WSBD: Freezing-Based Optimizer for Quantum Neural Networks

Christopher Kverne
Florida International University

Mayur Akewar
Florida International University

Yuqian Huo
Rice University

Tirthak Patel
Rice University

Janki Bhimani
Florida International University

Abstract

The training of Quantum Neural Networks (QNNs) is hindered by the high computational cost of gradient estimation and the barren plateau problem, where optimization landscapes become intractably flat. To address these challenges, we introduce Weighted Stochastic Block Descent (WSBD), a novel optimizer with a dynamic, parameter-wise freezing strategy. WSBD intelligently focuses computational resources by identifying and temporarily freezing less influential parameters based on a gradient-derived importance score. This approach significantly reduces the number of forward passes required per training step and helps navigate the optimization landscape more effectively. Unlike pruning or layer-wise freezing, WSBD maintains full expressive capacity while adapting throughout training. Our extensive evaluation shows that WSBD converges on average 63.9% faster than Adam for the popular ground-state-energy problem, an advantage that grows with QNN size. We provide a formal convergence proof for WSBD and show that parameter-wise freezing outperforms traditional layer-wise approaches in QNNs. Project page: `https://github.com/Damrl-lab/WSBD-Stochastic-Freezing-Optimizer`.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

The promise of quantum neural networks (QNNs) (Abbas et al., 2021) is currently shackled by a fundamental roadblock: the cost and impracticality of training them. This challenge is twofold. First, the primary method for gradient evaluation, the Parameter-Shift Rule (PSR) (Crooks, 2019; Wierichs et al., 2022), demands at least two full circuit evaluations for every single parameter at every training step. This creates a crippling computational bottleneck that scales linearly with the model size, making gradient-based optimization prohibitively expensive for large-scale QNNs. Adding to this issue is the notorious barren plateau problem, where optimization landscapes flatten exponentially with qubit count, causing gradients to vanish and stall training progress (McClean et al., 2018; Larocca et al., 2025). Trainers are thus caught in a double bind: the gradients that are essential for optimization are both incredibly expensive to compute and often non-impactful when obtained. To reduce the computational burden of training, parameter freezing is a well-established strategy in classical machine learning. However, classical freezing methods, which are often static or operate at a coarse, layer-wise level for transfer learning, are fundamentally unsuited for the unique optimization landscape of QNNs. The highly entangled nature of quantum circuits means a parameter's influence is not neatly confined to a layer, and its importance can change dramatically throughout training. Therefore, a new approach is required, one that is granular, dynamic, and designed specifically to accelerate QNN training from scratch. To address this, we introduce Weighted Stochastic Block Descent (WSBD), a novel optimization algorithm that employs a dynamic, parameter-wise freezing strategy. By intelligently and temporarily allocating computational resources, WSBD directly confronts the high cost of gradient estimation in a manner tailored for

the quantum domain. Our major contributions are:

A Novel QNN-Specific Freezing Optimizer: We propose WSBD, which uses a gradient-derived importance score to temporarily freeze the least influential parameters during training. Its design overcomes the limitations of classical methods by offering a dynamic, stochastic, and parameter-wise approach that preserves the model's full expressive capacity while significantly reducing the number of required circuit evaluations. **Scalable Efficiency:** Our extensive experiments show WSBD is significantly more efficient, saving hundreds of thousands of circuit evaluations compared to optimizers like Adam. We demonstrate that this advantage scales with QNN size, highlighting WSBD's effectiveness for larger, more computationally demanding models. **Theoretical Guarantees and Critical Insights:** We provide a formal proof of convergence for WSBD, establishing its theoretical soundness. Furthermore, our ablation studies provide direct evidence for our central claim, revealing a critical insight for the field: a granular, parameter-wise freezing strategy is more effective than traditional layer-wise approaches for QNNs.

2 BACKGROUND AND RELATED WORK

2.1 QNNs and Key Training Obstacles

A QNN operates on the n-qubit ground state, $|0\rangle^{\otimes n}$ in an exponentially large Hilbert space $\mathcal{H}$. For a given input, $\hat{x_i} \in \mathbb{R}^m$, an encoding circuit, $A(\hat{x_i})$, applies a sequence of quantum gates whose parameters are functions of $\hat{x_i}$ mapping the classical input to a quantum state. Following this, a variational circuit or ansatz, $U(\boldsymbol{\theta})$, applies a sequence of gates organized into L layers. The total unitary transformation of the ansatz is the product of the operation of each layer:

$$U(\boldsymbol{\theta}) = \prod_{l=L}^{1} U_l(\boldsymbol{\theta}_l) W_l \tag{1}$$

where $U_l(\boldsymbol{\theta}_l)$ represents a set of parametrized gates and W_l represents a set of constant gates (e.g., CNOT, CZ, H, etc.) in a given layer l. The final state is given by $|\psi(\boldsymbol{\theta}, \hat{x_i})\rangle = U(\boldsymbol{\theta}) A(\hat{x_i}) |0\rangle^{\otimes n}$. To extract a classical output we must measure the system through a Hermitian observable, M, on the final state, which is the expectation value of the observable:

$$f(\boldsymbol{\theta}, \hat{x_i}) = \langle \psi(\boldsymbol{\theta}, \hat{x_i}) | M | \psi(\boldsymbol{\theta}, \hat{x_i}) \rangle \tag{2}$$

Now we can define an objective or cost function $\mathcal{C}(\boldsymbol{\theta}, \hat{x_i})$ which aims to quantify the difference of the label $\hat{y}_{true}$ and prediction $f(\boldsymbol{\theta}, \hat{x_i})$. A classical optimizer then iteratively updates $\boldsymbol{\theta}$ to minimize the cost. However, this process faces two significant hurdles: **(1) High Cost of Gradient Estimation:** As direct differentiation is infeasible, gradients are computed using the Parameter-Shift Rule (PSR) (Crooks, 2019). The gradient for a parameter θ_k is calculated as:

$$\frac{\partial \mathcal{C}(\boldsymbol{\theta}, \hat{x_i})}{\partial \theta_k} = \varsigma_P [\mathcal{C}(\boldsymbol{\theta} + \frac{\pi}{4\varsigma_P} \tilde{\mathbf{e}}_{\mathbf{k}}, \hat{x_i}) - \mathcal{C}(\boldsymbol{\theta} - \frac{\pi}{4\varsigma_P} \tilde{\mathbf{e}}_{\mathbf{k}}, \hat{x_i})] \tag{3}$$

Here ς_P represents a constant (for the Pauli-matrices, $\varsigma_P = \frac{1}{2}$), $\tilde{\mathbf{e}}_{\mathbf{k}}$ is the k-th standard basis vector (a vector with 1 in the k-th position and 0s elsewhere, meaning the shift is applied only to θ_k for each optimization step). Although exact, this means that computing $|\boldsymbol{\theta}|$ gradients requires $2|\boldsymbol{\theta}| + 1$ total evaluations, one to evaluate the cost, $\mathcal{C}(\boldsymbol{\theta}, \hat{x_i})$, and $2|\boldsymbol{\theta}|$ to evaluate all shifts ($\theta_k \pm \frac{\pi}{2}\ \forall \theta_k \in \boldsymbol{\theta}$), making gradient-based optimization expensive for large-scale QNNs. **(2) Barren Plateaus:** For many QNN architectures, particularly deep or wide ones, the optimization landscape becomes flat (McClean et al., 2018). This is formally expressed as a "barren plateau," where the variance of the cost function's gradient vanishes exponentially with the number of qubits, n:

$$\mathrm{Var}\left[\frac{\partial \mathcal{C}(\boldsymbol{\theta}, \hat{x_i})}{\partial \theta_k}\right] \propto \mathcal{O}\left(\frac{1}{\exp(n)}\right) \tag{4}$$

This means gradients become exponentially small, providing no meaningful direction for optimization and effectively stalling the training process. *These combined challenges make scaling QNNs a formidable task, necessitating new strategies to make training more efficient and effective.*

2.2 Strategies for Accelerating QNN Training

Several research directions have aimed to mitigate the high cost of QNN training through adaptive ansatz methods and circuit optimization (Huo et al., 2025). One popular approach is pruning and compression strategies which seeks to simplify the QNN by permanently removing non-influential parameters. This method, similar to freezing, reduces the number of forward passes required by 2 for each parameter removed. Frameworks like (Hu et al., 2022b) and algorithms such as (Kulshrestha et al., 2024; Kverne et al., 2025; Sim et al., 2021) have shown that removing a fraction of parameters can speed up training and improve performance. Similarly, some work has explored gradient compression techniques (Alistarh et al., 2017; Lin et al., 2017), however this fails to address the core computational bottleneck of gradient estimation itself. A more dynamic strategy is parameter freezing, which is the focus of this work. Unlike

pruning, freezing does not change the circuit architecture; instead, it temporarily deactivates the training of selected parameters, concentrating computational effort on a smaller, active subset. Prior work on quantum parameter freezing is limited. Layer-wise freezing has been employed to incrementally grow QNNs (Skolik et al., 2021), and dropout-inspired methods have been proposed to improve generalization (Scala et al., 2023). However, these approaches do not aim to accelerate training from scratch, nor do they leverage fine-grained, adaptive control over which parameters are frozen during training. In contrast, freezing strategies are well-established in classical machine learning. They are central to transfer learning (Hu et al., 2022a; Yosinski et al., 2014; Kornblith et al., 2018; He et al., 2019) and have also been used to reduce training time by avoiding backpropagation through certain layers (Brock et al., 2017). However, the idea of selectively freezing parameters in QNNs, from initialization and throughout training, as a way to improve training efficiency remains unexplored. *To address this gap, we draw inspiration from classical ML to introduce a method that accelerates QNN training without permanently altering the circuit architecture. We propose a dynamic, parameter-wise freezing strategy that intelligently adapts throughout the training process by concentrating computational resources on the most impactful parameters at each learning stage.*

3 WEIGHTED STOCHASTIC BLOCK DESCENT

In this section, we introduce our freezing-based QNN-specific optimizer, **Weighted Stochastic Block Descent (WSBD)**. The key idea is to intelligently focus computational resources on subsets of parameters that are most influential at different stages of the optimization process. Our optimizer is built upon the PSR (Eq. 3) and can be adapted to **work with any classical-based gradient optimizer** such as SGD, Adam, etc. At the core of WSBD is an importance score, $\mathcal{I}_p$, which quantifies the influence of each parameter θ_k over a given training window τ. While several metrics for this score are possible, we selected the Sum of Gradients which accumulates the gradients of each parameter in a given window:

$$\mathcal{I}_p(\theta_k) = \sum_{t=1}^{\tau} \frac{\partial \mathcal{C}(\boldsymbol{\theta}^{(t)}, \hat{x_t})}{\partial \theta_k} \tag{5}$$

Our empirical analysis, detailed in Appendix B, confirms this metric provides the best balance of performance and computational efficiency. It robustly identifies parameters with a consistent impact, unlike variance-based scores which can be misleading in barren plateaus, and avoids the significant overhead of second-order methods. After computing all importance scores we take its absolute value and add a small constant $\epsilon = 10^{-8}$ such that $\mathcal{I}_p(\theta_k) = |\mathcal{I}_p(\theta_k)| + \epsilon$, ensuring each importance score is greater than zero which will be crucial for proving convergence for WSBD (Appendix A). The WSBD optimizer is summarized in Algorithm 1. The algorithm proceeds in windows of size τ. Within each window (lines 4-14), it performs the following steps for each training instance: It starts by computing the gradients for the parameters in the active set $\mathbb{A}$ using PSR (lines 7-9). It then updates the active parameters using a classical optimizer (e.g., SGD, Adam, etc.) (line 10). Finally it accumulates the gradients for each active parameter to update their respective importance scores $\mathcal{I}_p$ (lines 11-13). After τ steps, the algorithm uses the calculated importance scores to stochastically determine which parameters to freeze. A probability distribution is formed where the probability, p_k, of a parameter θ_k remaining active is its normalized importance score: $p_k = \frac{|\mathcal{I}_p(\theta_k)|+\epsilon}{\sum_{i=1}^{|\boldsymbol{\theta}|}(|\mathcal{I}_p(\theta_i)|+\epsilon)}$ (line 16). Consequently, parameters with lower importance scores are assigned a higher probability of being frozen. Based on this distribution, the algorithm stochastically selects a new set of frozen parameters, with the total size of the set determined by the freeze percentile λ_f (line 18). This stochastic selection, in contrast to deterministically freezing the parameters with the lowest scores, promotes greater exploration of the optimization landscape by preventing permanent freezing.

A crucial feature of WSBD is how it manages the importance scores across freezing cycles. When a parameter becomes part of the new active set, its importance score is reset to zero (lines 19-21). However, the scores of parameters that remain frozen are cached (i.e., preserved). This reset mechanism is critical for the algorithm's adaptiveness, as it enables rapid assessment of newly activated parameters' current contributions while preventing legacy bias from parameters that were important early in training but have since become less influential. Our ablation studies confirm this empirically in Sec. 6, showing that the reset and stochastic freezing mechanisms are more effective. The classical overhead of our optimizer is linear, scaling as $O(|\boldsymbol{\theta}|)$ with the number of parameters for updating importance scores and executing the stochastic freeze. This is insignificant when compared to the substantial cost of the quantum circuit evaluations required for gradient estimation which is of order $O(\mathcal{M}2|\boldsymbol{\theta}|)$ where$\mathcal{M}$ represents the number of times each forward pass is run on quantum hardware (commonly called a shot or circuit evaluation) which is needed as the state collapse will only give you a sin-

Algorithm 1 Weighted Stochastic Block Descent (WSBD)

Input: Initial parameters $\boldsymbol{\theta}^{(0)}$, Freeze Threshold $\lambda_f \in [0, 1)$, Learning Rate $\eta > 0$, Training Window $\tau \in \mathbb{N}_+$
Initialize: Importance scores $\mathcal{I}_p(\theta_k) \leftarrow 0$ for all $k = 1, \ldots, |\boldsymbol{\theta}|$, Active set $\mathbb{A} \leftarrow \boldsymbol{\theta}$
while not converged **do**
  for $t = 1, \ldots, \tau$ **do** ▷ Iterate within the training window
    Sample data instance $(\hat{x}_t, \hat{y}_t)$
    Initialize gradient vector $g_t \leftarrow \mathbf{0} \in \mathbb{R}^{|\boldsymbol{\theta}|}$
    for each parameter $\theta_k \in \mathbb{A}$ **do** ▷ Compute gradients only for active parameters
      $g_{t,k} \leftarrow \varsigma_P[\mathcal{C}(\boldsymbol{\theta} + \frac{\pi}{4\varsigma_P}\tilde{\mathbf{e}}_\mathbf{k}, \hat{x}_t) - \mathcal{C}(\boldsymbol{\theta} - \frac{\pi}{4\varsigma_P}\tilde{\mathbf{e}}_\mathbf{k}, \hat{x}_t)]$
    end for
    $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \cdot g_t$ ▷ Update parameters with any optimizer (frozen parameter grads are 0)
    for each parameter $\theta_k \in \mathbb{A}$ **do** ▷ Update importance scores for active parameters
      $\mathcal{I}_p^{(t+1)}(\theta_k) \leftarrow \mathcal{I}_p^{(t)}(\theta_k) + g_{t,k}$ ▷ Using Sum of Gradients metric
    end for
  end for
  ▷ After window, stochastically select the new active set
  Define selection probability $P = [p_1, \cdots, p_k, \cdots, p_{|\boldsymbol{\theta}|}]$: $p_k \leftarrow \frac{|\mathcal{I}_p(\theta_k)|+\epsilon}{\sum_{i=1}^{|\boldsymbol{\theta}|}(|\mathcal{I}_p(\theta_i)|+\epsilon)}$ for all k
  $N_{active} \leftarrow \lceil (1 - \lambda_f) \cdot |\boldsymbol{\theta}| \rceil$ ▷ Calculate size of the new active set
  $\mathbb{A} \leftarrow \text{Sample}(N_{active}, \boldsymbol{\theta}, P)$ ▷ Sample N_{active} params using weights P
  for each parameter $\theta_k \in \mathbb{A}$ **do** ▷ Reset scores for newly active parameters
    $\mathcal{I}_p(\theta_k) \leftarrow 0$
  end for
end while
Return: Optimized parameters $\boldsymbol{\theta}$

gle output opposed to the entire output distribution needed to evaluate the cost. The higher value for $\mathcal{M}$ the more accurate estimate of $f(\boldsymbol{\theta}, \hat{x}_i)$ is given (commonly $\mathcal{M} \sim 1000$). We optimized the freeze threshold λ_f and training window τ via grid search, finding the best performance with λ_f=70% and τ=100. The full tuning process is detailed in Appendix B.

4 THEORETICAL CONVERGENCE FRAMEWORK

We extend the theoretical guarantees of WSBD to a broad class of optimizers $\mathcal{O}$. WSBD acts as a stochastic coordinate mask, scaling the expected descent direction without amplifying variance.

Generalized Update Rule. Let $\mathcal{O}$ generate an update vector u_t based on $\nabla\mathcal{C}^{(t)}$. The WSBD-augmented update is:

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t(\delta^{(t)} \odot u_t) \tag{6}$$

Assumptions. We assume: (1) $\mathcal{C}$ is L-smooth (proven for QNNs in Appendix A, Theorem 1) and bounded below by $\mathcal{C}^*$; (2) $\mathcal{O}$ produces a valid descent direction: $\mathbb{E}[\langle \nabla\mathcal{C}^{(t)}, u_t \rangle \mid \mathcal{F}_t] \geq c_1 \|\nabla\mathcal{C}^{(t)}\|^2$; (3) The update magnitude is bounded: $\mathbb{E}[\|u_t\|^2 \mid \mathcal{F}_t] \leq K$; and (4) The mask $\delta^{(t)}$ is independent of u_t with minimum selection probability $p_{\min} > 0$ which is ensured by ϵ.

Proof. By L-smoothness, the objective decreases as:

$$\mathcal{C}^{(t+1)} \leq \mathcal{C}^{(t)} - \eta_t \langle \nabla\mathcal{C}^{(t)}, \delta^{(t)} \odot u_t \rangle + \frac{L\eta_t^2}{2} \|\delta^{(t)} \odot u_t\|^2$$

Taking expectations conditioned on $\mathcal{F}_t$, we utilize the independence of the mask $\delta^{(t)}$ and the descent assumption to bound the first-order term by $-\eta_t p_{\min} c_1 \mathbb{E}[\|\nabla\mathcal{C}^{(t)}\|^2]$. The second-order term is a contraction, bounded by $\frac{L\eta_t^2}{2}K$. Summing these expectations over T steps:

$$p_{\min} c_1 \sum_{t=0}^{T-1} \eta_t \mathbb{E}[\|\nabla\mathcal{C}^{(t)}\|^2] \leq \mathcal{C}^{(0)} - \mathcal{C}^* + \frac{LK}{2} \sum_{t=0}^{T-1} \eta_t^2$$

Given standard step-size conditions ($\sum \eta_t = \infty, \sum \eta_t^2 < \infty$), the RHS remains finite while the LHS accumulates. This implies $\liminf_{t\to\infty} \mathbb{E}[\|\nabla\mathcal{C}^{(t)}\|^2] = 0$. (See Appendix A for full derivations including SGD and AMSGrad variants). This means WSBD will not change the convergence guarantees of other optimizers as the active parameter set changes throughout. □

5 EXPERIMENTAL SETUP

We benchmark the performance of our proposed WSBD optimizer across three representative tasks: MNIST classification, the parity problem, and a Variational Quantum Eigensolver (VQE) ground-state problem. Our goal is to demonstrate that WSBD integrates with classical optimizers to improve convergence efficiency. To this end, we compare it against a suite of standard optimizers (summarized in Table 1) using the QNN architecture shown in Fig. 1. To assess both near-term practicality and future potential, we conduct all experiments in two settings: an idealized, noise-free simulation and a realistic noisy environment that emulates current hardware by incorporating decoherence, depolarizing noise, and readout errors calibrated from the IBM Heron quantum device (IBM, 2025).

5.1 Tasks and Datasets

We evaluate WSBD on three tasks that represent different challenges common in QML. The first task is MNIST image classification (LeCun et al., 2002), where each image is preprocessed with Principal Component Analysis to match the number of qubits and is then encoded into the quantum state space using angle encoding. For a reduced feature vector $\hat{x_i}$ the operator $A(\hat{x_i})$ applies single-qubit rotations: $A(\hat{x_i})\left|0\right\rangle^{\otimes n} = (\bigotimes_{i=1}^{n} R_X(\hat{x_i}))\left|0\right\rangle^{\otimes n}$. The second task is the parity problem. For a binary input string $S = \langle x_1 x_2 \dots x_n \rangle$ with $x_k \in \{0, 1\}$ the parity function is $P(S) = (\sum_{k=1}^{n} x_k) \bmod 2$. The encoding operator $A(\hat{x_i})_{Parity}$ applies a Pauli-X gate to the k-th qubit if $x_k = 1$. The resulting initial state is: $A(\hat{x_i})_{Parity}\left|0\right\rangle^{\otimes n} = (\bigotimes_{k=1}^{n} X^{x_k})\left|0\right\rangle^{\otimes n}$.

The third task evaluates WSBD in a noisy, hardware-realistic setting through a Variational Quantum Eigensolver (VQE). VQE aims to approximate the ground state energy of a given Hamiltonian H by preparing a parametrized quantum state $|\psi(\boldsymbol{\theta})\rangle$ and minimizing the energy expectation value $E(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})| M |\psi(\boldsymbol{\theta})\rangle$. Here we study the one-dimensional transverse-field Ising model (TFIM) Hamiltonian, defined for n qubits as: $H_{TFIM} = -J\sum_{i=1}^{n-1} Z_i Z_{i+1} - h\sum_{i=1}^{n} X_i$. This Hamiltonian is widely used as a benchmark in VQE studies because it is non-trivial yet efficiently simulatable, making it ideal for analyzing optimizer performance under noise (Peruzzo et al., 2014; Kandala et al., 2017). Unlike MNIST and parity, the VQE problem does not require an explicit data encoding stage; the ansatz circuit directly parameterizes candidate ground states. The cost function is simply the expectation value of H_{TFIM}, and optimization seeks to minimize this energy. Because each parameter-shift evaluation involves stochastic measurement outcomes and is further distorted by decoherence and gate errors, this task provides a stringent test of WSBD under realistic noise conditions. We simulate noise using calibration data from the IBM Heron backend, including amplitude damping (T_1), phase damping (T_2), depolarizing noise on one- and two-qubit gates, idle errors, and readout assignment errors. This models the primary error channels of near-term superconducting hardware (IBM, 2025).

5.2 Experimental Design

We conduct experiments on quantum neural networks with varying architectures, including a 4-qubit 2-layer model, an 8-qubit 3-layer model, and a 10-qubit 5-layer model. These architectures represent challenging, large-scale benchmarks by current quantum standards, chosen specifically to demonstrate our optimizer's advantage where training costs become prohibitive. For MNIST we use an ansatz with R_X and R_Z single-qubit rotations and cyclic entanglement, while for the parity problem the encoding $A(\hat{x_i})_{Parity}$ is used and only a single qubit is measured. All experiments are performed on simulated quantum hardware using the PennyLane framework (Bergholm et al., 2018). We repeat each experiment five times for optimizers with stochastic elements (such as the freezing mechanism in WSBD). WSBD consistently shows stability, with a coefficient of variation in convergence between 0–3%. Reported results are averaged over these runs. For the noisy ground-state energy experiments, we use the same circuit structure and test 1-, 2-, and 4-qubit systems with varying depth.

While wall-clock time is generally confounded by backend throughput, queueing delays, and simulator performance, we performed a small *real-hardware calibration* to contextualize FP savings. Using 100 random QNNs (4q-2l, 8q-3l, 10q-5l), we measured forward-pass latencies on the real **IBM Heron R2 processor** as **1.33 ± 0.24 seconds of QPU execution time** and **4.74 ± 0.76 seconds of end-to-end wall time** (including submission and return). Thus, *each forward pass saved by WSBD corresponds to roughly five seconds of wall time* on current hardware (IBM, 2025). These numbers can fluctuate depending on the backend and system load, since queue lengths and calibration states vary, but they nevertheless illustrate that FP reductions translate directly into practical time savings. For this reason, we retain FP-to-target as our *primary evaluation metric*, since it provides a reproducible, hardware-agnostic measure of algorithmic efficiency, while our calibration confirms that FP reductions yield substantial wall-clock benefits in practice.

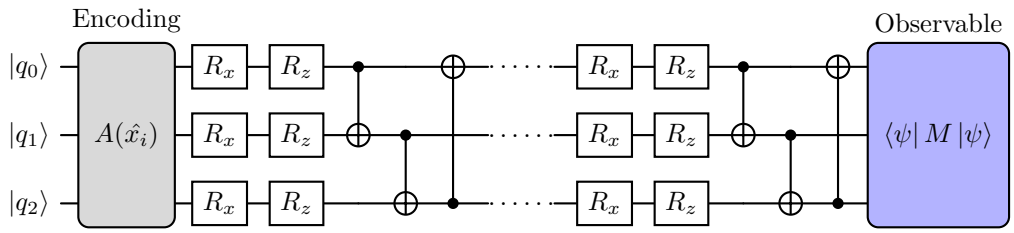


Figure 1: VQA architecture used in this study with data encoding, variational layers, and measurement. The circuit begins with an encoding block $A(\hat{x}_i)$ that maps classical data to quantum states, followed by alternating parametrized rotations and entangling CNOT gates. The final measurement observable $\langle\psi| M |\psi\rangle$ extracts the computational result.

Table 1: Comparison of optimizer characteristics. $|\boldsymbol{\theta}|$ is the total number of parameters, and $\mathbb{A}$ is the set of active (non-frozen) parameters, where $|\mathbb{A}| \leq |\boldsymbol{\theta}|$. $\mathcal{M}$ represents the number of shots (circuit evaluations) each forward pass is run on the circuit, where a higher value for $\mathcal{M}$ gives more precise outputs.

		Selection		**Freezing**		**Circuit Evals**
Optimizer	**Gradient**	**Stochastic**	**Det.**	**Layer**	**Parameter**	**Per Step**
SGD	✓					$\mathcal{M}(2\|\boldsymbol{\theta}\|+1)$
WSBD-SGD	✓	✓			✓	$\mathcal{M}(2\|\mathbb{A}\|+1)$
DBD-SGD	✓		✓		✓	$\mathcal{M}(2\|\mathbb{A}\|+1)$
SBD-SGD	✓	✓			✓	$\mathcal{M}(2\|\mathbb{A}\|+1)$
L-WSBD-SGD	✓	✓		✓		$\mathcal{M}(2\|\mathbb{A}\|+1)$
WSBD-NO-RESET	✓	✓			✓	$\mathcal{M}(2\|\mathbb{A}\|+1)$
ADAM	✓					$\mathcal{M}(2\|\boldsymbol{\theta}\|+1)$
WSBD-ADAM	✓	✓			✓	$\mathcal{M}(2\|\mathbb{A}\|+1)$
SPSA		✓				$2\mathcal{M}$
NELDER-MEAD			✓			Evals $\in (\mathcal{M}, \mathcal{M}\|\boldsymbol{\theta}\|]$
BAYESIAN		✓				$\mathcal{M}$

5.3 Optimizers Considered

We compare WSBD against both classical gradient-based and gradient-free optimizers. Standard baselines include stochastic gradient descent (SGD) and Adam (Kingma, 2014). We also evaluate Nelder-Mead (Nelder and Mead, 1965), Simultaneous Perturbation Stochastic Approximation (SPSA) (Spall, 2002), and Bayesian optimization (Jones et al., 1998), which are widely used in quantum machine learning due to their efficiency in the absence of explicit gradients. In addition, we perform ablation studies with several WSBD variants. Deterministic Block Descent (DBD) freezes parameters with the lowest importance scores deterministically; this variant is to showcase the value of the stochastic freezing element in WSBD. Stochastic Block Descent (SBD) randomly freezes parameters without importance scores, this variant showcases that WSBD performance improvement doesn't come from freezing directly but its intelligent importance tracking. Layer-wise WSBD freezes whole layers based on summed importance scores. WSBD without reset does not reset importance scores of recently active parameters, this variant showcases that resetting scores enables more rapid assessment of parameter importance to speed up training. These comparisons help isolate the contribution of each design choice in WSBD.

6 EVALUATION AND RESULTS

In this section we evaluate WSBD across three representative QNN tasks. Our analysis focuses on three themes: (i) scalable efficiency as circuit size increases, (ii) robustness under noise in hardware-realistic environments, and (iii) the importance of WSBD's design choices through ablation studies. Together these experiments provide both empirical validation and insight into the mechanisms that make WSBD effective.

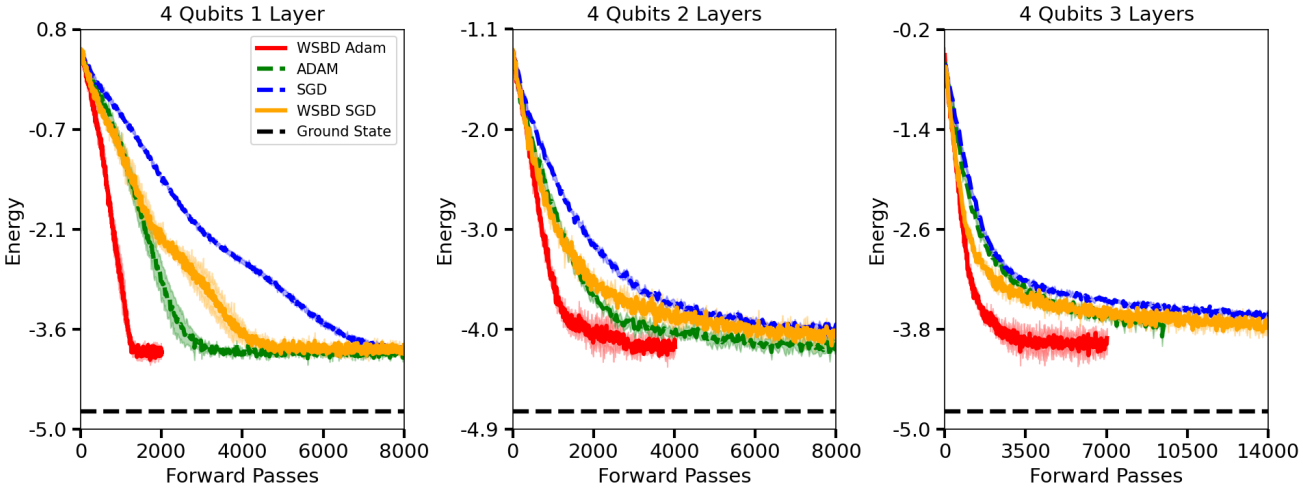


Figure 2: Training curves using Adam, SGD and their WSBD counterpart optimizers on the VQE problem. The black dotted line represents the ground state energy each optimizer aims to reach (closer is better). Each optimizer was trained until convergence. For the full training curves of all QNN sizes see Figure 4. Table 2a shows the percentage reduction in forward passes needed to converge using both WSBD SGD and WSBD Adam.

6.1 Scalable Efficiency and Practical Savings in QNN Training

A central motivation for WSBD is to accelerate training in large QNNs, where the cost of gradient evaluation scales linearly in the number of parameters $|\boldsymbol{\theta}|$. Table 3 shows the raw number of forward passes (FP) required for convergence across models of increasing size and how many forward passes are saved for all problems. While improvements are already visible on 4-qubit systems, the relative benefit of WSBD compounds with scale: on 8- and 10-qubit models, WSBD-SGD and WSBD-Adam save tens of thousands of forward passes compared to their baselines. The reason for this trend is twofold. First, larger circuits require proportionally more parameter-shift evaluations per update, which magnifies the savings achieved when WSBD focuses only on a subset of parameters. Second, barren plateaus — regions where gradients vanish exponentially with qubit count — become more prevalent as the system grows. In these landscapes, standard optimizers expend computational effort on parameters whose gradients are effectively noise. By concentrating updates on parameters with consistently strong gradient signals, WSBD both reduces overhead and steers optimization away from flat directions. This explains why WSBD not only improves convergence speed but also enables successful optimization where SGD or Adam begin to stagnate (e.g., the 4-qubit parity task in Fig. 5d where WSBD-SGD converged and SGD didn't). In summary, WSBD's efficiency gains are not merely numerical savings but a reflection of how parameter-wise freezing interacts with the geometry of high-dimensional quantum optimization landscapes. The advantage therefore scales with problem size, making WSBD particularly suitable for the next generation of deeper QNNs.

Based on our real-hardware calibration on the IBM Heron R2 processor (Section 5.2), each forward pass requires on average ≈5 seconds of end-to-end wall time. For example, in the VQE ground-state energy problem, WSBD-Adam reaches the target energy in roughly $3,252$ fewer forward passes on average compared to Adam, corresponding to a reduction of ≈ 4.5 hours of wall-clock compute. For the MNIST problem the savings are more extreme with WSBD-Adam reaching its maximum accuracy while saving on average 36.3 hours of wall-clock compute and WSBD-SGD saving 89.9 hours of compute. Similarly for the parity problem, WSBD-Adam reached 100% accuracy saving 8.8 hours of wall-clock compute, while WSBD-SGD saved **216** hours. These back-of-the-envelope calculations, while approximate and dependent on backend conditions, underscore the practical value of FP reductions: training speedups of tens of percent (Tables 2) directly translate into *hours of quantum compute saved*. Even at modest qubit counts, these reductions are practically significant: current quantum hardware is both limited and in high demand, with academic users often allocated only a few hours of device access per month. Saving several hours of wall-clock compute per training run can therefore determine whether an experiment is feasible at all, and such savings will compound as circuit widths and depths scale.

Table 2: Side-by-side results. Left: percentage reduction in forward passes (FP) to reach target energy for the noisy VQE task (higher is better). Right: max accuracy for MNIST and Parity, plus speedup when adding WSBD. See Figures 4, 5 and 6 for the respective training curves on all problems.

(a) VQE: % FP reduction needed to reach target energy. Target energy represents the converged values of Adam and SGD.

Model Size	**WSBD-SGD v.s. SGD**	**WSBD-Adam v.s. Adam**
1q, 2l	**25.9%**	**43.9%**
1q, 4l	**20.6%**	**83.1%**
1q, 8l	**21.5%**	**69.0%**
2q, 1l	**28.1%**	**55.3%**
2q, 2l	**58.2%**	**82.8%**
2q, 4l	**40.0%**	**65.5%**
4q, 1l	**34.0%**	**59.9%**
4q, 2l	**23.3%**	**39.2%**
4q, 3l	**46.4%**	**76.5%**

(b) MNIST & Parity: Max accuracy reached for each optimizer. The 'Speedup' compares the forward passes required for each optimizer to reach its own reported maximum accuracy. As the maximum accuracies achieved by WSBD and its baseline counterparts were consistently similar, this metric provides a direct comparison of training efficiency.

	Standard		WSBD		Speedup (%)
Model	SGD	Adam	SGD	Adam	(SGD, Adam)
MNIST Classification Max Accuracy (%)					
4q 2l	58.1	70.2	59.9	70.4	80.3, 46.2
8q 3l	49.6	68.5	49.9	68.7	38.5, 30.2
10q 5l	56.6	73.7	57.5	74.2	29.3, 17.6
Parity Problem Max Accuracy					
4q 2l	50	100	100	100	∞, 7.6
8q 3l	100	100	100	100	48.8, 3.0
10q 5l	50	100	50	100	N/A, 20.7

6.2 Robustness Under Noise

We next examine WSBD under noisy, hardware-realistic conditions. For the VQE task with transverse-field Ising Hamiltonians, we simulate errors calibrated from IBM Heron devices, including amplitude/phase damping, depolarization, and readout errors. Results in Tables 2a and 3 reveal that WSBD retains significant advantages even when every gate and measurement is perturbed. For example, on the 2-qubit, 2-layer VQE problem, WSBD-SGD required 58% fewer forward passes than SGD, while WSBD-Adam reduced forward passes by more than 80%.

These results can be interpreted mathematically. Let $\nabla f(\boldsymbol{\theta})$ denote the true gradient of the cost function in the VQE problem, and let ε represent a noise channel acting on each circuit evaluation. Under noise, the effective gradient becomes:$\tilde{\nabla} f(\boldsymbol{\theta}) = \nabla f(\boldsymbol{\theta}) + \varepsilon(\boldsymbol{\theta})$, where $\varepsilon(\boldsymbol{\theta})$ is a stochastic disturbance induced by the various error sources. Standard optimizers attempt to estimate all components of ∇f, but many of these are small in magnitude and easily overwhelmed by ε. WSBD, by design, biases updates toward parameters with consistently large cumulative gradients. This adaptive focus acts as a form of noise filtering: resources are less likely to be wasted on low-signal parameters, and optimization proceeds along directions where the true signal dominates the noise. The practical consequence is that WSBD remains effective even under the severe error models of near-term devices. In fact, its bias toward "signal-rich" parameters may be even more valuable in noisy settings than in idealized simulations, as evidenced by the divergence between baseline and WSBD performance in Tables 2a and 2b where the performance improvements were often greater in the noisy problem or the training Figures 2 and 4 where WSBD often reached a lower target energy overall.

6.3 Why Each Component Matters: Ablation Study

Finally, we analyze the contribution of WSBD's design choices through ablations (Figure 5). Four variants were considered: Deterministic Block Descent (DBD): freezes the lowest-importance parameters deterministically. Stochastic Block Descent (SBD): freezes parameters randomly without importance scores. Layer-wise WSBD: aggregates importance scores per layer and freezes entire layers. WSBD without reset: retains past importance scores when parameters are reactivated. The results highlight several key insights. First, stochastic selection is crucial: DBD achieves convergence but often plateaus at higher losses, while stochastic freezing avoids premature elimination of parameters that may regain importance later. Second, importance scores provide the main source of WSBD's advantage; SBD performs similarly to plain SGD, confirming that freezing alone is insufficient without intelligent parameter ranking. Third, parameter-wise granularity is essential for QNNs, where influence is not neatly aligned with layer boundaries. WSBD-

SGD was nearly 50% faster than its layer-wise counterpart, demonstrating that coarse freezing strategies cannot capture the entangled parameter dependencies of quantum circuits. Finally, the reset mechanism prevents stale importance values from dominating; without resets, optimization stagnates on both parity and MNIST tasks. Together, these ablations show that each design element of WSBD is necessary. Far from being an incidental combination of heuristics, WSBD's success depends on the interplay of stochasticity, fine-grained importance tracking, and adaptive resetting. Finally for the gradient-free optimizers, although Simultaneous Perturbation Stochastic Approximation (SPSA) (Spall, 2002), Nelder-Mead (Nelder and Mead, 1965), and Bayesian optimization (Jones et al., 1998) require fewer circuit evaluations per step, they consistently stagnated on larger models. This suggests that their low-cost updates do not provide enough directional information for challenging optimization landscapes.

7 CONCLUSION

In this work, we have introduced Weighted Stochastic Block Descent, a dynamic parameter freezing optimizer designed to directly confront the burdens that hinder QNN training. Our results showcase that by intelligently focusing on a subset of the most influential parameters, WSBD consistently accelerates convergence and achieves superior performance compared to standard optimizers and gradient-free methods. Our ablation studies validate the core principles behind WSBD's design: the efficacy of its stochastic selection process, the use of gradient-based importance scores, the critical need for a granular parameter-wise approach, and the adaptive score-resetting mechanism. These elements, supported by a formal proof of convergence, establish dynamic parameter freezing as a potent and practical strategy for QNN optimization. Beyond these immediate results, this work opens up several exciting future directions. The principles of WSBD could be extended to create hardware-aware optimizers, where freezing is tailored to the noise profiles of specific quantum devices. Furthermore, the ability to make training more tractable may enable researchers to explore novel QNN architectures that are currently considered too deep or wide to train effectively. *Ultimately, by transforming the training process from a brute-force calculation into an intelligent allocation of scarce resources, WSBD represents a significant step toward making QML a practical reality.*

Acknowledgements

This work was supported by the following NSF grants: CSR-2402328, CAREER-2338457, CSR-2406069, CSR-2323100, and HRD-2225201, as well as by the DOE Office of Science User Facility, supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, using NERSC award DDR-ERCAP0035598. We also acknowledge support from Rice University and Florida International University.

References

(2025). Ibm quantum platform. `https://quantum.cloud.ibm.com/`.

Abbas, A., Sutter, D., Zoufal, C., Lucchi, A., Figalli, A., and Woerner, S. (2021). The power of quantum neural networks. *Nature Computational Science*, 1(6):403–409.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. (2017). Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30.

Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Ahmed, S., Ajith, V., Alam, M. S., Alonso-Linaje, G., AkashNarayanan, B., Asadi, A., et al. (2018). Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*.

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. (2017). Freezeout: Accelerate training by progressively freezing layers. *arXiv preprint arXiv:1706.04983*.

Crooks, G. E. (2019). Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv preprint arXiv:1905.13311*.

He, K., Girshick, R., and Dollár, P. (2019). Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4918–4927.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2022a). Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Hu, Z., Dong, P., Wang, Z., Lin, Y., Wang, Y., and Jiang, W. (2022b). Quantum neural network compression. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9.

Huo, Y., Wei, J., Kverne, C., Akewar, M., Bhimani, J., and Patel, T. (2025). Revisiting noise-adaptive

transpilation in quantum computing: How much impact does it have? *arXiv preprint arXiv:2507.01195*.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.

Kandala, A., Mezzacapo, A., Temme, K., Takita, M., Brink, M., Chow, J. M., and Gambetta, J. M. (2017). Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *nature*, 549(7671):242–246.

Kingma, D. P. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kornblith, S., Shlens, J., and Le, Q. V. (2018). Do better imagenet models transfer better? *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2656–2666.

Kulshrestha, A., Liu, X., Ushijima-Mwesigwa, H., Bach, B., and Safro, I. (2024). Qadaprune: Adaptive parameter pruning for training variational quantum circuits. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 2, pages 120–125. IEEE.

Kverne, C., Akewar, M., Huo, Y., Patel, T., and Bhimani, J. (2025). Quantum neural networks need checkpointing. In *Proceedings of the 17th ACM Workshop on Hot Topics in Storage and File Systems*, pages 93–99.

Larocca, M., Thanasilp, S., Wang, S., Sharma, K., Biamonte, J., Coles, P. J., Cincio, L., McClean, J. R., Holmes, Z., and Cerezo, M. (2025). Barren plateaus in variational quantum computing. *Nature Reviews Physics*, pages 1–16.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (2002). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, W. J. (2017). Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*.

McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., and Neven, H. (2018). Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308–313.

Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., Aspuru-Guzik, A., and O'brien, J. L. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):4213.

Reddi, S. J., Kale, S., and Kumar, S. (2019). On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*.

Scala, F., Ceschini, A., Panella, M., and Gerace, D. (2023). A general approach to dropout in quantum neural networks. *Advanced Quantum Technologies*, page 2300220.

Sim, S., Romero, J., Gonthier, J. F., and Kunitsa, A. A. (2021). Adaptive pruning-based optimization of parameterized quantum circuits. *Quantum Science and Technology*, 6(2):025019.

Skolik, A., McClean, J. R., Mohseni, M., Van Der Smagt, P., and Leib, M. (2021). Layerwise learning for quantum neural networks. *Quantum Machine Intelligence*, 3(1):5.

Spall, J. C. (2002). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341.

Welford, B. P. (1962). Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420.

Wierichs, D., Izaac, J., Wang, C., and Lin, C. Y.-Y. (2022). General parameter-shift rules for quantum gradients. *Quantum*, 6:677.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes.** WSBD is summarized in Algorithm 1 with proof and assumptions given in Appendix A.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes.** The time complexity for WSBD is given in Section 3 being $O(|\boldsymbol{\theta}|)$.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **Yes.** All source code

is provided in the supplementary material with hardware and dependencies specification given in Appendix C.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. **Yes.** See Appendix A.
   (b) Complete proofs of all theoretical results. **Yes.** L-smoothness and convergence is proven in Appendix A.
   (c) Clear explanations of any assumptions. **Yes.** Standard assumptions for convergence is given in Appendix A.1.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Yes.** All code to reproduce our findings are available on GitHub.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes.** Section 3 and Appendix B shows the importance score, freezing percentile, and training window used.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes.** See Section 5.2 with a variation in convergence less than 3%.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes.** Both quantum and classical hardware used is reported in Appendix C.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. **Yes.** MNIST dataset is properly cited.
   (b) The license information of the assets, if applicable. **Not Applicable.**
   (c) New assets either in the supplemental material or as a URL, if applicable. **Not Applicable.**
   (d) Information about consent from data providers/curators. **Not Applicable.**
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not Applicable.**

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. **Not Applicable.**
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable.**
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable.**

APPENDIX

A THEORETICAL ANALYSIS

In this section, we formally prove that the objective functions of parametrized quantum neural networks (QNNs) are L-smooth. Rather than computing the Hessian explicitly or bounding it term by term, our strategy will be to study how the gradient changes along parameter paths. By carefully tracking commutator structures and using only boundedness of the generators and the observable, we arrive at a direct and conceptually transparent proof of smoothness.

Theorem 1 (Smoothness of QNN objectives)**.** *Let the objective function be denoted by*

$$f(\boldsymbol{\theta}) \;=\; \langle \psi_0 \mid U(\boldsymbol{\theta})^\dagger M U(\boldsymbol{\theta}) \mid \psi_0 \rangle, \qquad U(\boldsymbol{\theta}) = \prod_{j=1}^{P} U_j(\boldsymbol{\theta}_j), \quad U_j(\boldsymbol{\theta}_j) = e^{-i\,\boldsymbol{\theta}_j G_j},$$

where M is a bounded Hermitian observable, and each G_j is a bounded Hermitian generator. Then the gradient $\nabla f(\boldsymbol{\theta})$ is globally Lipschitz continuous, i.e., there exists a constant $L < \infty$ such that

$$\|\nabla f(\boldsymbol{\theta} + h) - \nabla f(\boldsymbol{\theta})\| \;\le\; L\,\|h\| \qquad \forall \boldsymbol{\theta}, h \in \mathbb{R}^P.$$

Proof. We proceed in steps.

Step 1: Expressing derivatives as commutators. Let $M(\boldsymbol{\theta}) = U(\boldsymbol{\theta})^\dagger M U(\boldsymbol{\theta})$ denote the Heisenberg-evolved observable. A straightforward calculation yields

$$\partial_j f(\boldsymbol{\theta}) = i\,\langle \psi_0 \mid U(\boldsymbol{\theta})^\dagger [M, \tilde{G}_j(\boldsymbol{\theta})] U(\boldsymbol{\theta}) \mid \psi_0 \rangle,$$

where $\tilde{G}_j(\boldsymbol{\theta})$ is the generator G_j conjugated by part of the circuit. Since conjugation by a unitary preserves operator norm, we always have $\|\tilde{G}_j(\boldsymbol{\theta})\| = \|G_j\|$.

Step 2: Following the gradient along a path. Fix $\boldsymbol{\theta}, h \in \mathbb{R}^P$, and consider the interpolation $\boldsymbol{\theta}(t) = \boldsymbol{\theta} + th$, $t \in [0, 1]$. Define

$$\Phi(t) = \nabla f(\boldsymbol{\theta}(t)).$$

By construction,

$$\nabla f(\boldsymbol{\theta} + h) - \nabla f(\boldsymbol{\theta}) = \Phi(1) - \Phi(0) = \int_0^1 \Phi'(t)\, dt.$$

Hence the Lipschitz property will follow once we show that $\|\Phi'(t)\|$ can be bounded uniformly by a constant multiple of $\|h\|$.

Step 3: Structure of $\Phi'(t)$. Differentiating $\Phi_j(t)$ with respect to t produces expressions that are linear combinations of expectation values of *double commutators* of the form

$$\langle \psi_0 \mid U(\boldsymbol{\theta}(t))^\dagger [A, [B, C]] U(\boldsymbol{\theta}(t)) \mid \psi_0 \rangle,$$

where A, B, C are drawn from the bounded set $\{M\} \cup \{\tilde{G}_k(\boldsymbol{\theta}(t))\}_{k=1}^P$. The coefficients of these terms depend linearly on the components of h.

Step 4: Uniform boundedness. By the standard inequality $\|[A,[B,C]]\| \leq 4\|A\|\|B\|\|C\|$ and the invariance of operator norm under conjugation, each of these expectation values is uniformly bounded by a constant multiple of $\|h\|$. Importantly, the constant depends only on $\|M\|$ and the norms of the generators $\|G_j\|$, and is independent of $\boldsymbol{\theta}$, h, or t.

Thus, for all $t \in [0,1]$,

$$\|\Phi'(t)\| \;\leq\; C\,\|h\|$$

for some finite constant C determined solely by the problem data.

Step 5: Conclusion. Combining the previous displays gives

$$\|\nabla f(\boldsymbol{\theta}+h) - \nabla f(\boldsymbol{\theta})\| \leq \int_0^1 \|\Phi'(t)\|\,dt \leq C\,\|h\|.$$

This shows that ∇f is globally Lipschitz, i.e. f is L-smooth, with $L = C$. □

In summary, we have shown that QNN objectives are L-smooth under the mild assumption that the observable and all generators are bounded. The proof did not require constructing the Hessian or computing explicit constants; instead, it relied on the pathwise behavior of the gradient and the boundedness of commutator structures. This establishes smoothness in a direct and conceptually simple way.

A.1 Proof of Convergence WSBD-SGD

This proof of convergence is specifically tailored to the standard gradient descent update rule, thereby formally guaranteeing convergence for the WSBD-SGD variant. While our extensive empirical results demonstrate strong performance and faster convergence for the WSBD-Adam variant across all tasks, a formal convergence proof for adaptive optimizers like Adam is considerably more complex, as it relies on a different set of assumptions to bound the update steps. We believe a formal proof for the adaptive case is achievable and propose its development as a promising direction for future work.

Variable Names: Let $\mathcal{C}^{(t)}$represent the objective function and $\nabla\mathcal{C}^{(t)}$ represent the gradient for the current parameter vector $\boldsymbol{\theta}^{(t)}$. Let $p_k^{(t)} = \mathbb{P}(\theta_k \in \mathbb{A}^{(t)}|\boldsymbol{\theta}^{(t)})$ be the probability that parameter θ_k is in the active set. Let $\delta^{(t)}$ be a vector with entries $\delta_k^{(t)} = 1$ if $\theta_k \in \mathbb{A}^{(t)}$ and 0 if $\theta_k \notin \mathbb{A}^{(t)}$.

Assumptions:

- *Cost Function is L-Smooth*: We prove this property in Theorem 1.
- *Strictly Positive Selection Probability*: $p_k^{(t)} \geq p_{\min} > 0$, which is guaranteed as $p_{min} = \frac{\epsilon}{\sum_{i=1}^{|\boldsymbol{\theta}|}(|\mathcal{I}_p(\theta_i)|+\epsilon)}$.
- *The cost function $\mathcal{C}^{(t)}$ is bounded below optimum $\mathcal{C}^*$.*

Proof. We begin the proof with Assumption 1.

$$\mathcal{C}^{(t+1)} \leq \mathcal{C}^{(t)} + \langle\nabla\mathcal{C}^{(t)}, \boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\rangle + \frac{L}{2}||\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}||^2$$

The WSBD update rule is given by:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta\delta^{(t)}\nabla\mathcal{C}^{(t)}$$

Substituting the update rule gives:

$$\mathcal{C}^{(t+1)} \leq \mathcal{C}^{(t)} - \eta\langle\nabla\mathcal{C}^{(t)}, \delta^{(t)}\nabla\mathcal{C}^{(t)}\rangle + \frac{L\eta^2}{2}||\delta^{(t)}\nabla\mathcal{C}^{(t)}||^2$$

Taking the expectation over the stochasticity of the process (both in the gradient estimation and $\delta^{(t)}$):

$$\mathbb{E}[\mathcal{C}^{(t+1)}] \leq \mathbb{E}[\mathcal{C}^{(t)}] - \eta\mathbb{E}[\langle\nabla\mathcal{C}^{(t)}, \delta^{(t)}\nabla\mathcal{C}^{(t)}\rangle] + \frac{L\eta^2}{2}\mathbb{E}[||\delta^{(t)}\nabla\mathcal{C}^{(t)}||^2]$$

We analyze the inner product term. Using the law of total expectation, conditioned on $\boldsymbol{\theta}^{(t)}$:

$$\mathbb{E}[\langle \nabla \mathcal{C}^{(t)}, \delta^{(t)} \nabla \mathcal{C}^{(t)} \rangle] = \mathbb{E}\left[\mathbb{E}\left[\sum_{k=1}^{|\boldsymbol{\theta}|} (\nabla_k \mathcal{C}^{(t)})^2 \delta_k^{(t)} \middle| \boldsymbol{\theta}^{(t)} \right] \right] = \mathbb{E}\left[\sum_{k=1}^{|\boldsymbol{\theta}|} (\nabla_k \mathcal{C}^{(t)})^2 \mathbb{E}[\delta_k^{(t)} | \boldsymbol{\theta}^{(t)}] \right]$$

$\mathbb{E}[\delta_k^{(t)} | \boldsymbol{\theta}^{(t)}]$ can be simplified to $p_k^{(t)} \geq p_{min}$, thus:

$$\mathbb{E}[\langle \nabla \mathcal{C}^{(t)}, \delta^{(t)} \nabla \mathcal{C}^{(t)} \rangle] \geq p_{\min} \mathbb{E}[||\nabla \mathcal{C}^{(t)}||^2]$$

Next, we bound the final term $||\delta^{(t)} \nabla \mathcal{C}^{(t)}||^2$. Since $\delta_k^{(t)} \in \{0, 1\}$, we have $(\delta_k^{(t)})^2 = \delta_k^{(t)} \leq 1$, thus:

$$\mathbb{E}[||\delta^{(t)} \nabla \mathcal{C}^{(t)}||^2] \leq \mathbb{E}[||\nabla \mathcal{C}^{(t)}||^2]$$

Substituting these bounds back into the main inequality:

$$\mathbb{E}[\mathcal{C}^{(t+1)}] \leq \mathbb{E}[\mathcal{C}^{(t)}] - \eta p_{\min} \mathbb{E}[||\nabla \mathcal{C}^{(t)}||^2] + \frac{L\eta^2}{2} \mathbb{E}[||\nabla \mathcal{C}^{(t)}||^2] = \mathbb{E}[\mathcal{C}^{(t)}] - \eta \left(p_{\min} - \frac{L\eta}{2} \right) \mathbb{E}[||\nabla \mathcal{C}^{(t)}||^2]$$

For convergence, we require the learning rate η to be chosen such that $p_{\min} > \frac{L\eta}{2}$. Let $C_\eta = \eta(p_{\min} - \frac{L\eta}{2})$. Rearranging the inequality:

$$C_\eta \mathbb{E}[||\nabla \mathcal{C}^{(t)}||^2] \leq \mathbb{E}[\mathcal{C}^{(t)}] - \mathbb{E}[\mathcal{C}^{(t+1)}]$$

Now, we sum this inequality from $t = 0$ to $T - 1$:

$$\sum_{t=0}^{T-1} C_\eta \mathbb{E}[||\nabla \mathcal{C}^{(t)}||^2] \leq \sum_{t=0}^{T-1} (\mathbb{E}[\mathcal{C}^{(t)}] - \mathbb{E}[\mathcal{C}^{(t+1)}])$$

This simplifies to:

$$C_\eta \sum_{t=0}^{T-1} \mathbb{E}[||\nabla \mathcal{C}^{(t)}||^2] \leq \mathbb{E}[\mathcal{C}^{(0)}] - \mathbb{E}[\mathcal{C}^{(T)}]$$

Following assumption 3, we have $\mathbb{E}[\mathcal{C}^{(T)}] \geq \mathcal{C}^*$.

$$\sum_{t=0}^{T-1} \mathbb{E}[||\nabla \mathcal{C}^{(t)}||^2] \leq \frac{\mathcal{C}^{(0)} - \mathcal{C}^*}{C_\eta}$$

Dividing by T and taking the limit as $T \to \infty$:

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[||\nabla \mathcal{C}^{(t)}||^2] \leq \lim_{T \to \infty} \frac{\mathcal{C}^{(0)} - \mathcal{C}^*}{T C_\eta} = 0$$

Thus, the expected squared norm of the gradient converges to zero on average, which completes the proof of convergence for WSBD. Convergence is therefore guaranteed for any importance metric $\mathcal{I}_p$ that ensures $p_{min} > 0$, training window $\tau > 0$, and for any freezing threshold λ_f that ensures $|\mathbb{A}| \geq 1$, making WSBD highly tunable. □

A.2 Proof of Convergence for WSBD-ADAM (AMSGrad Variant)

While standard Adam lacks general convergence guarantees in certain non-convex settings (Reddi et al., 2019), its AMSGrad variant restores convergence by enforcing non-increasing adaptive learning rates. As AMSGrad and Adam behave similarly in practice, we provide a convergence proof for **WSBD-AMSGrad**, which serves as a proxy for the WSBD-Adam optimizer used in the main text.

Update Rule. Let $g_t = \nabla \mathcal{C}^{(t)}$ be the (stochastic) gradient at iteration t, and let $\delta^{(t)} \in \{0,1\}^d$ be the coordinate-wise WSBD freezing mask. The masked adaptive moments evolve as:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)(\delta^{(t)} \odot g_t), \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\delta^{(t)} \odot g_t)^2, \\ \hat{v}_t &= \max(\hat{v}_{t-1}, v_t) \qquad \text{(AMSGrad correction)}, \\ \theta^{(t+1)} &= \theta^{(t)} - \eta_t\, \delta^{(t)} \odot \frac{m_t}{\sqrt{\hat{v}_t} + \epsilon}. \end{aligned}$$

Let H_t denote the diagonal preconditioner with entries $(H_t)_{kk} = 1/(\sqrt{\hat{v}_{t,k}} + \epsilon)$.

Additional Assumptions. In addition to the smoothness and bounded-below assumptions from Appendix A.1, we assume:

- **Bounded Gradients:** $\|g_t\|_\infty \le G_\infty$. This implies

$$0 < c_\ell \;\le\; (H_t)_{kk} \;\le\; c_u < \infty.$$

- **Decaying Step Size:** $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$.
- **Mask Selection Probability:** Each coordinate is active with probability at least $p_{\min} > 0$:

$$\mathbb{P}\Big[(\delta^{(t)})_k = 1\Big] \ge p_{\min}.$$

  The mask is independent of stochastic gradient noise conditioned on past iterates.

Proof. By L-smoothness of $\mathcal{C}$,

$$\mathcal{C}^{(t+1)} \le \mathcal{C}^{(t)} + \big\langle \nabla \mathcal{C}^{(t)}, \theta^{(t+1)} - \theta^{(t)} \big\rangle + \frac{L}{2}\big\|\theta^{(t+1)} - \theta^{(t)}\big\|^2. \tag{7}$$

Substituting the update $\theta^{(t+1)} - \theta^{(t)} = -\eta_t\, \delta^{(t)} \odot H_t m_t$, we obtain

$$\mathcal{C}^{(t+1)} \le \mathcal{C}^{(t)} - \eta_t \big\langle \nabla \mathcal{C}^{(t)},\, \delta^{(t)} \odot H_t m_t \big\rangle + \frac{L\eta_t^2}{2}\big\|\delta^{(t)} \odot H_t m_t\big\|^2.$$

Bounding the Quadratic Term. Since $\|g_t\|_\infty \le G_\infty$ and AMSGrad ensures monotone $\hat{v}_t$, standard arguments (Reddi et al., 2019) imply that m_t and $H_t m_t$ are uniformly bounded. Thus, there exists $K > 0$ such that

$$\big\|\delta^{(t)} \odot H_t m_t\big\|^2 \le K,$$

and therefore

$$\frac{L\eta_t^2}{2}\big\|\delta^{(t)} \odot H_t m_t\big\|^2 \le \frac{LK}{2}\eta_t^2.$$

Analyzing the Descent Term. Taking conditional expectation over both gradient noise and mask randomness:

$$\mathbb{E}\Big[\big\langle \nabla \mathcal{C}^{(t)},\, \delta^{(t)} \odot H_t m_t \big\rangle \mid \mathcal{F}_t\Big] = \Big\langle \nabla \mathcal{C}^{(t)},\, \mathbb{E}[\delta^{(t)} \mid \mathcal{F}_t] \odot H_t m_t \Big\rangle.$$

Since each coordinate satisfies $\mathbb{E}[(\delta^{(t)})_k \mid \mathcal{F}_t] \ge p_{\min}$,

$$\mathbb{E}[\delta^{(t)} \mid \mathcal{F}_t] \succeq p_{\min} I,$$

and thus

$$\mathbb{E}\Big[\big\langle \nabla \mathcal{C}^{(t)},\, \delta^{(t)} \odot H_t m_t \big\rangle\Big] \ge p_{\min}\, \mathbb{E}\Big[\big\langle \nabla \mathcal{C}^{(t)},\, H_t m_t \big\rangle\Big].$$

Invoking the AMSGrad Descent Lemma. Non-convex AMSGrad analysis (e.g. (Reddi et al., 2019)) guarantees constants $c > 0$ and $B < \infty$ such that

$$\sum_{t=0}^{T-1} \eta_t \, \mathbb{E}\Big[\big\langle \nabla \mathcal{C}^{(t)},\, H_t m_t \big\rangle\Big] \;\geq\; c \sum_{t=0}^{T-1} \eta_t \, \mathbb{E}\|\nabla \mathcal{C}^{(t)}\|^2 - B.$$

Combining with the bound above:

$$\sum_{t=0}^{T-1} \eta_t \, \mathbb{E}\Big[\big\langle \nabla \mathcal{C}^{(t)},\, \delta^{(t)} \odot H_t m_t \big\rangle\Big] \;\geq\; p_{\min} \left(c \sum_{t=0}^{T-1} \eta_t \, \mathbb{E}\|\nabla \mathcal{C}^{(t)}\|^2 - B \right).$$

Final Convergence Argument. Taking expectations in (7), summing from $t = 0$ to $T - 1$, and using the bounds above yields:

$$p_{\min} c \sum_{t=0}^{T-1} \eta_t \, \mathbb{E}\|\nabla \mathcal{C}^{(t)}\|^2 \;\leq\; \mathcal{C}^{(0)} - \mathcal{C}^* + p_{\min} B + \frac{LK}{2} \sum_{t=0}^{\infty} \eta_t^2 \;=\; M < \infty.$$

Because $\sum_t \eta_t = \infty$, finiteness of the weighted sum implies

$$\liminf_{t \to \infty} \mathbb{E}\|\nabla \mathcal{C}^{(t)}\|^2 = 0,$$

i.e. the WSBD-AMSGrad iterates converge to a stationary point in expectation. □

A.3 General Convergence Framework for WSBD

Having established convergence for SGD and AMSGrad, we now extend the theoretical guarantees to a broad class of optimizers. The key observation is that WSBD acts as a stochastic coordinate mask that scales the expected descent direction of the base optimizer without amplifying its variance. Under mild and standard assumptions on the base optimizer $\mathcal{O}$, this suffices to preserve convergence.

Generalized Update Rule. Let $\mathcal{O}$ be any optimizer that, at step t, generates a proposed update vector $u_t \in \mathbb{R}^{|\theta|}$ based on the gradient $\nabla \mathcal{C}^{(t)}$ and internal states (e.g., momentum buffers). The standard update would be $\theta^{(t+1)} = \theta^{(t)} - \eta_t u_t$. The **WSBD-augmented update** applies the freezing mask $\delta^{(t)} \in \{0, 1\}^{|\theta|}$ to this vector:

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t (\delta^{(t)} \odot u_t) \tag{8}$$

Assumptions on Optimizer $\mathcal{O}$. Let $\mathcal{F}_t$ denote the filtration generated by all randomness up to step t (gradients, mini-batches, internal states). We assume:

1. **L-Smoothness:** The objective function $\mathcal{C}$ is L-smooth and bounded below by $\mathcal{C}^*$.

2. **Descent Condition:** The base optimizer produces a valid descent direction in expectation. Specifically, there exists a constant $c_1 > 0$ such that:

$$\mathbb{E}[\langle \nabla \mathcal{C}^{(t)}, u_t \rangle \mid \mathcal{F}_t] \geq c_1 \|\nabla \mathcal{C}^{(t)}\|^2 \tag{9}$$

3. **Bounded Update Magnitude:** The magnitude of the update vector is bounded. There exists $K > 0$ such that:

$$\mathbb{E}[\|u_t\|^2 \mid \mathcal{F}_t] \leq K \tag{10}$$

4. **Mask Independence and Minimum Probability:** The mask draw satisfies $\mathbb{E}[(\delta^{(t)})_k] = p_k^{(t)} \geq p_{\min} > 0$. Furthermore, the random draw of the current mask $\delta^{(t)}$ is conditionally independent of the stochastic noise in u_t given the history $\mathcal{F}_t$.

Proof. We start with the L-smoothness inequality:

$$\mathcal{C}^{(t+1)} \leq \mathcal{C}^{(t)} - \eta_t \langle \nabla \mathcal{C}^{(t)}, \delta^{(t)} \odot u_t \rangle + \frac{L\eta_t^2}{2} \|\delta^{(t)} \odot u_t\|^2$$

1. Bounding the Descent Term: We analyze the expected inner product using the Law of Total Expectation. Due to the conditional independence of the mask draw $\delta^{(t)}$ and the update u_t:

$$\begin{aligned}\mathbb{E}[\langle \nabla \mathcal{C}^{(t)}, \delta^{(t)} \odot u_t \rangle] &= \mathbb{E}\left[\sum_k (\nabla \mathcal{C}^{(t)})_k (u_t)_k \mathbb{E}[\delta_k^{(t)} \mid \mathcal{F}_t]\right] \\ &\geq p_{\min} \mathbb{E}[\langle \nabla \mathcal{C}^{(t)}, u_t \rangle]\end{aligned}$$

Applying the Descent Condition (Assumption 2):

$$\mathbb{E}[\langle \nabla \mathcal{C}^{(t)}, \delta^{(t)} \odot u_t \rangle] \geq p_{\min} c_1 \mathbb{E}[\|\nabla \mathcal{C}^{(t)}\|^2] \tag{11}$$

2. Bounding the Second-Order Term: Since $\delta_k^{(t)} \in \{0, 1\}$, element-wise multiplication by the mask is a contraction in the Euclidean norm ($\|\delta^{(t)} \odot u_t\|^2 \leq \|u_t\|^2$). Using Assumption 3:

$$\mathbb{E}[\|\delta^{(t)} \odot u_t\|^2] \leq \mathbb{E}[\|u_t\|^2] \leq K \tag{12}$$

3. Final Convergence Argument: Substituting these into the smoothness bound and taking full expectations:

$$\mathbb{E}[\mathcal{C}^{(t+1)}] \leq \mathbb{E}[\mathcal{C}^{(t)}] - \eta_t p_{\min} c_1 \mathbb{E}[\|\nabla \mathcal{C}^{(t)}\|^2] + \frac{L\eta_t^2}{2} K$$

Summing from $t = 0$ to $T - 1$:

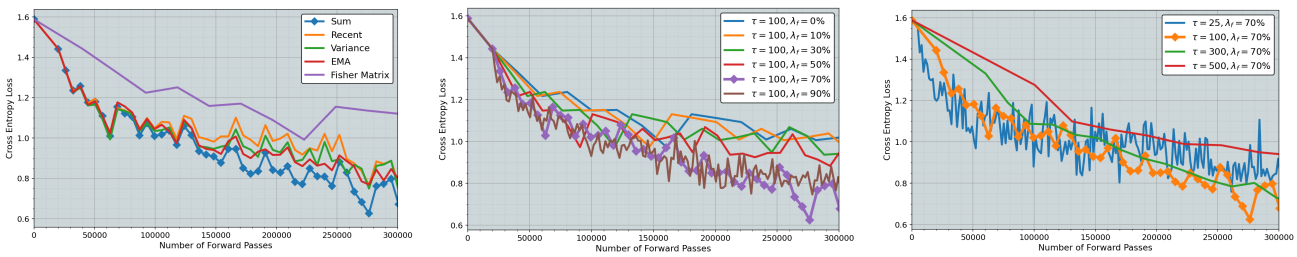
$$p_{\min} c_1 \sum_{t=0}^{T-1} \eta_t \mathbb{E}[\|\nabla \mathcal{C}^{(t)}\|^2] \leq \mathcal{C}^{(0)} - \mathcal{C}^* + \frac{LK}{2} \sum_{t=0}^{T-1} \eta_t^2$$

Given standard step-size conditions ($\sum \eta_t = \infty, \sum \eta_t^2 < \infty$), taking the limit as $T \to \infty$ implies:

$$\liminf_{t \to \infty} \mathbb{E}[\|\nabla \mathcal{C}^{(t)}\|^2] = 0$$

Thus, WSBD-$\mathcal{O}$ converges to a stationary point, provided the base optimizer $\mathcal{O}$ satisfies the descent and boundedness conditions. □

B IMPORTANCE SCORE METRICS AND HYPERPARAMETER TUNING



(a) Importance Scores (b) Freeze Thresholds (λ_f) (c) Window Sizes (τ)

Figure 3: Hyperparameter tuning for WSBD. Each sub figure shows how the choice in importance score, freezing threshold and training window affects the optimization process. This was done for a 10 qubit, 5 layer QNN on the MNIST problem.

To identify the most effective metric for our purposes, we investigated several candidates:

Sum of Gradients (Sum): This metric accumulates the absolute sum of gradients over the window. A large value suggests a consistent and strong impact on the cost function.

$$\mathcal{I}_p(\theta_k) = \Big| \sum_{t=1}^{\tau} \frac{\partial \mathcal{C}(\boldsymbol{\theta}^{(t)}, \hat{x_t})}{\partial \theta_k} \Big| + \epsilon \tag{13}$$

Most Recent Gradient (Recent): This metric uses only the final gradient (most recent influence) in the window.

$$\mathcal{I}_p(\theta_k) = \Big| \frac{\partial \mathcal{C}(\boldsymbol{\theta}^{(\tau)}, \hat{x_\tau})}{\partial \theta_k} \Big| + \epsilon \tag{14}$$

Gradient Variance (Variance): This metric gauges the consistency of a parameter's gradient throughout the training window. A low variance suggests a stable, though not necessarily large, gradient. It is computed using Welford's online algorithm Welford (1962), an efficient single-pass method.

$$\mathcal{I}_p(\theta_k) = \Big| \mathrm{Var}_{t\in[1,\tau]} \Big(\frac{\partial \mathcal{C}(\boldsymbol{\theta}^{(t)}, \hat{x_t})}{\partial \theta_k} \Big) \Big| + \epsilon \tag{15}$$

Exponential Moving Average (EMA): This provides a smoothed average of gradients, giving more weight to recent updates. It requires an extra hyperparameter, the decay rate β. First, we define the exponential moving average of the gradient, $m_k^{(t)}$ or a specific parameter θ_k at training step t:

$$\mathcal{I}_p(\theta_k)^{(t)} = \beta \mathcal{I}_p(\theta_k)^{(t-1)} + (1-\beta) \frac{\partial \mathcal{C}(\boldsymbol{\theta}^{(t)}, \hat{x_t})}{\partial \theta_k} \tag{16}$$

The final importance score is then given by:

$$\mathcal{I}_p(\theta_k)^{(t)} = |\mathcal{I}_p(\theta_k)^{(t)}| + \epsilon \tag{17}$$

Diagonal Fisher (Fisher Matrix): This is approximated by the average of the squared gradients and is often used as a proxy for the second-order derivative information.

$$\mathcal{I}_p(\theta_k) = \frac{1}{\tau} \sum_{t=1}^{\tau} \Big(\frac{\partial \mathcal{C}(\boldsymbol{\theta}^{(t)}, \hat{x_t})}{\partial \theta_k} \Big)^2 + \epsilon \tag{18}$$

Note that a small constant ϵ (e.g., 10^{-8}) is added to each score ensuring $\forall \theta_k \in \boldsymbol{\theta}, \mathcal{I}_p(\theta_k) > 0$ which will be crucial for proving convergence for WSBD (Sec. A). To select the optimal importance score for WSBD, we compared the five metrics. Each metric was used to train an identical 10-qubit, 5-layer QNN on the MNIST problem, with the performance comparison shown in Fig. 3a. We make multiple observations.

- The **Fisher Matrix** metric, while theoretically powerful, showed poor performance as the diagonal elements is likely not enough second order information for the optimization process and parameters influence each other.
- The **Gradient Variance** metric was found to be ill-suited for QNNs due to the barren plateau problem. A parameter with a consistently near-zero gradient would have a low variance, but this is a sign of it being stuck on a plateau, not a reliable indicator of its true importance.
- The **Exponential Moving Average (EMA)** performed similarly to the Sum of Gradients but was less effective and required tuning an additional hyperparameter, adding unnecessary complexity.
- The **Sum of Gradients** metric provides a straightforward and highly effective measure of a parameter's impact. It correctly identifies parameters with strong, consistent gradients as influential while effectively flagging those on barren plateaus as unimportant.

This makes it the superior choice for navigating the challenges of QNN training. Future work could involve exploring other importance metrics, such as novel hardware-aware metrics that incorporate factors like device-specific error rates to make the optimizer more robust in noisy settings.

B.1 Hyperparameter Tuning and its Effect on the Optimization Landscape

The WSBD optimizer has three primary hyperparameters: the importance score metric ($\mathcal{I}_p$), the training window size (τ), and the freeze percentile (λ_f). While our proof of convergence (Appendix A.1) guarantees that the optimizer will converge for any valid setting of these parameters (i.e., $\mathcal{I}_p(\theta)_k > 0 \; \forall \theta_k, \; \tau > 0, \lambda_f < 100\%$), their values directly influence the training dynamics and overall efficiency. This section explains the practical trade-offs involved with tuning each hyperparameter. We tune WSBD using grid search on two parameters, the freezing threshold λ_f and the time window size τ, using the MNIST task on a 10-qubit 5-layer model. First, we fix $\tau = 100$ and vary λ_f across $\{0\%, 10\%, 30\%, 50\%, 70\%, 90\%\}$. Then we fix $\lambda_f = 70\%$ and vary τ across $\{25, 100, 300, 500\}$. The best performance is obtained with $\lambda_f = 70\%$ and $\tau = 100$. This supports the hypothesis that freezing a large fraction of less important parameters accelerates convergence and that shorter time windows help the optimizer adapt more effectively. We notice that these hyperparameter values are generally stable with good performance improvements for τ being between 100-500 and λ_f being between 30-90%.

B.1.1 Training Window Size (τ)

The training window, τ, dictates the number of training steps for which the active set of parameters remains fixed and being trained. This hyperparameter controls the balance between focused optimization and algorithmic adaptivity.

- A **high τ value** means the active set is trained for a longer duration. This allows the optimizer to thoroughly explore the subspace defined by the active parameters. However, if a suboptimal set is chosen, it can lead to stagnation before the algorithm has a chance to re-evaluate and select a more influential set of parameters. Additionally the importance of a parameter might stagnate before the window is complete.
- A **low τ value** results in frequent re-shuffling of the active and frozen sets. This enhances the algorithm's adaptivity, allowing it to quickly pivot its focus. However, excessively frequent updates may prevent the optimizer from training any single parameter long enough to accurately gauge its true influence, and the overhead of constantly re-calculating importance scores can diminish computational gains.

Our experiments found that values between **100-300** were optimal, providing a sweet spot that balances deep training of an active set with the flexibility to adapt to the changing optimization landscape.

B.1.2 Freeze Percentile (λ_f)

The freeze percentile, λ_f, determines the aggressiveness of the resource allocation strategy by setting the proportion of parameters to be frozen.

- A **high λ_f value** results in a smaller active set. This offers the greatest computational savings, as fewer forward passes are required for gradient estimation at each step. However, an overly aggressive percentile risks freezing parameters that may become important later in training, potentially hindering the optimizer's ability to explore the full parameter space.
- A **low λ_f value** makes the optimizer behave more like a standard gradient-based method. While this ensures all parameters are explored more frequently, it diminishes the primary benefit of WSBD, as the number of forward passes remains high.

Our empirical results showed that a fairly aggressive freeze percentile of **70% performed best**. This suggests that at any given stage of training, a relatively small subset of parameters is responsible for the most significant progress, validating WSBD's core hypothesis. Additionally as WSBD enables exploration through its reset mechanism and stochastic selection the high freezing percentile is well justified.S

C REPRODUCIBILITY

To ensure full reproducibility and encourage adoption and extension by the Quantum Machine Learning (QML) community, all source code, datasets, and analysis scripts used to generate the results presented in this study will be made publicly available on GitHub upon publication.

C.1 Software Environment and Dependencies

Our implementation is built upon Python (v3.12.5) and leverages several standard, open-source libraries for scientific computing and machine learning.

- **Core QML Framework:** The Quantum Neural Networks (QNNs) were constructed, simulated, and trained using the `PennyLane` library (v 0.40.0) and training using non-gradient optimizers used `scikit_optimize` (v 0.10.2).
- **Data Handling and Processing:** We utilized `NumPy` (v 2.3.2) for numerical operations and `Scikit-learn` (v 1.7.1) for sourcing and preprocessing the MNIST dataset. `Pandas` (v 2.3.1) was used for organizing experimental data.
- **Visualization:** All plots and figures were generated with `Matplotlib` (v 3.9.2). The training progress was monitored using `tqdm` (v 4.66.5).

A complete `requirements.txt` file will be included in the root of our public repository to allow for one-step replication of the software environment.

Hardware Configuration and Quantum Computer Used

The experiments were executed on a high-performance computing server equipped with dual Intel(R) Xeon(R) Gold 6258R CPUs, 1.5 TB of RAM, and eight NVIDIA A100 (40GB) GPUs.

While this information is provided for completeness, our primary performance metric—the number of forward passes required for convergence—is a **hardware-independent** measure of algorithmic efficiency. The conclusions drawn from our results are therefore transferable across different computing infrastructures.

For the quantum hardware wall-clock experiment we utilized the IBM quantum platform `ibm_kingston` quantum computer which has the `Heron r2` quantum processing unit. We ran 100 random circuits for the different QNN sizes with $\mathcal{M} = 1000$ shots. We used these error rates for the VQE training problem.

C.2 Experimental Protocol and Control of Randomness

Dataset Integrity: To ensure a fair and rigorous comparison between all optimizers, we implemented strict controls over stochastic elements. The training and testing datasets for both the MNIST and Parity problems are static and will be provided in our repository. For every experiment, all optimizers were fed data points in the exact same sequence, eliminating any performance variations due to random data shuffling.

Parameter Initialization: For each QNN architecture, the initial variational parameters were randomly generated once and then saved. This fixed set of starting parameters was used for all optimizers being tested on that architecture, guaranteeing an identical starting point for every experimental run.

Stochasticity in WSBD: The only intentionally stochastic component in our experiments was the parameter selection mechanism within the WSBD, SBD, and L-WSBD optimizers, which is a core feature of their design. We found that despite this stochasticity, the performance and convergence behavior of WSBD remained highly stable and consistent across multiple independent runs ($\sim$5), validating the robustness of our approach.

Access and Usage

The codebase is structured to enable straightforward replication of our findings. The code is organized into two primary directories corresponding to the experimental problems: one for MNIST and one for Parity. Within each directory, we provide a separate Python script for each optimizer (e.g., `WSBD-SGD.py`, `SGD.py`, `DBD.py`, etc.). To reproduce a specific result, the QNN architecture (the number of qubits and layers) can be configured directly within the relevant script. The experiment is then launched by executing that specific file. For instance, to run the 10-qubit, 5-layer Parity problem with WSBD-SGD, a user would modify the parameters inside the `WSBD-SGD.py` file in the `/XOR/` folder and then run the script. Detailed instructions for setup and execution will be available in the `README.md` file in the project's root directory.

Table 3: Forward passes to reach max accuracy and target energies for the MNIST, parity and noisy VQE problems. $\mathcal{M}$ represents the number of circuit evaluations each forward pass is ran. For the VQE problem $\mathcal{M} = 1000$. In all problems and QNN sizes, WSBD shows performance improvements reaching its target faster. For the parity problem 0 represents an optimizer not solving or improving on the task.

QNN Size	Max Accuracy (SGD/Adam WSBD-SGD/WSBD-Adam)	Standard Optimizers		WSBD Optimizers	
		SGD	ADAM	WSBD-SGD	WSBD-ADAM
MNIST Classification					
4q, 2l	(58.1, 70.2 59.9, 70.4)	39,600$\mathcal{M}$	41,250$\mathcal{M}$	**7,800**$\mathcal{M}$	**22,200**$\mathcal{M}$
8q, 3l	(49.6, 68.5 49.9, 68.7)	223,100$\mathcal{M}$	126,100$\mathcal{M}$	**137,300**$\mathcal{M}$	**88,000**$\mathcal{M}$
10q, 5l	(56.6, 73.7 57.5, 74.2)	261,300$\mathcal{M}$	120,600$\mathcal{M}$	**184,800**$\mathcal{M}$	**99,400**$\mathcal{M}$
Parity Problem					
4q, 2l	(50.0, 100.0 100.0, 100.0)	0	3,960$\mathcal{M}$	**51,720**$\mathcal{M}$	**3,660**$\mathcal{M}$
8q, 3l	(100.0, 100.0 100.0, 100.0)	318,160$\mathcal{M}$	13,580$\mathcal{M}$	**162,820**$\mathcal{M}$	**13,180**$\mathcal{M}$
10q, 5l	(50.0, 100.0 50.0, 100.0)	0	88,440$\mathcal{M}$	0	**70,120**$\mathcal{M}$
Ground State Energy Problem					
	Target Energy (SGD/Adam)				
1q, 2l	-0.85990 / -0.88768	1,072$\mathcal{M}$	624$\mathcal{M}$	**794**$\mathcal{M}$	**350**$\mathcal{M}$
1q, 4l	-0.90606 / -0.94134	1,280$\mathcal{M}$	5,456$\mathcal{M}$	**1,016**$\mathcal{M}$	**920**$\mathcal{M}$
1q, 8l	-0.71530 / -0.90632	2,080$\mathcal{M}$	9,888$\mathcal{M}$	**1,632**$\mathcal{M}$	**3,064**$\mathcal{M}$
2q, 1l	-1.48394 / -1.47540	2,416$\mathcal{M}$	976$\mathcal{M}$	**1,738**$\mathcal{M}$	**436**$\mathcal{M}$
2q, 2l	-2.00472 / -2.03302	3,664$\mathcal{M}$	4,464$\mathcal{M}$	**1,532**$\mathcal{M}$	**768**$\mathcal{M}$
2q, 4l	-1.81826 / -1.90246	6,848$\mathcal{M}$	4,896$\mathcal{M}$	**4,112**$\mathcal{M}$	**1,688**$\mathcal{M}$
4q, 1l	-3.88842 / -3.90748	7,216$\mathcal{M}$	3,280$\mathcal{M}$	**4,760**$\mathcal{M}$	**1,316**$\mathcal{M}$
4q, 2l	-4.12360 / -4.09478	11,936$\mathcal{M}$	4,384$\mathcal{M}$	**9,152**$\mathcal{M}$	**2,664**$\mathcal{M}$
4q, 3l	-3.68399 / -3.75891	13,440$\mathcal{M}$	8,496 $\mathcal{M}$	**7,200**$\mathcal{M}$	**1,994**$\mathcal{M}$

D ADDITIONAL RESULTS

This section includes detailed training plots and tables related to the comparison of optimizers.

QNN Size	SGD	ADAM	WSBD SGD	WSBD ADAM
1q 2l	0.01389 (±1.39%)	0.01163 (±1.16%)	0.01945 (±1.95%)	0.01696 (±1.70%)
1q 4l	0.01154 (±1.15%)	0.01012 (±1.01%)	0.01465 (±1.46%)	0.01314 (±1.31%)
1q 8l	0.00956 (±0.96%)	0.01325 (±1.32%)	0.02179 (±2.18%)	0.02156 (±2.16%)
2q 1l	0.03574 (±1.60%)	0.03630 (±1.62%)	0.03796 (±1.70%)	0.03945 (±1.76%)
2q 2l	0.03533 (±1.58%)	0.07152 (±3.20%)	0.05569 (±2.49%)	0.04123 (±1.84%)
2q 4l	0.03664 (±1.64%)	0.03352 (±1.50%)	0.04100 (±1.83%)	0.05009 (±2.24%)
4q 1l	0.03172 (±0.67%)	0.08518 (±1.79%)	0.09618 (±2.02%)	0.10447 (±2.20%)
4q 2l	0.02535 (±0.53%)	0.05149 (±1.08%)	0.05418 (±1.14%)	0.07743 (±1.63%)
4q 3l	0.02587 (±0.54%)	0.04386 (±0.92%)	0.05690 (±1.20%)	0.09522 (±2.00%)

Table 4: Standard Deviation (STD) and Normalized STD (NSTD = $\frac{\text{STD}}{\text{Ground State Energy}}$) for SGD/ADAM and WSBD variants on the noisy VQE problem. As seen the weighted-stochastic freezing element of WSBD still remains highly stable, with a Normalized STD around 1-2% typically.

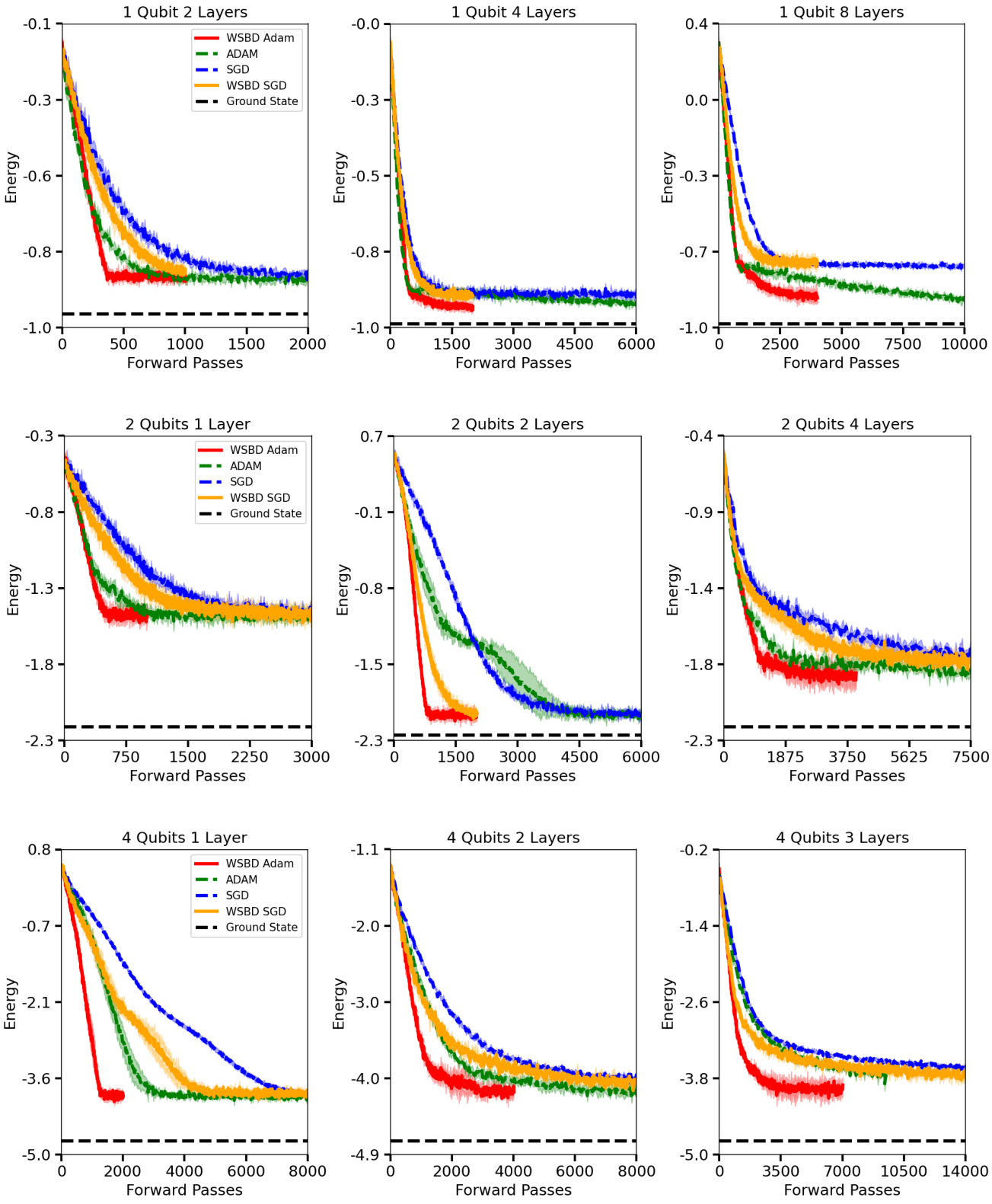
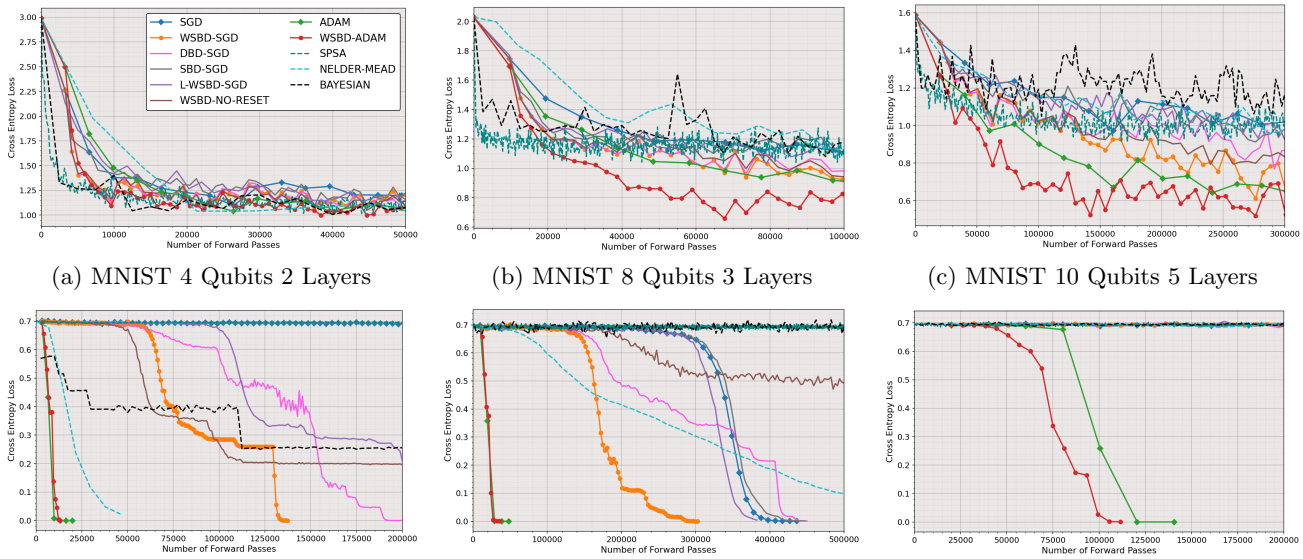


Figure 4: Training curves for the ground state energy problem using Adam, SGD and their WSBD counterpart optimizers. The black dotted line represents the ground state energy each optimizer aims to reach (closer is better). The models were trained in realistic noisy environments until convergence was reached.



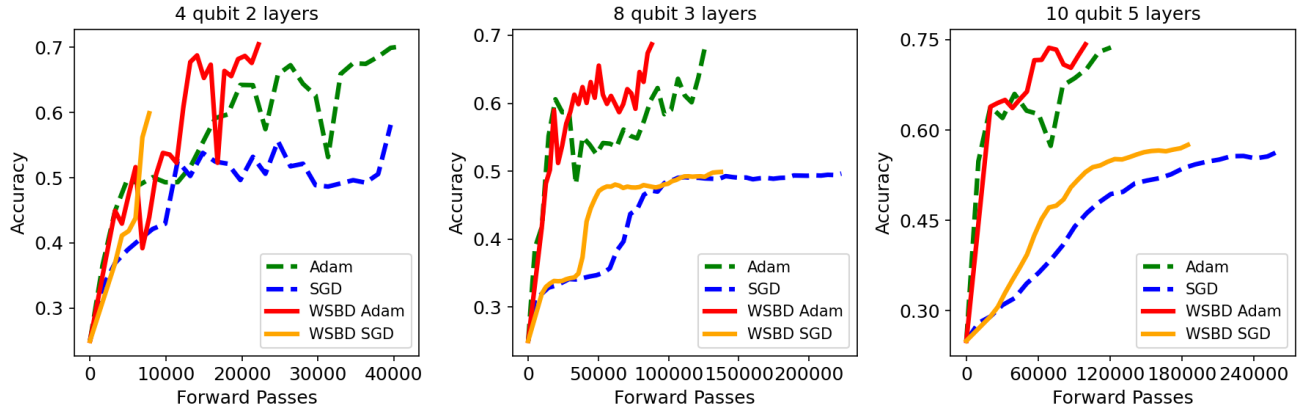
(a) MNIST 4 Qubits 2 Layers (b) MNIST 8 Qubits 3 Layers (c) MNIST 10 Qubits 5 Layers

(d) Parity 4 Qubits 2 Layers (e) Parity 8 Qubits 3 Layers (f) Parity 10 Qubits 5 Layers

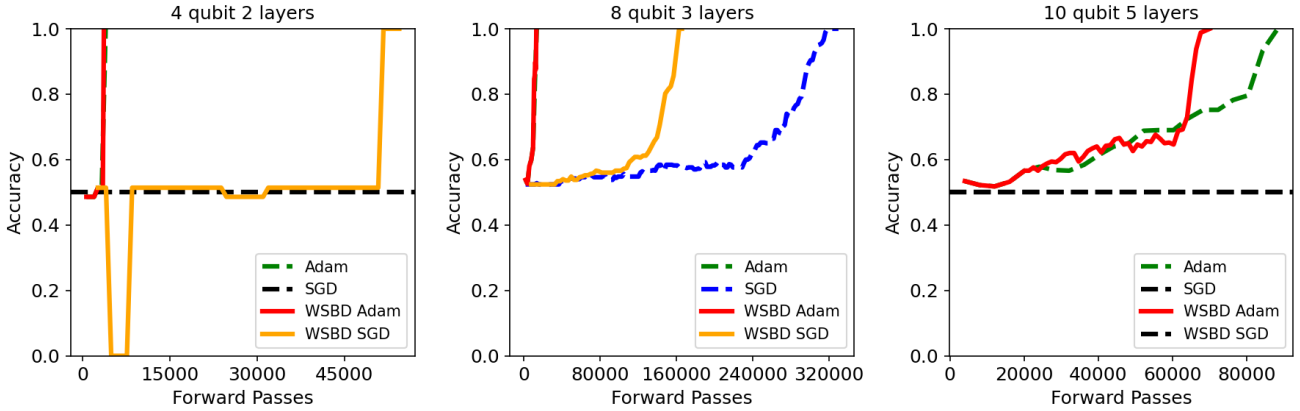
Figure 5: Comparisons of QNN training on optimizers. Top row: MNIST classification problem. Bottom row: Parity problem. We compare all optimizers summarized in Table 1 on these two tasks. WSBD shows clear performance improvements having both a faster decrease in loss and often reaching a lower loss overall for many models.

QNN Size	WSBD SGD (STD, CV)	WSBD ADAM (STD, CV)
4q 2l	0.0228 $(\pm 3.25 \cdot 10^{-4}\%)$	0.0146 $(\pm 2.08 \cdot 10^{-4})\%$
8q 3l	0.0028 $(\pm 7.01 \cdot 10^{-6})\%$	0.0132 $(\pm 3.35 \cdot 10^{-5})\%$
10q 5l	0.0044 $(\pm 8.82 \cdot 10^{-6})\%$	0.0076 $(\pm 2.04 \cdot 10^{-5})\%$

Table 5: Standard Deviation (STD) and Coefficient of Variation (CV) for WSBD SGD and WSBD ADAM on the MNIST problem. As seen the weighted-stochastic freezing element of WSBD still remains highly stable, with a CV close to 0. Note: Both the parameters and data order was fixed for all runs. Therefore, this experiment tests the stability of the weighted-stochastic freezing aspect directly.



(a) MNIST accuracy training curves with the SGD, Adam and WSBD optimizers. We show the training until the highest accuracy is reached using each optimizer.



(b) Parity problem accuracy training curves with the SGD, Adam and WSBD optimizers. The black dotted line represents when a given optimizer wasn't able to solve or improve on the problem.

Figure 6: Training curves showing how accuracy increases for a 5000 image and 500 bit-string testing set using Adam, SGD and the WSBD optimizers. WSBD reaches its max accuracy faster in all curves and often reaches a higher accuracy overall.