TITAN: A Trajectory-Informed Technique for Adaptive Parameter Freezing in Large-Scale VQE

Yifeng Peng[1,†], Xinyi Li[1], Samuel Yen-Chi Chen[2], Kaining Zhang[3], Zhiding Liang[4], Ying Wang[1,*], Yuxuan Du[3,‡]

[1] Stevens Institute of Technology [2] Wells Fargo [3] Nanyang Technological University
[4] Rensselaer Polytechnic Institute

ypeng21@stevens.edu, xli215@stevens.edu, ycchen1989@ieee.org,
kaining.zhang@ntu.edu.sg, zlianghahaha@gmail.com, ywang6@stevens.edu,
duyuxuan123@gmail.com

Appendix

A Related Works

A.1 (i) Measurement–Cost Mitigation

Commuting–set and graph–based grouping. Early works exploited qubit–wise commuting (QWC) partitions, typically found via minimum–clique–cover heuristics on Pauli–term graphs. While grouping can slash shot counts, the NP–hard clique search still scales poorly and demands one distinct circuit per group. Recent "overlapped–grouping" frameworks generalise QWC by allowing *shared* observables between groups; the resulting convex allocation of shots outperforms prior heuristics across 16-qubit molecular Hamiltonians[1].

Classical–shadow variants. Classical shadows randomise measurement bases and reconstruct many expectation values simultaneously, but uniform sampling wastes shots on insignificant terms. Derandomised and importance–weighted shadows address this by pre-computing tailored bases, at the cost of *classical* post-processing that scales quadratically in the number of measured terms[2, 3]. The very recent ROGS algorithm blends overlapped grouping with shadow tomography and optimally allocates a fixed shot budget via a confidence-bound objective, lowering *both* shot *and* unique-circuit counts on benchmarks up to 20 qubits[4].

Learning-based grouping. Graph neural networks and reinforcement learning have been used to predict near-optimal partitions directly from Pauli–Hamiltonian structure[5]. These methods amortise the combinatorial search after training, but require sizeable data sets of Hamiltonians and incur non-negligible offline costs.

Measurement strategies trade quantum shots for classical runtime; none alter the *optimization* landscape itself, leaving barren-plateau issues untouched.

A.2 (ii) Ansatz Design and Compression

Hardware-efficient vs. problem-inspired circuits. Hardware-efficient ansätze (HEA) minimise depth on specific devices but often suffer from barren plateaus (BPs)[6]. In contrast, chemically or symmetry inspired ansätze (e.g., UCC or symmetry-preserving layouts) improve trainability yet dramatically increase two-qubit-gate count[7].

Adaptive and search-based construction. ADAPT-VQE builds circuits iteratively by adding operators with the largest energy gradient, offering compact, system-specific wavefunctions but

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

requiring hundreds of rounds of gradient evaluations [8]. Architecture-search frameworks, both evolutionary[9] and Bayesian[10], scan supercircuit spaces to identify expressive, shallow layouts, yet introduce an outer optimization loop whose cost grows combinatorially with qubit number.

Pruning and parameter sparsification. Once trained, many ansätze contain redundant gates. Gradient-based pruning[11] and Hessian-guided methods such as QADAPRUNE[12] delete low-saliency parameters on the fly, reducing depth without re-training. The *Pruned-ADAPT-VQE* refinement removes near-zero-amplitude operators from adaptive circuits, shrinking molecular ansätze by up to 30 %[13].

Circuit-level improvements fight either expressibility or depth; they do not reduce the *dimension* of the parameter space fed to the classical optimizer.

A.3 (iii) Advanced Classical Optimizers

Quantum-aware gradients. Natural-gradient and quantum–Fisher techniques exploit the information-geometry of parameterised unitaries but double (or worse) the number of hardware evaluations per step. QBANG approximates the Fisher matrix via a Broyden update and mixes it with momentum, matching natural-gradient accuracy with gradient-descent cost[14].

Meta-learning and hyper-opt. Meta-optimizers learn update rules across VQE tasks[15] and can warm-start large-scale instances, yet inject their own nested objective and hyperparameter. RL-based schedulers dynamically switch between gradient-free and gradient-based steps, but still operate in the *full* parameter space.

Noise-aware and Bayesian schemes. Techniques that re-weight gradients by empirical shot noise or adopt Bayesian optimization have improved robustness on real hardware, though at the price of surrogate-model maintenance and additional variance estimates[16].

Take-away. Optimizer innovations reshape the search trajectory yet remain bottlenecked by high-dimensional parameter landscapes.

A.4 (iv) Parameter-Space Reduction

A complementary axis—*shrinking the search space itself*. Early works froze parameters whose gradients fall below a threshold, but fixed thresholds risk misclassifying temporarily stagnant yet ultimately useful parameters [17]. TITAN extends this idea by predicting freezing *before training* via a CFCSA encoder and APFA labels, reducing trainable parameters and measurement shots simultaneously. Unlike pruning (which acts *after* or *during* optimization), predictive freezing is *proactive*, feeding a leaner search space to *any* optimizer or measurement scheme.

Synergy with prior axes. Because freezing removes variables rather than adding gates or shots, TITAN layers orthogonally atop (i)–(iii): fewer parameters lower the classical cost of natural-gradient estimators, shorten architecture-search evaluations, and even tighten confidence bounds in shadow-based measurement allocation.

B APFA Implementations Details

The APFA mechanism dynamically identifies and *freezes* low-saliency parameters in a VQE to record the freezing trajectory as shown in Figure 1. For a VQE with parameter vector $\boldsymbol{\theta}^t \in \mathbb{R}^P$ at iteration t, we define the stochastic gradient

$$\mathbf{g}_t \;=\; \nabla_{\boldsymbol{\theta}}\, f\big(\boldsymbol{\theta}^t\big) \;+\; \boldsymbol{\xi}_t, \quad \boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \gamma^2 \mathbf{I}), \tag{1}$$

where $\boldsymbol{\xi}^t$ is an isotropic noise term (with scale γ) designed to improve exploration. APFA *dynamically* toggles each parameter θ_i $(i = 1, \ldots, P)$ between an *active* and a *frozen* state, according to the four steps below:

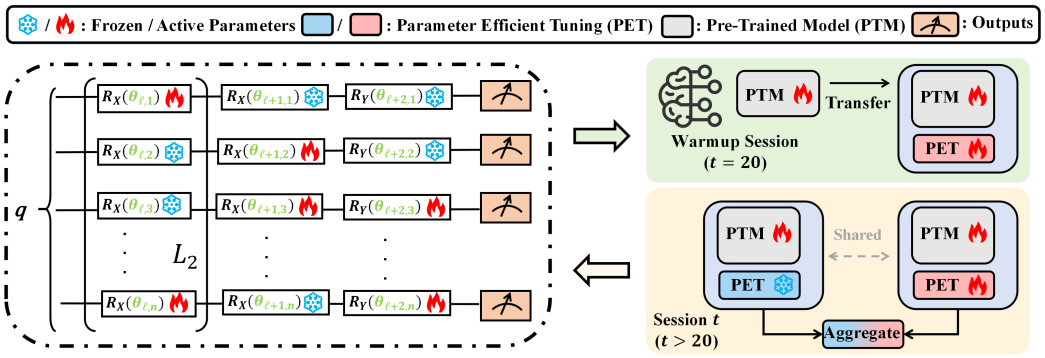


Figure 1: An illustration of the TITAN architecture employing APFA to freeze or reactivate parameters. The red flames mark active (trainable) parameters, while blue snowflakes indicate frozen coordinates.

Gradients and Exponential Moving Average (EMA) To quantify parameter *saliency*, an exponential moving average (EMA) of the absolute gradient is maintained for each coordinate $i \in \{1, \ldots, P\}$:

$$\widehat{g}_t^{(i)} \;=\; \alpha\, \widehat{g}_{t-1}^{(i)} \;+\; (1-\alpha)\, \big|g_t^{(i)}\big|, \qquad 0 < \alpha < 1, \tag{2}$$

with $\widehat{g}_0^{(i)} = |g_0^{(i)}|$ and $g_t^{(i)}$ the ith component of $\mathbf{g}_t$. This EMA captures recent trends in gradient magnitudes while smoothing out short-term fluctuations.

Global Magnitude and Dynamic Thresholding We track two global metrics at iteration t. First, the *mean* of the EMA of absolute gradients, $\bar{g}_t = \frac{1}{P}\sum_{i=1}^{P} \widehat{g}_t^{(i)}$, provides a representative measure of overall gradient magnitude. Second, we compute a *decay ratio* r_t defined by

$$r_t \;=\; \frac{\|\mathbf{g}_t\|_2}{\|\mathbf{g}_0\|_2 + \varepsilon}, \quad \varepsilon \approx 10^{-12}, \tag{3}$$

which compares the current global gradient norm to its initial value. This ratio serves as an adaptive control signal for scaling the *freeze* factor $\lambda_f^{(t)}$ and *activate* factor $\lambda_a^{(t)}$:

$$\lambda_f^{(t)} = \begin{cases} 3\,\lambda_f^{(0)}, & r_t < 0.2, \\ \lambda_f^{(0)}\big[1 + 2\,(1 - r_t)/0.8\big], & r_t \geq 0.2, \end{cases} \qquad \lambda_a^{(t)} = \begin{cases} 3\,\lambda_a^{(0)}, & r_t < 0.2, \\ \lambda_a^{(0)}\big[1 + 2\,(1 - r_t)/0.8\big], & r_t \geq 0.2. \end{cases} \tag{4}$$

Thus, if r_t becomes sufficiently small (i.e., the global gradient norm decays sharply), the factors $\lambda_f^{(t)}$ and $\lambda_a^{(t)}$ both increase, encouraging more aggressive freezing and reactivation thresholds.

The resulting *freeze* and *activate* thresholds are given by$\tau_f^{(t)} = \lambda_f^{(t)}\,\bar{g}_t, \tau_a^{(t)} = \lambda_a^{(t)}\,\tau_f^{(t)}$. Therefore, any coordinate whose EMA magnitude $\widehat{g}_t^{(i)}$ falls below $\tau_f^{(t)}$ repeatedly is considered low-saliency and may be *frozen*, while coordinates rising above $\tau_a^{(t)}$ sufficiently often are *reactivated*.

State Machine for Freezing and Activating Parameters To implement the above thresholds, we maintain two counters, $c_f^{(i)}$ and $c_a^{(i)}$, and a binary mask $m_t^{(i)} \in \{0, 1\}$ for each coordinate i. If $m_t^{(i)} = 0$, the parameter is *frozen*, whereas if $m_t^{(i)} = 1$, it is *active*. The counters record how many consecutive iterations $\widehat{g}_t^{(i)}$ has stayed below $\tau_f^{(t)}$ (for freezing) or exceeded $\tau_a^{(t)}$ (for activating). Formally, for user-defined integers N_f (freeze) and N_a (activate), the update rule is:

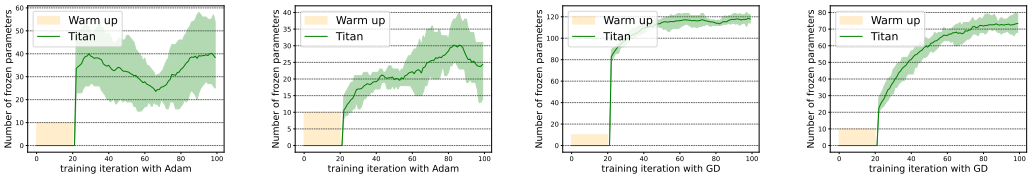
$$\begin{aligned} &\textbf{if } m_{t-1}^{(i)} = 1 \text{ (active):} && \begin{cases} c_f^{(i)} \leftarrow c_f^{(i)} + 1, & \text{if } \widehat{g}_t^{(i)} < \tau_f^{(t)}, \\ c_f^{(i)} \leftarrow 0, & \text{otherwise,} \end{cases} && \text{freeze if } c_f^{(i)} \geq N_f; \\ &\textbf{if } m_{t-1}^{(i)} = 0 \text{ (frozen):} && \begin{cases} c_a^{(i)} \leftarrow c_a^{(i)} + 1, & \text{if } \widehat{g}_t^{(i)} > \tau_a^{(t)}, \\ c_a^{(i)} \leftarrow 0, & \text{otherwise,} \end{cases} && \text{activate if } c_a^{(i)} \geq N_a. \end{aligned} \tag{5}$$

As soon as a counter $c_f^{(i)}$ or $c_a^{(i)}$ crosses its respective threshold N_f or N_a, we update $m_t^{(i)}$ to 0 or 1 (i.e., freeze or activate). Letting $\mathbf{m}_t$ denote the vector of masks at iteration t, a generic first-order optimizer subsequently performs the following masked parameter update:

$$\boldsymbol{\theta}_{t+1} \;=\; \boldsymbol{\theta}_t \;-\; \eta_t \left(\mathbf{m}_t \odot \mathbf{g}_t\right), \tag{6}$$

where η_t is the learning rate and $\odot$ denotes the Hadamard product. Only active parameters (i.e., coordinates with $m_t^{(i)} = 1$) are updated, while frozen ones remain fixed.

Convergence and Practical Considerations Under standard smoothness and bounded-below assumptions on $\mathcal{L}$ (i.e., $\mathcal{L}$ being L-smooth and coercive), with learning rate sequence $\{\eta_t\}$ satisfying $\sum_t \eta_t = \infty,\ \sum_t \eta_t^2 < \infty$, and $\lambda_f^{(t)} \to 0$ no faster than η_t, the analysis in [18] can be adapted to show $\liminf_{t\to\infty} \left\|\nabla\mathcal{L}\big(\boldsymbol{\theta}_t\big)\right\|_2 \;=\; 0$, i.e., APFA retains the standard stochastic gradient convergence guarantees. In practice, once parameters are frozen, the optimization is effectively running in a reduced-dimensional subspace, which can lead to faster progress on significant coordinates while still allowing reactivation when needed. Consequently, this *parameter-efficient* approach can offer both computational savings and improved generalization performance by reducing overfitting on low-saliency dimensions.



(a) (b) (c) (d)

Figure 2: Evolution of the number of frozen parameters under different optimizers and noise settings of the Heisenberg model (HEA, XX+YY+ZZ, Qubits: 10, Layers: 15) (a) Adam without noise. (b) Adam with noise. (c) Gradient Descent without noise. (d) Gradient Descent with noise.

Figure. 2 tracks the cumulative number of gates identified as redundant by APFA a 10–qubit, 15–layer HEA circuit for the XX + YY + ZZ Heisenberg Hamiltonian. The yellow band marks the *warm-up phase*($t \le T_{\text{warm}}$), whereas the green curve (with $\pm\sigma$ envelope, $n = 5$ seeds) shows the ensuing Titan-mediated freezing trajectory. Overall, Titan adapts its freezing rate to the optimizer–noise landscape: stochastic Adam benefits from *incremental* freezes that hedge against noisy gradients, whereas deterministic GD permits *aggressive* early pruning.

C Proof for Theorem 1

For convenience, we list two useful lemmas before the proof of Theorem 1.

lemma 1. *Consider the unitary U sampled from the unitary 2-design in $\mathcal{U}(2)$. Let $A, B \in \{I, X, Y, Z\}$. Then we have*

$$\mathop{\mathbb{E}}_{U}(U^\dagger A U) \otimes (U^\dagger B U) = \begin{cases} I \otimes I, & \text{when } A = B = I, \\ \dfrac{1}{3}\left(X \otimes X + Y \otimes Y + Z \otimes Z\right), & \text{when } A = B = X/Y/Z, \\ 0, & \text{for other cases.} \end{cases} \tag{7}$$

Proof. Lemma 1 can be obtained by using Eq. (53) in Ref. [19] for $A, B \in \{I, X, Y, Z\}$.

□

lemma 2. *Consider the function $f = \mathrm{Tr}[(O_1 \otimes O_2)(R \otimes A)B(R^\dagger \otimes A^\dagger)]$, where $R = \exp[-i\theta G/2]$ and the Hamiltonian G is either commuting or anti-commuting with O_1. Then*

$$\mathop{\mathbb{E}}_{\theta} f^2 = \begin{cases} \left(\mathrm{Tr}[(O_1 \otimes O_2)(I \otimes A)B(I \otimes A^\dagger)]\right)^2, & \text{when } GO_1 = O_1G, \\ \frac{1+\exp[-2\gamma^2]}{2}\left(\mathrm{Tr}\left[(O_1 \otimes O_2)(I \otimes A)B(I \otimes A^\dagger)\right]\right)^2 & \\ +\frac{1-\exp[-2\gamma^2]}{2}\left(\mathrm{Tr}\left[(iGO_1 \otimes O_2)(I \otimes A)B(I \otimes A^\dagger)\right]\right)^2, & \text{when } GO_1 = -O_1G, \end{cases}$$

$$\mathop{\mathbb{E}}_{\theta}\left(\frac{\partial f}{\partial \theta}\right)^2 = \begin{cases} 0, & \text{when } GO_1 = O_1G, \\ \frac{1-\exp[-2\gamma^2]}{2}\left(\mathrm{Tr}\left[(O_1 \otimes O_2)(I \otimes A)B(I \otimes A^\dagger)\right]\right)^2 & \\ +\frac{1+\exp[-2\gamma^2]}{2}\left(\mathrm{Tr}\left[(iGO_1 \otimes O_2)(I \otimes A)B(I \otimes A^\dagger)\right]\right)^2, & \text{when } GO_1 = -O_1G, \end{cases}$$

where θ is sampled from $\mathcal{N}(0, \gamma^2)$.

Proof. For the case of $GO_1 = O_1G$, the result yields trivially. We focus on the case of $GO_1 = -O_1G$, where the function f has the format

$$\begin{aligned} f &= \mathrm{Tr}\left[(O_1 \otimes O_2)(\exp[-i\theta G/2] \otimes I)(I \otimes A)B(I \otimes A^\dagger)(\exp[i\theta G/2] \otimes I)\right] \\ &= \mathrm{Tr}\left[(\exp[i\theta G] \otimes I)(O_1 \otimes O_2)(I \otimes A)B(I \otimes A^\dagger)\right] \\ &= \cos\theta \mathrm{Tr}\left[(O_1 \otimes O_2)(I \otimes A)B(I \otimes A^\dagger)\right] + \sin\theta \mathrm{Tr}\left[(iGO_1 \otimes O_2)(I \otimes A)B(I \otimes A^\dagger)\right]. \end{aligned}$$

By calculating the expectation of $\cos^2\theta$, $\sin^2\theta$, and $\sin\theta\cos\theta$, we obtain Lemma 2.

□

Theorem 3 (Enhanced Gaussian Initialization)**.** *Consider the VQC $V(\boldsymbol{\theta}) = (\otimes_{n=1}^N U_n)\prod_{i=1}^L \mathrm{CZ}_i\mathrm{RY}_i(\boldsymbol{\theta}^{2i-1})\mathrm{RX}_i(\boldsymbol{\theta}^{2i})$, where each U_n is sampled from unitary 2-design in $\mathcal{U}(2)$ and each parameter in $\boldsymbol{\theta}$ is sampled from $\mathcal{N}(0, \gamma^2)$ independently with $\gamma \leq 1$. Consider the S-local observable with the Pauli decomposition $O = \sum_{\boldsymbol{i}} \alpha_{\boldsymbol{i}} P_{\boldsymbol{i}}$. Let $f = \mathrm{Tr}[OV(\boldsymbol{\theta})|0\rangle\langle 0|V(\boldsymbol{\theta})^\dagger]$ be the loss function. Then for each partial derivative $\frac{\partial f}{\partial \theta^{m,n}}$, the expected squared gradient (with respect to both the random unitaries $U = \otimes_{n=1}^N U_n$ and the random parameters $\boldsymbol{\theta}$) admits the following lower bound:*

$$\mathop{\mathbb{E}}_{U,\boldsymbol{\theta}}\left(\frac{\partial f}{\partial \theta^{m,n}}\right)^2 \geq \frac{1}{3^S}\sum_{\boldsymbol{i}} \alpha_{\boldsymbol{i}}^2 (1 - \delta_{i_n 0})\left(1-\gamma^2\right)^{2LS}\frac{2}{3}\gamma^2, \tag{8}$$

where $S = \max_{\boldsymbol{i}}(\|\boldsymbol{i}\|_0)$. Moreover, choosing $\gamma^2 = c/L$ ensures that

$$\mathop{\mathbb{E}}_{U,\boldsymbol{\theta}} \|\nabla_{\boldsymbol{\theta}} f\|^2 \geq \mathcal{O}(1) \tag{9}$$

for the case of local observable $O = \sum_{\boldsymbol{i}} \alpha_{\boldsymbol{i}} P_{\boldsymbol{i}}$ with $S = \mathcal{O}(1)$ and $\alpha_{\boldsymbol{i}} = \mathcal{O}(1)$.

Proof. For convenience, we denote intermediate states

$$\rho_k := \left(\prod_{i=L+1-k}^{L} \mathrm{CZ}_i\mathrm{RY}_i(\boldsymbol{\theta}^{2i-1})\mathrm{RX}_i(\boldsymbol{\theta}^{2i})\right)\rho_0\left(\prod_{i=L+1-k}^{L} \mathrm{CZ}_i\mathrm{RY}_i(\boldsymbol{\theta}^{2i-1})\mathrm{RX}_i(\boldsymbol{\theta}^{2i})\right)^\dagger, \tag{10}$$

where $\rho_0 = |0\rangle\langle 0|$. Therefore, we can rewrite the loss function as

$$f(\boldsymbol{\theta}) = \mathrm{Tr}\left[OU\rho_L(\boldsymbol{\theta})U^\dagger\right], \tag{11}$$

where $U = \otimes_{n=1}^N U_n$ is the final single-qubit unitary layer. Based on the parameter-shift rule, the partial derivative has the format

$$\begin{aligned} \frac{\partial f}{\partial \theta^{m,n}} &= \frac{1}{2}\left[f(\boldsymbol{\theta}_+) - f(\boldsymbol{\theta}_-)\right] \\ &= \frac{1}{2}\left[\mathrm{Tr}\left[OU\rho_L(\boldsymbol{\theta}_+)U^\dagger\right] - \mathrm{Tr}\left[OU\rho_L(\boldsymbol{\theta}_-)U^\dagger\right]\right] \\ &:= \mathrm{Tr}\left[OU\rho_L' U^\dagger\right], \end{aligned} \tag{12}$$

where $\boldsymbol{\theta}_+$ and $\boldsymbol{\theta}_-$ differ from $\boldsymbol{\theta}$ by adding $\pi/2$ or $-\pi/2$ on the term $\theta^{m,n}$, and $\rho'_L := \frac{\rho_L(\boldsymbol{\theta}_+)-\rho_L(\boldsymbol{\theta}_-)}{2}$.

Next, we consider the expectation of the square of the partial derivative term under the distribution of U. By using the Pauli decomposition of the observable $O = \sum_{\boldsymbol{i}} \alpha_{\boldsymbol{i}} P_{\boldsymbol{i}}$, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{U}\left(\frac{\partial f}{\partial \theta^{m,n}}\right)^2 &= \mathop{\mathbb{E}}_{U}\left(\mathrm{Tr}\left[OU\rho'_L U^\dagger\right]\right)^2 \\
&= \sum_{\boldsymbol{i},\boldsymbol{j}} \alpha_{\boldsymbol{i}}\alpha_{\boldsymbol{j}} \mathop{\mathbb{E}}_{U} \mathrm{Tr}\left[P_{\boldsymbol{i}}U\rho'_L U^\dagger\right]\mathrm{Tr}\left[P_{\boldsymbol{j}}U\rho'_L U^\dagger\right] \qquad (13)\\
&= \sum_{\boldsymbol{i}} \alpha_{\boldsymbol{i}}^2 \frac{1}{3^{\|\boldsymbol{i}\|_0}} \sum_{\boldsymbol{j}\in \mathrm{Supp}(\boldsymbol{i})} \mathrm{Tr}\left[P_{\boldsymbol{j}}\rho'_L\right]^2 \qquad (14)\\
&\geq \frac{1}{3^S}\sum_{\boldsymbol{i}} \alpha_{\boldsymbol{i}}^2 \mathrm{Tr}\left[Z_{\boldsymbol{i}}\rho'_L\right]^2, \qquad (15)
\end{aligned}$$

where $\mathrm{Supp}(\boldsymbol{i})$ denotes the set

$$\{(k_1, k_2, \cdots k_N) | k_n \in \{1,2,3\} \text{ if } i_n \neq 0;\ k_n = 0 \text{ if } i_n = 0\},$$

and Eq. (14) is derived by using Lemma 1. Eq. (15) is derived since each Pauli basis in O with non-zero coefficient has at most S non-identity elements (S-local). The term $Z_{\boldsymbol{i}}$ denotes the observable consisted of Z and I, where Z is applied when the corresponding coefficient in $\boldsymbol{i}$ is not zero.

Next, we proceed to the analysis of the expectation of Eq. (15) with respect to $\boldsymbol{\theta}$. Let $j := \lfloor \frac{m+1}{2} \rfloor$. For the case of $j > 1$, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{\theta}^1,\boldsymbol{\theta}^2} \mathrm{Tr}\left[Z_{\boldsymbol{i}}\rho'_L\right]^2 &= \mathop{\mathbb{E}}_{\boldsymbol{\theta}^1,\boldsymbol{\theta}^2} \mathrm{Tr}\left[Z_{\boldsymbol{i}}\mathrm{CZ}_1\mathrm{RY}_1(\boldsymbol{\theta}^1)\mathrm{RX}_1(\boldsymbol{\theta}^2)\rho'_{L-1}\mathrm{RX}_1(\boldsymbol{\theta}^2)^\dagger\mathrm{RY}_1(\boldsymbol{\theta}^1)^\dagger\mathrm{CZ}_1^\dagger\right]^2 \\
&= \mathop{\mathbb{E}}_{\boldsymbol{\theta}^1,\boldsymbol{\theta}^2} \mathrm{Tr}\left[Z_{\boldsymbol{i}}\mathrm{RY}_1(\boldsymbol{\theta}^1)\mathrm{RX}_1(\boldsymbol{\theta}^2)\rho'_{L-1}\mathrm{RX}_1(\boldsymbol{\theta}^2)^\dagger\mathrm{RY}_1(\boldsymbol{\theta}^1)^\dagger\right]^2 \qquad (16)\\
&\geq \mathop{\mathbb{E}}_{\boldsymbol{\theta}^2}\left(\frac{1+\exp[-2\gamma^2]}{2}\right)^S \mathrm{Tr}\left[Z_{\boldsymbol{i}}\mathrm{RX}_1(\boldsymbol{\theta}^2)\rho'_{L-1}\mathrm{RX}_1(\boldsymbol{\theta}^2)^\dagger\right]^2 \qquad (17)\\
&\geq \left(\frac{1+\exp[-2\gamma^2]}{2}\right)^{2S} \mathrm{Tr}\left[Z_{\boldsymbol{i}}\rho'_{L-1}\right]^2, \qquad (18)
\end{aligned}$$

where Eq. (16) is derived by noticing that $Z_{\boldsymbol{i}}$ commutes with CZ_1. Eqs. (17) and (18) yields from Lemma 2. By applying the derivation in Eqs. (16-18) for $j-1$ times, we obtain

$$\mathop{\mathbb{E}}_{\boldsymbol{\theta}^1,\boldsymbol{\theta}^2,\cdots,\boldsymbol{\theta}^{2j-2}} \mathrm{Tr}\left[Z_{\boldsymbol{i}}\rho'_L\right]^2 \geq \left(\frac{1+\exp[-2\gamma^2]}{2}\right)^{2(j-1)S} \mathrm{Tr}\left[Z_{\boldsymbol{i}}\rho'_{L+1-j}\right]^2. \qquad (19)$$

Next, we deal with parameters $\boldsymbol{\theta}^{2j-1}$ and $\boldsymbol{\theta}^{2j}$. For the case of $m = 2j-1$, i.e., the partial derivative corresponds to the RY layer, by denoting $\sigma := \mathrm{RX}_j(\boldsymbol{\theta}^{2j})\rho_{L-j}\mathrm{RX}_j(\boldsymbol{\theta}^{2j})^\dagger$, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\boldsymbol{\theta}^{2j-1},\boldsymbol{\theta}^{2j}} \mathrm{Tr}\left[Z_{\boldsymbol{i}}\rho'_{L+1-j}\right]^2 \\
=& \mathop{\mathbb{E}}_{\boldsymbol{\theta}^{2j-1},\boldsymbol{\theta}^{2j}} \frac{1}{4}\Big(\mathrm{Tr}\left[Z_{\boldsymbol{i}}\mathrm{CZ}_j\mathrm{RY}_j(\boldsymbol{\theta}_+^{2j-1})\sigma\mathrm{RY}_j(\boldsymbol{\theta}_+^{2j-1})^\dagger\mathrm{CZ}_j^\dagger\right] \\
&-\mathrm{Tr}\left[Z_{\boldsymbol{i}}\mathrm{CZ}_j\mathrm{RY}_j(\boldsymbol{\theta}_-^{2j-1})\sigma\mathrm{RY}_j(\boldsymbol{\theta}_-^{2j-1})^\dagger\mathrm{CZ}_j^\dagger\right]\Big)^2 \\
=& \mathop{\mathbb{E}}_{\boldsymbol{\theta}^{2j-1},\boldsymbol{\theta}^{2j}} \frac{1}{4}\left(\mathrm{Tr}\left[Z_{\boldsymbol{i}}\mathrm{RY}_j(\boldsymbol{\theta}_+^{2j-1})\sigma\mathrm{RY}_j(\boldsymbol{\theta}_+^{2j-1})^\dagger\right] - \mathrm{Tr}\left[Z_{\boldsymbol{i}}\mathrm{RY}_j(\boldsymbol{\theta}_-^{2j-1})\sigma\mathrm{RY}_j(\boldsymbol{\theta}_-^{2j-1})^\dagger\right]\right)^2 \qquad (20)\\
=& \mathop{\mathbb{E}}_{\boldsymbol{\theta}^{2j-1},\boldsymbol{\theta}^{2j}} \left(\frac{\partial}{\partial\theta^{2j-1,n}}\mathrm{Tr}\left[Z_{\boldsymbol{i}}\mathrm{RY}_j(\boldsymbol{\theta}^{2j-1})\sigma\mathrm{RY}_j(\boldsymbol{\theta}^{2j-1})^\dagger\right]\right)^2 \qquad (21)\\
\geq& (1-\delta_{i_n 0})\left(\frac{1+\exp[-2\gamma^2]}{2}\right)^{S-1}\frac{1-\exp[-2\gamma^2]}{2}\mathop{\mathbb{E}}_{\boldsymbol{\theta}^{2j}}\mathrm{Tr}\left[Z_{\boldsymbol{i}}\sigma\right]^2 \qquad (22)\\
\geq& (1-\delta_{i_n 0})\left(\frac{1+\exp[-2\gamma^2]}{2}\right)^{2S-1}\frac{1-\exp[-2\gamma^2]}{2}\mathrm{Tr}\left[Z_{\boldsymbol{i}}\rho_{L-j}\right]^2, \qquad (23)
\end{aligned}$$

where Eq. (20) is derived by noticing that $Z_{\boldsymbol{i}}$ commutes with CZ_j. Eq. (21) is derived by using the parameter-shift rule. Eqs. (22) and (23) are obtained by using Lemma 2. For the case of $m = 2j$, the result in Eq. (23) can be derived similarly. Next, by combining the derivation in Eqs. (13-23), we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{\theta}} \mathrm{Tr}\left[Z_{\boldsymbol{i}}\rho_L'\right]^2 &\geq (1-\delta_{i_n 0})\left(\frac{1+\exp[-2\gamma^2]}{2}\right)^{2LS-1}\frac{1-\exp[-2\gamma^2]}{2}\mathrm{Tr}\left[Z_{\boldsymbol{i}}\rho_0\right]^2 \\
&= (1-\delta_{i_n 0})\left(\frac{1+\exp[-2\gamma^2]}{2}\right)^{2LS-1}\frac{1-\exp[-2\gamma^2]}{2} \qquad (24)\\
&= (1-\delta_{i_n 0})\left(\frac{1+\exp[-2\gamma^2]}{2}\right)^{2LS}\frac{1-\exp[-2\gamma^2]}{1+\exp[-2\gamma^2]} \\
&\geq (1-\delta_{i_n 0})\left(1-\gamma^2\right)^{2LS}\frac{2}{3}\gamma^2, \qquad (25)
\end{aligned}$$

where Eq. (24) follows from $\mathrm{Tr}[Z_{\boldsymbol{i}}|0\rangle\langle 0|] = 1$. Eq. (25) is obtained by using $\exp[-x] \geq 1-x$ and $\frac{1-\exp[-x]}{1+\exp[-x]} \geq \frac{x}{3}$ for $0 \leq x \leq 2$. Applying Eq. (24) in Eq. (15) yields

$$\mathop{\mathbb{E}}_{U,\boldsymbol{\theta}}\left(\frac{\partial f}{\partial \theta^{m,n}}\right)^2 \geq \frac{1}{3^S}\sum_{\boldsymbol{i}}\alpha_{\boldsymbol{i}}^2(1-\delta_{i_n 0})\left(1-\gamma^2\right)^{2LS}\frac{2}{3}\gamma^2. \qquad (26)$$

Moreover, for the case of local observables with $S = \mathcal{O}(1)$ and $\alpha_{\boldsymbol{i}} = \mathcal{O}(1)$, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{U,\boldsymbol{\theta}}\|\nabla_{\boldsymbol{\theta}} f\|^2 &= \sum_{m=1}^{2L}\sum_{n=1}^{N}\mathop{\mathbb{E}}_{U,\boldsymbol{\theta}}\left(\frac{\partial f}{\partial \theta^{m,n}}\right)^2 \\
&\geq \sum_{m=1}^{2L}\sum_{n=1}^{N}\frac{1}{3^S}\sum_{\boldsymbol{i}}\alpha_{\boldsymbol{i}}^2(1-\delta_{i_n 0})\left(1-\gamma^2\right)^{2LS}\frac{2}{3}\gamma^2 \\
&\geq \mathcal{O}(1) \qquad (27)
\end{aligned}$$

by choosing $\gamma^2 = c/L$.

□

D Backbone ResNet-18 Training

Problem Statement for Anisotropic Hamiltonian

Let $\{(\mathbf{x}_i, \mathbf{c}_i)\}_{i=1}^N$ be a training dataset, where each $\mathbf{x}_i \in \mathbb{R}^{C\times H\times W}$ is a multi-channel 2D input image/tensor (in our case, $C = 5$ for the coordinate channels plus the parameters (a, b, c)). Each corresponding ground truth $\mathbf{c}_i \in \mathbb{R}^{1\times H\times W}$ is the target intensity map.

We denote the model parameters collectively by θ. The model, which we write as a function $f_\theta(\mathbf{x})$, outputs a predicted intensity map of size $\mathbb{R}^{1\times H\times W}$. Specifically, the network architecture can be expressed as a composition of layers:

$$f_\theta(\mathbf{x}) \;=\; \mathrm{ConvOut}\Big(\mathrm{MHAttn2D}\big(\mathrm{ResNet18}(\mathbf{x})\big)\Big),$$

where:

- ResNet18Backbone denotes the modified ResNet18 layers (first convolution, no max pooling, and adjusted strides).
- MHAttn2D is the multi-head self-attention module in 2D.
- ConvOut is the final 1×1 convolution layer projecting to 1 channel.

Loss Function

We use the Mean Squared Error (MSE) as the training loss. For a single training example $(\mathbf{x}, \mathbf{c})$, the MSE loss is given by

$$\mathcal{L}(\mathbf{c}, f_\theta(\mathbf{x})) \;=\; \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} \Big(y_{h,w} - \hat{c}_{h,w}(\theta) \Big)^2 \;=\; \frac{1}{HW} \left\| \mathbf{c} - f_\theta(\mathbf{x}) \right\|^2,$$

where $\hat{c}_{h,w}(\theta)$ represents the predicted pixel value at position (h, w), and $c_{h,w}$ is the corresponding ground truth pixel value.

Given a minibatch of size B (the default in our script is often $B = 1$, but let us write it in general form), the total MSE loss is the average over the batch:

$$\mathcal{L}_{\text{batch}}(\theta) = \frac{1}{B} \sum_{i=1}^{B} \frac{1}{HW} \left\| \mathbf{c}_i - f_\theta(\mathbf{x}_i) \right\|^2.$$

Optimization via Gradient Descent

We seek to find parameters θ that minimize $\mathcal{L}_{\text{batch}}(\theta)$. In gradient-based optimization, we iteratively update θ in the direction of the negative gradient of $\mathcal{L}_{\text{batch}}$. If we let η denote the learning rate, then the parameter update rule in standard gradient descent is:

$$\theta \leftarrow \theta \;-\; \eta \, \nabla_\theta \mathcal{L}_{\text{batch}}(\theta).$$

In our code, we use the Adam optimizer , which introduces adaptive learning rates and momentum terms. Conceptually, the Adam update can be written as:

$$\begin{aligned}
m_t &\leftarrow \beta_1 \, m_{t-1} + (1 - \beta_1) \, \nabla_\theta \mathcal{L}_{\text{batch}}(\theta), \\
v_t &\leftarrow \beta_2 \, v_{t-1} + (1 - \beta_2) \, \big(\nabla_\theta \mathcal{L}_{\text{batch}}(\theta) \big)^2, \\
\hat{m}_t &\leftarrow \frac{m_t}{1 - \beta_1^t}, \\
\hat{v}_t &\leftarrow \frac{v_t}{1 - \beta_2^t}, \\
\theta &\leftarrow \theta - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon},
\end{aligned}$$

where β_1, β_2 are hyperparameters (often 0.9 and 0.999), and ϵ is a small constant (e.g., 10^{-8}) to prevent division by zero. These moment estimates help achieve faster and more stable convergence in practice.

Forward Pass

1. **Input Preparation:** Each input $\mathbf{x}_i$ is a 5-channel tensor of dimension $(5, H, W)$. For instance:
$$\mathbf{x}_i \;=\; \big(x_{i,\text{coordLayer}},\, x_{i,\text{coordCol}},\, x_{i,a},\, x_{i,b},\, x_{i,c} \big).$$
2. **ResNet18 Backbone:** The first convolutional layer, batch norm, and subsequent residual blocks produce an intermediate feature map:
$$\mathbf{f}_{\text{res}} = \text{ResNet18Backbone}(\mathbf{x}_i).$$
3. **2D Multi-Head Self-Attention:** We apply a 2D multi-head self-attention module to $\mathbf{f}_{\text{res}}$. This module computes queries $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$ across spatial positions:
$$\mathbf{Q} = W_q * \mathbf{f}_{\text{res}}, \quad \mathbf{K} = W_k * \mathbf{f}_{\text{res}}, \quad \mathbf{V} = W_v * \mathbf{f}_{\text{res}},$$
where $*$ denotes a 1×1 convolution, and W_q, W_k, W_v are the learned parameters. We split these into H_{heads} heads, each of dimension d_k per head. The attention weights for each head h are given by:
$$\alpha_h(\mathbf{Q}, \mathbf{K}) \;=\; \text{Softmax}\big(\mathbf{Q}_h \mathbf{K}_h^\top \big),$$
and the output of the attention for head h is
$$\mathbf{O}_h = \mathbf{V}_h \, \alpha_h^\top.$$
The combined output across all heads is then projected and added to the input feature map (via a residual connection multiplied by a learnable scalar γ):
$$\mathbf{f}_{\text{attn}} \;=\; \mathbf{f}_{\text{res}} + \gamma \Big[W_o \big(\text{Concat}_{h=1}^{H_{\text{heads}}}(\mathbf{O}_h) \big) \Big].$$

4. **Final 1×1 Convolution:** We pass $\mathbf{f}_{\text{attn}}$ through a final convolution to predict a single-channel intensity map:
$$\hat{\mathbf{c}}_i \;=\; \text{Conv}_{1\times1}\big(\mathbf{f}_{\text{attn}}\big).$$
5. **Upsampling (if needed):** If the ResNet blocks reduce spatial resolution, we may apply bilinear upsampling to match the original height H and width W:
$$\hat{\mathbf{c}}_i \;=\; \text{Upsample}\big(\hat{\mathbf{c}}_i, (H, W)\big).$$

Backward Pass

We define the MSE loss for the i-th example:
$$\mathcal{L}(\theta; \mathbf{x}_i, \mathbf{c}_i) \;=\; \frac{1}{HW} \left\|\mathbf{c}_i - \hat{\mathbf{c}}_i\right\|^2.$$
Then, we compute the gradient with respect to θ by applying the chain rule:
$$\nabla_\theta \mathcal{L}(\theta; \mathbf{x}_i, \mathbf{c}_i) \;=\; \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{c}}_i} \frac{\partial \hat{\mathbf{c}}_i}{\partial \mathbf{f}_{\text{attn}}} \frac{\partial \mathbf{f}_{\text{attn}}}{\partial \mathbf{f}_{\text{res}}} \frac{\partial \mathbf{f}_{\text{res}}}{\partial \theta}.$$
In practice, modern deep learning libraries (e.g., PyTorch) automate this via backpropagation, computing partial derivatives layer by layer. The final step is an optimizer update (e.g., Adam) based on these gradients.

The overall training loop can be summarized as follows:

1. **Initialize model parameters** θ.
2. **For each epoch** $e \in \{1, 2, \ldots, E\}$:
   (a) Shuffle dataset $\{(\mathbf{x}_i, \mathbf{c}_i)\}_{i=1}^N$ and (optionally) form mini-batches of size B.
   (b) **For each mini-batch** $\mathcal{B}$:
      i. Perform a forward pass: $\hat{\mathbf{c}}_i = f_\theta(\mathbf{x}_i)$ for each $(\mathbf{x}_i, \mathbf{c}_i) \in \mathcal{B}$.
      ii. Compute the MSE loss over the batch:
$$\mathcal{L}_{\text{batch}}(\theta) = \frac{1}{B} \sum_{i \in \mathcal{B}} \frac{1}{HW} \left\|\mathbf{c}_i - \hat{\mathbf{c}}_i\right\|^2.$$
      iii. Backpropagate to obtain $\nabla_\theta \mathcal{L}_{\text{batch}}(\theta)$.
      iv. Update parameters θ using Adam or standard gradient descent.

After convergence (or after a chosen number of epochs), we obtain a trained model f_θ that can be used to predict freeze intensity maps for new inputs.

E TITAN: Complementary to other methods

E.1 Layer-wise Greedy Gradient Descent (LGGD)

Let a hardware-efficient VQE ansatz acting on N qubits be written as a depth-factorised unitary
$$U(\boldsymbol{\theta}) \;=\; \prod_{\ell=1}^{L} U_\ell\big(\boldsymbol{\theta}^{(\ell)}\big), \quad U_\ell = \big[\text{CZ}\cdot\text{RX}\cdot\text{RY}\big]^{\otimes N}, \tag{28}$$
where $\boldsymbol{\theta} = \big(\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(L)}\big)$ and $\boldsymbol{\theta}^{(\ell)} \in \mathbb{R}^{2N}$ contains the RX/RY angles of layer ℓ. Throughout we denote by $\boldsymbol{\theta}_{1:\ell} = \big(\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(\ell)}\big)$ the *prefix* up to depth ℓ. The VQE objective for a target Hamiltonian $H = \sum_j h_j P_j$ is
$$\mathcal{L}(\boldsymbol{\theta}) = \langle 0|U(\boldsymbol{\theta})^\dagger H U(\boldsymbol{\theta})|0\rangle, \quad \text{to be } \textit{minimised}. \tag{29}$$

Greedy depth-growth schedule. Starting from a shallow prefix of depth $\ell = 1$ we minimise
$$\mathcal{L}_\ell\big(\boldsymbol{\theta}_{1:\ell}\big) := \mathcal{L}\big(\boldsymbol{\theta}_{1:\ell}, \underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{L-\ell \text{ frozen layers}}\big), \tag{30}$$
keeping all yet-to-be-added layers fixed at 0. After T gradient steps we **freeze** $\boldsymbol{\theta}^{(\ell)}$ and expose $\boldsymbol{\theta}^{(\ell+1)}$; the process repeats until $\ell = L$. The update rule for the active prefix is the usual GD step,
$$\boldsymbol{\theta}_{t+1}^{(\ell)} = \boldsymbol{\theta}_t^{(\ell)} - \eta \nabla_{\boldsymbol{\theta}^{(\ell)}} \mathcal{L}_\ell\big(\boldsymbol{\theta}_t\big), \tag{31}$$
where η is the learning rate.

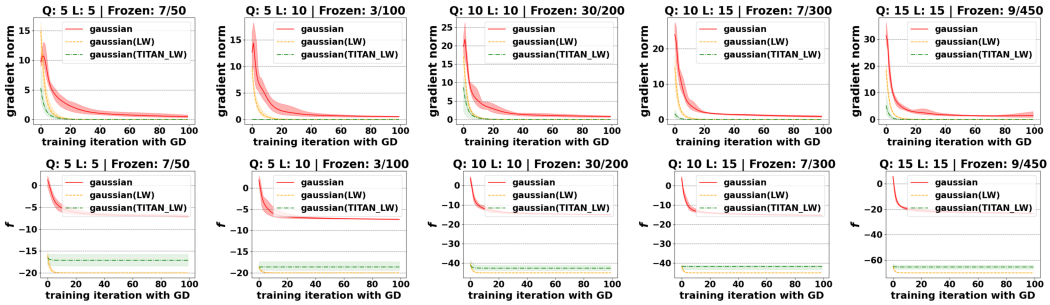


Figure 3: Convergence behavior of TITAN + LAYERWISE (green dotted) versus the *Baseline* (solid red) and a *Layerwise* control (yellow dotted) HEA isotropic Hamiltonian with Gaussian initialization [20]. The **top row** plots the ℓ_2-norm, while the **bottom row** tracks f (final energy). Shaded envelopes denote $\pm\sigma$ across **5 independent runs** with threshold ($\tau = 80$).

Gradient consistency. Because the expectation in (30) depends on the prefix *only through* $U_{1:\ell} \equiv \prod_{k=1}^{\ell} U_k$, one has

$$\nabla_{\boldsymbol{\theta}^{(\ell)}} \mathcal{L} = \nabla_{\boldsymbol{\theta}^{(\ell)}} \mathcal{L}_\ell, \qquad \nabla_{\boldsymbol{\theta}^{(k)}} \mathcal{L}_\ell = \mathbf{0}, \ \forall\, k > \ell,$$

hence LGGD reproduces the exact gradients of the full objective *restricted to* the current layer set and incurs no bias. The per-iteration cost is therefore $\mathcal{O}(\ell\,\mathrm{poly}(N))$ instead of $\mathcal{O}(L\,\mathrm{poly}(N))$, yielding a cumulative complexity $\sum_{\ell=1}^{L} \mathcal{O}(\ell) = \mathcal{O}(L^2)$ circuit evaluations—quadratically smaller than standard depth-L training.

Convergence guarantee (sketch). Assume each layer unitary $U_\ell(\cdot)$ is β-smooth and the objective is lower-bounded by $\mathcal{L}_\star$. For a fixed depth ℓ and step size $0 < \eta \le 1/\beta$,

$$\mathcal{L}_\ell\big(\boldsymbol{\theta}_t\big) - \mathcal{L}_\star \ \le\ \frac{\|\nabla \mathcal{L}_\ell(\boldsymbol{\theta}_0)\|^2}{2\beta t},$$

mirroring classical GD. Thus each stage achieves ε-accuracy in $T = \mathcal{O}\big(\beta/\varepsilon\big)$ steps, and the overall runtime is $\mathcal{O}\big(L^2\beta/\varepsilon\big)$.

Discussion. LGGD based Layerwise (LW) method bears a resemblance to curriculum-style training in deep networks and to the *Disentangled ADAPT-VQE*[8] strategy, but differs in that (i) it does not search for new gates; (ii) its depth growth is deterministic; and (iii) intermediate gradients are exact for the active sub-circuit. As illustrated in Figure 3, the LGGD-based layer-wise (LW) initialization surpasses conventional Gaussian initialization, achieving both faster convergence and a lower final energy. More critically, the TITAN framework interfaces seamlessly with either LGGD-based LW or Gaussian initialization. When coupled with LGGD-based LW, TITAN preserves the same minimum-energy solution quality while *further* accelerating convergence; in addition, the parameter-freezing mechanism embedded in TITAN reduces the measurement overhead, thereby enhancing experimental efficiency without sacrificing accuracy.

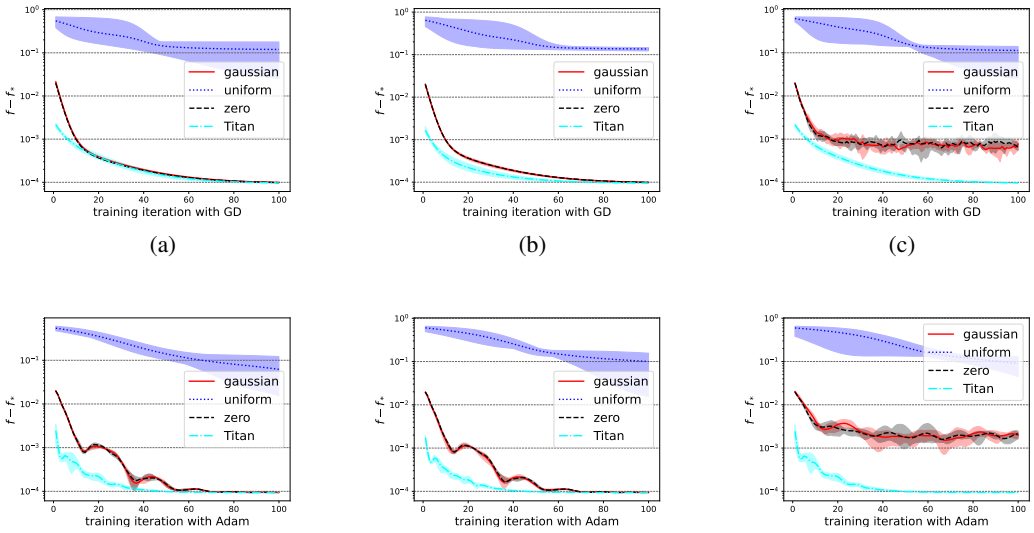
F Experiments

TITAN Training Settings For Dataset Construction, all runs sweep the circuit width $n_{\text{qubits}} \in \{5, \ldots, 15\}$ and depth $L \in \{5, \ldots, 15\}$, train each setting for 100 gradient steps, repeat the experiment 20 times, and print diagnostics every 20 epochs. Learning rates are fixed at $\eta = 0.01$. For gradient-descent with APFA, we decay the per-parameter EMA with $\alpha = 0.7$ and freeze a weight once its EMA stays below $\tau_f = 0.2\,\bar{g}$ for $N_f = 3$ consecutive steps; re-activation occurs when the EMA exceeds $\tau_a = 0.4\,\tau_f$ for $N_a = 2$ steps after an initial 20-step warm-up. The Adam variant inherits the same α and patience lengths but tightens the thresholds to $\tau_f = 0.5\,\bar{g}$ and $\tau_a = 1.0\,\phi_f$; its momentum parameters are $\beta_1 = 0.9,\ \beta_2 = 0.99$ with numerical stabilizer $\varepsilon = 10^{-8}$.

For model training, the learning rate is set to be 0.001 and training epochs are 1200, optimizer is Adam.

Isotropic Hamiltonian & Anisotropic Hamiltonian Train each setting for 100 gradient steps, repeat the experiment 20 times, and print diagnostics every 20 epochs. Learning rates are fixed at $\eta = 0.01$. Optimizer is Gradient Descent (GD).

F.1 Supplementary experiments and results: Molecule



(a) (b) (c)

(d) (e) (f)

Figure 4: Numerical results of loss f minus global minimum f_* of the molecule LiH. The first and second rows show training results with the gradient descent and the Adam optimizer, respectively. The left, the middle, and the right columns show results using accurate gradients, noisy gradients with adaptive-distributed noises, and noisy gradients with constant-distributed noises. The variance of noises in the middle line (Figures 4b and 4e), while the variance of noises in the right line (Figures 4c and 4f) is 0.001. Each line denotes the average of 5 rounds of optimizations.

For an active space of n_e electrons in n_0 spin-orbitals, the second-quantized encoding employs $N = n_0$ qubits, each representing a single spin-orbital occupation. The mean-field reference is$|\phi_{\mathrm{HF}}\rangle = \bigotimes_{j=1}^{n_e} |1\rangle_j \otimes \bigotimes_{j=n_e+1}^{n_0} |0\rangle_j$, where $|1\rangle$ ($|0\rangle$) denotes an occupied (vacant) spin-orbital. Electron correlation is introduced via a sequence of Givens rotations, $V_{\mathrm{Givens}}(\boldsymbol{\theta}) = \prod_{l=1}^{L} R^{(l)}_{\mathrm{Givens}}(\theta_l)$, with L two- or four-qubit gates; the value of L is determined by the chosen orbital mapping and basis. The molecular electronic Hamiltonian in second quantization reads

$$H = \sum_{pq} h_{pq}\, a_p^\dagger a_q + \tfrac{1}{2} \sum_{pqrs} h_{pqrs}\, a_p^\dagger a_q^\dagger a_r a_s, \tag{32}$$

which, after a Jordan–Wigner or Bravyi–Kitaev transform, becomes a sum of Pauli strings on N qubits. Ground-state energy with respect to the parameter vector $\boldsymbol{\theta}$ is approximated by minimizing

$$f(\boldsymbol{\theta}) = \langle \phi_{\mathrm{HF}}|\, V^\dagger_{\mathrm{Givens}}(\boldsymbol{\theta})\, H\, V_{\mathrm{Givens}}(\boldsymbol{\theta})\, |\phi_{\mathrm{HF}}\rangle\,. \tag{33}$$

Active Electrons and Spin Orbitals For a molecule with n_e active electrons and n_0 spin orbitals. The dimension of the corresponding Hilbert space on a quantum computer is determined by n_0, leading to $N = n_0$, qubits for representing the molecular wavefunction in the second-quantized form.

Hartree–Fock Reference State A common starting point for a variational algorithm is the Hartree–Fock (HF) reference state,

$$|\phi_{\mathrm{HF}}\rangle = \underbrace{|1\rangle \otimes |1\rangle \otimes \cdots \otimes |1\rangle}_{n_e \text{ times}} \otimes \underbrace{|0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle}_{n_0 - n_e \text{ times}}. \tag{34}$$

This state encodes the mean-field approximation in which the n_e active electrons occupy the lowest-energy orbitals, and the remaining $n_0 - n_e$ orbitals are left unoccupied.

Parameterization via Givens Rotations To capture electron correlation effects beyond the HF approximation, we apply a parameterized quantum circuit composed of Givens rotations. Denote the unitary operator as $V_{\text{Givens}}(\theta) = \prod_{i=1}^{L} R^{(i)}_{\text{Givens}}(\theta_i)$, where each $R^{(i)}_{\text{Givens}}(\theta_i)$ is either a 2-qubit or 4-qubit Givens rotation gate, parameterized by an angle θ_i. The total number of such gates, L, depends on the specific molecule (e.g., the basis set, chosen orbital mapping, etc.).

Molecular Hamiltonian The second-quantized electronic structure Hamiltonian for a given molecule can be written generally as

$$H_{\text{mol}} = \sum_{p,q} h_{pq}\, a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s} h_{pqrs}\, a_p^\dagger a_q^\dagger a_r a_s, \tag{35}$$

where $a_p^\dagger$ (a_p) are the fermionic creation (annihilation) operators, and h_{pq}, h_{pqrs} are the one- and two-electron integrals, respectively. After performing a Jordan-Wigner or Bravyi-Kitaev transformation, H_{mol} can be expressed as a sum of Pauli operators on N qubits.

Variational Loss Function Our goal is to minimize the expectation value of H_{mol} with respect to the parameterized state. Formally, we define the loss function

$$f(\theta) = \text{Tr}\Big[H_{\text{mol}}\, V_{\text{Givens}}(\theta)\ |\phi_{\text{HF}}\rangle \langle\phi_{\text{HF}}|\ V_{\text{Givens}}(\theta)^\dagger\Big]. \tag{36}$$

By varying the set of parameters $\{\theta_i\}$ and minimizing $f(\theta)$, one effectively searches for the lowest eigenvalue of H_{mol}, which corresponds to the ground-state energy of the molecule.

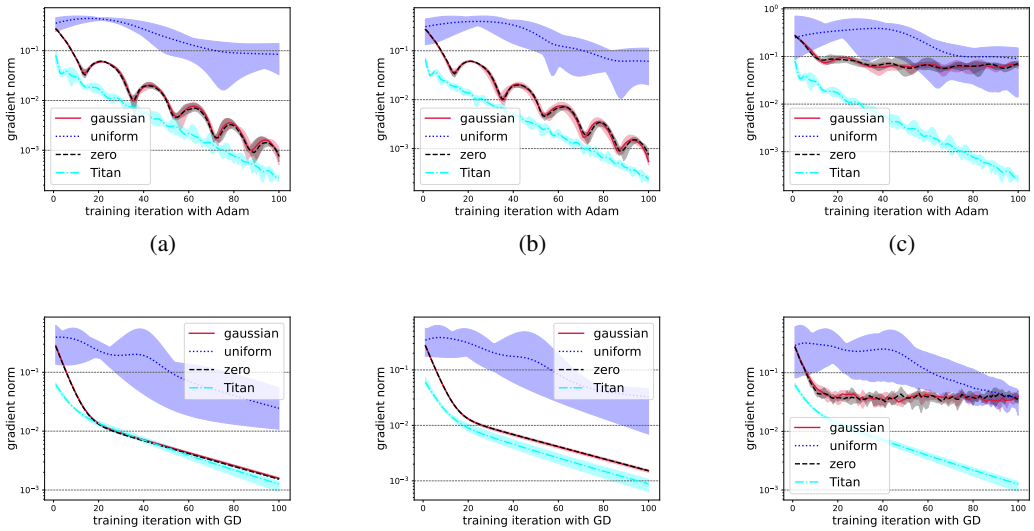
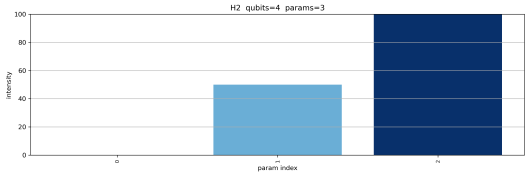


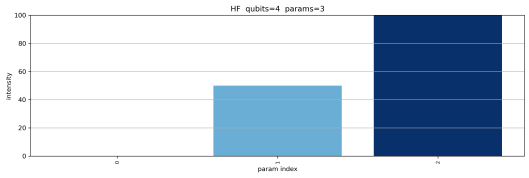
Figure 5: Numerical results of finding the ground energy of the molecule LiH. The first and second rows show training results with the gradient descent and the Adam optimizer, respectively. The left, the middle, and the right columns show results using accurate gradients, noisy gradients with adaptive-distributed noises, and noisy gradients with constant-distributed noises. The variance of noises in the middle line (Figures 5b and 5e), while the variance of noises in the right line (Figures 5c and 5f) is 0.001. Each line denotes the average of 5 rounds of optimizations.

Table 1: Representative Molecules

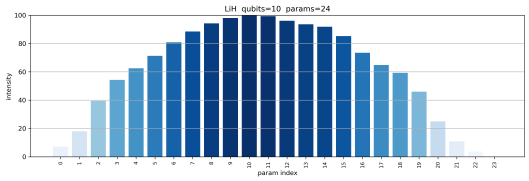
Molecule	N_{e}	N_{orb}	Qubits (N)
H_2	2	2	4
HF	2	5	4
LiH	2	5	10
BeH_2	4	6	12
H_2O	6	7	10
N_2	6	6	12
CO	6	7	12



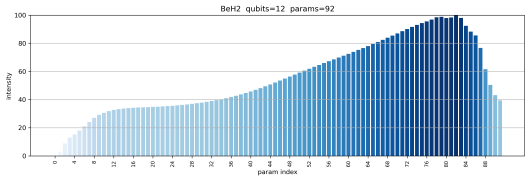
(a) H2



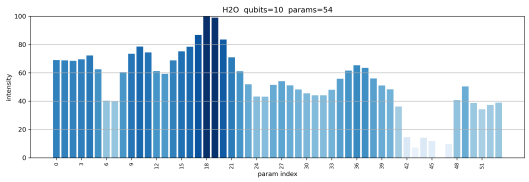
(b) HF



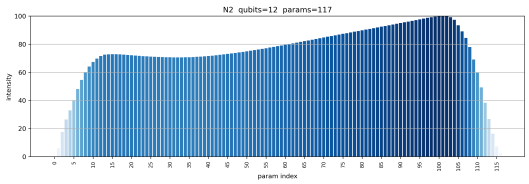
(c) LiH



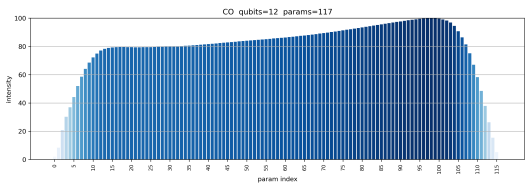
(d) BeH2



(e) H2O



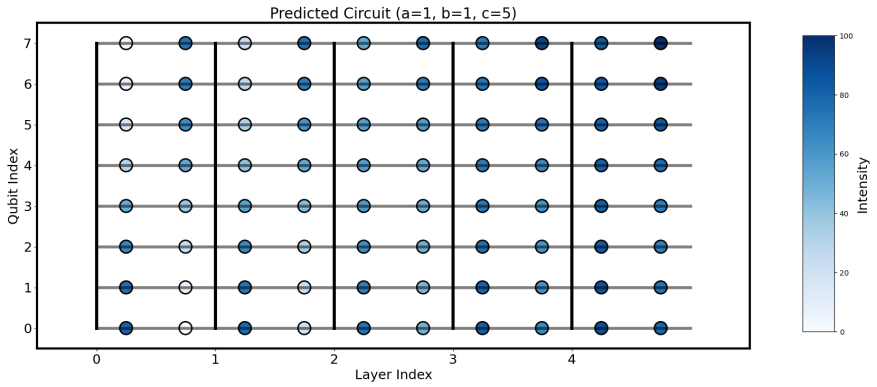
(f) N2



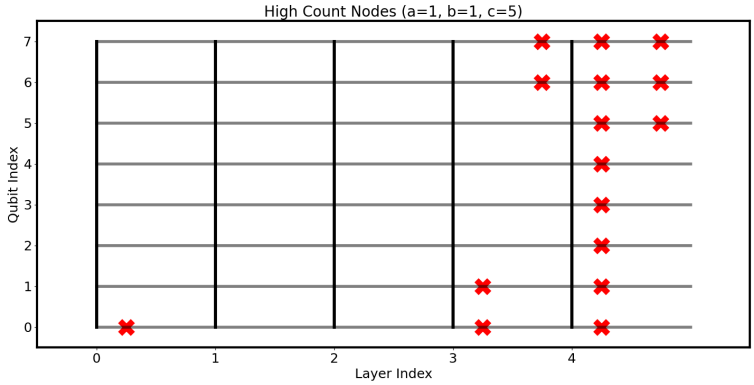
(g) CO

Figure 6: Titan Predicted intensity of different molecules.

F.2 Supplementary experiments and results: Anisotropy Hamiltonian

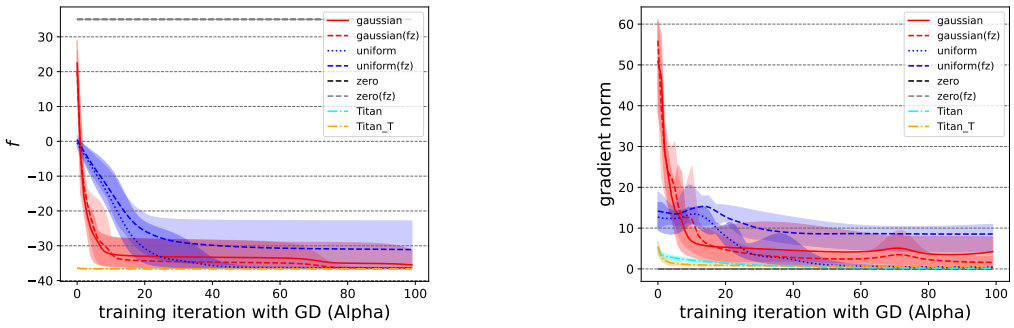


(a)



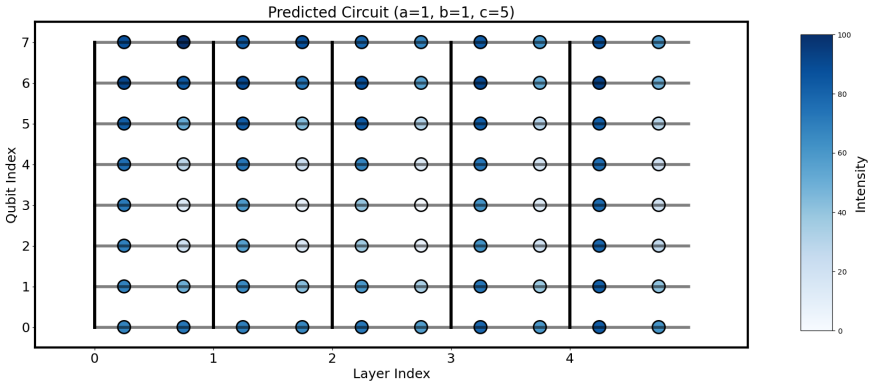
(b)

Figure 7: (a) TITAN predicted intensity. (b) TITAN selected frozen gates (**16/80**) (20%). Qubits 8, Layers 5, Threshold: $\tau = 80$. HEA-based anisotropic Hamiltonian solver: $1 \times$ XX + $1 \times$ YY + $5 \times$ ZZ.

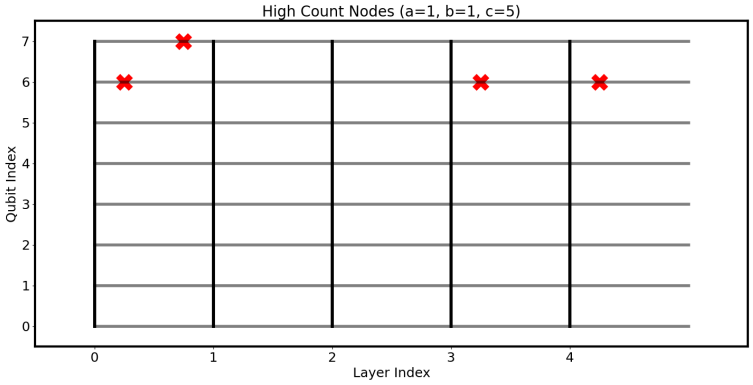


(a) (b)

Figure 8: (a) Loss GD Noiseless. (b) Gradient Norm GD Noiseless. Qubits 8, Layers 5, Threshold: $\tau = 80$. Selected Frozen Gates (**16/80**) (20%). HEA-based anisotropic Hamiltonian solver: $1 \times$ XX + $1 \times$ YY + $5 \times$ ZZ. TITAN: Enhanced Gaussian. TITAN_T: APFA Only without freezing.

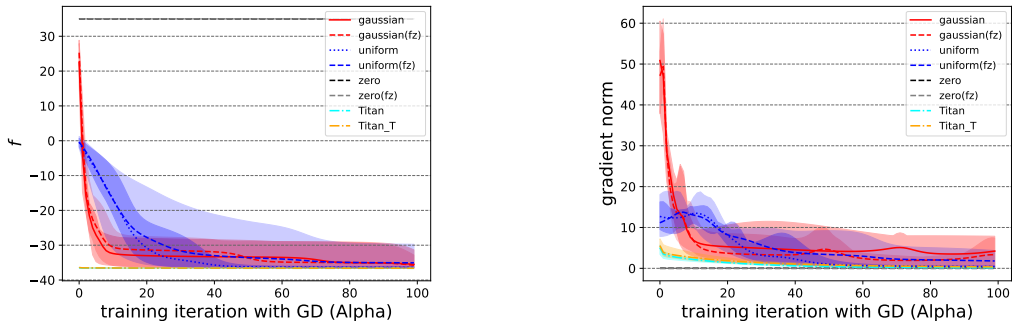


(a)



(b)

Figure 9: (a) TITAN predicted intensity. (b) TITAN selected Frozen Gates (**4/80**) (5%). Qubits 8, Layers 5, Threshold: 90. HEA-based anisotropic Hamiltonian solver: $1 \times$ XX + $1 \times$ YY + $5 \times$ ZZ.



(a) (b)

Figure 10: (a) Loss GD Noiseless. (b) Gradient Norm GD Noiseless. Qubits 8, Layers 5, Threshold: 90. Selected Frozen Gates (**4/80**) (5%). HEA-based anisotropic Hamiltonian solver: $1 \times$ XX + $1 \times$ YY + $5 \times$ ZZ. TITAN: Enhanced Gaussian. TITAN_T: APFA Only without freezing.

F.3 Supplementary experiments and results: Isotropy Hamiltonians

Table 2: Frozen performance comparison between TITAN and Random Freezing (ansätze: HEA, Optimizer: GD). Frozen-parameter counts that exceed 10% are highlighted in blue; entries with $\Delta E \leq 0$ are highlighted in green.

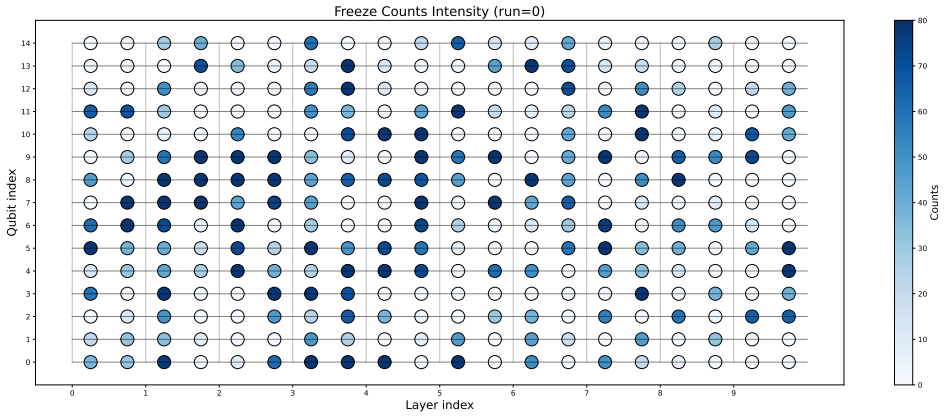
	5	6	7	8	9	10	11	12	13	14	15
Proposed TITAN: Final Energy Comparison ($\Delta E = E_{\text{Titan Gauss}} - E_{\text{Baseline Gauss}}$)											
5	0.161	0.145	0.152	0.079	0.374	0.198	**−0.092**	0.315	0.465	0.203	0.260
6	**−0.084**	0.109	0.100	0.301	**−0.014**	0.147	0.216	0.329	0.096	0.922	0.222
7	**−0.222**	**−0.025**	0.268	0.235	0.202	0.264	**−0.002**	**−0.008**	0.254	0.088	0.324
8	0.108	**−0.022**	0.123	0.066	0.102	0.089	**−0.029**	0.147	0.152	0.035	0.215
9	0.019	**−0.024**	**−0.114**	**−0.030**	0.017	**−0.120**	0.220	**−0.114**	**−0.177**	-0.025	**−0.063**
10	**−0.084**	0.027	0.029	0.002	0.055	0.104	**−0.033**	0.131	0.115	0.113	0.289
Frozen Parameters (threshold $\tau = 80$)											
5	**7**$^{(14.0\%)}$	5$^{(8.3\%)}$	**9**$^{(12.9\%)}$	5$^{(6.3\%)}$	**15**$^{(16.7\%)}$	7$^{(7\%)}$	**12**$^{(10.9\%)}$	**13**$^{(10.8\%)}$	10$^{(7.7\%)}$	**20**$^{(14.3\%)}$	**16**$^{(10.7\%)}$
6	4$^{(6.7\%)}$	4$^{(5.6\%)}$	**15**$^{(17.9\%)}$	8$^{(8.3\%)}$	10$^{(9.3\%)}$	8$^{(6.7\%)}$	12$^{(9.1\%)}$	**25**$^{(17.3\%)}$	**25**$^{(16.0\%)}$	**31**$^{(18.5\%)}$	**27**$^{(15.0\%)}$
7	5$^{(7.1\%)}$	5$^{(6.0\%)}$	4$^{(4.1\%)}$	10$^{(8.9\%)}$	9$^{(7.1\%)}$	2$^{(1.4\%)}$	6$^{(3.9\%)}$	10$^{(6.0\%)}$	7$^{(3.9\%)}$	5$^{(2.6\%)}$	19$^{(9.1\%)}$
8	4$^{(5.0\%)}$	9$^{(9.4\%)}$	4$^{(3.6\%)}$	**14**$^{(10.9\%)}$	7$^{(4.9\%)}$	11$^{(6.9\%)}$	**22**$^{(12.5\%)}$	**21**$^{(10.9\%)}$	8$^{(3.9\%)}$	6$^{(2.7\%)}$	8$^{(3.3\%)}$
9	2$^{(2.2\%)}$	1$^{(0.9\%)}$	10$^{(7.9\%)}$	2$^{(1.4\%)}$	8$^{(4.9\%)}$	9$^{(5.0\%)}$	**21**$^{(10.6\%)}$	10$^{(4.6\%)}$	11$^{(4.7\%)}$	8$^{(3.2\%)}$	10$^{(3.7\%)}$
10	3$^{(3\%)}$	12$^{(10.0\%)}$	7$^{(5.0\%)}$	**18**$^{(11.3\%)}$	11$^{(6.1\%)}$	**30**$^{(15.0\%)}$	8$^{(3.6\%)}$	4$^{(1.7\%)}$	6$^{(2.3\%)}$	24$^{(8.6\%)}$	18$^{(6.0\%)}$
Final Energy Comparison ($\Delta E = E_{\text{Random Gauss}} - E_{\text{Baseline Gauss}}$)											
5	0.184	0.153	0.099	0.042	0.118	0.122	0.026	0.089	0.380	0.615	0.466
6	0.106	0.160	0.107	0.139	0.103	0.090	0.126	0.094	0.147	0.460	0.216
7	0.118	0.004	0.047	0.083	0.027	0.025	0.159	0.029	0.163	0.255	0.375
8	0.041	0.123	0.022	0.066	0.026	0.018	0.047	0.075	0.222	0.025	0.173
9	0.052	0.005	0.095	0.072	0.010	**−0.069**	0.049	0.051	0.223	0.042	0.037
10	0.113	0.022	0.003	0.032	0.015	0.015	**−0.068**	0.041	**−0.097**	0.046	0.268

Table 3: Frozen performance comparison between TITAN and Random Freezing (ansätze: HEA, Optimizer: GD). Frozen parameters exceeding between 10% and 30% are colored with blue. The value in each cell is written as $\Delta E^{(|\Delta E/E_{\text{baseline}}|\%)}$; cells where $\Delta E \leq 0$ are shaded with green of Titan and red of random freezing.

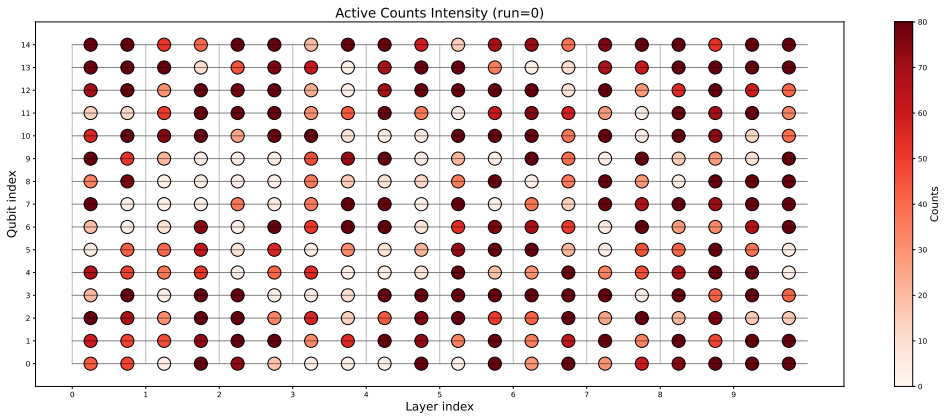
Layers \ Qubit	**Qubit**: 5	**Qubit**: 6	**Qubit**: 7	**Qubit**: 8	**Qubit**: 9	**Qubit**: 10	**Qubit**: 11	**Qubit**: 12	**Qubit**: 13	**Qubit**: 14	**Qubit**: 15
Proposed TITAN: Final Energy Comparison: $\Delta E = E_{\text{Titan Gaussian}} - E_{\text{Baseline Gaussian}}$											
Layers: **5**	$0.161^{(2.3\%)}$	$0.145^{(1.8\%)}$	$0.152^{(1.5\%)}$	$0.079^{(0.7\%)}$	$0.374^{(2.9\%)}$	$0.198^{(1.4\%)}$	$-0.092^{(0.6\%)}$	$0.315^{(1.8\%)}$	$0.465^{(2.5\%)}$	$0.203^{(1.0\%)}$	$0.260^{(1.2\%)}$
Layers: **6**	$-0.084^{(1.2\%)}$	$0.109^{(1.3\%)}$	$0.100^{(1.0\%)}$	$0.301^{(2.6\%)}$	$-0.014^{(0.1\%)}$	$0.147^{(1.0\%)}$	$0.216^{(1.3\%)}$	$0.329^{(1.9\%)}$	$0.096^{(0.5\%)}$	$0.922^{(4.5\%)}$	$0.222^{(1.0\%)}$
Layers: **7**	$-0.222^{(3.1\%)}$	$-0.025^{(0.3\%)}$	$0.268^{(2.6\%)}$	$0.235^{(2.0\%)}$	$0.202^{(1.5\%)}$	$0.264^{(1.8\%)}$	$-0.002^{(0.0\%)}$	$-0.008^{(0.0\%)}$	$0.254^{(1.3\%)}$	$0.088^{(0.4\%)}$	$0.324^{(1.4\%)}$
Layers: **8**	$0.108^{(1.5\%)}$	$-0.022^{(0.3\%)}$	$0.123^{(1.2\%)}$	$0.066^{(0.6\%)}$	$0.102^{(0.8\%)}$	$0.089^{(0.6\%)}$	$-0.029^{(0.2\%)}$	$0.147^{(0.8\%)}$	$0.152^{(0.8\%)}$	$0.035^{(0.2\%)}$	$0.215^{(1.0\%)}$
Layers: **9**	$0.019^{(0.3\%)}$	$-0.024^{(0.3\%)}$	$-0.114^{(1.1\%)}$	$-0.030^{(0.3\%)}$	$0.017^{(0.1\%)}$	$-0.120^{(0.8\%)}$	$0.220^{(1.3\%)}$	$-0.114^{(0.6\%)}$	$-0.177^{(0.9\%)}$	$-0.025^{(0.1\%)}$	$-0.063^{(0.3\%)}$
Layers: **10**	$-0.084^{(1.1\%)}$	$0.027^{(0.3\%)}$	$0.029^{(0.3\%)}$	$0.002^{(0.0\%)}$	$0.055^{(0.4\%)}$	$0.104^{(0.7\%)}$	$-0.033^{(0.2\%)}$	$0.131^{(0.7\%)}$	$0.115^{(0.6\%)}$	$0.113^{(0.5\%)}$	$0.289^{(1.3\%)}$
Layers: **11**	$0.022^{(0.3\%)}$	$-0.002^{(0.0\%)}$	$0.118^{(1.1\%)}$	$-0.010^{(0.1\%)}$	$-0.169^{(1.3\%)}$	$0.103^{(0.7\%)}$	$0.050^{(0.3\%)}$	$0.123^{(0.7\%)}$	$0.206^{(1.0\%)}$	$0.096^{(0.4\%)}$	$0.095^{(0.4\%)}$
Layers: **12**	$0.022^{(0.3\%)}$	$-0.001^{(0.0\%)}$	$-0.095^{(0.9\%)}$	$0.052^{(0.4\%)}$	$0.075^{(0.6\%)}$	$0.148^{(1.0\%)}$	$0.298^{(1.8\%)}$	$-0.088^{(0.5\%)}$	$-0.023^{(0.1\%)}$	$-0.206^{(1.0\%)}$	$0.083^{(0.4\%)}$
Layers: **13**	$0.033^{(0.4\%)}$	$0.039^{(0.4\%)}$	$0.262^{(2.4\%)}$	$-0.018^{(0.2\%)}$	$-0.157^{(1.2\%)}$	$0.053^{(0.3\%)}$	$-0.073^{(0.4\%)}$	$0.037^{(0.2\%)}$	$-0.016^{(0.1\%)}$	$-0.227^{(1.1\%)}$	$-0.131^{(0.6\%)}$
Layers: **14**	$-0.034^{(0.5\%)}$	$0.018^{(0.2\%)}$	$0.212^{(2.0\%)}$	$-0.121^{(1.0\%)}$	$0.008^{(0.1\%)}$	$-0.001^{(0.0\%)}$	$0.093^{(0.6\%)}$	$-0.215^{(1.2\%)}$	$0.186^{(0.9\%)}$	$0.042^{(0.2\%)}$	$0.201^{(0.9\%)}$
Layers: **15**	$-0.000^{(0.0\%)}$	$0.076^{(0.9\%)}$	$-0.066^{(0.6\%)}$	$0.002^{(0.0\%)}$	$-0.154^{(1.1\%)}$	$0.050^{(0.3\%)}$	$-0.141^{(0.8\%)}$	$0.094^{(0.5\%)}$	$-0.061^{(0.3\%)}$	$-0.005^{(0.0\%)}$	$0.033^{(0.1\%)}$
Frozen Parameters: Threshold: $\tau = 80$											
Layers: **5**	$7^{(14.0\%)}$	$5^{(8.3\%)}$	$9^{(12.9\%)}$	$5^{(6.3\%)}$	$15^{(16.7\%)}$	$7^{(7\%)}$	$12^{(10.9\%)}$	$13^{(10.8\%)}$	$10^{(7.7\%)}$	$20^{(14.3\%)}$	$16^{(10.7\%)}$
Layers: **6**	$4^{(6.7\%)}$	$4^{(5.6\%)}$	$15^{(17.9\%)}$	$8^{(8.3\%)}$	$10^{(9.3\%)}$	$8^{(6.7\%)}$	$12^{(9.1\%)}$	$25^{(17.3\%)}$	$25^{(16.0\%)}$	$31^{(18.5\%)}$	$27^{(15.0\%)}$
Layers: **7**	$5^{(7.1\%)}$	$5^{(6.0\%)}$	$4^{(4.1\%)}$	$10^{(8.9\%)}$	$9^{(7.1\%)}$	$2^{(1.4\%)}$	$6^{(3.9\%)}$	$10^{(6.0\%)}$	$7^{(3.9\%)}$	$5^{(2.6\%)}$	$19^{(9.1\%)}$
Layers: **8**	$4^{(5.0\%)}$	$9^{(9.4\%)}$	$4^{(3.6\%)}$	$14^{(10.9\%)}$	$7^{(4.9\%)}$	$11^{(6.9\%)}$	$22^{(12.5\%)}$	$21^{(10.9\%)}$	$8^{(3.9\%)}$	$6^{(2.7\%)}$	$8^{(3.3\%)}$
Layers: **9**	$2^{(2.2\%)}$	$1^{(0.9\%)}$	$10^{(7.9\%)}$	$2^{(1.4\%)}$	$8^{(4.9\%)}$	$9^{(5.0\%)}$	$21^{(10.6\%)}$	$10^{(4.6\%)}$	$11^{(4.7\%)}$	$8^{(3.2\%)}$	$10^{(3.7\%)}$
Layers: **10**	$3^{(3\%)}$	$12^{(10.0\%)}$	$7^{(5.0\%)}$	$18^{(11.3\%)}$	$11^{(6.1\%)}$	$30^{(15.0\%)}$	$8^{(3.6\%)}$	$4^{(1.7\%)}$	$6^{(2.3\%)}$	$24^{(8.6\%)}$	$18^{(6.0\%)}$
Layers: **11**	$29^{(26.4\%)}$	$11^{(8.3\%)}$	$14^{(9.1\%)}$	$4^{(2.3\%)}$	$2^{(1.0\%)}$	$11^{(5.0\%)}$	$5^{(2.1\%)}$	$5^{(1.9\%)}$	$5^{(1.8\%)}$	$17^{(5.5\%)}$	$18^{(5.5\%)}$
Layers: **12**	$13^{(10.8\%)}$	$11^{(7.6\%)}$	$6^{(3.6\%)}$	$21^{(10.9\%)}$	$26^{(12.0\%)}$	$4^{(1.7\%)}$	$12^{(4.6\%)}$	$13^{(4.5\%)}$	$10^{(3.2\%)}$	$8^{(2.4\%)}$	$4^{(1.11\%)}$
Layers: **13**	$8^{(6.2\%)}$	$13^{(8.3\%)}$	$21^{(11.5\%)}$	$13^{(6.3\%)}$	$14^{(6.0\%)}$	$8^{(3.1\%)}$	$23^{(8.0\%)}$	$15^{(4.8\%)}$	$20^{(5.9\%)}$	$32^{(8.8\%)}$	$12^{(3.1\%)}$
Layers: **14**	$4^{(2.9\%)}$	$7^{(4.2\%)}$	$22^{(11.2\%)}$	$14^{(6.3\%)}$	$4^{(1.6\%)}$	$12^{(4.3\%)}$	$41^{(13.3\%)}$	$13^{(3.9\%)}$	$9^{(2.5\%)}$	$15^{(3.8\%)}$	$24^{(5.7\%)}$
Layers: **15**	$15^{(10.0\%)}$	$15^{(8.3\%)}$	$15^{(7.1\%)}$	$6^{(2.5\%)}$	$15^{(5.6\%)}$	$7^{(2.3\%)}$	$13^{(3.9\%)}$	$8^{(2.2\%)}$	$9^{(2.3\%)}$	$30^{(7.1\%)}$	$9^{(2.0\%)}$
Final Energy Comparison: $\Delta E = E_{\text{random Gaussian}} - E_{\text{baseline Gaussian}}$											
Layers: **5**	$0.184^{(2.6\%)}$	$0.153^{(1.9\%)}$	$0.099^{(1.0\%)}$	$0.042^{(0.4\%)}$	$0.118^{(1.0\%)}$	$0.122^{(0.9\%)}$	$0.026^{(0.3\%)}$	$0.089^{(0.5\%)}$	$0.380^{(2.0\%)}$	$0.615^{(3.0\%)}$	$0.466^{(2.1\%)}$
Layers: **6**	$0.106^{(1.5\%)}$	$0.160^{(1.9\%)}$	$0.107^{(1.0\%)}$	$0.139^{(1.2\%)}$	$0.103^{(0.7\%)}$	$0.090^{(0.6\%)}$	$0.126^{(0.7\%)}$	$0.094^{(0.5\%)}$	$0.147^{(0.7\%)}$	$0.460^{(2.2\%)}$	$0.216^{(1.0\%)}$
Layers: **7**	$0.118^{(1.7\%)}$	$0.004^{(0.0\%)}$	$0.047^{(0.5\%)}$	$0.083^{(0.8\%)}$	$0.027^{(0.2\%)}$	$0.025^{(0.2\%)}$	$0.159^{(1.5\%)}$	$0.029^{(0.2\%)}$	$0.163^{(0.8\%)}$	$0.255^{(1.2\%)}$	$0.375^{(1.7\%)}$
Layers: **8**	$0.041^{(0.6\%)}$	$0.123^{(1.4\%)}$	$0.022^{(0.2\%)}$	$0.066^{(0.6\%)}$	$0.026^{(0.2\%)}$	$0.018^{(0.1\%)}$	$0.047^{(0.3\%)}$	$0.075^{(0.4\%)}$	$0.222^{(1.1\%)}$	$0.025^{(0.1\%)}$	$0.173^{(0.8\%)}$
Layers: **9**	$0.052^{(0.7\%)}$	$0.005^{(0.1\%)}$	$0.095^{(1.0\%)}$	$0.072^{(0.7\%)}$	$0.010^{(0.1\%)}$	$-0.069^{(0.5\%)}$	$0.049^{(0.3\%)}$	$0.051^{(0.3\%)}$	$0.223^{(1.1\%)}$	$0.042^{(0.2\%)}$	$0.037^{(0.2\%)}$
Layers: **10**	$0.113^{(1.5\%)}$	$0.022^{(0.3\%)}$	$0.003^{(0.0\%)}$	$0.032^{(0.3\%)}$	$0.015^{(0.1\%)}$	$0.015^{(0.1\%)}$	$-0.068^{(0.4\%)}$	$0.041^{(0.2\%)}$	$-0.097^{(0.5\%)}$	$0.046^{(0.2\%)}$	$0.268^{(1.2\%)}$
Layers: **11**	$0.237^{(3.2\%)}$	$-0.064^{(0.7\%)}$	$0.050^{(0.5\%)}$	$0.029^{(0.3\%)}$	$-0.175^{(1.4\%)}$	$0.078^{(0.5\%)}$	$0.053^{(0.4\%)}$	$0.152^{(0.9\%)}$	$0.155^{(0.8\%)}$	$0.221^{(1.0\%)}$	$0.006^{(0.0\%)}$
Layers: **12**	$0.070^{(0.9\%)}$	$0.080^{(0.9\%)}$	$-0.096^{(1.0\%)}$	$0.004^{(0.0\%)}$	$0.010^{(0.1\%)}$	$0.033^{(0.2\%)}$	$0.140^{(0.8\%)}$	$0.099^{(0.6\%)}$	$-0.215^{(1.0\%)}$	$0.237^{(1.0\%)}$	$0.237^{(1.0\%)}$
Layers: **13**	$0.010^{(0.1\%)}$	$0.030^{(0.4\%)}$	$0.048^{(0.5\%)}$	$0.022^{(0.2\%)}$	$-0.112^{(0.9\%)}$	$0.043^{(0.3\%)}$	$-0.071^{(0.4\%)}$	$0.055^{(0.3\%)}$	$0.188^{(0.9\%)}$	$-0.194^{(0.8\%)}$	$0.144^{(0.6\%)}$
Layers: **14**	$-0.045^{(0.6\%)}$	$-0.058^{(0.7\%)}$	$0.027^{(0.3\%)}$	$-0.021^{(0.2\%)}$	$0.003^{(0.0\%)}$	$0.007^{(0.0\%)}$	$0.044^{(0.3\%)}$	$-0.110^{(0.6\%)}$	$-0.134^{(0.6\%)}$	$0.081^{(0.4\%)}$	$0.228^{(1.0\%)}$
Layers: **15**	$0.026^{(0.3\%)}$	$0.089^{(1.0\%)}$	$-0.159^{(1.5\%)}$	$0.029^{(0.2\%)}$	$-0.184^{(1.3\%)}$	$0.047^{(0.3\%)}$	$-0.098^{(0.6\%)}$	$0.254^{(1.4\%)}$	$0.008^{(0.0\%)}$	$0.089^{(0.4\%)}$	$0.038^{(0.2\%)}$

Layers \ Qubit	**Qubit**: 5	**Qubit**: 6	**Qubit**: 7	**Qubit**: 8	**Qubit**: 9	**Qubit**: 10	**Qubit**: 11	**Qubit**: 12	**Qubit**: 13	**Qubit**: 14	**Qubit**: 15
Threshold:70											
Layers: **5**	$13^{26.0\%}$	$10^{16.7\%}$	$17^{24.3\%}$	$27^{33.8\%}$	$30^{33.3\%}$	$17^{17.0\%}$	$17^{15.5\%}$	$26^{21.7\%}$	$21^{16.2\%}$	$33^{23.6\%}$	$35^{23.3\%}$
Layers: **6**	$12^{20.0\%}$	$9^{12.5\%}$	$26^{31.0\%}$	$16^{16.7\%}$	$24^{22.2\%}$	$24^{20.0\%}$	$25^{18.9\%}$	$45^{31.3\%}$	$54^{34.6\%}$	$59^{35.1\%}$	$55^{30.6\%}$
Layers: **7**	$11^{15.7\%}$	$9^{10.7\%}$	$10^{10.2\%}$	$18^{16.1\%}$	$17^{13.5\%}$	$7^{5.0\%}$	$8^{5.2\%}$	$18^{10.7\%}$	$21^{11.5\%}$	$11^{5.6\%}$	$32^{15.2\%}$
Layers: **8**	$5^{6.3\%}$	$12^{12.5\%}$	$5^{4.5\%}$	$29^{22.7\%}$	$14^{9.7\%}$	$30^{18.8\%}$	$57^{32.4\%}$	$42^{21.9\%}$	$19^{9.1\%}$	$20^{8.9\%}$	$27^{11.3\%}$
Layers: **9**	$8^{8.9\%}$	$4^{3.7\%}$	$22^{17.5\%}$	$4^{2.8\%}$	$16^{9.9\%}$	$23^{12.8\%}$	$44^{22.2\%}$	$19^{8.8\%}$	$25^{10.7\%}$	$25^{9.9\%}$	$20^{7.4\%}$
Layers: **10**	$5^{5.0\%}$	$27^{22.5\%}$	$11^{7.9\%}$	$39^{24.4\%}$	$30^{16.7\%}$	$60^{30.0\%}$	$26^{11.8\%}$	$8^{3.3\%}$	$7^{2.7\%}$	$36^{12.9\%}$	$38^{12.7\%}$
Layers: **11**	$48^{43.6\%}$	$16^{12.1\%}$	$34^{22.1\%}$	$13^{7.39\%}$	$7^{3.5\%}$	$31^{14.1\%}$	$21^{8.7\%}$	$33^{12.5\%}$	$58^{20.3\%}$	$45^{14.6\%}$	$41^{12.4\%}$
Layers: **12**	$18^{15.0\%}$	$30^{20.8\%}$	$15^{8.9\%}$	$46^{24.0\%}$	$58^{26.9\%}$	$8^{3.3\%}$	$13^{4.9\%}$	$26^{9.0\%}$	$41^{13.1\%}$	$20^{6.0\%}$	$16^{4.4\%}$
Layers: **13**	$13^{10.0\%}$	$24^{15.4\%}$	$41^{22.5\%}$	$29^{13.9\%}$	$38^{16.2\%}$	$18^{6.9\%}$	$58^{20.3\%}$	$41^{13.1\%}$	$42^{12.4\%}$	$87^{23.9\%}$	$29^{7.4\%}$
Layers: **14**	$7^{5.0\%}$	$12^{7.1\%}$	$49^{25.0\%}$	$30^{13.4\%}$	$4^{1.6\%}$	$30^{10.7\%}$	$71^{23.1\%}$	$32^{9.5\%}$	$48^{13.2\%}$	$45^{11.5\%}$	$75^{17.9\%}$
Layers: **15**	$35^{23.3\%}$	$28^{15.6\%}$	$38^{18.1\%}$	$31^{12.9\%}$	$30^{11.1\%}$	$23^{7.7\%}$	$29^{8.8\%}$	$16^{4.4\%}$	$30^{7.7\%}$	$72^{17.1\%}$	$36^{8.0\%}$
Threshold:80											
Layers: **5**	$7^{(14.0\%)}$	$5^{(8.3\%)}$	$9^{(12.9\%)}$	$5^{(6.3\%)}$	$15^{(16.7\%)}$	$7^{(7\%)}$	$12^{(10.9\%)}$	$13^{(10.8\%)}$	$10^{(7.7\%)}$	$20^{(14.3\%)}$	$16^{(10.7\%)}$
Layers: **6**	$4^{(6.7\%)}$	$4^{(5.6\%)}$	$15^{(17.9\%)}$	$8^{(8.3\%)}$	$10^{(9.3\%)}$	$8^{(6.7\%)}$	$12^{(9.1\%)}$	$25^{(17.3\%)}$	$25^{(16.0\%)}$	$31^{(18.5\%)}$	$27^{(15.0\%)}$
Layers: **7**	$5^{(7.1\%)}$	$5^{(6.0\%)}$	$4^{(4.1\%)}$	$10^{(8.9\%)}$	$9^{(7.1\%)}$	$2^{(1.4\%)}$	$6^{(3.9\%)}$	$10^{(6.0\%)}$	$7^{(3.9\%)}$	$5^{(2.6\%)}$	$19^{(9.1\%)}$
Layers: **8**	$4^{(5.0\%)}$	$9^{(9.4\%)}$	$4^{(3.6\%)}$	$14^{(10.9\%)}$	$7^{(4.9\%)}$	$11^{(6.9\%)}$	$22^{(12.5\%)}$	$21^{(10.9\%)}$	$8^{(3.9\%)}$	$6^{(2.7\%)}$	$8^{(3.3\%)}$
Layers: **9**	$2^{(2.2\%)}$	$1^{(0.9\%)}$	$10^{(7.9\%)}$	$2^{(1.4\%)}$	$8^{(4.9\%)}$	$9^{(5.0\%)}$	$21^{(10.6\%)}$	$10^{(4.6\%)}$	$11^{(4.7\%)}$	$8^{(3.2\%)}$	$10^{(3.7\%)}$
Layers: **10**	$3^{(3\%)}$	$12^{(10.0\%)}$	$7^{(5.0\%)}$	$18^{(11.3\%)}$	$11^{(6.1\%)}$	$30^{(15.0\%)}$	$8^{(3.6\%)}$	$4^{(1.7\%)}$	$6^{(2.3\%)}$	$24^{(8.6\%)}$	$18^{(6.0\%)}$
Layers: **11**	$29^{(26.4\%)}$	$11^{(8.3\%)}$	$14^{(9.1\%)}$	$4^{(2.3\%)}$	$2^{(1.0\%)}$	$11^{(5.0\%)}$	$5^{(2.1\%)}$	$5^{(1.9\%)}$	$5^{(1.8\%)}$	$17^{(5.5\%)}$	$18^{(5.5\%)}$
Layers: **12**	$13^{(10.8\%)}$	$11^{(7.6\%)}$	$6^{(3.6\%)}$	$21^{(10.9\%)}$	$26^{(12.0\%)}$	$4^{(1.7\%)}$	$12^{(4.6\%)}$	$13^{(4.5\%)}$	$10^{(3.2\%)}$	$8^{(2.4\%)}$	$4^{(1.11\%)}$
Layers: **13**	$8^{(6.2\%)}$	$13^{(8.3\%)}$	$21^{(11.5\%)}$	$13^{(6.3\%)}$	$14^{(6.0\%)}$	$8^{(3.1\%)}$	$23^{(8.0\%)}$	$15^{(4.8\%)}$	$20^{(5.9\%)}$	$32^{(8.8\%)}$	$12^{(3.1\%)}$
Layers: **14**	$4^{(2.9\%)}$	$7^{(4.2\%)}$	$22^{(11.2\%)}$	$14^{(6.3\%)}$	$4^{(1.6\%)}$	$12^{(4.3\%)}$	$41^{(13.3\%)}$	$13^{(3.9\%)}$	$9^{(2.5\%)}$	$15^{(3.8\%)}$	$24^{(5.7\%)}$
Layers: **15**	$15^{(10.0\%)}$	$15^{(8.3\%)}$	$15^{(7.1\%)}$	$6^{(2.5\%)}$	$15^{(5.6\%)}$	$7^{(2.3\%)}$	$13^{(3.9\%)}$	$8^{(2.2\%)}$	$9^{(2.3\%)}$	$30^{(7.1\%)}$	$9^{(2.0\%)}$

Table 4: Number of frozen parameters predicted by TITAN on HEA based Isotropic Hamiltonian solver. Frozen parameters exceeding 30% is colored with red and exceeding between 10% and 30% is colored with blue.

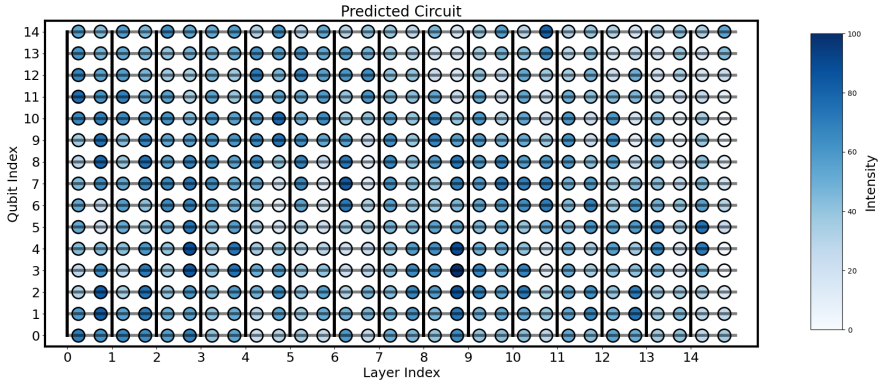


(a)

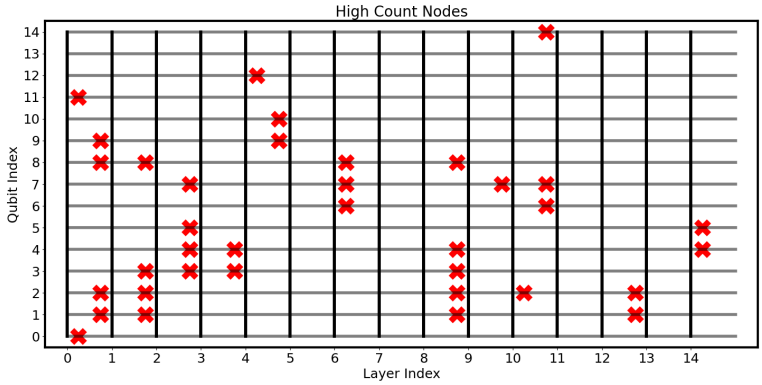


(b)

Figure 11: APFA records parameters trajectory with HEA-based isotropic Hamiltonian solver. (a) Freeze Counts Intensity: In the figure (a) above, the color of the dot indicates the number of times the parameter was "frozen" (freeze counts) during the optimization process. The darker the color, the more times the parameter was frozen; the lighter the color, the fewer times it was frozen. The thin black line on the outer edge of each dot is only used to emphasize the position of different parameters and has nothing to do with intensity. The color scale bar on the right indicates the value range and mapping relationship. (b) Active Counts Intensity: In the figure (b) below, the color of the dot indicates the number of times the parameter was "activated" (active counts) during the training process. Similarly, the darker the color, the more times it was activated; the lighter the color, the fewer times it was activated. Other marking methods are the same as in (a).

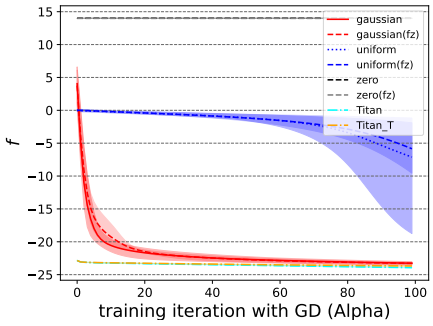


(a)

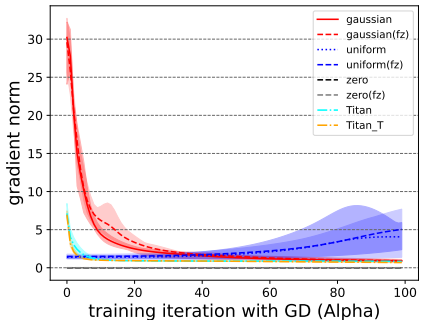


(b)

Figure 12: (a) TITAN predicted intensity. (b) TITAN selected Frozen Gates (**36/450**) (8%). Qubits 15, Layers 15, Threshold: 70. HEA-based isotropic Hamiltonian solver: XX + YY + ZZ.

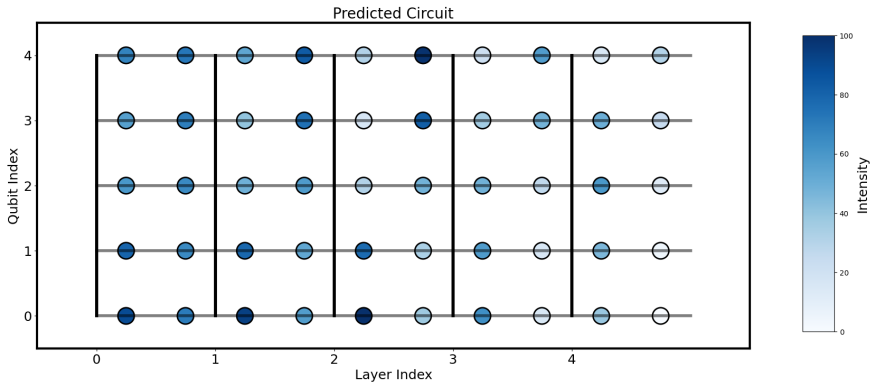


(a)

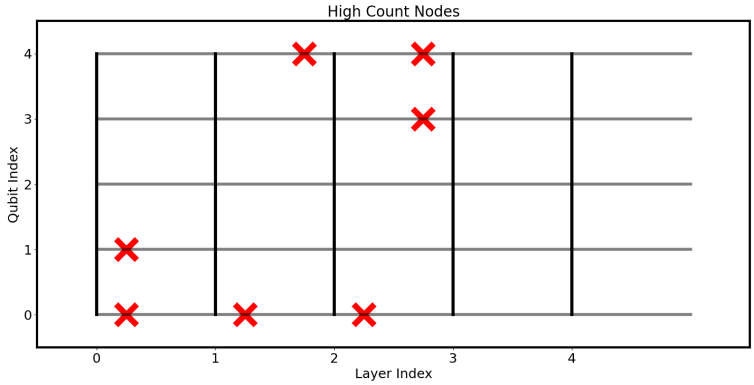


(b)

Figure 13: (a) Loss GD Noiseless. (b) Gradient Norm GD Noiseless. Qubits 15, Layers 15, Threshold: 70. Selected Frozen Gates (**36/450**) (8%). HEA-based isotropic Hamiltonian solver: XX + YY + ZZ. TITAN: Enhanced Gaussian. TITAN_T: APFA Only without freezing.

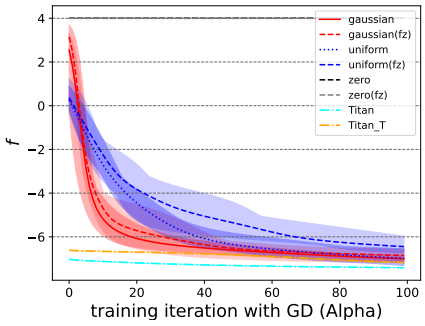


(a)

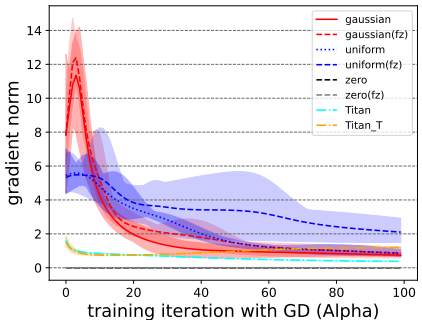


(b)

Figure 14: (a) TITAN predicted intensity. (b) TITAN selected Frozen Gates (**7/50**) (14%). Qubits 5, Layers 5, Threshold: $\tau = 80$. HEA-based isotropic Hamiltonian solver: XX + YY + ZZ.

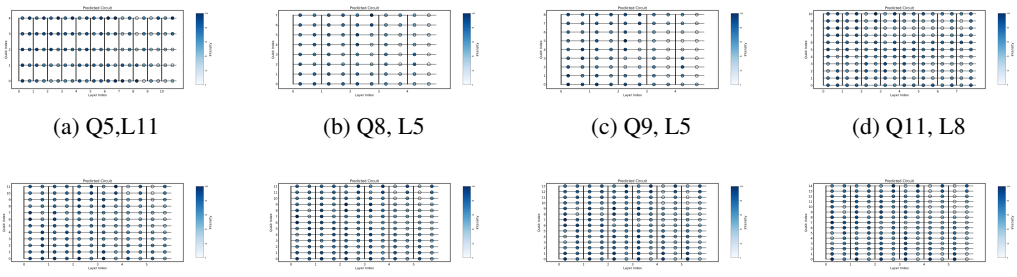


(a)



(b)

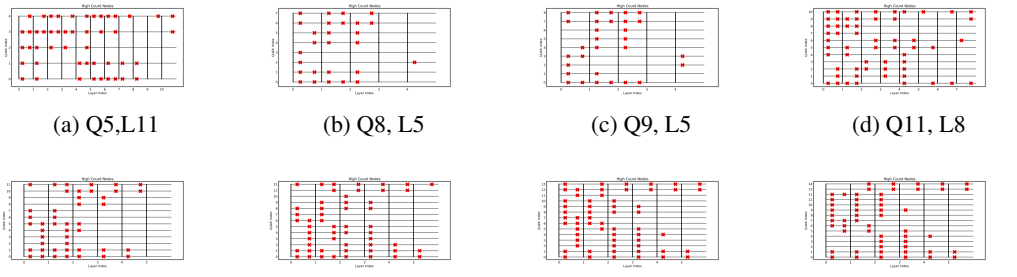
Figure 15: (a) Loss GD Noiseless. (b) Gradient Norm GD Noiseless. Qubits 5, Layers 5, Threshold: $\tau = 80$. Selected Frozen Gates (**7/50**) (14%). HEA-based isotropic Hamiltonian solver: XX + YY + ZZ. TITAN: Enhanced Gaussian. TITAN_T: APFA Only without freezing.



(a) Q5,L11 (b) Q8, L5 (c) Q9, L5 (d) Q11, L8

(e) Q12, L6 (f) Q13, L6 (g) Q14, L6 (h) Q15, L6

Figure 16: TITAN predicted freezing intensity for HEA-based isotropic Hamiltonian solver with threshold: 70.



(a) Q5,L11 (b) Q8, L5 (c) Q9, L5 (d) Q11, L8

(e) Q12, L6 (f) Q13, L6 (g) Q14, L6 (h) Q15, L6

Figure 17: TITAN selected frozen Parameters for HEA-based isotropic Hamiltonian solver with threshold: 70.

Layers \ Qubit	**Qubit**: 5	**Qubit**: 6	**Qubit**: 7	**Qubit**: 8	**Qubit**: 9	**Qubit**: 10	**Qubit**: 11	**Qubit**: 12	**Qubit**: 13	**Qubit**: 14	**Qubit**: 15
Threshold:50											
Layers: **5**	$26^{52.0\%}$	$33^{55.0\%}$	$42^{60.0\%}$	$55^{68.8\%}$	$60^{66.7\%}$	$50^{50.0\%}$	$48^{43.6\%}$	$52^{43.3\%}$	$47^{36.2\%}$	$70^{50.0\%}$	$88^{58.7\%}$
Layers: **6**	$25^{41.7\%}$	$26^{36.1\%}$	$46^{54.8\%}$	$46^{47.9\%}$	$56^{51.9\%}$	$54^{45.0\%}$	$60^{45.5\%}$	$107^{74.3\%}$	$111^{71.2\%}$	$118^{70.2\%}$	$117^{65.0\%}$
Layers: **7**	$31^{44.3\%}$	$37^{44.1\%}$	$29^{29.6\%}$	$40^{35.7\%}$	$45^{35.7\%}$	$44^{31.4\%}$	$34^{22.1\%}$	$57^{33.9\%}$	$67^{36.8\%}$	$48^{24.5\%}$	$97^{46.2\%}$
Layers: **8**	$34^{42.5\%}$	$34^{35.4\%}$	$39^{34.8\%}$	$82^{64.1\%}$	$61^{42.4\%}$	$81^{50.6\%}$	$122^{69.3\%}$	$101^{52.6\%}$	$78^{37.5\%}$	$96^{42.9\%}$	$128^{53.3\%}$
Layers: **9**	$40^{44.4\%}$	$32^{29.6\%}$	$58^{46.0\%}$	$51^{35.4\%}$	$59^{36.4\%}$	$79^{43.9\%}$	$114^{57.6\%}$	$71^{32.9\%}$	$111^{47.4\%}$	$97^{38.5\%}$	$93^{34.4\%}$
Layers: **10**	$40^{40.0\%}$	$66^{55.0\%}$	$50^{35.7\%}$	$83^{51.9\%}$	$78^{43.3\%}$	$148^{74.0\%}$	$93^{42.3\%}$	$51^{21.3\%}$	$51^{19.6\%}$	$105^{37.5\%}$	$163^{54.3\%}$
Layers: **11**	$84^{76.4\%}$	$39^{29.6\%}$	$85^{55.2\%}$	$62^{35.2\%}$	$40^{20.2\%}$	$96^{43.6\%}$	$103^{42.6\%}$	$81^{30.7\%}$	$80^{28.0\%}$	$145^{47.1\%}$	$165^{50.0\%}$
Layers: **12**	$51^{42.5\%}$	$69^{47.9\%}$	$55^{32.7\%}$	$110^{57.3\%}$	$131^{60.7\%}$	$71^{29.6\%}$	$124^{47.0\%}$	$98^{34.0\%}$	$133^{42.6\%}$	$115^{34.2\%}$	$125^{34.7\%}$
Layers: **13**	$49^{37.7\%}$	$80^{51.3\%}$	$119^{65.4\%}$	$77^{37.0\%}$	$118^{50.4\%}$	$79^{30.4\%}$	$147^{51.4\%}$	$154^{49.4\%}$	$150^{44.4\%}$	$222^{61.0\%}$	$125^{32.1\%}$
Layers: **14**	$58^{41.4\%}$	$38^{22.6\%}$	$120^{61.2\%}$	$92^{41.1\%}$	$68^{27.0\%}$	$111^{39.6\%}$	$168^{54.6\%}$	$142^{42.3\%}$	$208^{57.1\%}$	$187^{47.7\%}$	$231^{55.0\%}$
Layers: **15**	$103^{68.7\%}$	$88^{48.9\%}$	$96^{45.7\%}$	$90^{37.5\%}$	$87^{32.2\%}$	$96^{32.0\%}$	$139^{42.1\%}$	$97^{26.9\%}$	$144^{36.9\%}$	$211^{50.2\%}$	$206^{45.8\%}$
Threshold:60											
Layers: **5**	$18^{36.0\%}$	$21^{35.0\%}$	$28^{40.0\%}$	$41^{51.3\%}$	$42^{46.7\%}$	$33^{33.0\%}$	$33^{30.0\%}$	$36^{30.0\%}$	$27^{20.8\%}$	$44^{31.4\%}$	$48^{32.0\%}$
Layers: **6**	$16^{26.7\%}$	$19^{26.4\%}$	$36^{42.9\%}$	$27^{28.1\%}$	$39^{36.1\%}$	$38^{31.7\%}$	$44^{33.3\%}$	$76^{52.8\%}$	$84^{53.8\%}$	$92^{54.8\%}$	$86^{47.8\%}$
Layers: **7**	$19^{27.1\%}$	$21^{25.0\%}$	$18^{18.4\%}$	$30^{26.8\%}$	$29^{23.0\%}$	$23^{16.4\%}$	$20^{13.0\%}$	$35^{20.8\%}$	$36^{19.8\%}$	$22^{11.2\%}$	$60^{28.6\%}$
Layers: **8**	$15^{18.8\%}$	$18^{18.8\%}$	$15^{13.4\%}$	$54^{42.2\%}$	$31^{21.5\%}$	$60^{37.5\%}$	$95^{54.0\%}$	$65^{33.9\%}$	$43^{20.7\%}$	$45^{20.1\%}$	$72^{30.0\%}$
Layers: **9**	$24^{26.7\%}$	$15^{13.9\%}$	$36^{28.6\%}$	$20^{13.9\%}$	$32^{19.8\%}$	$49^{27.2\%}$	$76^{38.4\%}$	$34^{15.7\%}$	$60^{25.6\%}$	$51^{20.2\%}$	$48^{17.8\%}$
Layers: **10**	$22^{22.0\%}$	$48^{40.0\%}$	$24^{17.1\%}$	$65^{40.6\%}$	$53^{29.4\%}$	$108^{54.0\%}$	$52^{23.6\%}$	$29^{12.1\%}$	$20^{7.7\%}$	$63^{22.5\%}$	$85^{28.3\%}$
Layers: **11**	$72^{65.5\%}$	$21^{15.9\%}$	$61^{39.6\%}$	$38^{21.6\%}$	$21^{10.6\%}$	$57^{25.9\%}$	$55^{22.7\%}$	$33^{12.5\%}$	$29^{10.1\%}$	$82^{26.6\%}$	$95^{28.8\%}$
Layers: **12**	$31^{25.8\%}$	$52^{36.1\%}$	$32^{19.1\%}$	$81^{42.2\%}$	$92^{42.6\%}$	$24^{10.0\%}$	$87^{33.0\%}$	$47^{16.3\%}$	$75^{24.0\%}$	$51^{15.2\%}$	$59^{16.4\%}$
Layers: **13**	$29^{22.3\%}$	$47^{30.1\%}$	$80^{44.0\%}$	$46^{22.1\%}$	$80^{34.2\%}$	$35^{13.5\%}$	$107^{37.4\%}$	$82^{26.3\%}$	$90^{26.6\%}$	$143^{39.3\%}$	$58^{14.9\%}$
Layers: **14**	$29^{20.7\%}$	$24^{14.3\%}$	$86^{43.9\%}$	$62^{27.7\%}$	$22^{8.7\%}$	$61^{21.8\%}$	$120^{39.0\%}$	$79^{23.5\%}$	$106^{29.1\%}$	$107^{27.3\%}$	$140^{33.3\%}$
Layers: **15**	$67^{44.7\%}$	$52^{28.9\%}$	$63^{30.0\%}$	$58^{24.2\%}$	$55^{20.4\%}$	$59^{19.7\%}$	$76^{23.0\%}$	$44^{12.2\%}$	$76^{19.5\%}$	$135^{32.1\%}$	$106^{23.6\%}$

Table 5: TITAN predicted frozen parameter with HEA-based isotropic Hamiltonian solver. Frozen parameters exceeding 30% is colored with red and exceeding between 10% and 30% is colored with blue and exceeding 60% is colored with darker red.

F.4 Limitations

All empirical evidence is obtained on state–vector *classical* simulators. As a consequence, noise processes are injected only synthetically (Gaussian perturbations to gradients) rather than captured from *in situ* device characterization. The absence of crosstalk, SPAM errors, and non-Markovian decoherence. Scalability of APFA thresholds and EMA hyperparameters beyond this regime remains untested. This study remains limited to classical simulators and benchmark circuits do not exceed 100 qubits, however, in the future, we plan to scale to deeper circuits in real-world quantum sensors, ultimately enabling resource-efficient VQE deployments for chemically and physically relevant systems.

References

[1] Bujiao Wu, Jinzhao Sun, Qi Huang, and Xiao Yuan. Overlapped grouping measurement: A unified framework for measuring quantum states. *Quantum*, 7:896, 2023.

[2] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, 16(10):1050–1057, 2020.

[3] Ho Lun Tang, VO Shkolnikov, George S Barron, Harper R Grimsley, Nicholas J Mayhall, Edwin Barnes, and Sophia E Economou. qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor. *PRX Quantum*, 2(2):020310, 2021.

[4] Min Li, Mao Lin, and Matthew JS Beach. Resource-optimized grouping shadow for efficient energy estimation. *Quantum*, 9:1694, 2025.

[5] Guillermo García-Pérez, Matteo AC Rossi, Boris Sokolov, Francesco Tacchino, Panagiotis Kl Barkoutsos, Guglielmo Mazzola, Ivano Tavernelli, and Sabrina Maniscalco. Learning to measure: Adaptive informationally complete generalized measurements for quantum algorithms. *Prx quantum*, 2(4):040342, 2021.

[6] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *nature*, 549(7671):242–246, 2017.

[7] Francisco JR Ruiz, Tuomas Laakkonen, Johannes Bausch, Matej Balog, Mohammadamin Barekatain, Francisco JH Heras, Alexander Novikov, Nathan Fitzpatrick, Bernardino Romera-Paredes, John van de Wetering, et al. Quantum circuit optimization with alphatensor. *Nature Machine Intelligence*, pages 1–12, 2025.

[8] Harper R Grimsley, Sophia E Economou, Edwin Barnes, and Nicholas J Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature communications*, 10(1):3007, 2019.

[9] Yuxuan Du, Tao Huang, Shan You, Min-Hsiu Hsieh, and Dacheng Tao. Quantum circuit architecture search for variational quantum algorithms. *npj Quantum Information*, 8(1):62, 2022.

[10] Weiwei Zhu, Jiangtao Pi, and Qiuyuan Peng. A brief survey of quantum architecture search. In *Proceedings of the 6th international conference on algorithms, computing and systems*, pages 1–5, 2022.

[11] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[12] Ankit Kulshrestha, Xiaoyuan Liu, Hayato Ushijima-Mwesigwa, Bao Bach, and Ilya Safro. 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, 2024.

[13] Nonia Vaquero-Sabater, Abel Carreras, and David Casanova. Pruned-adapt-vqe: compacting molecular ans\" atze by removing irrelevant operators. *arXiv preprint arXiv:2504.04652*, 2025.

[14] David Fitzek, Robert S Jonsson, Werner Dobrautz, and Christian Schäfer. Optimizing variational quantum algorithms with qbang: Efficiently interweaving metric and momentum to navigate flat energy landscapes. *Quantum*, 8:1313, 2024.

[15] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[16] Di Luo, Jiayu Shen, Rumen Dangovski, and Marin Soljacic. Quack: accelerating gradient-based quantum optimization with koopman operator learning. *Advances in Neural Information Processing Systems*, 36:25662–25692, 2023.

[17] Jeihee Cho, Junyong Lee, Daniel Justice, and Shiho Kim. Enhancing circuit trainability with selective gate activation strategy, 2025.

[18] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

[19] Marco Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature communications*, 12(1):1791, 2021.

[20] Kaining Zhang, Liu Liu, Min-Hsiu Hsieh, and Dacheng Tao. Escaping from the barren plateau via gaussian initializations in deep variational quantum circuits. *Advances in Neural Information Processing Systems*, 35:18612–18627, 2022.