LIEQAS: DIFFERENTIABLE QUANTUM ARCHITECTURE SEARCH VIA CONTINUOUS LIE-ALGEBRAIC PARAMETERIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantum Architecture Search (QAS) aims to automate variational circuit design. While classical differentiable Neural Architecture Search (NAS) can continuously relax candidate operations, its standard weighted operation-mixing relaxation does not directly transfer to quantum circuits. Existing differentiable QAS methods therefore introduce differentiability over discrete circuit-operation choices through sampling, categorical relaxation, or mixed-state representations. These mechanisms make architecture optimization indirect: sampling yields stochastic gradient estimates, while surrogate or ensemble objectives may mismatch with the discrete circuit ultimately selected. We propose LieQAS, a differentiable QAS framework that instead parameterizes the searched unitary directly in continuous Lie-algebraic coordinates using Pauli strings. Gradients are taken directly with respect to these Pauli-string coefficients, while the architecture is defined by the active Pauli-string support and updated through progressive pruning. We further introduce hardware-cost-aware regularization and non-commutativity-aware search with singleton finalization, so that the finalized architecture admits exact direct extraction. Experiments on molecular VQE benchmarks show that LieQAS achieves chemical accuracy on all four VQE instances while reducing compiled gate count by up to 64% relative to QuantumDARTS, with lower circuit depth and no more CX gates. On MNIST 0/1 classification, LieQAS reduces CX count by up to 89% while maintaining competitive mean accuracy.

1 INTRODUCTION

Recent years have witnessed growing applications of Variational Quantum Algorithms (VQAs), including Quantum Machine Learning (QML) (Hur et al., 2022; Jeswal & Chakraverty, 2019) and Variational Quantum Eigensolvers (VQEs) (Tilly et al., 2022; Kandala et al., 2017). These algorithms rely on parameterized quantum circuits, or quantum ansätze, whose architectures directly affect trainability and hardware feasibility (Cerezo et al., 2021). Existing Ansatz design mainly follows problem-inspired and hardware-efficient approaches. Problem-inspired ansätze such as UCCSD (Romero et al., 2019) incorporate physical priors but often require deep circuits (McCaskey et al., 2019), whereas hardware-efficient ansätze (HEAs) (Kandala et al., 2017) reduce circuit depth but may suffer from barren plateaus (McClean et al., 2018; Cerezo et al., 2021). These limitations make effective quantum circuit architecture design increasingly challenging.

Inspired by Neural Architecture Search (NAS), Quantum Architecture Search (QAS) aims to automate ansatz design by searching over candidate quantum operations and circuit structures (Zhang et al., 2021; 2022; Wu et al., 2023). In differentiable NAS, discrete operation choices are continuously relaxed, for example through weighted operation mixing (Liu et al., 2018), allowing architecture variables and network parameters to be optimized using gradient-based methods (Liu et al., 2018; Wu et al., 2019). Following this paradigm, differentiable QAS introduces gradient-based optimization into quantum architecture search.

However, the standard weighted operation-mixing relaxation used in differentiable NAS cannot be directly transferred to quantum circuits (Wu et al., 2023). Existing differentiable QAS methods therefore retain discrete circuit-operation choices and introduce different mechanisms for gradient-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

based optimization. DQAS (Zhang et al., 2022) estimates architecture gradients via Monte Carlo sampling over discrete circuit architectures, whereas QuantumDARTS (Wu et al., 2023) employs Gumbel-Softmax with one-hot gate selection during the forward pass. Recently, ρDARTS (Kumar et al., 2025) avoids explicit circuit sampling by optimizing a differentiable mixed-state ensemble over discrete gate architectures.

Despite these different formulations, their architecture search remains defined over discrete gate-level candidates, rather than continuous coordinates of the searched unitary itself. The architecture-optimization signal is therefore mediated by a sampled distribution, a differentiable categorical surrogate, or a mixed-state ensemble. Consequently, the resulting optimization signal can be stochastic or indirect, and improvement of the search-time objective need not be preserved exactly after discrete architecture selection. This motivates the following question: *Can differentiable QAS instead parameterize the searched unitary directly in a continuous Lie-algebraic coordinate space, rather than introducing continuous selection variables over a discrete operation space?*

To define the architecture search directly in continuous coordinates, we use the Lie-algebraic representation of unitary evolution (Hall, 2013). Up to a global phase, an n-qubit unitary U can be represented by an element of the Lie group $SU(2^n)$, with corresponding Lie algebra $\mathfrak{su}(2^n)$. Let $\{P_k\}_{k=1}^{4^n-1}$ denote the non-identity n-qubit Pauli strings. The corresponding skew-Hermitian operators $\{-iP_k\}_{k=1}^{4^n-1}$ form a basis of $\mathfrak{su}(2^n)$ (Khaneja & Glaser, 2001). Then an n-qubit unitary can be parameterized as equation 1, where $H(\boldsymbol{\theta}) = \sum_k \theta_k P_k$ and $\theta_k \in \mathbb{R}$ is the continuous coefficient of Pauli string P_k. Thus, $\boldsymbol{\theta}$ directly parameterizes the searched unitary through continuous Pauli-basis coefficients. Because $H(\boldsymbol{\theta})$ is Hermitian for any real $\boldsymbol{\theta}$, $U(\boldsymbol{\theta})$ remains unitary by construction. Moreover, the matrix exponential is differentiable with respect to these coefficients (Wilcox, 1967; Al-Mohy & Higham, 2009), enabling direct gradient-based optimization in this continuous space. **This motivates LieQAS, a differentiable quantum architecture search framework that parameterizes the searched unitary directly in Lie-algebraic coordinates and defines the architecture by the active Pauli-string support.**

$$U(\boldsymbol{\theta}) = \exp\left(-i\sum_k \theta_k P_k\right) = \exp\left(-iH(\boldsymbol{\theta})\right). \tag{1}$$

This Pauli-basis representation introduces two practical challenges. **Challenge 1 (Search):** The Pauli-string candidate pool contains $4^n - 1$ non-identity Pauli strings, so the number of candidate coordinates per layer grows exponentially with the number of qubits. Thus, the candidate pool must be reduced while accounting for the higher hardware cost of two-qubit interactions (Krantz et al., 2019). **Challenge 2 (Realization):** If multiple non-commuting Pauli strings remain active within the same layer, the searched unitary generally cannot be reproduced exactly by applying the corresponding Pauli rotations sequentially (Hall, 2013). The final architecture must therefore support exact gate-level extraction.

LieQAS addresses these challenges through progressive Pauli-string pruning and non-commutativity-aware search and finalization. The main contributions are:

- **Continuous Lie-Algebraic Pauli-Basis Representation**: We parameterize each layer in continuous Lie-algebraic coordinates using Pauli strings, with a binary mask indicating the active Pauli-string support. Restricting the candidate pool to Pauli strings of weight at most two (i.e., acting nontrivially on at most two qubits) reduces the number of candidates per layer from $\mathcal{O}(4^n)$ to $\mathcal{O}(n^2)$ (§3.1).
- **Progressive Architecture Search**: We progressively sparsify the active Pauli-string support using a first-order Taylor saliency criterion, with asymmetric penalties based on Pauli weight as a simple hardware-cost proxy (§3.2).
- **Non-Commutativity-Aware Search and Finalization**: We introduce congestion-adjusted scoring to bias search toward lower layer occupancy and singleton finalization that enforces one active Pauli string per occupied layer to guarantee exact direct extraction (§3.3).

Experiments on molecular VQE benchmarks show that LieQAS achieves chemical accuracy on all four VQE instances while reducing compiled gate count by up to 64% relative to QuantumDARTS, with lower circuit depth and no more CX gates on every instance. On MNIST 0/1 classification, LieQAS reduces CX count by up to 89% while maintaining competitive mean accuracy.

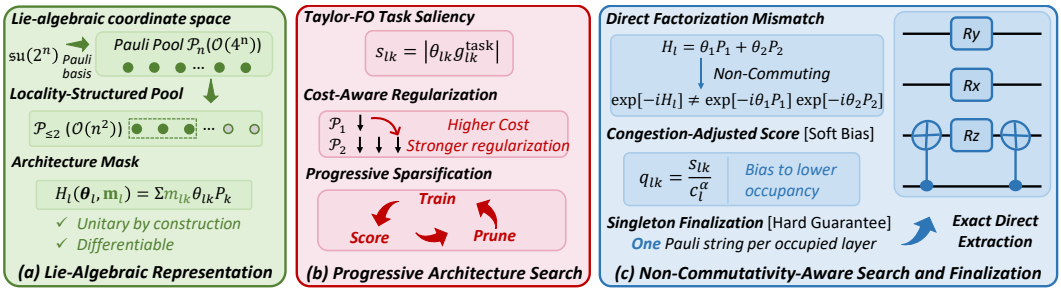

Figure 1: **Overview of LieQAS.** (a) Continuous Lie-algebraic Pauli-basis representation with a $\mathcal{P}_{\leq 2}$ Pauli-string candidate pool. (b) Progressive support sparsification through Taylor-based pruning and hardware-cost-aware regularization. (c) Congestion-adjusted scoring and singleton finalization for exact direct extraction.

2 Related Work

Fixed and Adaptive Ansatz Construction. Fixed ansätze like UCCSD (Romero et al., 2019) and HEA (Kandala et al., 2017) embed physical or hardware priors, but fix the circuit topology prior to optimization. Instead, adaptive schemes iteratively adjust structures, either by dynamically inserting and removing gates (VAns (Bilkis et al., 2023)) or greedily selecting maximal-gradient operators from fermionic or qubit pools (ADAPT-VQE (Grimsley et al., 2019), Qubit-ADAPT-VQE (Tang et al., 2021)). While these adaptive methods can produce compact circuits, their structural updates are still performed through discrete gate or operator insertion, removal, and selection. ADAPT-style methods additionally rely on predefined operator pools.

Discrete Quantum Architecture Search. Discrete QAS directly searches over gate-level circuit structures. Representative strategies include evolutionary search over circuit encodings (Ding & Spector, 2022; Giovagnoli et al., 2023; Zhang & Zhao, 2023; Wang et al., 2022), reinforcement-learning-based search over discrete architecture decisions (Ostaszewski et al., 2021; Patel et al., 2024), and dynamic sparse gate-topology exploration (Chen et al., 2023). Despite their different search strategies, architecture changes are realized as discrete modifications of gate-level circuits.

Differentiable Quantum Architecture Search. Representative differentiable QAS methods introduce continuous variables over architecture spaces defined by discrete circuit-operation choices. DQAS (Zhang et al., 2022) parameterizes a probabilistic model over discrete circuit architectures and estimates architecture gradients through Monte Carlo sampling. QuantumDARTS (Wu et al., 2023) instead uses Gumbel-Softmax, with a one-hot gate selection in the forward pass and a continuous relaxation for gradient back-propagation. More recently, ρDARTS (Kumar et al., 2025) avoids circuit sampling by propagating a differentiable density-matrix mixture over the discrete architecture space. Across these methods, continuous architecture variables govern distributions or relaxations over predefined circuit-operation choices. LieQAS instead directly parameterizes the searched unitary with continuous Pauli-string coefficients and searches over their active support.

Lie-Algebraic Formulations and Circuit Pruning. Lie-algebraic tools have been used for controllability and unitary synthesis in quantum control (Khaneja & Glaser, 2001), trainability analysis in variational quantum algorithms (Larocca et al., 2022), and parameterized-circuit optimization and synthesis (Goh et al., 2025). These works use Lie algebra as a useful representation for quantum transformations, but do not search the active Pauli-string support as circuit architecture. In parallel, structural pruning ranks parameters or gates according to their estimated effect on the objective. Classical methods use relevance and curvature information (Mozer & Smolensky, 1988; LeCun et al., 1989). In quantum circuits, QuantumSEA (Chen et al., 2023) uses gate saliency, while QAdaPrune (Kulshrestha et al., 2024) uses gradient-variation information as a curvature proxy for parameter pruning. LieQAS connects these directions by continuously optimizing Pauli-string coefficients while progressively pruning the active Pauli-string support as the searched architecture.

3 METHODOLOGY

Figure 1 summarizes three components of LieQAS. First, **Lie-Algebraic Pauli-Basis Representation** (§3.1) represents each layer through continuous coordinates over the Pauli basis, with architecture defined by the active Pauli-string support, and constructs a candidate pool with Pauli strings of weight at most two. Second, **Progressive Architecture Search** (§3.2) progressively sparsifies the Pauli-string support through pruning while hardware-cost-aware regularization biases the search toward lower-cost Pauli strings. Third, **Non-Commutativity-Aware Search and Finalization** (§3.3) biases the search toward lower layer occupancy and enforces singleton occupancy, resolving the direct-factorization mismatch caused by non-commuting Pauli strings within the same layer.

3.1 CONTINUOUS LIE-ALGEBRAIC PAULI-BASIS REPRESENTATION

We first define the Lie-algebraic architecture representation used by LieQAS. For an n-qubit, L-layer circuit $U = U_{L-1} \cdots U_0$, let $\mathcal{P}_n = \{I, X, Y, Z\}^{\otimes n} \setminus \{I^{\otimes n}\}$ denote the non-identity Pauli strings. Their skew-Hermitian counterparts $\{-iP : P \in \mathcal{P}_n\}$ form a basis of $\mathfrak{su}(2^n)$. Equivalently, using the Hermitian Pauli basis, LieQAS parameterizes each layer as equation 2, where $\theta_{lk} \in \mathbb{R}$ is the continuous Pauli-string coefficient and $m_{lk} \in \{0, 1\}$ indicates whether P_k is active. The searched architecture is defined by the active Pauli-string support encoded by the mask, while $\boldsymbol{\theta}$ continuously parameterizes the searched unitary. The mask is not optimized as a continuous architecture variable; it is updated when Pauli strings are removed during progressive pruning.

$$H_l(\boldsymbol{\theta}_l, \mathbf{m}_l) = \sum_{P_k \in \mathcal{P}_n} m_{lk} \theta_{lk} P_k, \qquad U_l = \exp[-iH_l(\boldsymbol{\theta}_l, \mathbf{m}_l)]. \tag{2}$$

Because Pauli strings are Hermitian and their coefficients are real, $H_l(\boldsymbol{\theta}_l, \mathbf{m}_l)$ is Hermitian for any mask configuration, so each layer is unitary by construction. The matrix exponential is differentiable with respect to the active Pauli-string coefficients, allowing $\boldsymbol{\theta}$ to be optimized by gradient-based training (Wilcox, 1967; Al-Mohy & Higham, 2009). We next define the explicit Pauli-string candidate pool used for search.

The complete pool contains $4^n - 1$ non-identity Pauli strings per layer, which grows exponentially with n. Gate-based quantum hardware is typically built from one- and two-qubit operations (Krantz et al., 2019). Motivated by search-space tractability and the prevalence of such primitives, we restrict the explicit search pool to Pauli strings of weight at most two, where the Pauli weight is the number of non-identity factors in a Pauli string (Quek et al., 2024). Specifically, $\mathcal{P}_{\leq 2} = \mathcal{P}_1 \cup \mathcal{P}_2$, where $\mathcal{P}_1$ and $\mathcal{P}_2$ denote the weight-one and weight-two Pauli-string subsets, respectively. With three weight-one Pauli strings per qubit and nine weight-two Pauli strings per qubit pair, the number of searchable coordinates per layer becomes:

$$K = |\mathcal{P}_{\leq 2}| = 3n + 9\binom{n}{2} = \frac{9n^2 - 3n}{2} = \mathcal{O}(n^2). \tag{3}$$

Thus, the number of explicit candidate Pauli strings per layer is reduced from exponential to quadratic scaling. This reduction concerns the number of explicit search coordinates rather than the complexity of dense classical simulation. All subsequent search variables are indexed by $P_k \in \mathcal{P}_{\leq 2}$, with $k = 1, \ldots, K$.

Although the explicit pool contains only weight-one and weight-two Pauli strings, the Lie algebra generated by their skew-Hermitian counterparts is not restricted to weight-two directions. Repeated commutators of these generators produce higher-weight Pauli directions, and $\{-iP : P \in \mathcal{P}_{\leq 2}\}$ generates the full $\mathfrak{su}(2^n)$ Lie algebra (Khaneja & Glaser, 2001; Dodd et al., 2002). Thus, $\mathcal{P}_{\leq 2}$ retains universal controllability while using only $\mathcal{O}(n^2)$ explicit search coordinates. Appendix B.3 evaluates the effect of explicitly including higher-weight Pauli strings.

3.2 PROGRESSIVE ARCHITECTURE SEARCH

Given the Pauli-basis representation and candidate pool defined in §3.1, LieQAS progressively selects which Pauli strings remain active. Architecture updates are pruning decisions informed by gradients with respect to the continuously optimized Pauli-string coefficients.

3.2.1 First-Order Taylor Saliency for Pauli-String Pruning

With the candidate pool fixed, the next question is which active coordinates should be retained in the architecture. Removing Pauli string P_k from layer l changes its mask entry from $m_{lk} = 1$ to $m_{lk} = 0$, which is equivalent to setting its coefficient θ_{lk} to zero. We therefore estimate the importance of an active Pauli string by the task-loss change induced by this deletion. Let $\mathcal{L}_{\text{task}}(\boldsymbol{\theta}, \mathbf{m})$ denote the task-specific objective evaluated on the current architecture. Following first-order Taylor pruning criteria commonly used for structural importance estimation (Molchanov et al., 2019; Chen et al., 2023), we locally approximate the loss change caused by $\Delta\theta_{lk} = -\theta_{lk}$ as:

$$\Delta\mathcal{L}_{\text{task}} \approx -\theta_{lk} g_{lk}^{\text{task}}, \qquad g_{lk}^{\text{task}} = \frac{\partial \mathcal{L}_{\text{task}}}{\partial \theta_{lk}}. \tag{4}$$

We refer to this first-order approximation as *Taylor-FO* and define the corresponding Pauli-string saliency s_{lk} as:

$$s_{lk} = \left| \theta_{lk} g_{lk}^{\text{task}} \right|. \tag{5}$$

A smaller s_{lk} means that deleting the corresponding Pauli string is locally estimated to cause a smaller change in task loss. Taylor-FO therefore provides the task-sensitivity component for ranking candidate deletions during progressive search.

3.2.2 Hardware-Cost-Aware Regularization

Taylor-FO neglects second- and higher-order terms, whereas Pauli-string deletion is a finite perturbation $\Delta\theta_{lk} = -\theta_{lk}$. By Taylor's theorem (Nocedal & Wright, 2006), the first omitted term is second order in the perturbation magnitude and depends on local curvature along the deletion direction. Hence, for comparable curvature, larger $|\theta_{lk}|$ can lead to larger truncation error. We therefore introduce L_1 regularization to shrink active coefficients toward zero (Tibshirani, 1996), moving deletion toward a smaller-perturbation regime. For comparable task gradients, this shrinkage also tends to reduce Taylor-FO saliency, making the corresponding coordinates more likely to be removed.

Rather than penalizing all Pauli strings equally, we further use the regularization term to reflect their different hardware costs. In particular, rotations generated by weight-two Pauli terms generally require entangling operations (Cowtan et al., 2019), while two-qubit operations generally incur greater hardware overhead than single-qubit operations on near-term devices (Krantz et al., 2019). We therefore use Pauli-string weight as a simple hardware-cost proxy and assign a larger regularization penalty to weight-two Pauli strings. The search objective becomes:

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{task}} + \sum_{l=0}^{L-1} \sum_{k=1}^{K} m_{lk} \lambda_k |\theta_{lk}|, \qquad \lambda_k = \begin{cases} \lambda_{1q}, & P_k \in \mathcal{P}_1, \\ \lambda_{2q}, & P_k \in \mathcal{P}_2, \end{cases} \tag{6}$$

where λ_{1q} and λ_{2q} denote the regularization strengths for weight-one and weight-two Pauli strings and $\lambda_{2q} > \lambda_{1q} \geq 0$. We use the task-loss gradient rather than the regularized training gradient to calculate saliency, so that it reflects only task sensitivity. The regularizer serves two roles: coefficient shrinkage moves Pauli-string deletion toward the regime in which the first-order approximation is expected to be more accurate, while the larger penalty on weight-two Pauli strings biases the search against weight-two Pauli-string support. We examine these effects separately in §4.4.1 and §4.4.2.

With the candidate pool, saliency criterion, and training objective defined, LieQAS alternates coefficient optimization and progressive pruning. In each round, the surviving coefficients are optimized under equation 6. Candidate deletions are scored and selected for pruning, after which optimization resumes on the reduced support. This process progressively reduces the active support while allowing the surviving coefficients to adapt. The task-based saliency is converted into the congestion-adjusted pruning score introduced in §3.3.

3.3 Non-Commutativity-Aware Search and Finalization

The search mechanisms above account for task sensitivity and Pauli-string cost but do not constrain how surviving Pauli strings are distributed across layers. Multiple Pauli strings may therefore remain located within the same layer. When such Pauli strings do not commute, the Baker-Campbell-Hausdorff (BCH) expansion of the corresponding sequential product contains additional commutator

terms (Hall, 2013), so in general:

$$\exp[-i(\theta_1 P_1 + \theta_2 P_2)] \neq \exp(-i\theta_1 P_1)\exp(-i\theta_2 P_2). \tag{7}$$

Thus, the searched unitary generally cannot be reproduced exactly by applying the corresponding Pauli rotations sequentially. A more general circuit synthesis can realize the composite unitary, but need not preserve a direct correspondence with the selected Pauli-string support. LieQAS instead seeks a final support that admits exact direct extraction.

A sufficient condition for such direct extraction is to leave at most one active Pauli string in each layer. To describe this structure, we define:

$$c_l = \sum_{k=1}^{K} m_{lk}, \qquad M = \sum_{l=0}^{L-1} c_l, \qquad N_{\mathrm{occ}} = \sum_{l=0}^{L-1} \mathbf{1}[c_l > 0], \tag{8}$$

where c_l is the Pauli-string occupancy of layer l, M is the total number of active Pauli strings, and N_{occ} is the number of occupied layers. The indicator function $\mathbf{1}[c_l > 0]$ equals 1 when $c_l > 0$ and 0 otherwise. Let G denote the target number of occupied layers. Requiring $c_l \leq 1$ for every layer is a conservative sufficient condition for exact direct extraction: commuting multi-Pauli-string layers could also be factorized exactly, but singleton occupancy avoids explicit pairwise commutation checks. Non-commutativity across different layers does not affect exact extraction because the original ordering of layer unitaries is preserved.

To incorporate this structural preference into the progressive search, we adjust the Taylor-FO saliency from equation 5 according to layer occupancy. For each active coordinate, we define the congestion-adjusted score:

$$q_{lk} = \frac{s_{lk}}{c_l^{\alpha}}, \qquad \alpha \geq 0. \tag{9}$$

Pruning selects coordinates with smaller q_{lk}, so for comparable task saliency s_{lk}, a Pauli string in a highly occupied layer receives a smaller score and is more likely to be removed. The exponent α controls the strength of this occupancy bias, with $\alpha = 0$ recovering the Taylor-FO ranking without congestion adjustment. Importantly, this mechanism is only a soft bias: it encourages low-occupancy structures during progressive search but does not guarantee singleton layers.

Progressive pruning continues until the target occupied-layer count $N_{\mathrm{occ}} = G$ is reached. At this point, the number of active Pauli strings may still satisfy $M > G$ because some occupied layers can contain multiple Pauli strings. LieQAS therefore performs a finalization stage in which further deletions are restricted to layers with $c_l > 1$. LieQAS removes the Pauli string with the smallest q_{lk}. Each deletion reduces M while preserving N_{occ}, so the process terminates at

$$M = N_{\mathrm{occ}} = G, \tag{10}$$

which implies that every occupied layer contains exactly one active Pauli string. Then the Pauli rotation $\exp(-i\theta_{lk}P_k)$ admits an exact decomposition into standard one- and two-qubit gate primitives (Cowtan et al., 2019). Because no occupied layer contains multiple Pauli strings, no intra-layer factorization is required, and the extracted circuit preserves each finalized layer unitary exactly. Thus, congestion-adjusted scoring introduces a soft structural bias toward lower layer occupancy, while singleton finalization provides the hard guarantee required for exact direct extraction.

4 Experiments

4.1 Experimental Setup

We evaluate LieQAS across two widely used classes of Variational Quantum Algorithms (VQAs): ground state energy estimation via VQE and image classification via QML. We compare task performance and circuit resources against handcrafted ansätze (UCCSD (Romero et al., 2019) for VQE and QCNN (Hur et al., 2022) for QML) and representative differentiable QAS baselines, DQAS (Zhang et al., 2022) and QuantumDARTS (Wu et al., 2023). DQAS is evaluated using its official repository with modifications restricted to benchmark interfaces, while UCCSD, QuantumDARTS, and QCNN are independently implemented following their reported architectures and publicly available hyperparameters.

Table 1: **Ground state energy estimation on molecular VQE benchmarks.** Values in parentheses indicate the reduction ratio relative to the domain-inspired UCCSD baseline.

Molecule	Method	ΔE (Ha)	Gates (Red.)	Depth (Red.)	1Q (Red.)	CX (Red.)
H_2 (4q)	UCCSD	$\mathbf{5.5 \times 10^{-11}}$	116	76	73	43
	QuantumDARTS	3.3×10^{-6}	33 (71.6%)	17 (77.6%)	27 (63.0%)	6 (86.0%)
	LieQAS	1.7×10^{-7}	**12** (89.7%)	**7** (90.8%)	**10** (86.3%)	**2** (95.3%)
LiH (4q)	UCCSD	4.0×10^{-5}	331	238	188	143
	QuantumDARTS	$\mathbf{2.0 \times 10^{-6}}$	71 (78.5%)	35 (85.3%)	55 (70.7%)	16 (88.8%)
	LieQAS	6.0×10^{-4}	**53** (84.0%)	**32** (86.6%)	**41** (78.2%)	**12** (91.6%)
LiH (6q)	UCCSD	$\mathbf{4.0 \times 10^{-5}}$	466	346	234	232
	QuantumDARTS	9.1×10^{-4}	128 (72.5%)	50 (85.5%)	88 (62.4%)	40 (82.8%)
	LieQAS	9.1×10^{-4}	**74** (84.1%)	**42** (87.9%)	**56** (76.1%)	**18** (92.2%)
H_2O (8q)	UCCSD	$\mathbf{4.0 \times 10^{-6}}$	2,167	1,688	998	1169
	QuantumDARTS	1.4×10^{-3}	193 (91.1%)	62 (96.3%)	149 (85.1%)	**44** (96.2%)
	LieQAS	9.8×10^{-4}	**149** (93.1%)	**59** (96.5%)	**105** (89.5%)	**44** (96.2%)

The Lie-algebraic simulation and matrix-exponential operations are implemented in PyTorch (Paszke et al., 2019) with automatic differentiation support for matrix exponentials. Pauli-string coefficients $\boldsymbol{\theta}$ are optimized using Adam. All experiments are executed on NVIDIA RTX 4080D GPUs. All circuits used for resource comparison are synthesized using Qiskit (Javadi-Abhari et al., 2024) with optimization level 3 to the basis $\{\texttt{rx}, \texttt{ry}, \texttt{rz}, \texttt{cx}\}$ under all-to-all connectivity. We report the number of compiled single-qubit rotation gates as **1Q**. Reported architecture resource counts exclude data encoding and readout.

4.2 Task 1: Ground State Energy Estimation (VQE)

We evaluate LieQAS across four VQE instances: H_2 (4 qubits), LiH (4 and 6 qubits), and H_2O (8 qubits). Following the molecular benchmark settings of QuantumDARTS (Wu et al., 2023), we use aligned molecular Hamiltonians and reference energies across the evaluated methods. LieQAS architecture search minimizes the regularized energy objective, whereas final retraining optimizes the bare expectation $\langle H_{\mathrm{mol}} \rangle$. We evaluate performance using $\Delta E = |E(\boldsymbol{\theta}) - E_{\mathrm{ref}}|$, where E_{ref} is the exact ground-state energy, and use chemical accuracy ($\Delta E < 1.6 \times 10^{-3}$ Ha) as the target threshold (McArdle et al., 2020).

Table 1 summarizes the VQE results. LieQAS reaches chemical accuracy on all four benchmarks. Relative to our QuantumDARTS reproduction, LieQAS uses fewer compiled gates, lower circuit depth, and fewer single-qubit gates across all four systems, while using no more CX gates. Relative to the UCCSD baseline under the same compilation pipeline, LieQAS uses substantially fewer gates and lower circuit depth while remaining within chemical accuracy. These results show that LieQAS identifies substantially more compact VQE circuits while retaining the target energy accuracy.

4.3 Task 2: Quantum Machine Learning Classification (QML)

We evaluate two architecture budgets for the MNIST 0/1 binary classification on four feature encodings in QuantumDARTS Wu et al. (2023): LieQAS-S uses $G = 10$ occupied singleton layers, while LieQAS-L uses $G = 50$. As shown in Table 2, compact **LieQAS-S** consistently uses fewer compiled single-qubit gates, CX gates, and total compiled gates than DQAS and QuantumDARTS across all four feature encodings, while maintaining competitive classification accuracy. Relative to QuantumDARTS, LieQAS-S reduces CX gates by up to 89% (requiring 2.0 CXs on PCA encodings) and total gates by up to 65%. While achieving similar mean accuracy on PCA-angle and AE-dense, it achieves higher mean accuracy on PCA-dense and AE-angle. These results show that the small-budget search can reduce circuit resources without a consistent loss in classification performance.

We further evaluate scaled **LieQAS-L** against the handcrafted QCNN (Hur et al., 2022) to examine whether the resource savings persist when model size is increased. LieQAS-L achieves higher mean accuracy on three of the four encodings and lower mean accuracy on AE-dense, while requiring fewer compiled single-qubit gates, CX gates, total gates, and lower circuit depth in all four settings. These results show that the resource savings persist for the larger LieQAS architecture budget.

Table 2: **Classification performance and compiled circuit complexity on MNIST 0/1.** Results are reported over five independent seeds. QCNN uses the same configuration across encodings.

Encoding	Method	Acc. (%)	Gates	Depth	1Q	CX
PCA-angle	DQAS	81.9 ± 14.2	58.8	18.2	37.4	21.4
	QuantumDARTS	$\mathbf{97.4 \pm 3.1}$	48.0	19.0	30.0	18.0
	LieQAS-S	97.2 ± 4.3	**16.8**	**15.4**	**14.8**	**2.0**
	QCNN	98.9 ± 0.3	436.0	137.0	376.0	60.0
	LieQAS-L	$\mathbf{99.3 \pm 0.1}$	**108.6**	**83.4**	**89.6**	**19.0**
PCA-dense	DQAS	89.1 ± 12.8	60.4	18.0	39.8	20.6
	QuantumDARTS	88.8 ± 14.4	46.0	17.2	30.0	16.0
	LieQAS-S	$\mathbf{93.4 \pm 5.2}$	**20.0**	**16.0**	**18.0**	**2.0**
	QCNN	99.3 ± 0.1	436.0	137.0	376.0	60.0
	LieQAS-L	$\mathbf{99.5 \pm 0.1}$	**133.8**	**88.4**	**106.2**	**27.6**
AE-angle	DQAS	85.5 ± 8.9	60.4	19.4	35.8	24.6
	QuantumDARTS	71.9 ± 7.6	40.6	**14.0**	30.0	10.6
	LieQAS-S	$\mathbf{94.8 \pm 0.1}$	**25.2**	20.6	**18.8**	**6.4**
	QCNN	97.4 ± 3.6	436.0	137.0	376.0	60.0
	LieQAS-L	$\mathbf{98.9 \pm 0.5}$	**123.0**	**88.4**	**106.0**	**17.0**
AE-dense	DQAS	$\mathbf{92.7 \pm 6.6}$	61.6	18.6	41.2	20.4
	QuantumDARTS	90.9 ± 1.5	47.0	19.0	30.0	17.0
	LieQAS-S	92.2 ± 2.9	**25.2**	**18.2**	**19.2**	**6.0**
	QCNN	$\mathbf{97.3 \pm 1.8}$	436.0	137.0	376.0	60.0
	LieQAS-L	96.1 ± 0.9	**123.0**	**97.6**	**100.2**	**22.8**

4.4 Ablation Studies and Mechanistic Diagnostics

We conduct ablations on the Fashion-MNIST (dress–shirt) benchmark across five random seeds to examine three mechanisms of LieQAS: (1) the interaction between Taylor approximation order and L_1 shrinkage, (2) hardware-cost bias under asymmetric regularization, and (3) congestion-adjusted search and singleton finalization.

4.4.1 Ablation Study 1: Effect of Regularization on Taylor Approximation

We first examine how structural regularization affects the accuracy of Taylor-based deletion scores. For this diagnostic, we denote the Taylor-FO score from equation 5 by $S_{lk}^{(1)} = |\theta_{lk} g_{lk}^{\mathrm{task}}|$ and additionally define the second-order score, which we term Taylor-SO:

$$S_{lk}^{(2)} = \left| -\theta_{lk} g_{lk}^{\mathrm{task}} + \frac{1}{2} \theta_{lk}^2 h_{lk}^{\mathrm{task}} \right|, \qquad h_{lk}^{\mathrm{task}} = \frac{\partial^2 \mathcal{L}_{\mathrm{task}}}{\partial \theta_{lk}^2}. \tag{11}$$

Taylor-SO is used only as a diagnostic comparator and is not part of the LieQAS search. The exact deletion effect and MAE metric are defined in Appendix C. As shown in Figure 2(a), Taylor-FO exhibits increasing approximation error for larger $|\theta|$ in the unregularized regime, whereas Taylor-SO remains substantially more accurate, indicating that the second-order term explains much of the Taylor-FO error in this tested regime. After L_1 regularization shrinks the Pauli-string coefficients, the Taylor-FO error approaches the Taylor-SO error, consistent with the smaller deletion perturbations produced by shrinkage. The end-to-end results in Table 3 show the same pattern: the FO-SO mean-accuracy gap decreases from 5.3 pp without regularization to 0.1 pp under regularized training. Together, these results support using Taylor-FO pruning without Hessian evaluation in the regularized search.

4.4.2 Ablation Study 2: Asymmetric Regularization and Hardware Cost Trade-off

Holding the pruning engine to Taylor-FO, we evaluate the impact of asymmetric regularization $(\lambda_{1q}, \lambda_{2q})$. As listed in Table 3, configuring an asymmetric setting of $(0.02, 0.05)$ achieves similar mean accuracy to uniform shrinkage $(0.05, 0.05)$, while cutting both compiled CX count and depth by 27% and 22%, respectively. The more asymmetric setting $(0.01, 0.05)$ further reduces CX count and depth, illustrating the accuracy-resource trade-off. In contrast, the inverted setting $(0.05, 0.02)$ increases compiled CX count and depth. These results show that stronger two-qubit regularization can reduce compiled CX count and depth with little change in mean accuracy over the tested settings.

Table 3: **Ablation studies on Fashion-MNIST (dress-shirt).** Results are reported over five independent random seeds.

Pruning Criterion	$(\lambda_{1q}, \lambda_{2q})$	Test Acc. (%)	CX	Depth
Taylor-FO	$(0, 0)$	78.7 ± 4.0	8.4	15.0
Taylor-SO	$(0, 0)$	$\mathbf{84.0 \pm 0.8}$	5.6	10.4
Taylor-FO	$(0.05, 0.05)$	83.9 ± 0.7	6.6	11.8
Taylor-SO	$(0.05, 0.05)$	84.0 ± 0.8	6.2	12.0
Taylor-FO	$(0.01, 0.05)$	83.5 ± 1.1	**4.4**	**9.0**
Taylor-FO	$(0.02, 0.02)$	83.2 ± 0.9	6.6	12.2
Taylor-FO	$(0.02, 0.05)$	$\mathbf{83.9 \pm 0.8}$	4.8	9.2
Taylor-FO	$(0.02, 0.10)$	83.6 ± 0.8	5.8	10.8
Taylor-FO	$(0.05, 0.02)$	83.0 ± 1.0	7.4	13.8

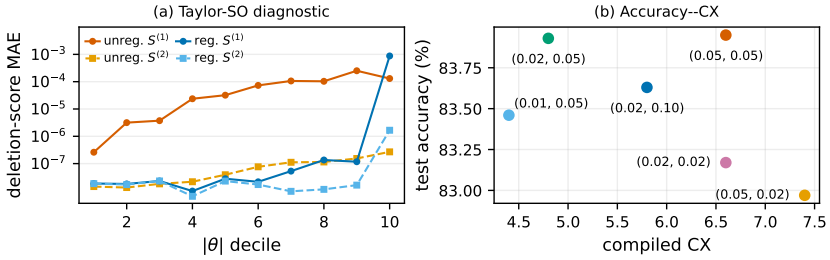

Figure 2: **Mechanistic validation of Taylor pruning and asymmetric regularization. (a)** Deletion-score MAE (defined in Appendix C) across $|\theta|$ deciles: without regularization, Taylor-FO error increases with parameter magnitude, consistent with a larger omitted second-order contribution, whereas Taylor-SO shows substantially lower error. Under regularization, Taylor-FO shows substantially lower approximation error across most $|\theta|$ deciles. **(b)** Accuracy-resource trade-off across $(\lambda_{1q}, \lambda_{2q})$: stronger two-qubit regularization can reduce compiled CX count with little change in mean accuracy in the tested range.

4.4.3 Ablation Study 3: Congestion-Aware Search and Singleton Finalization

We examine the respective roles of congestion-adjusted pruning and singleton finalization. Without either mechanism, the searched architectures retain 13.6 active Pauli strings for a target of $G = 10$ occupied layers, with a mean maximum layer occupancy of 2.6 and 1.4 non-commuting intra-layer pairs. Term-wise factorization yields a non-zero output discrepancy of $D_{\text{fac}}^{\text{QML}} = 0.5$. Using congestion-adjusted scoring reduces the active-Pauli-string count to 10.2 and maximum layer occupancy to 1.2. The remaining multi-Pauli-string layers are commuting in these runs, giving zero measured factorization discrepancy. Singleton finalization further enforces exactly 10 active Pauli strings with one Pauli string per occupied layer. Thus, congestion-adjusted scoring promotes low-occupancy structures, while finalization provides the deterministic singleton guarantee. Detailed statistics are reported in Table 7, with the discrepancy metric defined in Appendix D.

5 Conclusion

LieQAS directly optimizes valid unitary transformations in continuous Lie-algebraic coordinates over the Pauli basis while updating circuit architecture through gradient-informed pruning of the active Pauli-string support. Congestion-adjusted scoring biases the search toward low-occupancy layers, while singleton finalization guarantees one active Pauli string per occupied layer for exact direct extraction. Across molecular VQE and quantum classification benchmarks, the reported LieQAS architectures maintain competitive task performance, including chemical accuracy on all four VQE instances, while reducing compiled circuit resources relative to representative baselines. LieQAS therefore connects continuous Lie-algebraic optimization over a Pauli basis with structured architecture search and exact circuit extraction.

AI USE STATEMENT

In this work, we used generative AI tools only to assist with code debugging, literature search for background references, and polishing the manuscript. All AI-assisted debugging suggestions were manually reviewed and tested by the authors. Papers identified with AI assistance include only background references and were read and verified by the authors. All identified language and grammar issues were independently checked before any changes were made. Generative AI tools were not used to generate experimental data or to make final decisions regarding the methodology, experiments, analysis, or scientific claims. We take responsibility for the final content of this work, including any text, code, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Awad H Al-Mohy and Nicholas J Higham. Computing the fréchet derivative of the matrix exponential, with an application to condition number estimation. *SIAM Journal on Matrix Analysis and Applications*, 30(4):1639–1657, 2009.

Matias Bilkis, Marco Cerezo, Guillaume Verdon, Patrick J Coles, and Lukasz Cincio. A semi-agnostic ansatz with variable structure for variational quantum algorithms. *Quantum Machine Intelligence*, 5(2):43, 2023.

Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

Tianlong Chen, Zhenyu Zhang, Hanrui Wang, Jiaqi Gu, Zirui Li, David Z Pan, Frederic T Chong, Song Han, and Zhangyang Wang. Quantumsea: In-time sparse exploration for noise adaptive quantum circuits. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 1, pp. 51–62. IEEE, 2023.

Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons, and Seyon Sivarajah. Phase gadget synthesis for shallow circuits. *arXiv preprint arXiv:1906.01734*, 2019.

Li Ding and Lee Spector. Evolutionary quantum architecture search for parametrized quantum circuits. In *Proceedings of the genetic and evolutionary computation conference companion*, pp. 2190–2195, 2022.

Jennifer L Dodd, Michael A Nielsen, Michael J Bremner, and Robert T Thew. Universal quantum computation and simulation using any entangling hamiltonian and local unitaries. *Physical Review A*, 65(4):040301, 2002.

Alessandro Giovagnoli, Volker Tresp, Yunpu Ma, and Matthias Schubert. Qneat: Natural evolution of variational quantum circuit architecture. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pp. 647–650, 2023.

Matthew L Goh, Martin Larocca, Lukasz Cincio, Marco Cerezo, and Frédéric Sauvage. Lie-algebraic classical simulations for quantum computing. *Physical Review Research*, 7(3):033266, 2025.

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.

Brian C Hall. Lie groups, lie algebras, and representations. In *Quantum theory for mathematicians*, pp. 333–366. Springer, 2013.

Tak Hur, Leeseok Kim, and Daniel K Park. Quantum convolutional neural network for classical data classification. *Quantum Machine Intelligence*, 4(1):3, 2022.

Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D Nation, Lev S Bishop, Andrew W Cross, et al. Quantum computing with qiskit. *arXiv preprint arXiv:2405.08810*, 2024.

Sumit K Jeswal and S Chakraverty. Recent developments and applications in quantum neural network: A review. *Archives of Computational Methods in Engineering*, 26(4):793–807, 2019.

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.

Navin Khaneja and Steffen J Glaser. Cartan decomposition of su (2n) and control of spin systems. *Chemical Physics*, 267(1-3):11–23, 2001.

Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. A quantum engineer's guide to superconducting qubits. *Applied physics reviews*, 6(2): 021318, 2019.

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, pp. 120–125. IEEE, 2024.

Swagat Kumar, Jan-Nico Zaech, Colin Michael Wilmott, and Luc Van Gool. Rhodarts: Differentiable quantum architecture search with density matrix simulations. *arXiv preprint arXiv:2506.03697*, 2025.

Martin Larocca, Piotr Czarnik, Kunal Sharma, Gopikrishnan Muraleedharan, Patrick J Coles, and Marco Cerezo. Diagnosing barren plateaus with tools from quantum optimal control. *Quantum*, 6:824, 2022.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C Benjamin, and Xiao Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92(1):015003, 2020.

Alexander J McCaskey, Zachary P Parks, Jacek Jakowski, Shirley V Moore, Titus D Morris, Travis S Humble, and Raphael C Pooser. Quantum chemistry as a benchmark for near-term quantum computers. *npj Quantum Information*, 5(1):99, 2019.

Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812, 2018.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11256–11264. IEEE, 2019.

Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Advances in neural information processing systems*, 1, 1988.

J Nocedal and S Wright. *Numerical Optimization*. Springer, 2006.

Mateusz Ostaszewski, Lea M Trenkwalder, Wojciech Masarczyk, Eleanor Scerri, and Vedran Dunjko. Reinforcement learning for optimization of variational quantum circuit architectures. *Advances in neural information processing systems*, 34:18182–18194, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Yash J. Patel, Akash Kundu, Mateusz Ostaszewski, Xavier Bonet-Monroig, Vedran Dunjko, and Onur Danaci. Curriculum reinforcement learning for quantum architecture search under hardware errors. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 32298–32329, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/88cc8602d5e5a913e1c2a2142ca7377b-Paper-Conference.pdf.

Yihui Quek, Daniel Stilck França, Sumeet Khatri, Johannes Jakob Meyer, and Jens Eisert. Exponentially tighter bounds on limitations of quantum error mitigation. *Nature Physics*, 20(10): 1648–1658, 2024.

Jonathan Romero, Ryan Babbush, Jarrod R McClean, Cornelius Hempel, Peter J Love, and Alán Aspuru-Guzik. Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *Quantum Science and Technology*, 4(1):014008, 2019.

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.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H Booth, et al. The variational quantum eigensolver: a review of methods and best practices. *Physics Reports*, 986:1–128, 2022.

Hanrui Wang, Yongshan Ding, Jiaqi Gu, Yujun Lin, David Z Pan, Frederic T Chong, and Song Han. Quantumnas: Noise-adaptive search for robust quantum circuits. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 692–708. IEEE, 2022.

Ralph M Wilcox. Exponential operators and parameter differentiation in quantum physics. *Journal of Mathematical Physics*, 8(4):962–982, 1967.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10726–10734. IEEE, 2019.

Wenjie Wu, Ge Yan, Xudong Lu, Kaisen Pan, and Junchi Yan. Quantumdarts: differentiable quantum architecture search for variational quantum algorithms. In *International conference on machine learning*, pp. 37745–37764. PMLR, 2023.

Anqi Zhang and Shengmei Zhao. Evolutionary-based searching method for quantum circuit architecture. *Quantum Information Processing*, 22(7):283, 2023.

Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang, and Hong Yao. Neural predictor based quantum architecture search. *Machine Learning: Science and Technology*, 2(4):045027, 2021.

Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang, and Hong Yao. Differentiable quantum architecture search. *Quantum Science & Technology*, 7(4):045023, 2022.

A Full Search and Finalization Algorithm

The main text introduces progressive search and singleton extraction separately to clarify their respective roles. Here, we consolidate these components into the complete LieQAS procedure. Algorithm 1 summarizes the LieQAS search and extraction procedure. The search first applies congestion-adjusted progressive pruning until the number of occupied layers reaches the target G. If multiple Pauli strings remain within any occupied layer, a singleton finalization stage subsequently removes Pauli strings only from multi-Pauli-string layers until the total active Pauli string count also reaches G. The resulting architecture therefore contains exactly one active Pauli string in each of the G occupied layers.

Algorithm 1 Progressive Lie-Algebraic Architecture Search and Finalization

Require: Pauli-string pool $\mathcal{P}_{\leq 2}$, number of layers L, target occupied-layer number G, pruning rounds R with a removal schedule $\{\Delta M_r\}_{r=1}^{R}$ that reaches $M = N_{\mathrm{occ}} = G$, penalties $\lambda_{1q}, \lambda_{2q}$, and congestion exponent α

Initialize $\mathbf{m} \leftarrow \mathbf{1}_{L\times K}$ and Pauli-string coefficients $\boldsymbol{\theta}$
Compute $c_l \leftarrow \sum_k m_{lk}, \qquad M \leftarrow \sum_l c_l, \qquad N_{\mathrm{occ}} \leftarrow \sum_l \mathbf{1}[c_l > 0]$
for $r = 1, \ldots, R$ **do** ▷ *Stage I: progressive architecture search*
  if $N_{\mathrm{occ}} \leq G$ **then**
    break
  end if
  Optimize active coefficients under $\mathcal{L}_{\mathrm{train}} = \mathcal{L}_{\mathrm{task}} + \sum_{l,k} m_{lk}\lambda_k|\theta_{lk}|$
  Compute task gradients $g_{lk}^{\mathrm{task}} = \frac{\partial \mathcal{L}_{\mathrm{task}}}{\partial \theta_{lk}}$
  Compute Taylor-FO saliency $s_{lk} = |\theta_{lk} g_{lk}^{\mathrm{task}}|$
  Record current occupancies $c_l^{(r)} \leftarrow c_l$
  Compute $q_{lk}^{(r)} = s_{lk}/(c_l^{(r)})^{\alpha}$
  Sort active coordinates once by increasing $q_{lk}^{(r)}$
  for $j = 1, \ldots, \Delta M_r$ **do**
    Select the next active coordinate $(l^\star, k^\star)$ from the fixed ranking
    Remove the selected Pauli string: $m_{l^\star k^\star} \leftarrow 0$
    Update $c_{l^\star} \leftarrow c_{l^\star} - 1, \qquad M \leftarrow M - 1$
    Update $N_{\mathrm{occ}} \leftarrow \sum_l \mathbf{1}[c_l > 0]$
    if $N_{\mathrm{occ}} \leq G$ **then**
      break
    end if
  end for
end for ▷ *At this point,* $N_{\mathrm{occ}} = G$
while $M > G$ **do** ▷ *Stage II: singleton finalization*
  Optimize active coefficients under $\mathcal{L}_{\mathrm{train}} = \mathcal{L}_{\mathrm{task}} + \sum_{l,k} m_{lk}\lambda_k|\theta_{lk}|$
  Compute task gradients $g_{lk}^{\mathrm{task}} = \frac{\partial \mathcal{L}_{\mathrm{task}}}{\partial \theta_{lk}}$
  Compute Taylor-FO saliency $s_{lk} = |\theta_{lk} g_{lk}^{\mathrm{task}}|$
  Record current occupancies $\tilde{c}_l \leftarrow c_l$
  Construct the multi-Pauli-string candidate set $\mathcal{F}(\mathbf{m}) = \{(l,k) : m_{lk} = 1,\ \tilde{c}_l > 1\}$
  Compute $q_{lk} = s_{lk}/\tilde{c}_l^{\alpha}, \qquad (l,k) \in \mathcal{F}(\mathbf{m})$
  Sort $\mathcal{F}(\mathbf{m})$ once by increasing q_{lk}
  $d \leftarrow 0$
  for each (l,k) in the fixed ranking **do**
    if $M \leq G$ **or** $d \geq \Delta M_r$ **then**
      break
    end if
    if $m_{lk} = 0$ **or** $c_l \leq 1$ **then**
      continue
    end if
    Remove the selected Pauli string: $m_{lk} \leftarrow 0$
    Update $c_l \leftarrow c_l - 1, \qquad M \leftarrow M - 1$
    $d \leftarrow d + 1$
  end for
  $r \leftarrow r + 1$
end while
Retrain the final singleton architecture using $\mathcal{L}_{\mathrm{task}}$
return G occupied singleton Pauli rotations and their coefficients

Table 4: **Measured computational cost on the MNIST 0/1 benchmarks.** Results are reported over five independent random seeds.

Encoding	Method	Time (s)	Peak Mem. (MiB)
PCA-angle	DQAS	142.0 ± 12.2	449.2 ± 1.1
	QuantumDARTS	54.4 ± 4.4	807.2 ± 1.1
	LieQAS	522.4 ± 36.7	1598.0 ± 0.0
PCA-dense	DQAS	137.0 ± 14.6	450.4 ± 0.9
	QuantumDARTS	56.9 ± 6.1	806.8 ± 1.1
	LieQAS	512.9 ± 42.6	1598.0 ± 0.0
AE-angle	DQAS	126.2 ± 14.0	449.6 ± 0.9
	QuantumDARTS	35.4 ± 4.0	646.0 ± 0.0
	LieQAS	493.2 ± 31.5	1598.0 ± 0.0
AE-dense	DQAS	124.1 ± 13.7	449.2 ± 1.1
	QuantumDARTS	53.3 ± 5.1	807.2 ± 1.1
	LieQAS	469.8 ± 15.6	1598.0 ± 0.0

B Computational Complexity and Effect of Maximum Pauli Weight

B.1 Asymptotic Complexity

We first analyze the computational complexity of differentiable optimization in the Lie-algebraic representation. Let $d = 2^n$ denote the Hilbert-space dimension. If the explicit candidate pool contains all Pauli strings up to maximum weight w, the number of Pauli strings per layer is:

$$K_{\le w} = \sum_{r=1}^{w} 3^r \binom{n}{r}. \tag{12}$$

For fixed w, this gives $K_{\le w} = \mathcal{O}(n^w)$. The main LieQAS experiments use $w = 2$, for which:

$$K_{\le 2} = 3n + 9\binom{n}{2} = \frac{9n^2 - 3n}{2} = \mathcal{O}(n^2). \tag{13}$$

For dense simulation, evaluating a $d \times d$ matrix exponential and its reverse-mode derivative requires cubic matrix-algebra cost, $\mathcal{O}(d^3)$, up to algorithm-dependent constants. Once the gradient with respect to the Hermitian operator H_l is available, obtaining gradients for all $K_{\le w}$ Pauli-string coefficients requires an additional $\mathcal{O}(K_{\le w} d^2)$ contractions. Thus, one optimization step over L layers has asymptotic cost:

$$\mathcal{O}\big(Ld^3 + LK_{\le w} d^2\big). \tag{14}$$

For the weight-two pool used in the main experiments, this becomes

$$\mathcal{O}\big(L8^n + Ln^2 4^n\big). \tag{15}$$

The dominant cost therefore remains dense quantum simulation rather than the number of explicit search coordinates. The reduction from the complete $4^n - 1$ non-identity Pauli basis to the $\mathcal{O}(n^2)$ weight-two pool concerns the number of explicit candidate Pauli strings used by the architecture search, not the classical simulation complexity.

We next consider the overhead of progressive pruning. Let M denote the current number of active Pauli strings. Computing Taylor-FO saliency, layer occupancies, and congestion-adjusted scores costs $\mathcal{O}(M)$. Sorting the active coordinates once per pruning round costs $\mathcal{O}(M \log M)$, and processing ΔM_r deletions costs $\mathcal{O}(\Delta M_r)$. Thus, the pruning overhead per round is $\mathcal{O}(M \log M + \Delta M_r)$, excluding gradient evaluation. Hence, for the dense implementation used in this work, the matrix-exponential simulation remains the dominant computational cost.

B.2 Measured Runtime and Memory Cost

To complement the asymptotic analysis, we report wall-clock time and peak GPU memory for the evaluated implementations in Table 4. All results are measured over five independent seeds under the same GPU environment. These measurements are implementation-dependent and are intended to characterize the practical computational cost of the evaluated implementations rather than implementation-independent algorithmic complexity.

Table 5: **Measured wall-clock time on the broader QAS benchmarks.** Results are mean ± standard deviation over five independent seeds.

Task	LieQAS	QuantumSEA	QuantumNAS
Fashion-MNIST (3 vs 6)	243.2 ± 6.9	213.8 ± 10.9	2492.0 ± 103.1
Fashion-MNIST (0–3)	471.3 ± 7.9	214.0 ± 12.0	3584.7 ± 198.2
MNIST (1 vs 0)	186.7 ± 22.2	215.8 ± 7.1	2598.6 ± 147.9
MNIST (3 vs 6)	241.6 ± 12.6	214.6 ± 8.5	2474.9 ± 100.1
MNIST (0–3)	501.9 ± 31.3	211.2 ± 8.7	3545.0 ± 343.2
Vowel (0,5,1,6)	23.4 ± 3.1	181.1 ± 8.3	2554.0 ± 189.8

Table 6: **Effect of the maximum candidate Pauli weight.** Wall-clock time is measured under isolated GPU execution.

$w_{\max}$	Acc. (%)	Time (s)	Mem. (MiB)	Compiled Resources			
				Gates	Depth	1Q	CX
2	83.8 ± 1.1	132.2 ± 3.9	1292.0	21.6	13.4	17.6	4.0
3	83.4 ± 1.9	609.9 ± 24.7	2804.0	35.4	25.6	24.0	11.4
4	83.7 ± 0.4	2902.6 ± 68.1	8474.0	31.0	20.8	20.0	11.0

Results in Tabel 4 show that on the eight-qubit MNIST 0/1 benchmarks, LieQAS incurs higher wall-clock time and peak GPU memory than DQAS and QuantumDARTS. This overhead reflects both the dense matrix-exponential simulation and the dense-to-sparse nature of the search: LieQAS begins from a redundant Pauli-string pool and progressively prunes it to the final sparse architecture, therefore substantially more Pauli strings are active during early search than remain after finalization. The resulting compact circuits therefore do not imply lower classical search cost. LieQAS shows identical measured peak memory because its dense continuous parameterization keeps the main search tensors fixed in shape throughout optimization, with pruning changing only the mask and parameter values.

The measured cost also depends strongly on the problem size. On the broader-QAS benchmarks as shown in Table 5, LieQAS is substantially faster than QuantumNAS and is of a comparable order to QuantumSEA, whereas the eight-qubit MNIST experiments incur noticeably larger cost. This behavior is consistent with both the exponential dense-simulation dependence on qubit number and the temporary redundancy introduced by the dense-to-sparse pruning process.

Our current implementation uses dense matrix exponentials and does not explicitly exploit additional Lie-algebraic structure during simulation. Several implementation-level optimizations may reduce this overhead. For example, Lie-algebraic simulation methods can exploit sparse adjoint representations to accelerate evolution when the relevant dynamical Lie algebra admits a compact representation (Goh et al., 2025). More generally, matrix-free exponential actions and earlier active-set sparsification could reduce the cost of forming and differentiating dense operators. These directions are complementary to the present architecture-search formulation. We leave these implementation-level optimizations for future work.

B.3 Effect of Maximum Pauli Weight

We examine the effect of enlarging the explicit candidate pool beyond the weight-two Pauli strings used in the main experiments. The comparison uses the Fashion-MNIST binary classification task with classes 3 and 6 (Dress vs. Shirt) with a batch size of 16. Inputs are encoded on 8 qubits using PCA-angle encoding, with min-max scaling applied after PCA. We repeat the same task with maximum candidate Pauli weights $w_{\max} \in \{2, 3, 4\}$. All settings use five independent random seeds and are finalized to the same architecture size of 10 active Pauli strings. This isolates the effect of expanding the candidate Pauli-string pool while keeping the final architecture budget fixed.

As shown in Table 6, expanding the candidate pool beyond weight two does not improve mean test accuracy on this task, while substantially increasing both dense-simulation cost and compiled circuit complexity. Higher-weight Pauli rotations generally require longer decompositions into one- and two-qubit gates (Cowtan et al., 2019), and the $w_{\max} = 3$ and $w_{\max} = 4$ settings yield larger compiled circuits in this experiment. Meanwhile, relative to $w_{\max} = 2$, the dense-simulation

Table 7: **Ablation of congestion-adjusted pruning and singleton finalization with $G = 10$.** Each entry is mean $\pm$ standard deviation over five independent seeds. Factorized circuits use the fixed-order product of the surviving Pauli rotations.

Condition	Continuous Accuracy (%)	Factorized Accuracy (%)	Δ accuracy (pp)	Active Pauli Strings	$\max c_l$	Noncommuting Intra-Layer Pairs	$D_{\mathrm{fac}}^{\mathrm{QML}}$
No safeguard	84.1 ± 0.6	84.0 ± 0.7	-0.1 ± 0.1	13.6 ± 1.8	2.6 ± 0.9	1.4 ± 1.7	0.5 ± 0.9
Congestion only	83.9 ± 0.6	83.9 ± 0.6	0.0 ± 0.0	10.2 ± 0.5	1.2 ± 0.5	0.0 ± 0.0	0.0 ± 0.0
Congestion + Finalization	84.0 ± 0.7	84.0 ± 0.7	0.0 ± 0.0	10.0 ± 0.0	1.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0

cost increases to approximately $4.6\times/2.2\times$ the wall-clock time/peak memory for $w_{\max} = 3$ and $22.0\times/6.6\times$ for $w_{\max} = 4$. These timing and memory measurements reflect isolated GPU execution rather than a specific quantum-hardware gate-cost model. Overall, the observed accuracy-resource trade-off supports the weight-two candidate pool used in the main experiments.

C Exact Deletion Effect and Taylor Approximation Error

Ablation Study 1 evaluates whether Taylor-FO and Taylor-SO predict the finite loss change caused by Pauli-string deletion. To keep the main discussion focused on the observed approximation behavior, we define the exact deletion effect and the corresponding bin-wise MAE here. For coordinate (l, k), the exact deletion effect is

$$d_{lk} = \left|\mathcal{L}_{\mathrm{task}}(\boldsymbol{\theta} - \theta_{lk}\mathbf{e}_{lk}) - \mathcal{L}_{\mathrm{task}}(\boldsymbol{\theta})\right|, \tag{16}$$

where $\mathbf{e}_{lk}$ denotes the corresponding coordinate basis vector.

For the b-th $|\theta|$ decile $\mathcal{D}_b$, we measure the approximation error of the r-th order Taylor score by

$$\mathrm{MAE}_b^{(r)} = \frac{1}{|\mathcal{D}_b|} \sum_{(l,k)\in\mathcal{D}_b} \left|S_{lk}^{(r)} - d_{lk}\right|, \qquad r \in \{1, 2\}. \tag{17}$$

Lower MAE therefore indicates that the corresponding Taylor score more accurately predicts the finite loss change caused by Pauli-string deletion.

D Factorization Discrepancy Metric

Ablation Study 3 evaluates whether term-wise factorization changes the output of the searched continuous representation. We quantify this effect by comparing the continuous composite-exponential model with its factorized circuit realization under identical learned parameters.

For quantum classification workloads, let $z_c^{\mathrm{cont}}(x)$ and $z_c^{\mathrm{fac}}(x)$ denote the logit for class c under input x. The classification factorization discrepancy is defined as:

$$D_{\mathrm{fac}}^{\mathrm{QML}} = \max_{x\in\mathcal{D}_{\mathrm{test}}} \max_{c} \left|z_c^{\mathrm{cont}}(x) - z_c^{\mathrm{fac}}(x)\right|. \tag{18}$$

In this case, factorizing non-commuting multi-Pauli-string layers can produce non-zero output discrepancy through BCH commutator terms. The QML discrepancy reported in §4.4.3 is computed using equation 18. Table 7 reports the corresponding structural statistics and output discrepancies under different extraction settings. Here, "No safeguard" uses $\alpha = 0$ without finalization, "Congestion only" uses $\alpha = 1$ without finalization, and "Congestion + Finalization" additionally applies the singleton finalization. A non-zero $D_{\mathrm{fac}}^{\mathrm{QML}}$ indicates that term-wise factorization changes the model outputs on the evaluated data. Conversely, zero measured discrepancy can arise, for example, when every occupied layer is singleton or when the remaining co-located Pauli strings commute. Singleton finalization avoids this ambiguity by guaranteeing singleton occupancy, for which no intra-layer factorization is required.

E Broader Quantum Architecture Search Comparisons

Beyond the differentiable QAS baselines considered in the main text, we additionally compare with representative methods from broader QAS paradigms across several quantum classification tasks.

Table 8: **Benchmark comparison across additional quantum classification tasks.** All searched circuits are evaluated against QuantumSEA and QuantumNAS under the uniform Qiskit O3 compilation pipeline. Results are reported over five independent random seeds.

Task	Method	Test Acc. (%)	Compiled Metrics (O3)			
			1Q	Depth	CX	Total Gates
Fashion-MNIST (0, 1, 2, 3)	QuantumSEA	79.3 ± 1.7	124.0	67.4	24.0	148.0
	QuantumNAS	80.6 ± 0.9	208.6	154.8	25.2	233.8
	LieQAS	79.1 ± 0.9	96.6	98.2	47.2	143.8
Fashion-MNIST (3 vs 6)	QuantumSEA	89.2 ± 1.1	124.0	67.4	24.0	148.0
	QuantumNAS	87.3 ± 0.5	132.0	100.8	16.0	148.0
	LieQAS	87.2 ± 0.4	50.6	38.0	12.0	62.6
MNIST (0, 1, 2, 3)	QuantumSEA	71.1 ± 2.2	124.0	71.2	24.0	148.0
	QuantumNAS	71.9 ± 7.2	139.2	120.4	18.4	157.6
	LieQAS	71.4 ± 2.7	54.8	42.6	13.0	67.8
MNIST (1 vs 0)	QuantumSEA	97.5 ± 0.6	124.0	67.0	24.0	148.0
	QuantumNAS	97.5 ± 0.2	164.8	117.0	19.6	184.4
	LieQAS	95.9 ± 0.1	15.0	5.2	0.0	15.0
MNIST (3 vs 6)	QuantumSEA	92.3 ± 0.6	124.0	70.0	24.0	148.0
	QuantumNAS	94.9 ± 0.2	123.2	80.8	14.4	137.6
	LieQAS	93.3 ± 0.9	10.0	4.4	0.0	10.0
Vowel (0, 5, 1, 6)	QuantumSEA	46.7 ± 9.6	124.0	68.2	24.0	148.0
	QuantumNAS	49.5 ± 10.4	204.2	150.4	24.4	228.6
	LieQAS	47.2 ± 7.3	64.0	43.2	10.8	74.8

These experiments assess whether the accuracy-resource trade-off observed in the main experiments persists across additional datasets and search paradigms.

QuantumNAS. QuantumNAS (Wang et al., 2022) adopts an evolutionary algorithm-based one-shot SuperCircuit to decouple parameter training from architecture evaluation. Candidate SubCircuits inherit parameters from the trained SuperCircuit and are subsequently searched jointly with qubit mappings under hardware-noise information, followed by gate pruning and fine-tuning. Its search variables therefore correspond to discrete circuit structures and mappings rather than continuous coefficients over a Pauli basis.

QuantumSEA. QuantumSEA (Chen et al., 2023) performs dynamic sparse circuit exploration while maintaining a fixed sparsity budget during training. It alternates salience-based gate pruning with gradient-guided gate growth to jointly adapt circuit topology and parameters under device noise. In contrast to our progressive pruning over a continuous Pauli-basis representation, QuantumSEA explicitly explores sparse gate topologies through discrete prune-grow updates.

For comparison, we reproduce these broader QAS baselines under the same task definition and evaluation protocol wherever their search spaces permit a direct implementation. We benchmark quantum classification performance on MNIST, Fashion-MNIST, and Vowel datasets across both binary and multi-class tasks aligning with QuantumSEA and QuantumNAS.

As shown in Table 8, LieQAS remains competitive in classification accuracy across the evaluated tasks while consistently using fewer single-qubit gates and total compiled gates than QuantumSEA and QuantumNAS. On the binary MNIST tasks, particularly small architectures are obtained with only a modest decrease in mean accuracy relative to the best baseline. The multi-class and Vowel results show the same overall accuracy-resource trade-off across broader task settings.

F VQE Reporting Protocol and Additional Baseline Results

The molecular VQE results in Table 1 follow the single-result reporting convention used in prior differentiable QAS VQE evaluations (Wu et al., 2023; Kumar et al., 2025). For each QAS method and molecular instance, we evaluate three independent seeds and report the minimum-depth finalized architecture among the runs that reach chemical accuracy. The reported energy error and compiled resources therefore characterize a successful searched architecture rather than seed-averaged search robustness.

We additionally evaluated DQAS and ρDARTS under the aligned molecular benchmark setup. DQAS reached chemical accuracy for all three evaluated seeds on H_2 and 4-qubit LiH, but did not reach the threshold for 6-qubit LiH or 8-qubit H_2O. For ρDARTS, we used the released implementation, but some configuration details required to reproduce the reported VQE setting were not available. Under our reconstructed and aligned configuration, ρDARTS reached chemical accuracy for two of three seeds on H_2 and all three seeds on 4-qubit LiH, but not for 6-qubit LiH or 8-qubit H_2O. We therefore use QuantumDARTS as the primary differentiable QAS baseline for the four-instance VQE comparison in Table 1.