentially flat, RLQ-Grad retains only its computational advantage.

Poor local minima are not bypassed. Theorem 5 (Appendix G.1) shows that, independent of gradient quality, the density of good local minima is exponentially suppressed whenever the over-parameterization condition $p \geqslant \max_\alpha \beta_\alpha r_\alpha$ fails. Because RLQ-Grad still converges to a local minimum of the empirical loss, it inherits this obstruction. Figure 6 makes this visible: on the underparameterized F-MNIST configuration the loss surface fragments into sharp basins for every method, RLQ-Grad included.

Input-gradient decay propagates to classical pre-processing. The hybrid architecture we use funnels classical features through an FC layer $\mathrm{FC}_{\mathrm{in}}$ before the R_x encoding. Theorem 6 and Corollary 2 (Appendix G.2) show that whenever $U(\theta)$ forms (at least) a 2-design, the input-space Jacobian $\mathbb{E}_\theta[\|\nabla_x \mathcal{L}\|^2] \leqslant Cn/2^n$ collapses exponentially; all classical layers upstream of the quantum encoding therefore receive vanishing gradients.

Empirical scope. All experiments are conducted on statevector simulators with shot-noise-free measurement: we do not report results on physical hardware, under realistic noise channels, or on non-classification tasks (VQE, QAOA, generative modeling). The four-dataset benchmark of Table 2 spans $n = 2$ to 12, an early-onset rather than fully saturated barren plateau regime; only CIFAR-10 and the resource benchmarks of Figure 3 are extended to $n = 20$, which is where the barren plateau starts to become acute for global observables in generic HEAs [23]. At that scale RLQ-Grad matches, but does not exceed, dedicated mitigation methods such as layerwise learning (Table 3), so its advantage over them is computational rather than in accuracy. Finally, the fixed per-iteration overhead of the full RLQ-Grad pipeline makes it slower than analytic differentiation on small circuits (below 6 to 8 qubits on CPU), and adjoint differentiation remains more memory-efficient at every scale we tested.

Transferability and agent cost. The agents reported in Tables 2 and 3 are trained per configuration: a new agent is required for every choice of (n, L) and dataset. The cost of training the agent alongside the QNN (rollouts, actor and critic updates, optimizer states) is included in Figure 3, but it is not amortized across configurations. A learned-optimizer style meta-training regime that amortizes the agent across circuit scales and tasks (in the spirit of [37] and [1]) is a natural next step but is outside the present scope.

5 Conclusion

We propose RLQ-Grad, a reinforcement-learning optimizer that replaces explicit differentiation of parameterized quantum circuits with a classical policy that directly proposes parameter updates. The key insight is that barren plateaus arise from the distribution of $\partial_\mu \ell$ induced by $U(\theta)$; by sampling updates from a policy π_ϕ, RLQ-Grad bypasses this distribution entirely, avoiding the associated exponential variance collapse. Since π_ϕ is a finite-width MLP, its training follows standard classical concentration behavior independent of the number of qubits n, preventing analogous pathologies.

Trained with PPO and spectral normalization, RLQ-Grad achieves three main outcomes: (i) near-constant gradient variance (slope ≈ 0) across $n = 2$ to 12, compared to decay rates of -0.85, -0.30, and -0.14 for backpropagation, parameter-shift, and adjoint differentiation; (ii) end-to-end time and memory that grow only linearly in the number of QNN parameters and are independent of the Hilbert space dimension, yielding speedups of up to $7876\times$ and under 2 MB of memory at $n = 20$ once the full agent training pipeline is accounted for; and (iii) accuracy gains of 2% to 10% over gradient-based baselines across four supervised benchmarks, together with parity with dedicated barren plateau mitigation methods on CIFAR-10 up to 20 qubits, where evolutionary and gradient-free optimizers fail.

These results suggest that decoupling the optimization signal from quantum circuit geometry is a viable and complementary strategy to landscape-shaping approaches such as local costs, layerwise training, or structured initialization, matching their accuracy at a fraction of their simulation cost. However, RLQ-Grad does not address loss concentration, underparameterization, or gradient collapse in classical preprocessing layers. Future work includes meta-training across circuit scales, incorporating locality-aware rewards, extending to generative and variational tasks, and evaluating performance on noisy quantum hardware.

Acknowledgments and Disclosure of Funding

This research was financially sponsored in part by VinUniversity and The University of Aizu, Nhan Trong Luu was co-sponsored by both Can Tho University and VinUniversity.

References

[1] G. Verdon, M. Broughton, J. R. McClean, K. J. Sung, R. Babbush, Z. Jiang, H. Neven, and M. Mohseni, "Learning to learn with quantum neural networks via classical neural networks," *arXiv preprint arXiv:1907.05415*, 2019.

[2] M. Kölle, T. Witter, T. Rohe, G. Stenzel, P. Altmann, and T. Gabor, "A study on optimization techniques for variational quantum circuits in reinforcement learning," in *2024 IEEE International Conference on Quantum Software (QSW)*, pp. 157–167, IEEE, 2024.

[3] M. Ostaszewski, L. M. Trenkwalder, W. Masarczyk, E. Scerri, and V. Dunjko, "Reinforcement learning for optimization of variational quantum circuit architectures," *Advances in neural information processing systems*, vol. 34, pp. 18182–18194, 2021.

[4] S. Foderà, G. Turati, R. Nembrini, M. F. Dacrema, and P. Cremonesi, "Reinforcement learning for variational quantum circuits design," *arXiv preprint arXiv:2409.05475*, 2024.

[5] A. Kundu and S. Mangini, "Tensorrl-qas: Reinforcement learning with tensor networks for improved quantum architecture search," *arXiv preprint arXiv:2505.09371*, 2025.

[6] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature communications*, vol. 9, no. 1, p. 4812, 2018.

[7] S. Khairy, R. Shaydulin, L. Cincio, Y. Alexeev, and P. Balaprakash, "Learning to optimize variational quantum circuits to solve combinatorial problems," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 2367–2375, 2020.

[8] S. Khairy, R. Shaydulin, L. Cincio, Y. Alexeev, and P. Balaprakash, "Reinforcement-learning-based variational quantum circuits optimization for combinatorial problems," *arXiv preprint arXiv:1911.04574*, 2019.

[9] M. M. Wauters, E. Panizon, G. B. Mbeng, and G. E. Santoro, "Reinforcement-learning-assisted quantum optimization," *Physical Review Research*, vol. 2, no. 3, p. 033446, 2020.

[10] T. Fösel, M. Y. Niu, F. Marquardt, and L. Li, "Quantum circuit optimization with deep reinforcement learning," *arXiv preprint arXiv:2103.07585*, 2021.

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

[12] O. Lockwood, "Optimizing quantum variational circuits with deep reinforcement learning," *arXiv preprint arXiv:2109.03188*, 2021.

[13] X. Dai, T.-C. Wei, S. Yoo, and S. Y.-C. Chen, "Quantum machine learning architecture search via deep reinforcement learning," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 1, pp. 1525–1534, IEEE, 2024.

[14] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, "Data re-uploading for a universal quantum classifier," *Quantum*, vol. 4, p. 226, 2020.

[15] F. S. Gharehchopogh, "Quantum-inspired metaheuristic algorithms: comprehensive survey and classification," *Artificial Intelligence Review*, vol. 56, no. 6, pp. 5479–5543, 2023.

[16] A. Skolik, J. R. McClean, M. Mohseni, P. Van Der Smagt, and M. Leib, "Layerwise learning for quantum neural networks," *Quantum Machine Intelligence*, vol. 3, no. 1, p. 5, 2021.

[17] K. Zhang, L. Liu, M.-H. Hsieh, and D. Tao, "Escaping from the barren plateau via gaussian initializations in deep variational quantum circuits," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 18612–18627, 2022.

[18] T. Jones and J. Gacon, "Efficient calculation of gradients in classical simulations of variational quantum algorithms," *arXiv preprint arXiv:2009.02823*, 2020.

[19] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, p. 032309, 2018.

[20] L. Leone, S. F. Oliviero, L. Cincio, and M. Cerezo, "On the practical usefulness of the hardware efficient ansatz," *Quantum*, vol. 8, p. 1395, 2024.

[21] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, J. M. Arrazola, U. Azad, S. Banning, C. Blank, T. R. Bromley, B. A. Cordier, J. Ceroni, A. Delgado, O. D. Matteo, A. Dusko, T. Garg, D. Guala, A. Hayes, R. Hill, A. Ijaz, T. Isacsson, D. Ittah, S. Jahangiri, P. Jain, E. Jiang, A. Khandelwal, K. Kottmann, R. A. Lang, C. Lee, T. Loke, A. Lowe, K. McKiernan, J. J. Meyer, J. A. Montañez-Barrera, R. Moyard, Z. Niu, L. J. O'Riordan, S. Oud, A. Panigrahi, C.-Y. Park, D. Polatajko, N. Quesada, C. Roberts, N. Sá, I. Schoch, B. Shi, S. Shu, S. Sim, A. Singh, I. Strandberg, J. Soni, A. Száva, S. Thabet, R. A. Vargas-Hernández, T. Vincent, N. Vitucci, M. Weber, D. Wierichs, R. Wiersema, M. Willmann, V. Wong, S. Zhang, and N. Killoran, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2018.

[22] H. Wang, Y. Ding, J. Gu, Z. Li, Y. Lin, D. Z. Pan, F. T. Chong, and S. Han, "Quantumnas: Noise-adaptive search for robust quantum circuits," in *The 28th IEEE International Symposium on High-Performance Computer Architecture (HPCA-28)*, 2022.

[23] M. Larocca, S. Thanasilp, S. Wang, K. Sharma, J. Biamonte, P. J. Coles, L. Cincio, J. R. McClean, Z. Holmes, and M. Cerezo, "Barren plateaus in variational quantum computing," *Nature Reviews Physics*, vol. 7, no. 4, pp. 174–189, 2025.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[25] F. Gogianu, T. Berariu, M. C. Rosca, C. Clopath, L. Busoniu, and R. Pascanu, "Spectral normalisation for deep reinforcement learning: an optimisation perspective," in *International Conference on Machine Learning*, pp. 3734–3744, PMLR, 2021.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[28] E. R. Anschuetz, "A unified theory of quantum neural network loss landscapes," *arXiv preprint arXiv:2408.11901*, 2024.

[29] M. Zwitter and M. Soklic, "Breast Cancer." UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C51P4M.

[30] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[31] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[32] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[33] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[34] M. N. Omidvar and X. Li, "A comparative study of cma-es on large scale global optimisation," in *Australasian Joint Conference on Artificial Intelligence*, pp. 303–312, Springer, 2010.

[35] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *IEEE Transactions on aerospace and electronic systems*, vol. 34, no. 3, pp. 817–823, 1998.

[36] A. Arrasmith, M. Cerezo, P. Czarnik, L. Cincio, and P. J. Coles, "Effect of barren plateaus on gradient-free optimization," *Quantum*, vol. 5, p. 558, 2021.

[37] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," *Advances in neural information processing systems*, vol. 29, 2016.

[38] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[39] N. T. Luu, D. T. Luu, N. N. Pham, and T. C. Truong, "Parameter efficient hybrid spiking-quantum convolutional neural network with surrogate gradient and quantum data-reupload," *PeerJ Computer Science*, vol. 12, p. e3554, 2026.

[40] C.-Y. Park, M. Kang, and J. Huh, "Hardware-efficient ansatz without barren plateaus in any depth," *arXiv preprint arXiv:2403.04844*, 2024.

[41] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, "Cost function dependent barren plateaus in shallow parametrized quantum circuits," *Nature communications*, vol. 12, no. 1, p. 1791, 2021.

[42] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti, "An initialization strategy for addressing barren plateaus in parametrized quantum circuits," *Quantum*, vol. 3, p. 214, 2019.

[43] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[44] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, Pmlr, 2018.

[45] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, pp. 1587–1596, PMLR, 2018.

[46] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. P. Wierstra, "Continuous control with deep reinforcement learning," Sept. 15 2020. US Patent 10,776,692.

[47] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[48] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[49] N. Bjorck, C. P. Gomes, and K. Q. Weinberger, "Towards deeper deep reinforcement learning with spectral normalization," *Advances in neural information processing systems*, vol. 34, pp. 8242–8255, 2021.

[50] N. T. Luu, T. Moreaux, V. Khaustov, V. Khaustova, M. Mataar, T. Krueger, Z. S. Raphael, J. Wang, D. T. Luu, P. N. Nam, and T. C. Thang, "Robust reinforcement learning quadcopter control system for offshore cement wind turbine construction," *IEEE Access*, vol. 14, pp. 51918–51942, 2026.

[51] C. Lee, "Adjoint differentiation." PennyLane Demos, Nov 2021. Last updated: September 22, 2025.

[52] E. Fontana, D. Herman, S. Chakrabarti, N. Kumar, R. Yalovetzky, J. Heredge, S. H. Sureshbabu, and M. Pistoia, "Characterizing barren plateaus in quantum ansätze with the adjoint representation," *Nature Communications*, vol. 15, no. 1, p. 7171, 2024.

[53] M. Ragone, B. N. Bakalov, F. Sauvage, A. F. Kemper, C. Ortiz Marrero, M. Larocca, and M. Cerezo, "A lie algebraic theory of barren plateaus for deep parameterized quantum circuits," *Nature Communications*, vol. 15, no. 1, p. 7172, 2024.

[54] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, "Noise-induced barren plateaus in variational quantum algorithms," *Nature communications*, vol. 12, no. 1, p. 6961, 2021.

[55] L. Metz, N. Maheswaranathan, C. D. Freeman, B. Poole, and J. Sohl-Dickstein, "Tasks, stability, architecture, and compute: Training more effective learned optimizers, and using them to train themselves," *arXiv preprint arXiv:2009.11243*, 2020.

A Universal Approximation Capability of HEA

The expressive power of the HEA circuit adopted in this work is grounded in the Universal Approximation Theorem (UAT) [38], which we restate for completeness.

Theorem 3 ([38]). *Let $I_m = [0,1]^m$ and let $\varphi : \mathbb{R} \to \mathbb{R}$ be any nonconstant, bounded, and continuous function. Then for every $f \in \mathcal{C}(I_m)$ and every $\varepsilon > 0$, there exist $N \in \mathbb{N}$, $\alpha_i, b_i \in \mathbb{R}$, and $\vec{w}_i \in \mathbb{R}^m$ such that the single-hidden-layer network*

$$h(\vec{x}) = \sum_{i=1}^{N} \alpha_i \, \varphi(\vec{w}_i \cdot \vec{x} + b_i) \tag{9}$$

satisfies $\sup_{\vec{x} \in I_m} |h(\vec{x}) - f(\vec{x})| < \varepsilon$.

PQCs admit an analogous universality result. This construction endows even single-qubit QNNs with universal approximation capability over compact domains [14, 39]. The HEA used in this work inherits this property: with sufficient depth and data re-uploading, it can approximate arbitrary continuous classification functions to any prescribed precision.

B Quantum Barren Plateaus

Barren plateaus [23] refer to the exponential concentration of PQC loss landscapes, wherein both output and gradient variances vanish exponentially in the number of qubits n, rendering gradient-based optimization infeasible at scale.

Definition 1 ([23]). *Let $U(\theta)$ be a PQC acting on an n-qubit state ρ, and let $f_\theta(\rho, O) = \mathrm{Tr}[O\,U(\theta)\rho U^\dagger(\theta)]$ for an observable O. The circuit is said to exhibit a* barren plateau *over base b if*

$$\mathrm{Var}_\theta[f_\theta(\rho, O)] \;\; or \;\; \mathrm{Var}_\theta[\partial_\mu f_\theta(\rho, O)] \;\in\; \mathcal{O}\big(b^{-n}\big)\,, \tag{10}$$

where $b > 1$ depends on the choice of ρ and O.

The gradient variance bound in Equation 10 implies that the number of circuit evaluations required to estimate $\partial_\mu f$ to fixed precision grows exponentially in n, making training intractable on near-term hardware. This concentration is mechanistically linked to the degree to which the parameter distribution approximates a unitary t-design: deep HEA circuits approach a 2-design [40], which is sufficient to force $\mathrm{Var}_\theta[\partial_\mu f] \in \mathcal{O}(2^{-n})$ for global observables [23]. Mitigation strategies include the use of local cost functions [41], structured initialization [42], layer-wise training [16], and hybrid architectures [39].

C Proofs of main theorems

C.1 Proof of Theorem 1

In standard gradient-based QNN training, the parameter update is:

$$\theta_{t+1} = \theta_t - \eta\,\nabla_\theta \mathcal{L}(\theta_t), \tag{11}$$

and the barren plateau renders $\nabla_\theta \mathcal{L}(\theta_t) \approx 0$ exponentially in n, stalling optimisation[7]. In RLQ-Grad, the update is instead

$$\theta_{t+1} = \theta_t - \eta\, g_t, \qquad g_t \sim \pi_\phi(\cdot \mid s_t). \tag{12}$$

The quantity g_t is the output of the policy network π_ϕ, a purely classical computation. Its distribution is determined by the agent parameters ϕ and the state s_t, neither of which involves differentiating through $U(\theta)$. Consequently, g_t is statistically independent of $\nabla_\theta \mathcal{L}$, and the barren plateau variance bound $\mathrm{Var}_\theta[\partial_\mu \mathcal{L}] \in \mathcal{O}(b^{-n})$ places no constraint on g_t. The update therefore proceeds regardless of circuit depth or qubit count.

The barren plateau condition Equation 8 is a statement about the distribution of $\partial_\mu \ell$ over the parameter ensemble induced by $U(\theta)$. In RLQ-Grad, the gradient applied to θ_t is $g_t = \pi_\phi(s_t)$, a function of the RL agent's classical parameters ϕ and state s_t. Neither depends on differentiating through the quantum circuit and consequently, the Wishart process concentration result[8] has no bearing on g_t. The update proceeds regardless of the algebraic structure of $\mathcal{A}$. [28] also show that the precise scaling is governed by $\dim_{\mathbb{R}}(\mathrm{Aut}(\mathcal{A}_\alpha))$ and the entanglement structure of ρ with respect to $\mathcal{A}$ (through $\mathrm{Tr}(\rho^\alpha)^2$), not solely by circuit depth.

Note that Theorem 1 is also structural: it holds for any RL policy π_ϕ and any circuit $U(\theta)$, irrespective of whether the agent produces useful gradient estimates. It establishes only that the exponential vanishing pathology is absent from the update rule itself.

C.2 Proof of Theorem 2

The policy loss $\mathcal{L}_{\mathrm{RL}}(\phi)$ depends on ϕ only through the classical computation $\pi_\phi(s_t)$. Differentiating via the chain rule gives

$$\nabla_\phi \mathcal{L}_{\mathrm{RL}} = \sum_t \frac{\partial \mathcal{L}_{\mathrm{RL}}}{\partial g_t}\frac{\partial g_t}{\partial \phi}, \tag{13}$$

where $\partial g_t / \partial \phi$ is the Jacobian of the MLP output with respect to its weights. This quantity is governed by classical backpropagation through a feedforward network and satisfies standard gradient

[7] Noted in previous analysis of [6], where they characterised the barren plateau as $\mathrm{Var}_\theta[\partial_\mu \mathcal{L}] \in \mathcal{O}(2^{-n})$, arising from approximate 2-design behaviour at large depth.

[8] Which is Equation 8, originally derived from Corollary 4 of [28].

Table 4: Preliminary studies on different agent type with RLQ-Grad. l_{rollout} is correspondingly 400 for BC, 46800 for MNIST and FMNIST, 39000 for CIFAR10.

Agents	Buffer type	Buffer size	Params	MACs	Top 1 validation accuracy per dataset			
					BC	MNIST	F-MNIST	CIFAR10
SAC	Replay	1000000	17,038	16,640	95.61 ± 1.519	65.53 ± 0.521	72.27 ± 0.437	28.64 ± 0.516
TD3	Replay	1000000	16,648	16,256	96.71 ± 0.380	68.55 ± 0.989	73.52 ± 0.367	28.72 ± 0.287
DDPG	Replay	1000000	11,079	10,816	85.53 ± 6.837	55.23 ± 0.372	65.28 ± 0.098	23.14 ± 0.457
TRPO	Rollout	l_{rollout}	10,695	10,432	93.20 ± 1.899	60.27 ± 0.298	70.65 ± 0.428	26.46 ± 0.623
PPO	Rollout	l_{rollout}	5,575	5,440	90.05 ± 4.760	59.52 ± 0.212	68.52 ± 0.534	25.12 ± 0.961

concentration bounds for finite-width MLPs. In particular, its variance is polynomial in the network width d and depth L_{MLP}, not exponential in n. Since no quantum circuit evaluation appears in this computation, no gradient variance decay stemming from quantum barren plateau can arise.[9]

D Miscellaneous and settings

D.1 Preliminary studies and agent training settings

In our main experiments, we train the agent using the Proximal Policy Optimization (PPO) [24] algorithm with the default multilayer perceptron policy as implemented in Stable-Baselines3 (SB3) [43]. Justifications for this selection are made based on the results of our preliminary study across different model types, as noted in Table 4.

Table 4 shows a consistent performance hierarchy across all datasets, with off-policy methods (Soft Actor-Critic (SAC) [44], Twin Delayed Deep Deterministic (TD3) [45], Deep Deterministic Policy Gradient (DDPG) [46]) outperforming on-policy counterparts (Trust Region Policy Optimization (TRPO) [47] and PPO). In particular, TD3 achieves the best overall accuracy on all four datasets, followed closely by SAC, while DDPG exhibits the weakest performance among replay-based methods. On-policy methods (TRPO and PPO) consistently lag behind, indicating a performance gap between replay-buffer-based and rollout-based training strategies under the same architectural constraints.

A clear distinction emerges between replay and rollout buffers. Replay-based agents (SAC, TD3, DDPG) significantly outperform rollout-based agents (TRPO, PPO) across all datasets. This trend is especially pronounced on more complex datasets such as CIFAR10, where TD3 and SAC achieve approximately 28.7% accuracy, compared to 26.46% for TRPO and 25.12% for PPO.

Despite relatively small differences in parameter count and MACs, performance varies substantially. TD3 and SAC have comparable computational costs ($\sim$16k parameters and MACs), yet TD3 consistently achieves slightly better accuracy with lower variance. DDPG, while having fewer parameters ($\sim$11k), suffers from significantly degraded performance and higher variance.

All methods achieve high accuracy on the simplest dataset (BC), with TD3 reaching 96.71%, while performance drops significantly as dataset complexity increases. The relative ordering of methods, however, remains consistent across datasets. Variance across runs further differentiates the methods. TD3 demonstrates both high accuracy and low variance, indicating stable training dynamics. In contrast, DDPG and PPO exhibit larger standard deviations (e.g., 6.837 and 4.760 on BC).

On the other hand, PPO achieves competitive performance relative to TRPO despite having nearly half the parameters and MACs (5,575 vs. 10,695), making them an interesting candidate to further improve on. Furthermore, while achieving higher accuracy in overall, agents that used replay buffer (such as SAC, TD3 and DDPG) are significantly more memory consuming on CPU/GPU in comparison with those that use rollout buffers, eventually failing the promise of low resource consumption that we made initially.

All benchmarked policy and value networks are jointly optimized using stochastic gradient descent with a fixed learning rate of $5e-4$. The minibatch size is statically set to be equal with

[9] Still, note that Theorem 2 does not assert that $\nabla_\phi \mathcal{L}_{\text{RL}}$ is well-behaved in all respects. Classical MLPs can suffer from vanishing or exploding gradients when L_{MLP} is large, a problem addressed by standard techniques (batch normalisation, residual connections, careful initialisation). Critically, however, these pathologies are independent of n and do not worsen as the quantum system scales.

$\text{total_supervised_training_epoch}$[10]. To preserve memory efficient and avoid asynchronous gradient propose/model update problem, we only use 1 environment to train all the RL agents. At each training iteration, all agent that use rollout buffer collects a rollout of length (rollout buffer size) equal to the total training steps, calculated as:

$$l_{\text{rollout}} = \text{int}\left(\frac{\text{total_training_sample}}{\text{supervised_training_batch_size}}\right) \cdot \text{total_supervised_training_epoch}, \quad (14)$$

after which the collected transitions are used to perform multiple epochs of minibatch updates. This rollout configuration is motivated by the observation that larger values of l_{rollout} lead to more stable training dynamics and ultimately improved convergence, even over prolonged training durations. Benchmarking results that support this claim using TRPO and PPO on MNIST can be found in Table 5.

Table 5: Accuracy comparison of PPO and TRPO across l_{rollout} length on MNIST.

Agents	**Top 1 validation accuracy per dataset**				
	$0.2 \cdot l_{\text{rollout}}$	$0.4 \cdot l_{\text{rollout}}$	$0.6 \cdot l_{\text{rollout}}$	$0.8 \cdot l_{\text{rollout}}$	l_{rollout}
TRPO	57.13 ± 0.199	58.35 ± 0.467	58.73 ± 0.512	59.35 ± 0.408	60.27 ± 0.298
PPO	57.22 ± 0.204	57.62 ± 0.259	58.79 ± 0.315	58.94 ± 0.342	59.52 ± 0.212

We also fixed agent initialization random seed to ensure reproducibility across runs. Unless otherwise specified, all other agent hyperparameters (e.g., clipping range, number of epochs, GAE parameters, and entropy coefficient) are kept at their default values provided by SB3.

D.2 Stablization tricks with spectral normalization

Training RL agents with function approximation is notoriously unstable, particularly in the bootstrapped regime of temporal-difference (TD) learning where approximation errors compound through iterated Bellman updates [48]. Spectral normalization (SN) has emerged as a principled remedy for this instability, with demonstrated effectiveness in stabilizing Q-networks [25] and actor-critic architectures [49, 50]. To avoid this problem, we can adopt SN as a method for stabilizing the training process of our algorithm. Consider a feed-forward network $\mathcal{F}_\theta$ with L layers:

$$z_i = W_i\, a_{i-1} + b_i, \qquad a_i = \sigma(z_i), \qquad i = 1, \ldots, L, \quad (15)$$

where $\sigma(\cdot)$ is the ReLU activation for all hidden layers ($i < L$). For a designated subset of layers $\mathcal{S} \subseteq \{1, \ldots, L\}$, SN replaces each weight matrix with its spectrally normalized counterpart:

$$\hat{W}_i = \frac{W_i}{\rho_i}, \qquad \rho_i = \max\big(\rho(W_i),\ k\big), \quad (16)$$

where $\rho(W_i) = \sigma_{\max}(W_i)$ denotes the largest singular value (spectral norm) of W_i, $k > 0$ is a small constant preventing degenerate scaling, and a stop-gradient is applied to ρ_i during backpropagation so that normalization acts as a constraint rather than a differentiable transformation.

By capping the spectral norm of each layer, SN enforces a Lipschitz bound on the end-to-end mapping. Specifically, for a network with ReLU activations, the Lipschitz constant satisfies:

$$\text{Lip}(\mathcal{F}_\theta) \ \leqslant\ \prod_{i \in \mathcal{S}} \rho_i^{-1} \cdot \prod_{i=1}^{L} \|W_i\|_2 \ \leqslant\ \prod_{i \notin \mathcal{S}} \|W_i\|_2, \quad (17)$$

which directly limits how rapidly the network output can change with respect to its input. In the RL context, this Lipschitz constraint serves two critical purposes: (i) it bounds the propagation of approximation error through bootstrapped TD targets, preventing the runaway divergence observed in unconstrained networks [49]; and (ii) it promotes smoother value landscapes, improving generalization across nearby states.

[10] Which is around 200, meaning that the rollout buffer is shuffled and split into batches of 200 samples for agent gradient updates with SGD.

Beyond its regularization effect, SN induces a data-dependent modulation of gradient magnitudes that fundamentally alters the optimization dynamics. Under a bias-scaling reformulation [25], the pre-activations of the normalized network satisfy:

$$\hat{z}_i = \rho_{1:i}^{-1}\, z_i, \qquad \text{where} \quad \rho_{1:i} = \prod_{j \leqslant i,\ j \in \mathcal{S}} \rho_j. \tag{18}$$

The loss computed on the normalized network and the gradient with respect to the unnormalized parameters is therefore:

$$\hat{\mathcal{L}} = \mathcal{L}\big(\rho_{1:L}^{-1}\, z_L\big), \quad \frac{\partial \hat{\mathcal{L}}}{\partial W_i} = \rho_{1:L}^{-1}\, J_i\, \hat{\delta}_L\, a_{i-1}^{\top}, \tag{19}$$

where J_i is the Jacobian of all layers subsequent to layer i and $\hat{\delta}_L = \partial \hat{\mathcal{L}} / \partial \hat{z}_L$. Equation 19 reveals that SN imposes a *global* rescaling of all gradients by $\rho_{1:L}^{-1}$, the inverse product of spectral norms across normalized layers. As network weights grow during training (a natural consequence of fitting increasingly complex value landscapes), $\rho_{1:L}$ increases monotonically, progressively attenuating gradient magnitudes. This gradient rescaling has a particularly consequential interaction with adaptive optimizers. Recall the Adam update rule [26]:

$$\Delta\theta_t = \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}, \tag{20}$$

where $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates of the first and second gradient moments, respectively. When gradients are globally rescaled as $g'_t = \rho_t^{-1}\, g_t$ (with $\rho_t := \rho_{1:L}$ at iteration t), the moment estimates transform as:

$$\hat{m}'_t \approx \rho_t^{-1}\, \hat{m}_t, \qquad \hat{v}'_t \approx \rho_t^{-2}\, \hat{v}_t. \tag{21}$$

Substituting into the Adam update yields:

$$\Delta\theta'_t \approx \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \rho_t\, \varepsilon}. \tag{22}$$

The key observation is that SN effectively multiplies the damping constant ε by ρ_t, which grows monotonically over training. This induces a natural three-phase optimization trajectory:

1. **Early training** ($\rho_t \approx 1$): The modified update Equation 22 reduces to standard Adam. The optimizer retains its full adaptive behavior, enabling rapid initial learning and efficient exploration of the loss landscape.
2. **Mid training** (ρ_t growing): The inflated denominator uniformly reduces step sizes across all parameters, acting as a stabilizer that dampens sensitivity to noisy gradient estimates, which is particularly important in RL, where minibatch composition and advantage estimation introduce high variance.
3. **Late training** ($\rho_t\, \varepsilon \gg \sqrt{\hat{v}_t}$): The update asymptotically approaches

   $$\Delta\theta'_t \;\approx\; \frac{\eta}{\rho_t\, \varepsilon}\, \hat{m}_t, \tag{23}$$

   which is equivalent to SGD with momentum at a decaying effective learning rate $\eta_{\text{eff}} = \eta / (\rho_t\, \varepsilon)$. This automatic transition from adaptive to momentum-based updates is highly desirable: adaptive methods excel at navigating the complex early-training landscape but suffer from poor curvature estimates in the non-stationary TD regime, whereas SGD with momentum provides more stable convergence near optima [25].

This optimization-centric interpretation is empirically validated by [25], who show that explicit schedulers mimicking either the gradient rescaling (DIVGRAD) or the ε-scaling (MULEPS) of SN recover comparable performance gains, even in regimes where full SN fails to train.

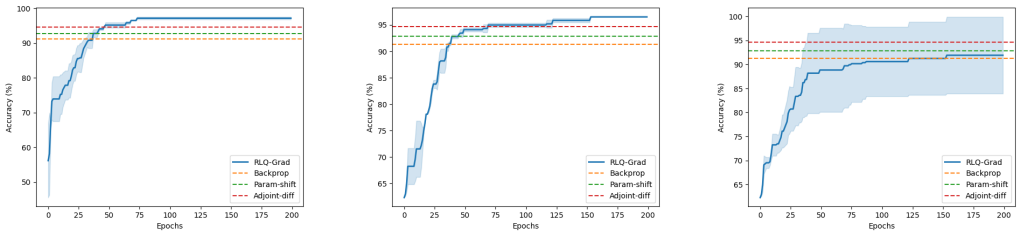
We apply SN selectively to the value and policy networks following the power iteration scheme described in Algorithm 2. Each normalized layer maintains a pair of auxiliary vectors $(\mathbf{u}, \mathbf{v})$ that approximate the leading left and right singular vectors of W_i via a single power iteration step per training update (with a negligible computational overhead).

To understand the contribution of SN on different network components, we evaluate three configurations on the BC dataset for the 2-qubit HEA problem (Figure 4): SN applied to the last layer of both

Algorithm 2 RL Training with Spectral Normalization

Require: Dataset $\mathcal{B}$ (replay buffer or on-policy batch), trainable networks $\mathcal{F}_\theta$, normalized layer set $\mathcal{S}$

for each training iteration **do**
  Sample batch $\mathcal{T} \sim \mathcal{B}$
  for each layer i in $\mathcal{F}_\theta$ **do**
    if $i \in \mathcal{S}$ **then**
      if $\mathbf{u}^{(i)}, \mathbf{v}^{(i)}$ not yet initialized **then**
        Initialize $\mathbf{u}^{(i)}, \mathbf{v}^{(i)} \sim \mathcal{N}(0, I)$
      end if
      $\tilde{\mathbf{v}} \leftarrow W_i^\top \mathbf{u}^{(i)}; \quad \mathbf{v}^{(i)} \leftarrow \tilde{\mathbf{v}}/\|\tilde{\mathbf{v}}\|$
      $\tilde{\mathbf{u}} \leftarrow W_i \mathbf{v}^{(i)}; \quad \rho_i \leftarrow \|\tilde{\mathbf{u}}\|; \quad \mathbf{u}^{(i)} \leftarrow \tilde{\mathbf{u}}/\rho_i$
      $\bar{W}_i \leftarrow W_i \,/\, \max(1,\ \rho_i)$ {Clamp spectral norm to $\leqslant 1$}
    end if
  end for
  Compute RL loss $\mathcal{L}_{\mathrm{RL}}(\mathcal{T}; \mathcal{F}_\theta)$ using normalized weights $\bar{W}$
  Update θ via gradient descent on $\mathcal{L}_{\mathrm{RL}}$
end for



(a) PolicySN[-1] + ValueSN[-1]. (b) ValueSN[-1]. (c) No SN.

Figure 4: Accuracy convergence of RLQ-Grad (with PPO) variants (with (a) both last layer of policy and value function SN, (b) only last layer of value function SN [25] and (c) no SN applied) on BC dataset (denoted blue with fluctuation region) when compare with maximum achievable accuracy of other gradient calculating methods on 2-qubits HEA.

the policy and value networks (PolicySN[-1] + ValueSN[-1]), SN applied to only the last layer of the value network (ValueSN[-1] [25]), and no SN applied.

The most striking difference across configurations is in convergence reliability. Configuration (a), with SN on both networks, achieves the fastest and most stable convergence, reaching near-optimal accuracy ($> 93\%$) within approximately 50 training epochs with minimal variance across runs. Configuration (b), with SN on the value network alone, also converges to comparable final accuracy, albeit with a slightly delayed onset, consistent with the value network being the primary source of bootstrap-induced instability. In contrast, configuration (c) without SN exhibits dramatically higher variance (visible as the wide blue fluctuation band) and slower, less reliable convergence, with some runs failing to reach the performance plateau within the allotted training budget.

The comparison between configurations (b) and (c) isolates the stabilizing effect of SN on the critic. The value network directly participates in computing TD targets, making it the primary channel through which approximation errors are amplified. By constraining the Lipschitz constant of the critic, SN bounds the magnitude of these compounding errors (Equation 17), preventing the catastrophic overestimation that often destabilizes training. The near-identical final performance of configurations (a) and (b) suggests that value network normalization accounts for the majority of the stability gains.

While policy network SN is not strictly necessary for convergence, configuration (a) demonstrates a measurable advantage in convergence speed over (b). We attribute this to the implicit learning rate scheduling effect that normalizing the policy network introduces an additional spectral radius

factor into the gradient rescaling (Equation 19), providing finer-grained control over the policy update magnitude.

Across all three configurations, the converged accuracy of our RLQ-Grad method matches or exceeds the maximum achievable accuracy of alternative gradient calculation methods (Backprop, Param-shift; shown as dashed horizontal lines in Figure 4). Notably, even the unstabilized variant (c) can achieve competitive final accuracy in successful runs, but its high variance makes it unreliable in practice.

Table 6: Accuracy comparison between ValueSN[-1] and non-SN PPO across datasets.

Agents	**Params**	**MACs**	**Top 1 validation accuracy per dataset**			
			BC	MNIST	F-MNIST	CIFAR10
PPO	5,575	5,440	90.05 ± 4.760	59.52 ± 0.212	68.52 ± 0.534	25.12 ± 0.961
PPO (SN)	11847	10,432	95.52 ± 0.080	61.36 ± 2.216	73.64 ± 0.415	27.17 ± 0.782

Table 6 demonstrates that incorporating SN into PPO yields consistent improvements across all datasets. PPO (SN) outperforms the baseline PPO in every case, with particularly notable gains on BC ($+5.47\%$) and F-MNIST ($+5.12\%$). Even on more challenging datasets such as CIFAR10, SN provides a measurable improvement ($+2.05\%$). The magnitude of improvement varies across datasets. Larger gains are observed on intermediate-complexity datasets (e.g., F-MNIST), while improvements are more modest on MNIST and CIFAR10.

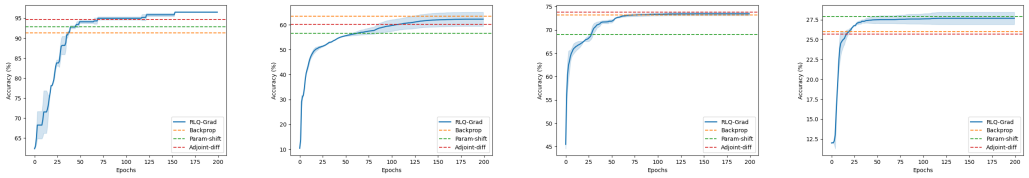
The improvements in accuracy come at the cost of increased model complexity. PPO (SN) approximately doubles both the parameter count (11,847 vs. 5,575) and MACs (10,432 vs. 5,440). Despite this, the performance gains are disproportionately large relative to the increase in computational cost, particularly on BC and F-MNIST.

D.3 Input dimensionality reduction

We employ Principal Component Analysis (PCA) to project the high-dimensional input into a lower-dimensional subspace while preserving maximal global variance. To determine the optimal number of features k, we evaluate the Cumulative Explained Variance Ratio (CEVR), defined as $\text{CEVR}(k) = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{d} \lambda_j}$ where λ_j represents the eigenvalue associated with the j-th principal component, d is the input dimension and captures approximately 95% of the total variance.

D.4 Accuracy convergence curves

Figure 5 shows the validation accuracy convergence of RLQ-Grad on the 2-qubit HEA for the four datasets of Table 2, compared with the maximum accuracy achieved by the other gradient computation methods.



(a) BC. (b) MNIST dataset. (c) F-MNIST dataset. (d) CIFAR-10 dataset.

Figure 5: Accuracy convergence of RLQ-Grad (denoted blue with fluctuation region) when compare with maximum achievable accuracy of other gradient calculating methods on 2-qubits HEA.

D.5 Hardware specifications

Since adjoint differentiation and parameter-shift rule currently lack efficient GPU implementations comparable to backpropagation and our proposed method, the resource allocation benchmark in Figure 3 was conducted on a CPU (AMD Ryzen 5 5600G) to assure fairness, averaging 12 timed trials after 4 warm-up iterations. The GPU comparison between RLQ-Grad and backpropagation (Table 9) was run on an NVIDIA RTX 3090 and averaged over 15 trials. In contrast, for accuracy evaluations

where minimizing training time is critical, experiments for our method and backpropagation were performed on an NVIDIA RTX 3090 (24GB) GPU.

D.6 Full resource benchmark values

Tables 7, 8, and 9 list the exact values behind Figure 3 and the GPU comparison. Rather than measuring only the actor's forward pass, all RLQ-Grad entries account for the complete training pipeline (PPO rollout, policy and value updates, gradient buffers, and Adam states).

Table 7: End-to-end CPU RAM usage of gradient computation methods ($n = 2$ to 20 qubits).

Method	2q1d	4q2d	6q3d	8q4d	10q5d	12q6d	14q7d	16q8d	18q9d	20q10d
Backpropagation	< 0.1 MB	187.15 KB	416.55 KB	619.74 KB	1.96 MB	10.10 MB	63.30 MB	282.70 MB	1227 MB	6174 MB
Parameter-shift	< 0.1 MB	282.89 KB	1.06 MB	6.38 MB	7.32 MB	9.98 MB	18.59 MB	22.37 MB	24.23 MB	22.55 MB
Adjoint-differentiation	< 0.1 MB	< 0.1 MB	< 0.1 MB	< 0.1 MB	< 0.1 MB	< 0.1 MB	< 0.1 MB	< 0.1 MB	< 0.1 MB	< 0.1 MB
RLQ-Grad (full)	**161.22 KB**	**215.64 KB**	**306.35 KB**	**433.33 KB**	**596.60 KB**	**796.14 KB**	**1.02 MB**	**1.30 MB**	**1.61 MB**	**1.95 MB**

Table 8: Per-iteration CPU wall-clock time (in seconds) of gradient computation methods.

Method	2q1d	4q2d	6q3d	8q4d	10q5d	12q6d	14q7d	16q8d	18q9d	20q10d
Backpropagation	0.0042	0.0123	0.0260	0.1295	0.1033	0.1662	0.2719	0.5112	2.3262	219.41
Parameter-shift	0.0067	0.0425	0.1333	0.3921	0.9860	2.3931	5.2019	16.2818	66.3288	693.96
Adjoint-differentiation	0.0027	0.0055	0.0109	0.0161	0.0266	0.0569	0.0836	0.6666	10.0897	59.3241
RLQ-Grad (full)	**0.0455**	**0.0545**	**0.0641**	**0.0565**	**0.0815**	**0.0858**	**0.0842**	**0.0862**	**0.0875**	**0.0881**

Table 9: Per-iteration GPU wall-clock time (in milliseconds, NVIDIA RTX 3090, 15 trials). Parameter-shift and adjoint differentiation are omitted as they lack efficient GPU kernels in current QNN frameworks.

Method	2q1d	4q2d	6q3d	8q4d	10q5d	12q6d	14q7d	16q8d	18q9d	20q10d
Backpropagation	23.37	65.54	126.96	218.95	322.98	451.28	605.74	807.99	1026.45	1457.06
RLQ-Grad (full)	**22.51**	**56.60**	**111.49**	**184.76**	**276.88**	**390.89**	**523.60**	**785.93**	**957.31**	**1190.83**

E Derivation process of quantum gradient calculation methods

Assuming we have a QNN with n qubits and L parameterized gates (or layers), let P denote the number of trainable parameters, typically $P = \mathcal{O}(L)$. A quantum state on n qubits is represented as a complex vector $|\psi\rangle \in \mathbb{C}^{2^n}$, so storing a single state requires a memory complexity of $\mathcal{O}_{\text{memory}}(2^n)$.

E.1 Forward pass complexity

Every gate acts on the full statevector, leading to a computational cost of $\mathcal{O}(2^n)$ per gate application. Therefore, the total cost of the forward pass is $\mathcal{O}(L \cdot 2^n)$, and this factor remain unchanged since benchmarked optimization algorithms only affect gradient computations.

E.2 Backward pass complexity

E.2.1 Backpropagation

Let the circuit be composed of unitary operators $\{U_k\}_{k=1}^{L}$. The forward evolution is $|\psi_k\rangle = U_k|\psi_{k-1}\rangle$ with $k = 1, \ldots, L$. Define the adjoint state $|\lambda_L\rangle = O|\psi_L\rangle$ where O is the observable; the backward recursion is:

$$|\lambda_{k-1}\rangle = U_k^\dagger |\lambda_k\rangle. \tag{24}$$

For a parameterized gate $U_k(\theta_k) = e^{-i\theta_k G_k}$, we have $\frac{dU_k}{d\theta_k} = -iG_k U_k$, so differentiating $\langle O\rangle = \langle\psi_L|O|\psi_L\rangle$ and collecting both the bra and ket contributions yields:

$$\frac{\partial\langle O\rangle}{\partial\theta_k} = 2\,\mathrm{Im}\left(\langle\lambda_k|G_k|\psi_k\rangle\right). \tag{25}$$

With L evolution steps we obtain a time complexity of $\mathcal{O}_{\text{time}}(L \cdot 2^n)$. For memory complexity, since all intermediate states $\{|\psi_k\rangle\}_{k=0}^{L}$ must be stored, we have $\mathcal{O}_{\text{memory}}(L \cdot 2^n)$.

E.2.2 Parameter-Shift Rule

When using parameter-shift [19], for a parameter θ_i the gradient is computed as:

$$\frac{\partial \langle O \rangle}{\partial \theta_i} = \frac{1}{2}\left[f\left(\theta_i + \frac{\pi}{2}\right) - f\left(\theta_i - \frac{\pi}{2}\right)\right], \tag{26}$$

where each evaluation requires a full forward pass. Thus, computing the gradient for all parameters requires $\mathcal{O}_{\text{time}}(P \cdot L \cdot 2^n)$. Since only one statevector is needed at a time per parameter update, the memory complexity is theoretically $\mathcal{O}_{\text{memory}}(2^n)$. Still, in practice many framework such as Pennylane [21] accumulate all parameters gradient and update the model as a whole, leading to a practical memory complexity of $\mathcal{O}_{\text{memory}}(P_{\text{QNN}} \cdot 2^n)$.

E.2.3 Adjoint differentiation

The reverse-mode gradient computation derived in [18] (or for short "adjoint differentiation" as noted in some materials [51]) is mathematically equivalent to traditional backpropagation, but exploits the reversible structure of quantum circuits to achieve improved memory efficiency.

In classical neural networks, backpropagation computes gradients via reverse-mode accumulation:

$$\boldsymbol{\delta}_k = \left(\frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k}\right)^\top \boldsymbol{\delta}_{k+1}, \tag{27}$$

requiring storage of all intermediate activations $\{\mathbf{h}_k\}_{k=1}^{L}$, leading to memory scaling proportionally with L. In quantum neural networks, the analogous reverse propagation is:

$$|\lambda_{k-1}\rangle = U_k^\dagger |\lambda_k\rangle, \tag{28}$$

which is directly analogous to multiplication by the transposed Jacobian in classical backpropagation. The gradient of the expectation value $\langle E \rangle$ with respect to parameter θ_i is:

$$\frac{\partial \langle E \rangle}{\partial \theta_i} = 2\,\mathrm{Re}\left(\langle \lambda_i | \frac{dU_i}{d\theta_i} | \phi_i \rangle\right), \tag{29}$$

where the backward state $|\lambda_i\rangle$ and the reconstructed forward state $|\phi_i\rangle$ satisfy the recursions (traversed from $k = L$ down to $k = 1$):

$$|\phi_i\rangle = U_i^\dagger |\phi_{i+1}\rangle, \quad |\lambda_i\rangle = U_{i+1}^\dagger |\lambda_{i+1}\rangle. \tag{30}$$

Here $|\phi_{i+1}\rangle$ is the reconstructed forward state at layer $i+1$, obtained by applying the inverse gate $U_{i+1}^\dagger$ to the current forward state rather than storing it from the forward pass.

Because quantum circuits are unitary and therefore reversible, the intermediate forward states $\{|\psi_k\rangle\}$ need not be stored; they are recomputed on-the-fly during the backward sweep. Consequently, all P gradients are obtained in a single reverse pass at a cost of $\mathcal{O}_{\text{time}}(P_{QNN} \cdot 2^n)$, matching backpropagation's time complexity while reducing the memory required beyond the forward statevector to $\mathcal{O}_{\text{memory}}(1)$, since only a fixed number of working buffers (the current forward and backward states) are kept regardless of L and P_{QNN}.

E.3 RLQ-Grad

In the case of our RLQ-Grad, gradient evaluation reduces to a forward pass through the MLP actor. For an L_{MLP}-layer network of hidden width d, whose input s_t and output g_t both have dimension $\mathcal{O}(P_{\text{QNN}})$, the input layer costs $\mathcal{O}(d\,P_{\text{QNN}})$, the $L_{\text{MLP}} - 2$ hidden layers cost $\mathcal{O}(d^2)$ each, and the output head costs $\mathcal{O}(d\,P_{\text{QNN}})$. Summing the three terms gives $\mathcal{O}_{\text{time}}(L_{\text{MLP}}d^2 + d\,P_{\text{QNN}})$, with activation memory $\mathcal{O}_{\text{memory}}(d + P_{\text{QNN}})$.

The full training pipeline additionally runs the critic, backpropagates through actor and critic during PPO updates, and maintains the Adam optimizer. Let $P_{\text{RL}} = \mathcal{O}(L_{\text{MLP}}d^2 + d\,P_{\text{QNN}})$ denote the joint actor and critic parameter count. The resident memory consists of the parameters (P_{RL}), their gradient buffers (P_{RL}), and the Adam first and second moments ($2P_{\text{RL}}$), i.e. $P_{\text{RL}} + P_{\text{RL}} + 2P_{\text{RL}} = 4P_{\text{RL}}$. This predicts the measured split of approximately 26% (parameters), 26% (gradients), and 52% (Adam states) reported in Section 3.1.1. Each PPO update step likewise costs $\mathcal{O}(P_{\text{RL}})$ per sample. Both quantities are linear in $P_{\text{QNN}} = 3nL$ and independent of 2^n.

F Further discussion of other properties

F.1 Optimal conditions for RLQ-Grad trainability

In this part, we further extend our theoretical framework by assessing the optimal condition for RLQ-Grad to converge upon training, viewed as the following theorem:

Theorem 4 (Optimal conditions for RLQ-Grad trainability)**.** *Let $U(\theta)$ be an n-qubit HEA circuit with associated JAWS $(\mathcal{A}, \{r_\alpha\}, \{\beta_\alpha\})$. RLQ-Grad achieves efficient training if and only if both of the following conditions hold:*

- ***Condition 1 (C1):*** *Structured QNN on supervise problem satisfy:*

$$\sum_{\alpha} \frac{\mathrm{Tr}(O_\alpha)^2 \,\mathrm{Tr}(\rho^\alpha)^2}{\dim_{\mathbb{R}}(\mathrm{Aut}(\mathcal{A}_\alpha))} = \Omega\big(\mathrm{poly}(\log N)^{-1}\big)\,, \tag{31}$$

  ensures the loss (and hence the reward r_t for RLQ-Grad) is sufficiently sensitive to parameter changes for the agent to receive a meaningful training signal (informativeness).

- ***Condition 2 (C2):*** *Selected QNN achieve overparameterization as:*

$$p \;\geqslant\; \max_{\alpha} \beta_\alpha r_\alpha, \tag{32}$$

  ensures local minima concentrate near the global minimum, so that any convergent optimizer finds a good solution.

Proof. We can prove this theorem by contradiction as:

- **Necessity of (C1).** If (C1) fails and the loss concentrates exponentially and ℓ_t is indistinguishable from a constant in the reward. By Proposition 1, the mutual information $I(g_t; r_t \mid s_t) \to 0$, and the agent cannot improve its policy.
- **Necessity of (C2).** If (C2) fails, by Theorem 5 the density of local minima near the global optimum is exponentially suppressed. Any convergent optimizer terminates at a poor local minimum with probability exponentially close to 1, regardless of gradient quality. This applies to RLQ-Grad without exception.
- **Sufficiency.** When (C1) holds, the reward is informative and the agent receives a nonvanishing gradient signal through the reward differences. When (C2) holds, good local minima are dense and any convergent optimizer finds a near-optimal solution. Since the agent's own training is governed by a classical MLP whose gradient pathologies are independent of n (Theorem 2 below), and the policy class is a universal approximator by the UAT (Appendix A), the combination is sufficient for efficient training.

□

F.2 Under Wishart process framework, how does this affect agent reward informativeness?

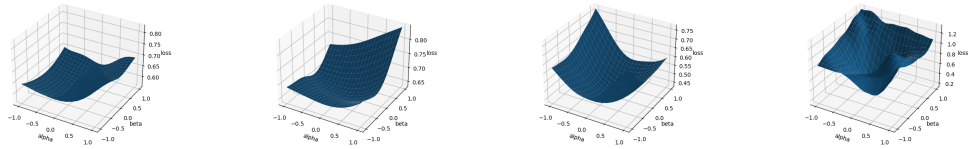
[28] also prove that in QNN, the parameter gradient distribution is not Gaussian but follows a Wishart process: the joint distribution of $\partial_{\theta_i}\ell$ conditioned on $\ell(\rho^\alpha;\theta) = z_\alpha$ is:

$$\hat{\ell}_i \;\asymp\; \sum_{\alpha} \frac{2 I_\alpha \sigma_\alpha \,\mathrm{Tr}_\alpha(\rho^\alpha)}{N_\alpha} \sqrt{\frac{\beta_\alpha z_\alpha}{I_\alpha o_\alpha}}\; G_{\alpha,i}\, \chi_{\alpha,i}, \tag{33}$$

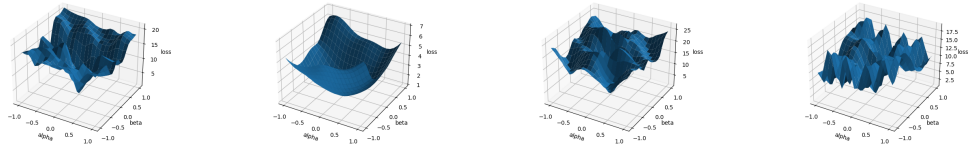
where $G_{\alpha,i}$ are i.i.d. standard normal and $\chi_{\alpha,i}$ are independent χ-distributed with $\max(2, \beta_\alpha)$ degrees of freedom. This has the following consequence for the informativeness of the RLQ-Grad reward:

Proposition 1. *The reward $r_t = \mathrm{acc}_t + (\ell_t + \epsilon)^{-1}$ is a monotone function of ℓ_t and acc_t. Under the Wishart process description, the loss value $z_\alpha = \ell(\rho^\alpha; \theta_t)$ is itself Wishart-distributed with degrees of freedom r_α[11], and the gradient is conditionally distributed as in Equation 33. Two consequences follow:*

[11] By Theorem 1 of [28]



(a) BC, backprop. (b) BC, param-shift. (c) BC, adjoint-diff. (d) BC, RLQ-Grad.



(e) F-MNIST, backprop. (f) F-MNIST, param-shift. (g) F-MNIST, adjoint-diff. (h) F-MNIST, RLQ-Grad.

Figure 6: Loss landscape visualization across benchmarked gradient computation methods over HEA-2b1q on BC and F-MNIST. Loss landscape change from smooth on BC dataset (overparameterized) to becomes more noisy (underparameterized) with sharp local minima in F-MNIST.

1. *The gradient magnitude scales as* $|\partial_{\theta_i}\ell| \propto \sqrt{z_\alpha}$*, so the gradient is correlated with the loss value. The reward* r_t*, which depends on* ℓ_t*, therefore carries information about the gradient magnitude even in the absence of direct gradient computation.*

2. *In the barren plateau regime,* $z_\alpha \to 0$ *exponentially, causing both the gradient and its correlation with the reward to vanish simultaneously. This is the fundamental obstruction: when the loss concentrates, so does the reward, and the agent receives no directional signal.*

Proof. Point 1 follows directly from Equation 33: $|\hat{\ell}_i| \propto \sqrt{z_\alpha / I_\alpha o_\alpha}$, and z_α is recoverable (up to noise) from $\ell_t \in s_t$. The mutual information $I(g_t; r_t \mid s_t) > 0$ therefore whenever $z_\alpha > 0$ at a distinguishable level above noise. In the barren plateau regime, $z_\alpha \in \mathcal{O}(\mathrm{poly}(\log N)^{-1})$, so the conditional gradient distribution of Equation 33 also concentrates at zero, making the reward differences unresolvable, eventually proved point 2[12]. □

Combining Theorems 1 and 2, RLQ-Grad offers a genuine advantage over conventional gradient methods precisely when the circuit satisfies conidtion 1 but not the gradient-variance condition. This corresponds to circuits with non-trivial Jordan algebraic structure (e.g., LASA circuits [52, 53]) where the loss landscape is non-flat but standard gradient computation is exponentially expensive. In circuits where condition 2 itself fails (i.e., the loss is truly exponentially flat) no method can efficiently train the quantum parameters, and the primary utility of RLQ-Grad is instead its computational advantage (Appendix E.3) rather than any landscape-related benefit.

G Further discussion of limitations

G.1 Poor local minima obstruction

The previous analysis identified only the barren plateau as an obstruction to RLQ-Grad. [28] identify a second, independent obstruction absent from earlier analyses:

Theorem 5 (Poor local minima obstruction)**.** *Under the JAWS framework [28], even when the barren plateau condition of Equation 8 is non-vanishing (i.e., gradients are not exponentially suppressed), the density of local minima near the global minimum is exponentially small if and only if the network is underparameterized:*

$$p < \max_\alpha \beta_\alpha r_\alpha, \tag{34}$$

[12] By Corollary 4 of [28].

where p is the number of trained parameters, $\beta_\alpha \in \{1, 2, 4\}$ corresponds to the field $\mathbb{F}_\alpha \in \{\mathbb{R}, \mathbb{C}, \mathbb{H}\}$ of the α-th Jordan component, and r_α is the degrees of freedom. In this regime, an optimizer (including RLQ-Grad) converges to a poor local minimum with overwhelming probability, regardless of gradient quality.

Proof. The density of local minima at loss value $z > 0$ is (to leading multiplicative order) a convolution of Gamma distributions[13]:

$$\kappa(z) = \underset{\alpha:\,\gamma_\alpha<1}{\circledast} f_\Gamma\left(\frac{z}{o_\alpha \operatorname{Tr}(\rho^\alpha)}; \frac{\beta_\alpha r_\alpha}{2}, \frac{2}{\beta_\alpha r_\alpha}\right), \tag{35}$$

where $\gamma_\alpha = p/\sqrt{\beta_\alpha r_\alpha}$ is the overparameterization ratio. When $\gamma_\alpha < 1$ for some α (Equation 34 violated), the Gamma distribution has exponential tails, placing an exponentially small fraction of local minima near $z = 0$. Since any gradient-based optimizer (and any surrogate that produces descent directions) converges to a local minimum, it converges to a poor one with probability exponentially close to 1. RLQ-Grad is not exempt from this, since the obstruction is in the loss landscape geometry, not in the gradient computation mechanism. □

More critically, barren plateaus and poor local minima are independent obstructions[14]. A circuit can be:

1. Barren plateau affected and underparameterized (doubly obstructed),
2. Barren plateau free but underparameterized (gradient information present, but all accessible local minima are poor),
3. Overparameterized but barren plateau affected (good minima exist but are unreachable by gradient methods),
4. Both barren plateau free and overparameterized.

While RLQ-Grad bypasses obstruction 3 by replacing the quantum gradient. It does not bypass obstruction 2, which is independent of gradient quality. A visualization of loss landscape across benchmarked gradient computation method can be found in Figure 6.

G.2 Barren plateau induced input gradient decay

Consider a QNN where the n-qubit input state is prepared via amplitude encoding using R_x rotations, followed by a trainable unitary $U(\boldsymbol{\theta})$ (e.g. a HEA in our experiments). The input gradient $\nabla_{\mathbf{x}}\mathcal{L}$ is structurally analogous to the parameter gradient, yet the two objects differ in a crucial respect: $\boldsymbol{\theta}$ is randomized during initialization, whereas $\mathbf{x}$ is a *fixed* data point. The question is therefore: whether $\partial\mathcal{L}/\partial x_i$, viewed as a random variable over the ensemble of weight initializations, also concentrates exponentially near zero?

Short answer is yes, the same Weingarten-calculus argument that governs weight barren plateaus applies to input gradients, provided the trainable block $U(\boldsymbol{\theta})$ forms (at least) a unitary 2-design. However, the interpretation differs: the concentration is over $\boldsymbol{\theta}$ at a fixed $\mathbf{x}$, not over the data distribution. To further understand this problem, we can formalize the following theorem:

Theorem 6 (Input quantum gradient decay)**.** *For a fixed input vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ define the encoding state:*

$$|\psi(\mathbf{x})\rangle = E(\mathbf{x})|0\rangle^{\otimes n}, \qquad E(\mathbf{x}) = \bigotimes_{i=1}^{n} R_x(x_i).$$

Let $U(\boldsymbol{\theta})$ be an L-layer HEA with parameters $\boldsymbol{\theta} \in [0, 2\pi)^M$ drawn uniformly, and let O be an observable with $\|O\| \leqslant 1$. The loss is:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\theta}) = \langle\psi(\mathbf{x})|U^\dagger(\boldsymbol{\theta})\, O\, U(\boldsymbol{\theta})|\psi(\mathbf{x})\rangle.$$

[13] By Corollary 6 of [28].
[14] By Definition 7 of [28].

The input gradient with respect to component x_i is:

$$\frac{\partial \mathcal{L}}{\partial x_i} = \langle \psi(\mathbf{x}) | \Big[O_{\boldsymbol{\theta}},\ \frac{\partial E(\mathbf{x})}{\partial x_i} E(\mathbf{x})^\dagger \Big]_+ | \psi(\mathbf{x}) \rangle,$$

where $O_{\boldsymbol{\theta}} = U^\dagger(\boldsymbol{\theta})\, O\, U(\boldsymbol{\theta})$ is the Heisenberg-picture observable, and $[\cdot,\cdot]_+$ denotes the symmetrised (anti-commutator-like) combination arising from the product rule. Equivalently, by the parameter-shift rule applied to x_i,

$$\frac{\partial \mathcal{L}}{\partial x_i} = \frac{1}{2} \Big[\mathcal{L}\big(\mathbf{x} + \tfrac{\pi}{2}\hat{e}_i, \boldsymbol{\theta}\big) - \mathcal{L}\big(\mathbf{x} - \tfrac{\pi}{2}\hat{e}_i, \boldsymbol{\theta}\big) \Big]. \tag{36}$$

Let the trainable block $U(\boldsymbol{\theta})$ with $L = \Omega(\log n)$ layers form an exact unitary 2-design on $U(2^n)$ when $\boldsymbol{\theta}$ is drawn uniformly from $[0, 2\pi)^M$. Fix any input $\mathbf{x} \in \mathbb{R}^n$ and any observable O with $\|O\| \leqslant 1$. Then for every coordinate $i \in \{1, \ldots, n\}$, input gradient converge toward zero mean $\mathbb{E}_{\boldsymbol{\theta}}\Big[\frac{\partial \mathcal{L}}{\partial x_i}\Big] = 0$ with exponentially small variance $\mathrm{Var}_{\boldsymbol{\theta}}\Big[\frac{\partial \mathcal{L}}{\partial x_i}\Big] \leqslant \frac{C}{2^n}$, for a universal constant $C > 0$ independent of $\mathbf{x}$, n, and L. The probabilistic concentration can also be quantified as $\Pr_{\boldsymbol{\theta}}\Big[\Big|\frac{\partial \mathcal{L}}{\partial x_i}\Big| \geqslant \varepsilon\Big] \leqslant \frac{C}{\varepsilon^2\, 2^n}$.

Equations 36 and the analogous formula for HEA parameter gradient $\partial \mathcal{L}/\partial \theta_\mu$ are formally identical: both are half the difference of two expectation values obtained by shifting a rotation angle by $\pm\pi/2$. The distinction is that θ_μ is a stochastic variable while x_i is fixed, consequently the randomness in the input gradient $\partial \mathcal{L}/\partial x_i$ comes entirely from $\boldsymbol{\theta}$. We can proceed to prove our theorem as follow:

Proof. From the parameter-shift representation in Equation 36:

$$\frac{\partial \mathcal{L}}{\partial x_i} = \frac{1}{2}\big(f^+ - f^-\big), \qquad f^\pm = \langle \psi_i^\pm | U^\dagger O\, U | \psi_i^\pm \rangle,$$

where $|\psi_i^\pm\rangle = E(\mathbf{x} \pm \frac{\pi}{2}\hat{e}_i)|0\rangle^{\otimes n}$ are two *fixed* product states (they depend on $\mathbf{x}$ but not on $\boldsymbol{\theta}$). Write $\rho^\pm = |\psi_i^\pm\rangle\langle\psi_i^\pm|$.

Because $U(\boldsymbol{\theta})$ forms a 1-design, we would have:

$$\mathbb{E}_{\boldsymbol{\theta}}\big[U^\dagger O\, U\big] = \frac{\mathrm{tr}(O)}{2^n}\, I,$$

and therefore:

$$\mathbb{E}_{\boldsymbol{\theta}}\big[f^\pm\big] = \mathrm{tr}\big(\rho^\pm\, \mathbb{E}[U^\dagger O U]\big) = \frac{\mathrm{tr}(O)}{2^n},$$

which is the same for both f^+ and f^-, giving:

$$\mathbb{E}_{\boldsymbol{\theta}}\left[\frac{\partial \mathcal{L}}{\partial x_i}\right] = \frac{1}{2}\big(\mathbb{E}[f^+] - \mathbb{E}[f^-]\big) = 0.$$

Since $\mathbb{E}[\partial \mathcal{L}/\partial x_i] = 0$, the variance equals the second moment:

$$\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_i}\right] = \mathbb{E}\left[\left(\frac{\partial \mathcal{L}}{\partial x_i}\right)^2\right] = \frac{1}{4}\Big(\mathbb{E}[(f^+)^2] + \mathbb{E}[(f^-)^2] - 2\,\mathbb{E}[f^+ f^-]\Big).$$

For any fixed normalised state $|\phi\rangle$ and any 2-design, the Weingarten formula gives

$$\mathbb{E}_U\big[\langle\phi|U^\dagger O U|\phi\rangle^2\big] = \frac{\mathrm{tr}(O^2) + \big(\mathrm{tr}(O)\big)^2}{2^n(2^n+1)}. \tag{37}$$

This formula holds for any fixed unit vector $|\phi\rangle$; it does not require $|\phi\rangle$ to be random. Substituting $|\phi\rangle = |\psi_i^\pm\rangle$ (both are unit vectors) into Equation 37 and using $\|O\| \leqslant 1$:

$$\mathbb{E}[(f^\pm)^2] = \frac{\mathrm{tr}(O^2) + (\mathrm{tr}(O))^2}{2^n(2^n+1)} \leqslant \frac{2^n + 2^{2n}}{2^n(2^n+1)} = \frac{2^n(1+2^n)}{2^n(2^n+1)} = 1.$$

A tighter bound follows from $\mathrm{tr}(O^2) \leqslant 2^n$ (since $\|O\| \leqslant 1$) and $|\mathrm{tr}(O)| \leqslant 2^n$:

$$\mathbb{E}[(f^{\pm})^2] \leqslant \frac{2 \cdot 2^n}{2^n(2^n+1)} = \frac{2}{2^n+1} \leqslant \frac{2}{2^n}.$$

For the cross term, the 2-design formula for two different states $|\phi\rangle, |\chi\rangle$ gives:

$$\mathbb{E}_U\big[\langle\phi|U^\dagger O U|\phi\rangle\langle\chi|U^\dagger O U|\chi\rangle\big] = \frac{\mathrm{tr}(O^2)\,|\langle\phi|\chi\rangle|^2 + (\mathrm{tr}(O))^2}{2^n(2^n+1)}.$$

Since $|\langle\psi_i^+|\psi_i^-\rangle| \leqslant 1$, the cross term satisfies $\mathbb{E}[f^+ f^-] \leqslant \frac{2}{2^n}$. Combining, and absorbing numerical prefactors into the constant $C = 3$:

$$\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_i}\right] \leqslant \frac{1}{4}\left(\frac{2}{2^n} + \frac{2}{2^n} + \frac{2}{2^n}\right) = \frac{3}{2 \cdot 2^n} \leqslant \frac{C}{2^n}.$$

By Chebyshev's inequality,

$$\Pr\left[\left|\frac{\partial \mathcal{L}}{\partial x_i}\right| \geqslant \varepsilon\right] \leqslant \frac{\mathrm{Var}[\partial \mathcal{L}/\partial x_i]}{\varepsilon^2} \leqslant \frac{C}{\varepsilon^2\, 2^n}.$$

This completes the proof for all claims. □

Let the setup be as in Theorem 6. The following distinctions hold between the input gradient $g_x = \partial \mathcal{L}/\partial x_i$ and the weight gradient $g_\theta = \partial \mathcal{L}/\partial \theta_\mu$:

- **Source of randomness:** Both g_x and g_θ are random variables over $\boldsymbol{\theta}$. The input $\mathbf{x}$ is deterministic in g_x while the parameter θ_μ itself is part of $\boldsymbol{\theta}$ and contributes additional variance to g_θ.
- **Variance bound:** Both satisfy $\mathrm{Var}[\cdot] \leqslant C/2^n$, with the same asymptotic scaling. The constant C may differ: for g_θ the gate G_μ participates in the 2-design argument on both sides of the circuit; for g_x the encoding gate is fixed.
- **Dependence on the data point:** $\mathrm{Var}[g_x]$ is independent of $\mathbf{x}$ (to leading order) because the Weingarten formula Equation 37 depends only on $\||\phi\rangle\| = 1$, not on which unit vector $|\phi\rangle$ is.
- **Practical implication:** The weight barren plateau obstructs training while the input-gradient barren plateau obstructs input-space optimisation, eventually affecting: quantum adversarial attacks, gradient-based feature attribution (quantum analogues of saliency maps), and hybrid quantum-classical architectures (in our case, the fully connected layers that encode map classical input to actual HEA input, formalize in Corollary 2) that backpropagate through the quantum layer.

Results from Theorem 6 also lead to following corollaries and remarks:

Corollary 1 (Full input Jacobian)**.** *Under the conditions of Theorem 6, the expected squared Frobenius norm of the input Jacobian $J = \nabla_{\mathbf{x}} \mathcal{L} \in \mathbb{R}^n$ satisfies*

$$\mathbb{E}_{\boldsymbol{\theta}}\big[\|J\|^2\big] = \sum_{i=1}^{n} \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_i}\right] \leqslant \frac{Cn}{2^n}.$$

Since $n/2^n \to 0$ exponentially, the entire Jacobian collapses to zero.

Corollary 2 (Backpropagation through a quantum layer)**.** *In a hybrid quantum-classical network, the classical layers preceding the quantum encoding receive gradients scaled by J. By the corollary above, $\mathbb{E}[\|J\|^2] \leqslant Cn/2^n$, so the classical pre-processing layers also experience exponentially vanishing gradients, regardless of their own architecture.*

Remark 1 (Noise exacerbates the effect)**.** *Under depolarising noise with per-gate error rate p, the effective observable seen by the input state is attenuated. By the same argument as [54] for weight gradients, one obtains:*

$$\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_i}\right] \in \mathcal{O}\big((1-p)^{2L} \cdot 2^{-n}\big),$$

which is doubly exponentially suppressed in L compared to the noiseless bound.

Remark 2 (Local cost functions offer no relief for input gradients)**.** *For weight gradients, replacing a global O with a local k-qubit observable reduces the variance from $\mathcal{O}(2^{-n})$ to $\mathcal{O}(2^{-k})$ [41]. This mitigation still applies to $\mathrm{Var}[g_x]$:*

$$\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_i}\right] \in \mathcal{O}(2^{-k}), \quad O \text{ acts on } k \text{ qubits}.$$

However, because the input dimension equals n, a reduction to $k \ll n$ qubits in the observable may destroy the expressiveness needed to solve the learning task, presenting a fundamental expressiveness-trainability trade-off for input gradients.

H Broader Impact

RLQ-Grad is a methodological contribution to the optimization of quantum neural networks and does not target a specific deployed application. We nevertheless identify several downstream considerations.

Positive impacts

By replacing $\mathcal{O}(L \cdot 2^n)$ statevector propagation with an $\mathcal{O}(L_{\mathrm{MLP}} d^2 + d\, P_{\mathrm{QNN}})$ forward pass through a small MLP, RLQ-Grad reduces the CPU cost of simulation-based QNN training by 2 to 4 orders of magnitude at $n = 20$ qubits in our benchmarks, once the full agent training pipeline is accounted for (while adding a small fixed overhead on circuits below roughly 8 qubits). This lowers the carbon footprint of QML research and widens access for groups without access to large-scale accelerators. Reliable QNN training is a prerequisite for variational approaches to quantum chemistry, condensed-matter simulation, and combinatorial optimization, all of which carry societal upside (e.g., catalyst design, battery materials, logistics). A stable, gradient-variance-flat optimizer reduces the run-to-run noise that currently complicates empirical comparisons in the field, supporting healthier scientific norms.

Negative and dual-use considerations

Any work that makes near-term variational quantum algorithms more practical contributes, incrementally, to the ecosystem progressing toward fault-tolerant quantum computers capable of breaking RSA and elliptic-curve cryptography. This is a diffuse, long-horizon concern rather than an immediate risk (RLQ-Grad operates on small HEAs and does not itself bring factoring-relevant circuits within reach) but it is the standard dual-use note for QML work. Organizations handling long-lived secrets should continue migrating to post-quantum cryptographic primitives. The update rule produced by π_ϕ is not an analytic gradient and cannot be inspected with the tools developed for parameter-shift or adjoint differentiation. In safety/compliance-critical hybrid pipelines, this opacity may hinder auditing, failure-mode analysis, and reward-specification validation (failure modes familiar from the learned-optimizer literature [37, 55]). As a general-purpose QNN optimizer, RLQ-Grad could be applied to any supervised objective, including ones with negative social externalities (surveillance, biometric identification, autonomous targeting). The same disclaimer applies as to any generic optimization advance; we do not see a specific misuse channel unique to the quantum setting at current hardware scales.

Mitigations

We recommend that practitioners deploying RLQ-Grad (a) log both the surrogate gradient g_t and the analytic gradient $\nabla_\theta \mathcal{L}$ on held-out probes during training to detect reward-hacking or silent drift, (b) retain analytic gradient verification in regimes where it remains tractable, and (c) report per-configuration agent training cost alongside the amortized QNN training cost for fair comparisons.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have presented our contributions accurately in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have briefly discuss limitations of our work in the Limitation section, with further details shown in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We have provided proof for all mentioned theoretical claims in the Appendix section.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: As far as we are concerned, the paper discloses all components required to reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Source code for our experiments are available publicly online on Zenodo as stated in the main text.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We have briefly explain our experimental setup in the main text and in details in the Appendix section.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report the mean and standard deviation for all primary experimental results, averaged over 4 independent runs (3 runs for the extended CIFAR-10 experiments at 14 to 20 qubits in Table 3). Each run utilized a distinct, documented random seed to capture variance in initialization and data shuffling, while specific non-stochastic architectural components remained fixed to ensure reproducibility. The $\pm$ values represent one standard deviation (1σ) from the mean. While the number of runs ($n = 4$) is constrained by computational resources, the reported error bars provide an initial measure of the stability and statistical significance of our performance gains.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided the hardware settings in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: As far as we are concerned, the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: A broader-impact discussion is provided in Appendix H. On the positive side, RLQ-Grad reduces the classical simulation cost of training QNNs from $\mathcal{O}(L \cdot 2^n)$ to $\mathcal{O}(L_{\text{MLP}} d^2 + d\, P_{\text{QNN}})$, lowering the energy footprint and hardware barrier for quantum-machine-learning research and potentially democratizing access for institutions without large GPU clusters. It also contributes to more reliable training of near-term variational algorithms, which are a prerequisite for quantum applications in drug discovery, materials science, and combinatorial optimization that carry clear societal benefit. On the negative side, advances that make QNN training more tractable at scale accelerate progress toward fault-tolerant quantum computing, which in turn accelerates concerns around cryptographically relevant quantum advantage (e.g., Shor-style attacks on RSA/ECC) and earlier-than-expected obsolescence of currently deployed public-key infrastructure. More narrowly, surrogate-gradient optimizers inherit the general risks of learned optimizers (reward hacking, silent divergence, and reduced interpretability of the update rule relative to analytic gradients) which may complicate safety auditing of hybrid quantum-classical pipelines. The work is foundational rather than application-specific and does not target any deployed system, but we flag these considerations to support informed downstream use.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: As far as we are concerned, the paper poses no such risk.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All related dependencies, assets and data in our work are open-source, which we have all cited properly and closely following license/terms of use in our work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have provided instructions to rerun experiments in our source code.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve human subjects or crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human subjects or crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: This work does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.