UNLEARNING-DRIVEN QUANTUM ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantum machine unlearning aims to remove the influence of specified training data from quantum models. However, existing quantum architecture search (QAS) primarily selects architectures based on predictive accuracy and does not account for potential post-deployment unlearning needs. To bridge this research gap, we conduct an empirical study and reveal for the first time that quantum model architecture itself can substantially affect unlearning capability, even among architectures with comparable predictive accuracy. Building on this observation, we propose Unlearning-driven Quantum Architecture Search (UQAS), a framework that incorporates unlearning capability into QAS to better prepare quantum models for post-deployment unlearning requests. In UQAS, we first introduce an Unlearning Capability Evaluator (UCE) to score the unlearning capability of a quantum architecture using synthetic unlearning requests. Then we design two complementary algorithms, UQAS-Select and UQAS-Design, to identify the quantum architecture with the best UCE score under two QAS settings (with and without an existing architecture pool) respectively. To evaluate the performance of the proposed UQAS, we conduct extensive experiments in both simulation and on IBM quantum computers, across sixteen evaluation scenarios spanning four diverse datasets and four accuracy-based QAS methods. The performance analysis reveals that UQAS effectively identifies quantum architectures with stronger unlearning capability while maintaining competitive predictive accuracy. Furthermore, UQAS-Select and UQAS-Design demonstrate complementary trade-offs between search cost and unlearning capabilities, enabling flexible selection under different computational budgets.

1 INTRODUCTION

Unlearning is emerging as a foundational capability for responsible Quantum Machine Learning (QML) systems that must adapt to evolving data and deployment requirements (Shaik et al., 2025). Its goal is to remove the influence of specified data without significantly degrading predictive performance on retained data (Su et al., 2026, Malik et al., 2026). To avoid the computational cost of retraining the model from scratch on the retained data, existing unlearning methods typically update the parameters of the deployed model in response to an unlearning request (Chen & Zhang, 2026, Crivoi & Ionescu, 2026). Such post-deployment update operates on a fixed quantum circuit architecture that has already been determined before deployment.

The circuit architecture is identified via quantum architecture search (QAS), which explores and evaluates candidate structures primarily based on predictive accuracy (Du et al., 2022, Lu et al., 2023), with additional consideration of circuit cost and robustness to hardware noise (Wang et al., 2022, Wu et al., 2023). However, existing QAS does not account for the potential need for post-deployment unlearning. This leaves a fundamental question unexplored: *does the quantum circuit architecture affect a QML model's unlearning capability*?

To answer this, we conduct an empirical study and reveal that, *quantum architectures exhibit substantially different unlearning capability, even when their predictive accuracy is comparable*. This finding indicates that unlearning capability is determined not solely by the post-deployment unlearning operation (e.g., parameter updating), but also by the architecture selected before deployment.

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.

Building on this key finding, we propose **U**nlearning-driven **Q**uantum **A**rchitecture **S**earch (UQAS), a novel framework that incorporates unlearning capability as a criterion into pre-deployment architecture selection while maintaining competitive predictive accuracy. In detail, since post-deployment unlearning requests are unknown during architecture selection, we first introduce an Unlearning Capability Evaluator (UCE) to assess the post-deployment unlearning capability of quantum architectures before deployment using synthetic unlearning requests. Guided by UCE, we then develop two complementary UQAS variants, UQAS-Select and UQAS-Design, for two practical QAS settings: with and without an existing architecture candidate pool. UQAS-Select selects the best architecture from an existing candidate pool, while UQAS-Design generates an architecture with optimized unlearning capability.

The main contributions of this work are threefold.

- To the best of our knowledge, we are the first to empirically reveal that quantum architectures with comparable predictive accuracy can exhibit substantially different unlearning capability. This finding establishes quantum architecture as a previously overlooked factor that affects post-deployment unlearning capability.
- Building on the above finding, we design UQAS to incorporate unlearning capability into QAS as a design objective before deployment. In UQAS, we propose a metric, *unlearning discrepancy*, to quantify the unlearning capability of quantum architecture. This metric is then adopted into UCE to score unlearning capability over synthetic unlearning requests. Guided by the UCE results, two UQAS variants are developed to identify an architecture with stronger unlearning capability while preserving competitive predictive accuracy.
- We evaluate UQAS through comprehensive experiments in simulation and on IBM quantum hardware, across sixteen evaluation scenarios spanning four datasets and four QAS methods. The numerical results demonstrate that UQAS significantly improves the unlearning capability of quantum architecture. Moreover, this improvement persists in real quantum computers.

2 Related Work

Quantum machine unlearning (QMU). Existing QMU studies primarily address post-deployment unlearning by updating trained model parameters within fixed circuit architectures (Chen & Zhang, 2026, Malik et al., 2026). To evaluate these methods under different unlearning needs, studies construct requests specifying the training data to forget (Crivoi & Ionescu, 2026). Randomly sampled subsets and entire classes serve as forget sets to assess methods such as Label-Complement Augmentation (LCA) on hybrid quantum models (Crivoi & Ionescu, 2026). For whole class requests, methods seek to preserve predictive performance on retained classes (Su et al., 2026), including approaches that guide model updates through target output distributions and constraints on retained predictions (Malik et al., 2026). Poison requests instead target the influence of samples whose labels are deliberately flipped before training (Chen & Zhang, 2026). However, existing studies use these request constructions primarily for post-deployment unlearning evaluation, leaving their role before deployment unexplored.

Quantum architecture search. QAS identifies and evaluates circuit architectures predominantly based on predictive accuracy. (Du et al., 2022, Lu et al., 2023, Martyniuk et al., 2024). The widely-used search strategies include reinforcement learning (Ostaszewski et al., 2021, Kuo et al., 2021, Zhu & Hou, 2023), tree search (Chen et al., 2024, Lipardi et al., 2025), evolutionary and adaptive search (Chivilikhin et al., 2020, Lourens et al., 2023, Huang et al., 2024, Grimsley et al., 2019, Tang et al., 2021), and gradient optimization over continuous architecture representations, as used in QuantumDARTS (Wu et al., 2023). Besides accuracy-oriented search, some studies focus on reducing candidate evaluation cost through learned performance predictors (Zhang et al., 2021, He et al., 2024b) or structural measures that avoid candidate training (He et al., 2024a). In addition, hardware noise has also motivated joint search over circuit structures and qubit mappings in QuantumNAS (Wang et al., 2022), as well as compact circuit construction under device noise (Patel et al., 2024). These studies advance QAS in terms of predictive performance, search efficiency, and robustness to hardware constraints. However, existing QAS methods do not incorporate unlearning capability into architecture selection or search.

Our approach. UQAS differs from the reviewed quantum machine unlearning and QAS methods in the role of unlearning evaluation. Whereas the existing quantum machine unlearning studies focus on developing unlearning procedures, our work examines unlearning differences across architectures with comparable predictive accuracy and uses these differences to guide architecture selection. Relative to QAS methods that emphasize predictive performance and hardware requirements, UQAS explicitly evaluates how candidate architectures respond to unlearning requests. Unlearning capability thus informs which architecture should be selected before deployment, complementing efforts to improve the unlearning procedure itself. This unlearning-driven design principle underlies both UQAS-Select and UQAS-Design. Table 1 summarizes these distinctions.

Table 1: **Feature comparison.**

Method	Architecture generation	Unlearning support	Hardware execution
LCA	—	✓	—
QuantumNAS	✓	—	✓
QuantumDARTS	✓	—	—
UQAS-Select (ours)	—	✓	✓
UQAS-Design (ours)	✓	✓	✓

3 Empirical Study: Quantum Architecture Affects Unlearning Capability

Our empirical study investigates whether unlearning capability varies across quantum architectures and whether such variation persists among architectures with comparable predictive accuracy. We study 256 randomly generated architectures on each of four diverse datasets. These span image classification on Digits (Alpaydin & Kaynak, 1998) and Fashion-MNIST (Xiao et al., 2017), tabular classification on Wisconsin Diagnostic Breast Cancer (WDBC) (Street et al., 1993), and quantum phase classification in the transverse-field Ising model (TFIM) (Sachdev, 1999).

Within each dataset, all 256 architectures share the same data split, training procedure, and unlearning update budget. For each unlearning request, we fine-tune a specified subset of model parameters on retained data and compare the unlearned model with the retrained model on that data (Bourtoule et al., 2021). We compare the models' Born distributions, which specify probabilities of computational basis outcomes (Nielsen et al., 2000), using total variation (TV) distance (Gibbs & Su, 2002). We average these distances over common evaluation inputs, requests, and initializations. Lower mean TV distance indicates better unlearning capability (closer agreement with retraining) under this measurement. Appendix C.1 provides details and full results, while Appendix A.5 explains the properties of TV distance.

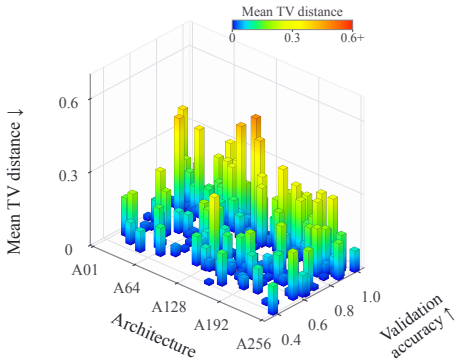


Figure 1: **Quantum architectures exhibit different unlearning capability on Digits.**

Figure 1 reveals substantial variation in mean TV distance across architectures on Digits. The same pattern appears across all four datasets in Figure 6 in Appendix C.1. Under the same procedure, unlearned models differ substantially from retrained models. Predictive accuracy varies across the population, motivating examination of architectures with comparable accuracy.

We then further investigate the unlearning capability of architectures whose accuracy falls within $[A^\star\text{-}3\%, A^\star]$, where $A^\star$ is the best mean validation accuracy of each dataset. Substantial variation still remains on all four datasets, with the full ranges and candidate counts reported in Appendix Table 8. For example, two Fashion-MNIST architectures attain the same mean validation accuracy of 98.5%, yet their mean TV distances are 0.0081 and 0.3894. Thus, *architectures that appear equally suitable for prediction can respond very differently to the same unlearning procedure*.

4 Problem Formulation and Proposed Metric

Building on the finding in Section 3, we propose to integrate unlearning capability and accuracy as key criteria in QAS. One of the main challenges is *how to quantify unlearning capability*. In this section, we define the metric (Unlearning Discrepancy) and present the formulated QAS objective.

4.1 Problem Setup

Let $D = \{(x_i, y_i)\}_{i=1}^{N}$ denote the training set of N examples, where x_i denotes the input features and y_i denotes the class label. Let n be the number of qubits and Ω the architecture search space. Each $\alpha \in \Omega$ specifies ordered unitary blocks, entangling connectivity, and trainable parameter locations (Schuld et al., 2020, Cerezo et al., 2021). For architecture α, let Θ_α denote its parameter space and $\theta \in \Theta_\alpha$ a parameter vector that specifies the parameterized unitary $U_\alpha(\theta)$.

For input x, let ρ_x denote the density operator of the encoded quantum state. A bit string $z \in \{0,1\}^n$ indexes the computational basis state $|z\rangle$. The Born rule (Nielsen et al., 2000) defines the model output distribution $p_\alpha(\theta, x)$, whose probability of outcome z is

$$p_\alpha(\theta, x)_z = \mathrm{Tr}\left[|z\rangle\langle z| U_\alpha(\theta)\rho_x U_\alpha^\dagger(\theta)\right], \qquad z \in \{0,1\}^n. \tag{1}$$

Training architecture α on D yields the trained parameters $\theta_D^\alpha = \mathrm{Train}(D, \alpha)$.

An unlearning request specifies a nonempty proper subset F of D. The request space is $\mathfrak{F} = \{F \subset D \mid 1 \le |F| < N\}$, and the retained dataset is $R = D \setminus F$. For request F, the unlearning operator $\mathcal{U}$ updates the trained parameters of architecture α to $\theta_{\mathcal{U}}^\alpha(F)$:

$$\theta_{\mathcal{U}}^\alpha(F) = \mathcal{U}(\theta_D^\alpha, F, R, \alpha). \tag{2}$$

To establish a reference for unlearning evaluation, we retrain the same architecture α from scratch on R and use the retrained model as a gold-standard reference (Guo et al., 2019, Neel et al., 2021, Georgiev et al., 2025). The retrained model is represented by its parameters θ_{-F}^α,

$$\theta_{-F}^\alpha = \mathrm{Train}(R, \alpha). \tag{3}$$

4.2 Unlearning Discrepancy

We define *unlearning discrepancy* (denoted as $\Delta_{\mathcal{U}}$), a metric to quantify the unlearning capability of a quantum architecture. The rationale is that if an unlearned model is effective, its Born distribution should be very close to that of a retrained model. Specifically, given two Born distributions p and q, we use the total variation (TV) distance to measure their similarity, denoted as $\mathrm{TV}(p, q) = \frac{1}{2}\sum_z |p_z - q_z|$, where p_z and q_z are the probabilities of outcome $z \in \{0,1\}^n$ (Gibbs & Su, 2002).

Given a nonempty evaluation set $\mathcal{B}$ of size $|\mathcal{B}|$, we then mathematically define the unlearning discrepancy of architecture α for unlearning request F as

$$\Delta_{\mathcal{U}}(\alpha, F, \mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{(x,y)\in\mathcal{B}} \mathrm{TV}\left(p_\alpha(\theta_{\mathcal{U}}^\alpha(F), x), p_\alpha(\theta_{-F}^\alpha, x)\right). \tag{4}$$

The discrepancy lies in $[0, 1]$, where a lower value indicates that the unlearned model is closer to the retrained model and therefore has stronger unlearning capability.

4.3 QAS Objective

To characterize unlearning capability across possible post-deployment requests, let $\mathcal{P}_{\mathrm{req}}$ denote an unlearning request distribution over the request space $\mathfrak{F}$ that captures variation in request type, unlearned data size, and data sampling. By aggregating $\Delta_{\mathcal{U}}(\alpha, F, \mathcal{B})$ over requests $F \sim \mathcal{P}_{\mathrm{req}}$, the expected deployment discrepancy of architecture α is defined as

$$R_{\mathcal{U}}(\alpha) = \mathbb{E}_{F\sim\mathcal{P}_{\mathrm{req}}}\left[\Delta_{\mathcal{U}}(\alpha, F, \mathcal{B})\right]. \tag{5}$$

A lower $R_{\mathcal{U}}(\alpha)$ indicates stronger expected post-deployment unlearning capability.

The formulated QAS aims to identify the architecture that minimizes $R_{\mathcal{U}}$ subject to a predictive accuracy constraint. Let $\alpha_{\mathrm{acc}} \in \Omega$ denote the baseline architecture that has the highest validation

accuracy among the candidates evaluated by accuracy-based QAS, and let A_{acc} denote this accuracy. A tolerance $\tau \geq 0$ specifies the maximum allowed decrease from A_{acc}. For a finite candidate set $\mathcal{S} \subseteq \Omega$ containing α_{acc}, let $\mathcal{S}_\tau$ denote the subset of architectures satisfying $A_{\text{val}}(\alpha) \geq A_{\text{acc}} - \tau$. The target architecture $\alpha^\star_{\mathcal{S}}$ is

$$\alpha^\star_{\mathcal{S}} \in \arg\min_{\alpha \in \mathcal{S}_\tau} R_{\mathcal{U}}(\alpha). \tag{6}$$

5 UQAS: Unlearning-driven Quantum Architecture Search

QAS occurs before post-deployment unlearning requests arrive, making it impossible to directly calculate $R_{\mathcal{U}}(\alpha)$ during QAS. To address this challenge, we propose UCE, an Unlearning Capability Evaluator that assesses architecture unlearning capability using a set of synthetic unlearning requests, and outputs an unlearning score. In this work, we consider two practical QAS settings. **1)** In well-studied quantum domains, existing QAS studies have established a pool of candidate architectures, allowing architecture selection from the available pool. **2)** In less-explored domains, no such pool exists and a new architecture needs to be designed. Accordingly, we develop UQAS-Select for the first setting and UQAS-Design for the second. Figure 2 illustrates the UQAS framework.

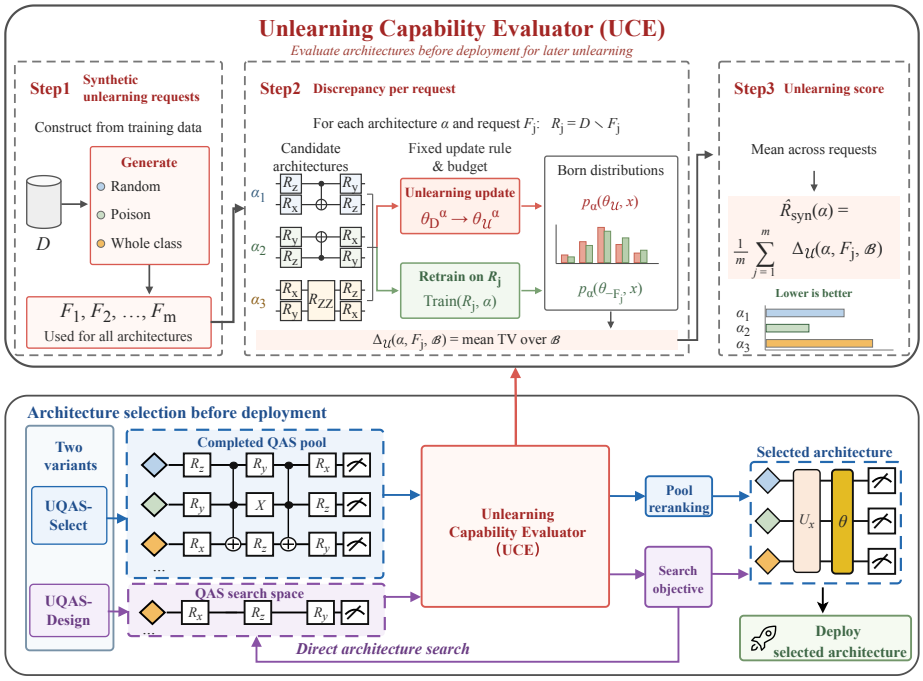


Figure 2: **Overview of the UQAS framework.**

5.1 Unlearning Capability Evaluator

UCE constructs synthetic unlearning requests from the available training data to represent potential unlearning needs after deployment. It integrates random, whole class, and poison requests into a unified formulation, covering sample removal, class removal, and removal of poisoned training samples. This integration enables UCE to characterize candidate architectures under different unlearning needs within the same framework. Appendix B.2 provides the construction details.

Let $\mathcal{P}_{\text{syn}}$ denote the distribution induced by the synthetic request construction. UCE samples $m \geq 1$ synthetic requests from $\mathcal{P}_{\text{syn}}$ and evaluates all candidate architectures using the constructed requests. UCE aggregates the unlearning discrepancies in Eq. (4) across these requests as

$$\widehat{R}_{\text{syn}}(\alpha) = \frac{1}{m} \sum_{j=1}^{m} \Delta_{\mathcal{U}}(\alpha, F_j, \mathcal{B}). \tag{7}$$

A lower UCE score indicates stronger estimated unlearning capability.

The UCE score therefore characterizes architecture behavior under $\mathcal{P}_{\mathrm{syn}}$. Let $\mathcal{P}_{\mathrm{req}}$ denote the request distribution after deployment. When $\mathcal{P}_{\mathrm{syn}} = \mathcal{P}_{\mathrm{req}}$, the expected UCE score corresponds to the deployment objective. When the distributions differ, UCE can still identify architecture advantages that transfer to unseen requests from $\mathcal{P}_{\mathrm{req}}$ under the conditions analyzed in Appendix A.4. The appendix provides the corresponding analysis of score estimation and request distribution shift.

5.2 UQAS-SELECT

By leveraging an existing pool of candidate architectures, UQAS-Select selects the final architecture directly from it, reducing the cost of generating candidates. Within the pool Ω_{pool}, we choose the architecture with the highest validation accuracy as the baseline α_{acc}. The feasible set under the accuracy tolerance τ is

$$\Omega_{\mathrm{pool},\tau} = \{\alpha \in \Omega_{\mathrm{pool}} \mid A_{\mathrm{val}}(\alpha) \geq A_{\mathrm{acc}} - \tau\}. \tag{8}$$

UCE evaluates each architecture in $\Omega_{\mathrm{pool},\tau}$, and UQAS-Select chooses the architecture with the lowest unlearning score:

$$\alpha_{\mathrm{Select}} \in \arg\min_{\alpha \in \Omega_{\mathrm{pool},\tau}} \widehat{R}_{\mathrm{syn}}(\alpha). \tag{9}$$

Thus, UQAS-Select reranks the accuracy-qualified candidates according to their estimated unlearning capability, and selects the top one. Appendix A.3 provides the detailed UQAS-Select algorithm.

5.3 UQAS-DESIGN

UQAS-Design is proposed to generate and search new architectures through mutation inspired by regularized evolution (Real et al., 2019). During this procedure, UQAS-Design uses both the UCE score and validation accuracy to guide architecture generation and population updates.

Let $\mathcal{S}_{\mathrm{Design}}$ denote the set of architectures evaluated during the generation. After the generation, UQAS-Design selects the accuracy-qualified architecture with the lowest UCE score:

$$\alpha_{\mathrm{Design}} \in \arg\min_{\substack{\alpha \in \mathcal{S}_{\mathrm{Design}} \\ A_{\mathrm{val}}(\alpha) \geq A_{\mathrm{acc}} - \tau}} \widehat{R}_{\mathrm{syn}}(\alpha). \tag{10}$$

To assess improvement beyond the synthetic requests used for generation, we perform a confirmation step by comparing α_{Design} and α_{acc} on the unseen synthetic requests from $\mathcal{P}_{\mathrm{syn}}$. We retain α_{Design} as the final choice $\hat{\alpha}_{\mathrm{Design}}$ only if the bootstrap lower confidence bound on its mean discrepancy reduction is positive. If this condition fails or no candidate satisfies the accuracy constraint, we retain α_{acc}. Appendix A.3 provides the UQAS-Design algorithm details.

6 EXPERIMENTS

We evaluate UQAS in simulation and on IBM quantum hardware. In simulation, we examine unlearning capability improvement, computational costs, and robustness to finite measurement shots. Hardware experiments evaluate whether the resulting unlearning gains persist under device noise.

6.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate binary image classification on Digits (Alpaydin & Kaynak, 1998) and Fashion-MNIST (Xiao et al., 2017), tabular classification on WDBC (Street et al., 1993), and phase classification of TFIM ground states (Carrasquilla & Melko, 2017, Sachdev, 1999).

Baseline. We use the architecture with the highest validation accuracy as the baseline. We also include a pool oracle, which selects the architecture with the lowest mean unlearning discrepancy among pool candidates satisfying the accuracy constraint.

Implementation details. We use NumPy (Harris et al., 2020) for simulation. UQAS-Select evaluates trained pools generated by random search (Random) (Bergstra & Bengio, 2012, Li & Talwalkar,

2020), regularized evolution (Evolution) (Real et al., 2019), Monte Carlo tree search (MCTS) (Browne et al., 2012), and differentiable quantum architecture search (DQAS) (Zhang et al., 2022). UQAS-Design uses regularized evolution over the architecture space. Computational cost is measured on an Intel Xeon Platinum 8352V processor with an NVIDIA GeForce RTX 4090 GPU. Hardware experiments use Qiskit (Javadi-Abhari et al., 2024) on `ibm_marrakesh`, `ibm_fez`, and `ibm_kingston` (IBM Quantum, 2026).

Unlearning setup. We use masked fine tuning on retained data as the default unlearning operator (Goel et al., 2022), freezing the first half of the trained parameters and updating the remainder. The validation accuracy tolerance is τ=0.03. Appendix B provides implementation details, and Appendix C.5 reports results with three additional unlearning operators: full fine tuning (Jia et al., 2023), NegGrad+ (Kurmanji et al., 2023), and adapted diagonal influence (Koh & Liang, 2017).

6.2 Simulation Experiments

Unlearning capability improvement of UQAS-Select. Table 2 summarizes unlearning capability improvements across sixteen dataset and search method combinations under three request types.

Table 2: **UQAS-Select across QAS pools and unlearning request types.**

		Mean unlearning discrepancy ↓							
		Random		Evolution		MCTS		DQAS	
Request type	Task	Baseline	Ours	Baseline	Ours	Baseline	Ours	Baseline	Ours
Random	Digits	0.062	0.030	0.084	0.052	0.157	0.037	0.182	0.054
	Fashion-MNIST	0.149	0.009	0.120	0.014	0.184	0.011	0.104	0.013
	WDBC	0.232	0.028	0.221	0.087	0.352	0.014	0.182	0.045
	TFIM	0.071	0.011	0.071	0.013	0.126	0.037	0.082	0.014
Poison	Digits	0.123	0.067	0.122	0.085	0.278	0.114	0.256	0.105
	Fashion-MNIST	0.239	0.036	0.168	0.069	0.284	0.038	0.185	0.050
	WDBC	0.246	0.139	0.233	0.120	0.359	0.038	0.239	0.093
	TFIM	0.155	0.029	0.155	0.035	0.351	0.075	0.382	0.087
Whole class	Digits	0.582	0.468	0.544	0.490	0.561	0.455	0.605	0.521
	Fashion-MNIST	0.449	0.323	0.434	0.346	0.505	0.268	0.443	0.298
	WDBC	0.718	0.630	0.597	0.587	0.772	0.575	0.668	0.606
	TFIM	0.373	0.310	0.373	0.303	0.596	0.472	0.557	0.420

UQAS-Select reduces mean unlearning discrepancy in all sixteen settings for all three request types. Averaged across the sixteen settings, the reductions relative to selection by the baseline are 80% for random requests, 69% for poison instances, and 19% for whole-class removal. For whole-class removal, architectures are selected using only synthetic random requests, yet UQAS-Select still improves all sixteen settings. UCE thus identifies architectures with lower unlearning discrepancy within existing trained pools, improving post-deployment unlearning without repeating architecture search. Appendix C.2 compares UCE with alternative scoring rules, while Appendices C.4 and C.6 provide additional evaluations on unseen data and synthetic requests, accuracy after unlearning, and unlearning fractions from 5% to 50%.

Unlearning capability improvement of UQAS-Design. We compare UQAS-Design, UQAS-Select, and a pool oracle. Across all tasks and request types in Table 3, UQAS-Design finds feasible architectures with lower mean discrepancy than the baseline, while mean validation accuracy differs by at most 0.012. Final mean gains after confirmation and baseline fallback remain positive in every setting. For random requests, final mean discrepancy is 63%–86% lower than the baseline. Appendix C.3 provides additional UQAS-Design results.

Table 4 compares UQAS-Select and UQAS-Design with the pool oracle. UQAS-Design outperforms UQAS-Select on all four tasks. It also improves over the pool oracle on Fashion-MNIST, WDBC, and TFIM, while remaining 0.003 higher on Digits. These results show that UCE can guide architecture search toward candidates with lower unlearning discrepancy beyond the existing pool.

Table 3: **UQAS-Design across unlearning request types.**

Request type	Task	Mean unlearning discrepancy ↓		Mean gain ↑		Mean validation accuracy ↑
		Baseline	Ours	Before confirmation	Final	Baseline / Ours
Random	Digits	0.120	0.042	+0.077	+0.069	0.975 / 0.970
	Fashion-MNIST	0.124	0.007	+0.117	+0.107	0.996 / 0.986
	WDBC	0.271	0.055	+0.217	+0.193	0.906 / 0.902
	TFIM	0.042	0.009	+0.033	+0.033	1.000 / 1.000
Poison	Digits	0.159	0.118	+0.041	+0.035	0.980 / 0.969
	Fashion-MNIST	0.143	0.034	+0.110	+0.098	0.996 / 0.984
	WDBC	0.287	0.086	+0.202	+0.177	0.906 / 0.907
	TFIM	0.133	0.037	+0.095	+0.102	1.000 / 1.000
Whole class	Digits	0.562	0.519	+0.043	+0.060	0.975 / 0.970
	Fashion-MNIST	0.431	0.282	+0.149	+0.157	0.996 / 0.986
	WDBC	0.648	0.552	+0.096	+0.092	0.906 / 0.902
	TFIM	0.378	0.275	+0.104	+0.126	1.000 / 1.000

Table 4: **Mean unlearning discrepancy within and beyond an existing QAS pool.**

Task	Mean unlearning discrepancy ↓			
	Baseline	UQAS-Select	Pool oracle	UQAS-Design
Digits	0.120	0.046	0.039	0.042
Fashion-MNIST	0.124	0.011	0.010	0.007
WDBC	0.271	0.144	0.100	0.055
TFIM	0.042	0.026	0.013	0.009

Computational cost. We compare the computational cost of UQAS-Select and UQAS-Design across circuit widths. Figure 3 reports runtime, cost function evaluations, and floating-point operations (FLOPs). Appendix B.3 provides the measurement details.

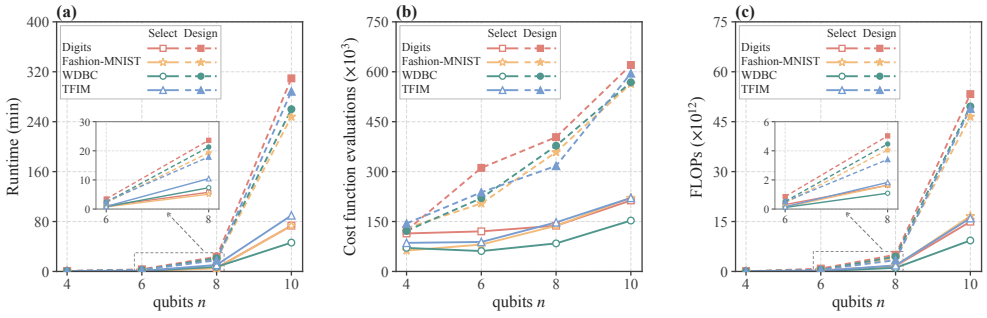


Figure 3: **Computational cost across qubit counts.**

Across the four datasets and circuit widths of four, six, eight, and ten qubits, UQAS-Select requires 29%–82% less runtime and 7%–78% fewer cost function evaluations than UQAS-Design. UQAS-Select also uses fewer FLOPs in all comparisons except Digits at four qubits. These results show the computational trade-off between pool reuse and direct architecture generation.

Robustness to finite shots. We evaluate whether UCE remains effective when its scores are estimated from finite-shot measurements. Across four datasets and four QAS pool sources, UQAS-Select lowers mean discrepancy in 15 of the 16 combinations at 128 shots, with reductions of 40.6%–76.7% when averaged across QAS methods within each dataset. All sixteen combinations improve at 1024, 2048, and 4096 shots. Appendix C.7 provides detailed finite-shot evaluation results.

At 2048 shots per distribution, UQAS-Design reduces mean unlearning discrepancy by 36%–90% relative to UQAS-Select across the four datasets, as shown in Table 5. UQAS-Design requires 1.79–

$6.67\times$ the measurement budget of UQAS-Select, showing that its lower discrepancy comes with higher sampling cost. Appendix C.7 provides the detailed shot accounting.

Table 5: **UQAS-Design versus UQAS-Select at 2048 shots.**

	Mean unlearning discrepancy ↓				
Task	Baseline	UQAS-Select	UQAS-Design	Reduction (%) ↑	Total shot ratio (Design/Select)
Digits	0.1839	0.0369	0.0134	63.6%	2.16×
Fashion-MNIST	0.1545	0.1008	0.0104	89.6%	1.79×
WDBC	0.2792	0.2786	0.0418	85.0%	6.67×
TFIM	0.0351	0.0168	0.0108	35.7%	1.91×

6.3 Experiments on IBM Quantum Hardware

To examine whether gains persist on quantum hardware on quantum hardware, we evaluate selected circuits on three IBM devices. Appendix B.4 provides hardware measurement details.

Unlearning performance across IBM devices. Table 6 shows that both UQAS variants reduce mean measured discrepancy on all four tasks. UQAS-Design achieves the largest relative reduction of 65% on Fashion-MNIST, while its gain on TFIM is 0.002. These results show that the unlearning gains remain observable on hardware, although their magnitude varies across tasks and devices.

Table 6: **Unlearning performance across three IBM devices.**

			Mean unlearning discrepancy ↓		
Method	Task	Device	Baseline	Ours	Mean gain ↑
UQAS-Select	Digits	Marrakesh	0.094	0.059	+0.035
	Fashion-MNIST	Kingston	0.059	0.047	+0.012
	WDBC	Kingston	0.295	0.190	+0.105
	TFIM	Kingston	0.146	0.131	+0.015
UQAS-Design	Digits	Fez	0.107	0.060	+0.047
	Fashion-MNIST	Fez	0.126	0.044	+0.082
	WDBC	Kingston	0.332	0.156	+0.176
	TFIM	Kingston	0.126	0.124	+0.002

Comparison of UQAS-Select and UQAS-Design. Table 7 directly compares the two variants on `ibm_kingston` using a shared baseline. Appendix B.4 provides the comparison details.

UQAS-Design reduces mean discrepancy by 7.2%–66.4% relative to the baseline and achieves the lowest discrepancy on all four tasks. UQAS-Select improves Digits and Fashion-MNIST, increases discrepancy by 9.0% on WDBC, and remains nearly unchanged on TFIM. The hardware results therefore show more consistent gains for UQAS-Design across the evaluated tasks.

Table 7: **Comparison on IBM Kingston.**

	Mean unlearning discrepancy ↓		
Task	Baseline	UQAS-Select	UQAS-Design
Digits	0.136	0.059	0.046
Fashion-MNIST	0.091	0.045	0.041
WDBC	0.199	0.217	0.101
TFIM	0.096	0.096	0.089

7 Conclusion

In this work, we have empirically revealed that quantum architecture can significantly influence a model's unlearning capability. Given this observation, we have proposed UQAS, a framework that integrates unlearning capability as a key search objective into QAS. Enabled by the designed UCE, UQAS effectively identifies architectures with strong unlearning capability while maintaining high predictive accuracy. Hardware experiments further show that these gains persist on real quantum devices, while UQAS-Select and UQAS-Design offer complementary trade-offs between search cost and unlearning capability.

REPRODUCIBILITY STATEMENT

Our code is available at https://anonymous.4open.science/r/uqas-code-CEFB/. Appendix A provides algorithmic details and theoretical analysis. Dataset preprocessing, experimental settings, and implementation details are described in Appendix B, with additional experimental details and evaluations provided in Appendix C.

AI USE STATEMENT

Generative AI tools assisted with the preparation of this work, including language polishing, grammar correction, and minor phrasing refinements. The authors reviewed the resulting materials and take full responsibility for the manuscript.

REFERENCES

Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*, 2021.

E. Alpaydin and C. Kaynak. Optical recognition of handwritten digits. UCI Machine Learning Repository, 1998. Dataset.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013. doi: 10.1093/acprof:oso/9780199535255.001.0001. URL https://doi.org/10.1093/acprof:oso/9780199535255.001.0001.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pp. 141–159. IEEE, 2021.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Juan Carrasquilla and Roger G Melko. Machine learning phases of matter. *Nature physics*, 13(5): 431–434, 2017.

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.

Jialin Chen, Zhiqiang Cai, Ke Xu, Di Wu, and Wei Cao. Qubit-wise architecture search method for variational quantum circuits. *arXiv preprint arXiv:2403.04268*, 2024.

Yu-Qin Chen and Shi-Xin Zhang. Superior resilience to poisoning and amenability to unlearning in quantum machine learning. *Nature Communications*, 17(1):3716, 2026.

D Chivilikhin, A Samarin, V Ulyantsev, I Iorsh, Artem R Oganov, and O Kyriienko. Mog-vqe: Multiobjective genetic variational quantum eigensolver. *arXiv preprint arXiv:2007.04424*, 2020.

Carla Crivoi and Radu Tudor Ionescu. Machine unlearning in the era of quantum machine learning: An empirical study. In *International Conference on Pattern Recognition*, pp. 31–47. Springer, 2026.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

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.

Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution*, pp. 569–593. Springer, 1992.

Kristian G Georgiev, Roy Rinberg, Sam Park, Shivam Garg, Andrew Ilyas, Aleksander Madry, and Seth Neel. Machine unlearning via simulated oracle matching. In *International Conference on Learning Representations*, volume 2025, pp. 49693–49731, 2025.

Alison L Gibbs and Francis Edward Su. On choosing and bounding probability metrics. *International statistical review*, 70(3):419–435, 2002.

Shashwat Goel, Ameya Prabhu, Amartya Sanyal, Ser-Nam Lim, Philip Torr, and Ponnurangam Kumaraguru. Towards adversarial evaluations for inexact machine unlearning. *arXiv preprint arXiv:2201.06640*, 2022.

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.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

Yanjun Han, Jiantao Jiao, and Tsachy Weissman. Minimax estimation of discrete distributions under ℓ_1 loss. *IEEE Transactions on Information Theory*, 61(11):6343–6354, 2015.

Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *nature*, 585(7825):357–362, 2020.

Tobias Haug, Kishor Bharti, and MS Kim. Capacity and quantum geometry of parametrized quantum circuits. *PRX Quantum*, 2(4):040309, 2021.

Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

Zhimin He, Maijie Deng, Shenggen Zheng, Lvzhou Li, and Haozhen Situ. Training-free quantum architecture search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 12430–12438, 2024a.

Zhimin He, Jiachun Wei, Chuangtao Chen, Zhiming Huang, Haozhen Situ, and Lvzhou Li. Gradient-based optimization for quantum architecture search. *Neural Networks*, 179:106508, 2024b.

Yuhan Huang, Siyuan Jin, Bei Zeng, and Qiming Shao. Adaptive diversity-based quantum circuit architecture search. *Physical Review Research*, 6(3):033033, 2024.

IBM Quantum. IBM Quantum Platform, 2026. URL `https://quantum.cloud.ibm.com/`. Accessed September 6, 2026.

Raban Iten, Roger Colbeck, Ivan Kukuljan, Jonathan Home, and Matthias Christandl. Quantum circuits for isometries. *Physical Review A*, 93(3):032318, 2016.

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.

Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsity can simplify machine unlearning. *Advances in neural information processing systems*, 36:51584–51605, 2023.

Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 374(2065):20150202, 2016.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. Pmlr, 2017.

Arjun Krishnakumar, Colin White, Arber Zela, Renbo Tu, Mahmoud Safari, and Frank Hutter. Nasbench-suite-zero: Accelerating research on zero cost proxies. *Advances in Neural Information Processing Systems*, 35:28037–28051, 2022.

En-Jui Kuo, Yao-Lung L Fang, and Samuel Yen-Chi Chen. Quantum architecture search via deep reinforcement learning. *arXiv preprint arXiv:2104.07715*, 2021.

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. *Advances in neural information processing systems*, 36:1957–1987, 2023.

Martin Larocca, Nathan Ju, Diego García-Martín, Patrick J Coles, and Marco Cerezo. Theory of overparametrization in quantum neural networks. *Nature Computational Science*, 3(6):542–551, 2023.

Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for nisq-era quantum devices. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems*, pp. 1001–1014, 2019.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, pp. 367–377. PMLR, 2020.

Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

Vincenzo Lipardi, Domenica Dibenedetto, Georgios Stamoulis, and Mark HM Winands. Quantum circuit design using a progressive widening enhanced monte carlo tree search. *Advanced Quantum Technologies*, 8(10):e2500093, 2025.

Matt Lourens, Ilya Sinayskiy, Daniel K Park, Carsten Blank, and Francesco Petruccione. Hierarchical quantum circuit representations for neural architecture search. *npj Quantum Information*, 9 (1):79, 2023.

Xudong Lu, Kaisen Pan, Ge Yan, Jiaming Shan, Wenjie Wu, and Junchi Yan. Qas-bench: rethinking quantum architecture search and a benchmark. In *International conference on machine learning*, pp. 22880–22898. PMLR, 2023.

Nausherwan Malik, Zubair Khalid, and Muhammad Faryad. Distribution-guided and constrained quantum machine unlearning. *arXiv preprint arXiv:2601.04413*, 2026.

Darya Martyniuk, Johannes Jung, and Adrian Paschke. Quantum architecture search: a survey. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 1, pp. 1695–1706. IEEE, 2024.

Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems*, pp. 1015–1029, 2019.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic learning theory*, pp. 931–962. PMLR, 2021.

Michael A Nielsen, Isaac L Chuang, et al. *Quantum computation and quantum information*, volume 1. Cambridge university press Cambridge, 2000.

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.

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 *International Conference on Learning Representations*, volume 2024, pp. 32298–32329, 2024.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pp. 606–610. IEEE, 2007.

Subir Sachdev. Quantum phase transitions. *Physics world*, 12(4):33–38, 1999.

Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

Maria Schuld, Alex Bocharov, Krysta M Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.

Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3):032430, 2021.

Thanveer Shaik, Xiaohui Tao, and Haoran Xie. Quantum machine unlearning: Foundations, mechanisms, and taxonomy. *arXiv preprint arXiv:2511.00406*, 2025.

Charles Spearman. The proof and measurement of association between two things. 1961.

W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pp. 861–870. SPIE, 1993.

Junjian Su, Runze He, Guanghui Li, Sujuan Qin, Zhimin He, Haozhen Situ, and Fei Gao. From membership-privacy leakage to quantum machine unlearning. *Physical Review Applied*, 25(4): 044056, 2026.

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.

Enayat Ullah, Tung Mai, Anup Rao, Ryan A Rossi, and Raman Arora. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*, pp. 4126–4142. PMLR, 2021.

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.

John Watrous. *The theory of quantum information*, volume 1. Cambridge university press Cambridge, 2018.

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.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International conference on machine learning*, pp. 7105–7114. PMLR, 2019.

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.

Xianchao Zhu and Xiaokai Hou. Quantum architecture search via truly proximal policy optimization. *Scientific Reports*, 13(1):5157, 2023.

A Model and Method Details

A.1 Conventional Quantum Machine Unlearning

In the conventional workflow, a quantum model is trained and deployed with a selected circuit architecture. When an unlearning request arrives, its trained parameters are updated while the architecture remains fixed, as illustrated in Figure 4. The figure uses α_{acc} for the architecture selected by validation accuracy and omits architecture superscripts from θ_D and $\theta_{\mathcal{U}}$. Here U_x encodes input x, θ_1, θ_2 are trainable parameters, R_x, R_y, R_z are single-qubit rotations, and X is a Pauli-X gate.

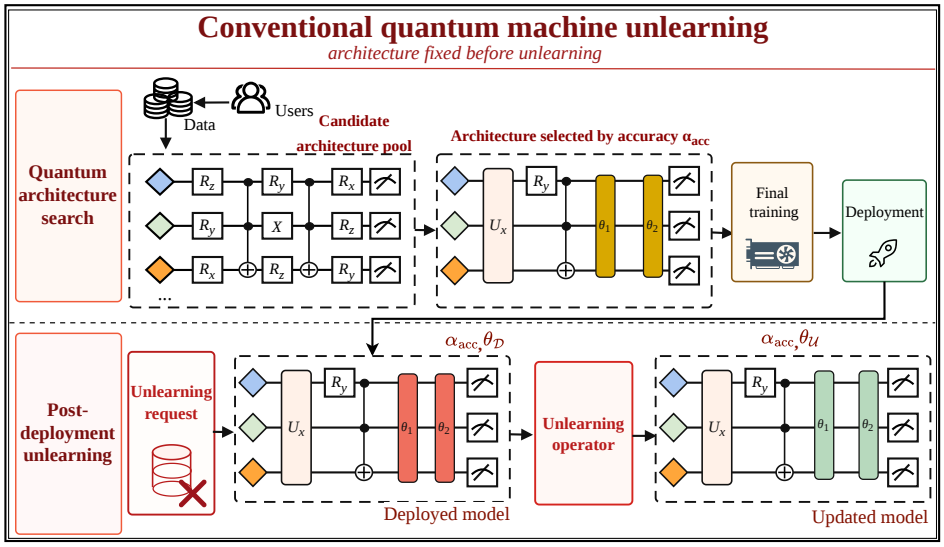


Figure 4: **Conventional quantum machine unlearning.**

A.2 Notation Table

Notation	Description
$D, D_{\text{val}}, D_{\text{eval}}$	Training, validation, and fixed discrepancy evaluation sets.
D_{new}	Separate evaluation set held out until architecture selection is complete.
$\mathcal{B}$	Generic nonempty evaluation set in Eq. (4), instantiated by D_{eval} or D_{new}.
$N, (x_i, y_i)$	Training set size and the input and label pair for example i.
C, c, k	Number of classes, class label, and number of readout bits $k = \lceil \log_2 C \rceil$.
$[x]_\ell$	Coordinate ℓ of a preprocessed input x.
$F \in \mathfrak{F}$	Requested training subset in the space of nonempty proper subsets of D.
R, R_j	Retained training data, with $R = D \setminus F$ and $R_j = D \setminus F_j$ for fixed D.
$\mathcal{P}_{\text{syn}}$	Distribution used to construct synthetic unlearning requests.
$\mathcal{P}_{\text{req}}$	Distribution of post-deployment unlearning requests.
$F_1, \ldots, F_m$	The m synthetic requests drawn from $\mathcal{P}_{\text{syn}}$ and used for all candidate architectures.
$F_j^{\text{req}}, M_{\text{req}}$	Post-deployment evaluation request j and the number of such requests.
$\widetilde{D}_j$	Poisoned training data for poison instance j, with retained set $R_j = \widetilde{D}_j \setminus F_j$.
$\alpha \in \Omega$	Quantum architecture in the search space Ω.
$\theta \in \Theta_\alpha$	Trainable parameter vector in the parameter space of architecture α.
$U_\alpha(\theta)$	Parameterized unitary specified by architecture α.
ρ_x	Density operator of the encoded input x.
$p_\alpha(\theta, x)$	Computational basis Born distribution in Eq. (1).
$q_\alpha(c \mid x, \theta)$	Class probability after grouping and normalizing Born probabilities.
$\mathcal{U}$	The same fixed unlearning operator applied to all architectures.
θ_D^α	Parameters of architecture α trained on D.
$\theta_{\mathcal{U}}^\alpha(F)$	Parameters after applying $\mathcal{U}$ to request F, as in Eq. (2).

Notation	Description (continued)
θ^{α}_{-F}	Parameters of the same architecture retrained on R, as in Eq. (3).
$\theta^{\alpha}_{\widetilde{D}_j}, \theta^{\alpha}_{R_j}$	Parameters trained on poisoned data and retrained on the retained data for poison instance j.
n	Number of qubits.
s	Number of measurement shots per Born distribution.
$\mathrm{TV}(p, q)$	Total variation distance between probability distributions p and q.
$\Delta_{\mathcal{U}}(\alpha, F, \mathcal{B})$	Mean TV between updated and retrained Born distributions over $\mathcal{B}$ for request F, as in Eq. (4).
$\Delta^{\mathrm{eval}}_{\mathrm{obs}}(\alpha, F)$	Abbreviation for $\Delta_{\mathcal{U}}(\alpha, F, D_{\mathrm{eval}})$.
$\Delta^{\mathrm{new}}_{\mathrm{obs}}(\alpha, F)$	Abbreviation for $\Delta_{\mathcal{U}}(\alpha, F, D_{\mathrm{new}})$.
$\Delta^{\mathrm{eval}}_{\mathrm{obs}}(\alpha, F_j \mid \widetilde{D}_j)$	Instance discrepancy using starting parameters trained on $\widetilde{D}_j$ and retraining on R_j.
$R_{\mathcal{U}}(\alpha)$	Expected discrepancy under $\mathcal{P}_{\mathrm{req}}$, as in Eq. (5).
$R_{\mathrm{syn}}(\alpha)$	Expected discrepancy under $\mathcal{P}_{\mathrm{syn}}$.
$\widehat{R}_{\mathrm{syn}}(\alpha)$	UCE unlearning score, the mean discrepancy over m synthetic requests in Eq. (7).
$A_{\mathrm{val}}(\alpha)$	Validation accuracy of architecture α.
$\alpha_{\mathrm{acc}}, A_{\mathrm{acc}}$	Accuracy baseline architecture, with $A_{\mathrm{acc}} = A_{\mathrm{val}}(\alpha_{\mathrm{acc}})$.
τ	Maximum allowed absolute decrease in validation accuracy from A_{acc}.
$\mathcal{S}, \mathcal{S}_\tau$	Candidate set and its subset satisfying $A_{\mathrm{val}}(\alpha) \geq A_{\mathrm{acc}} - \tau$.
$\alpha^{\star}_{\mathcal{S}}$	Architecture minimizing $R_{\mathcal{U}}$ over $\mathcal{S}_\tau$, as in Eq. (6).
$\Omega_{\mathrm{pool}}, \Omega_{\mathrm{pool},\tau}$	Trained QAS pool and its subset satisfying the accuracy requirement.
α_{Select}	Architecture selected from $\Omega_{\mathrm{pool},\tau}$ by Eq. (9).
$\lambda_{\mathrm{acc}}, [a]_+$	Accuracy penalty weight and positive part $[a]_+ = \max(a, 0)$.
$J_{\mathrm{Design}}(\alpha)$	Objective combining the UCE score and accuracy penalty, defined in Appendix A.3.
$\mathcal{S}_{\mathrm{Design}}, \mathcal{C}_{\mathrm{Design}}$	Architectures evaluated during Design generation and their subset satisfying the accuracy requirement.
$\alpha_{\mathrm{Design}}, \hat{\alpha}_{\mathrm{Design}}$	Provisional Design architecture and final architecture after confirmation or fallback to α_{acc}, as in Eq. (11).
Q_j, m_g, g_j	Confirmation request j, confirmation request count, and paired discrepancy gain over α_{acc}.
$q^{\mathrm{boot}}_{0.05}$	Fifth percentile of the bootstrap distribution of mean confirmation gain.
K, T	Number of eligible Select candidates and number of evaluated Design candidates.
$\tilde{\alpha}_{v,\kappa}$	Final architecture of method v in paired comparison κ. For Design, this is $\hat{\alpha}_{\mathrm{Design},\kappa}$.
$\bar{\Delta}_\kappa(\alpha), w_\kappa$	Mean post-deployment discrepancy for architecture α in comparison κ, and the comparison's aggregation weight.
$\widehat{G}^{v}_{\mathrm{req}}$	Mean paired discrepancy reduction on post-deployment requests for method v, defined in Eq. (17).
$\overline{\Delta}^{\mathrm{acc}}, \overline{\Delta}^{v}, r^v$	Weighted mean baseline and final discrepancies, and relative reduction in Eq. (19).
$\overline{\Delta}^{v}_{d}, r^v_d$	The same quantities restricted to dataset d.
$G_{\mathrm{new}}, \Delta A_{\mathrm{new}}$	Discrepancy gain and difference in test accuracy before unlearning on D_{new}, defined in Appendix C.4.
ρ, r	Spearman rank correlation and Pearson correlation.
$\sigma^2_\alpha, \bar{\sigma}^2$	Variance of request discrepancy under $\mathcal{P}_{\mathrm{syn}}$ and its maximum over the eligible pool.
δ, ϵ	Failure probability and uniform bound on score estimation error in Proposition A.1.
$J^{\mathrm{Born}}_{\alpha}(x)$	Jacobian of the Born distribution at θ^{α}_{D}.
$J^{\mathrm{class}}_{\alpha}$	Jacobian of class probabilities at θ^{α}_{D}.
e_j, S_α	Unit parameter direction j, and the set of coordinates allowed to change under the masked update.
$\delta^{\alpha}_{\mathcal{U}}(F), \delta^{\alpha}_{R}(F)$	Updated and retrained parameter displacements from θ^{α}_{D}.
$L_{\alpha,F}$	Bound on the summed absolute second directional derivatives used in Eq. (15).

A.3 Selection Algorithms and Confirmation

UQAS-Select. Algorithm 1 implements Eq. (9).

Algorithm 1 UQAS-Select.

Inputs QAS pool Ω_{pool}, trained parameters $\{\theta_D^\alpha\}_{\alpha\in\Omega_{\text{pool}}}$, and stored validation accuracies $\{A_{\text{val}}(\alpha)\}_{\alpha\in\Omega_{\text{pool}}}$ training set D and discrepancy evaluation set D_{eval} fixed unlearning operator $\mathcal{U}$ and synthetic request distribution $\mathcal{P}_{\text{syn}}$ tolerance τ and number of synthetic requests m

choose $\alpha_{\text{acc}} \in \arg\max_{\alpha\in\Omega_{\text{pool}}} A_{\text{val}}(\alpha)$ and set $A_{\text{acc}} \leftarrow A_{\text{val}}(\alpha_{\text{acc}})$
$\Omega_{\text{pool},\tau} \leftarrow \{\alpha \in \Omega_{\text{pool}} \mid A_{\text{val}}(\alpha) \geq A_{\text{acc}} - \tau\}$
sample $F_1, \ldots, F_m \overset{\text{i.i.d.}}{\sim} \mathcal{P}_{\text{syn}}$
$\widehat{R}_{\text{syn}}(\alpha) \leftarrow \frac{1}{m}\sum_{j=1}^{m} \Delta_{\text{obs}}^{\text{eval}}(\alpha, F_j)$ for every $\alpha \in \Omega_{\text{pool},\tau}$
choose $\alpha_{\text{Select}} \in \arg\min_{\alpha\in\Omega_{\text{pool},\tau}} \widehat{R}_{\text{syn}}(\alpha)$
return α_{Select}

UQAS-Design. Algorithm 2 generates candidates using the same synthetic requests and confirms the provisional choice on newly sampled requests. UQAS-Design and accuracy based QAS use the same architecture space and mutation operations with separate random seeds. Candidate generation minimizes

$$J_{\text{Design}}(\alpha) = \widehat{R}_{\text{syn}}(\alpha) + \lambda_{\text{acc}}\big[A_{\text{acc}} - \tau - A_{\text{val}}(\alpha)\big]_+,$$

where $[a]_+ = \max(a, 0)$ and $\lambda_{\text{acc}} > 0$ weights the accuracy penalty. The baseline and accuracy threshold are fixed before candidate generation.

Algorithm 2 UQAS-Design with confirmation.

Inputs training set D, validation set D_{val}, and discrepancy evaluation set D_{eval} QAS search space Ω and fixed unlearning operator $\mathcal{U}$ synthetic request distribution $\mathcal{P}_{\text{syn}}$, tolerance τ, penalty λ_{acc}, and request counts m, m_g candidate generation budget

perform QAS based on validation accuracy, fix α_{acc}, and set $A_{\text{acc}} \leftarrow A_{\text{val}}(\alpha_{\text{acc}})$
sample synthetic requests $F_1, \ldots, F_m \overset{\text{i.i.d.}}{\sim} \mathcal{P}_{\text{syn}}$
initialize regularized evolution and $\mathcal{S}_{\text{Design}} \leftarrow \varnothing$
while the generation budget remains **do**
  generate $\alpha \in \Omega$ with the evolution operator and train θ_D^α
  compute $A_{\text{val}}(\alpha)$, $\widehat{R}_{\text{syn}}(\alpha)$, and $J_{\text{Design}}(\alpha)$
  update the population and add α to $\mathcal{S}_{\text{Design}}$
end while
$\mathcal{C}_{\text{Design}} \leftarrow \{\alpha \in \mathcal{S}_{\text{Design}} \mid A_{\text{val}}(\alpha) \geq A_{\text{acc}} - \tau\}$
if $\mathcal{C}_{\text{Design}} = \varnothing$ **then**
  $\hat{\alpha}_{\text{Design}} \leftarrow \alpha_{\text{acc}}$
else
  choose $\alpha_{\text{Design}} \in \arg\min_{\alpha\in\mathcal{C}_{\text{Design}}} \widehat{R}_{\text{syn}}(\alpha)$
  sample confirmation requests $Q_1, \ldots, Q_{m_g} \overset{\text{i.i.d.}}{\sim} \mathcal{P}_{\text{syn}}$
  $g_j \leftarrow \Delta_{\text{obs}}^{\text{eval}}(\alpha_{\text{acc}}, Q_j) - \Delta_{\text{obs}}^{\text{eval}}(\alpha_{\text{Design}}, Q_j)$ for $j = 1, \ldots, m_g$
  set $\hat{\alpha}_{\text{Design}}$ from Eq. (11)
end if
return $\hat{\alpha}_{\text{Design}}$

The feasible subset $\mathcal{C}_{\text{Design}}$ in Algorithm 2 yields the provisional choice α_{Design} when nonempty. We draw $m_g \geq 1$ confirmation requests $Q_1, \ldots, Q_{m_g}$ independently from $\mathcal{P}_{\text{syn}}$, separate from those used for generation and selection, and compute gains $g_j = \Delta_{\text{obs}}^{\text{eval}}(\alpha_{\text{acc}}, Q_j) - \Delta_{\text{obs}}^{\text{eval}}(\alpha_{\text{Design}}, Q_j)$ on the same requests. Let $q_{0.05}^{\text{boot}}$ be the fifth percentile of the mean gain over 2,000 bootstrap resamples of these requests, giving a one-sided 95% bootstrap lower confidence bound (Efron, 1992). Empty

feasible sets skip confirmation, giving the final rule

$$\hat{\alpha}_{\text{Design}} = \begin{cases} \alpha_{\text{acc}}, & \mathcal{C}_{\text{Design}} = \varnothing, \\ \alpha_{\text{Design}}, & \mathcal{C}_{\text{Design}} \neq \varnothing \text{ and } q^{\text{boot}}_{0.05}\big(\{g_j\}_{j=1}^{m_g}\big) > 0, \\ \alpha_{\text{acc}}, & \text{otherwise.} \end{cases} \tag{11}$$

With $K = |\Omega_{\text{pool},\tau}|$ eligible candidates, Select computes Km discrepancies from a trained pool. Design trains $T = |\mathcal{S}_{\text{Design}}|$ candidates and computes Tm discrepancies during generation, plus $2m_g$ for confirmation when feasible. Each discrepancy requires one unlearning update and one retraining on retained data. Poison requests additionally require training the starting model on poisoned data.

A.4 Unlearning Scoring for Unknown Requests

The following two propositions assume fixed training data and parameters, exact Born probabilities, and randomness only in requests. They do not cover poison requests with varying training data. Throughout this analysis, $\mathcal{B} = D_{\text{eval}}$.

Define the population risk under synthetic requests

$$R_{\text{syn}}(\alpha) = \mathbb{E}_{F\sim\mathcal{P}_{\text{syn}}}\big[\Delta^{\text{eval}}_{\text{obs}}(\alpha, F)\big].$$

The empirical score in Eq. (7) averages m synthetic requests. Define the request discrepancy variance and its maximum over the eligible pool by

$$\sigma^2_\alpha = \text{Var}_{F\sim\mathcal{P}_{\text{syn}}}[\Delta^{\text{eval}}_{\text{obs}}(\alpha, F)], \qquad \bar{\sigma}^2 = \max_{\alpha\in\Omega_{\text{pool},\tau}} \sigma^2_\alpha.$$

Proposition A.1 (Uniform score estimation on an existing pool)**.** *Fix $\delta \in (0,1)$ and condition on the training, validation, and evaluation sets, trained parameters, and nonempty feasible pool $\Omega_{\text{pool},\tau}$ of size K. Draw $F_1, \ldots, F_m \overset{\text{i.i.d.}}{\sim} \mathcal{P}_{\text{syn}}$ after training, allowing dependence across architectures. Then $\widehat{R}_{\text{syn}}(\alpha)$ is unbiased for $R_{\text{syn}}(\alpha)$ with variance σ^2_α/m, and with probability at least $1-\delta$,*

$$\max_{\alpha\in\Omega_{\text{pool},\tau}} \left|\widehat{R}_{\text{syn}}(\alpha) - R_{\text{syn}}(\alpha)\right| \le \sqrt{\frac{2\bar{\sigma}^2\log(2K/\delta)}{m}} + \frac{2\log(2K/\delta)}{3m}. \tag{12}$$

When $\mathcal{P}_{\text{syn}} = \mathcal{P}_{\text{req}}$, the bound also holds for $R_{\mathcal{U}}$.

Proof. For each fixed α, $\widehat{R}_{\text{syn}}(\alpha)$ is the mean of m i.i.d. random variables in $[0,1]$, so its expectation and variance have the stated forms. Applying Bernstein's inequality (Boucheron et al., 2013) to each architecture mean and a union bound over the K architectures proves Eq. (12). □

Writing ϵ for the right side of Eq. (12), the empirical minimizer has excess synthetic population risk at most 2ϵ on this event. When $\mathcal{P}_{\text{syn}} = \mathcal{P}_{\text{req}}$, a population risk improvement over α_{acc} greater than 2ϵ therefore yields a lower score. This existing pool bound does not directly cover adaptive Design generation, which uses newly sampled requests for confirmation. Request distribution shift admits a separate bound.

Proposition A.2 (Bound under request shift)**.** *For distributions P, Q on $\mathfrak{F}$, define $\text{TV}(P,Q) = \sup_{E\subseteq\mathfrak{F}} |P(E) - Q(E)|$, where E is a request event (Gibbs & Su, 2002). For any fixed architecture α, let $\|\Delta^{\text{eval}}_{\text{obs}}(\alpha,\cdot)\|_\infty$ be its maximum discrepancy over requests. Then*

$$|R_{\text{syn}}(\alpha) - R_{\mathcal{U}}(\alpha)| \le \|\Delta^{\text{eval}}_{\text{obs}}(\alpha,\cdot)\|_\infty \,\text{TV}(\mathcal{P}_{\text{syn}}, \mathcal{P}_{\text{req}}). \tag{13}$$

Proof. Set $P = \mathcal{P}_{\text{syn}}$ and $Q = \mathcal{P}_{\text{req}}$. Let $f(F) = \Delta^{\text{eval}}_{\text{obs}}(\alpha, F)$ and $M = \|f\|_\infty$. Since f is nonnegative, $\mathbb{E}_P f - \mathbb{E}_Q f = \int_0^M \{P(f > t) - Q(f > t)\}\, dt$. The result follows by taking absolute values and applying the definition of total variation. □

Since $\Delta^{\text{eval}}_{\text{obs}}(\alpha, F) \in [0,1]$, the architecture with lower risk under $\mathcal{P}_{\text{syn}}$ also has lower risk under post-deployment requests when the gap between two fixed architectures exceeds $2\,\text{TV}(\mathcal{P}_{\text{syn}}, \mathcal{P}_{\text{req}})$.

Shot complexity. For 2^n outcomes and failure probability $\delta \in (0,1)$, empirical TV error scales as $O(\sqrt{2^n/s})$ (Han et al., 2015), plus a concentration term $O(\sqrt{\log(1/\delta)/s})$ (Boucheron et al., 2013). Full distribution estimation is practical for these small circuits, but larger systems may need estimators with lower measurement cost.

A.5 Quantum Architecture Dependence of Unlearning Discrepancy

Architecture affects how parameter displacements change output probabilities. As in local sensitivity analyses using influence functions (Koh & Liang, 2017), fix an architecture α, request F, and input x. Define the Born distribution Jacobian at the trained parameters, with outcome rows, parameter columns, and transpose denoted by $\top$,

$$J_\alpha^{\mathrm{Born}}(x) = \left.\frac{\partial p_\alpha(\theta, x)}{\partial \theta^\top}\right|_{\theta=\theta_D^\alpha}.$$

Suppose parameter coordinate θ_j occurs in exactly one rotation $\exp(-i\theta_j P_j/2)$, with Pauli generator $P_j^2 = I$, identity I, and imaginary unit i. Let $\rho_{x,j}$ be the state immediately after this rotation, $\mathcal{E}_{>j}^{(\alpha)}$ the downstream channel, $\Pi_z = |z\rangle\langle z|$ the measurement projector, and e_j the unit parameter direction. With $[P_j, \rho_{x,j}] = P_j\rho_{x,j} - \rho_{x,j}P_j$,

$$\partial_{\theta_j} p_\alpha(\theta, x)_z = \mathrm{Tr}\left[\Pi_z \mathcal{E}_{>j}^{(\alpha)}\left(-\frac{i}{2}[P_j, \rho_{x,j}]\right)\right] = \frac{p_\alpha(\theta + \frac{\pi}{2}e_j, x)_z - p_\alpha(\theta - \frac{\pi}{2}e_j, x)_z}{2}. \tag{14}$$

The second equality is the parameter shift rule (Mitarai et al., 2018, Schuld et al., 2019).

Define displacements $\delta_{\mathcal{U}}^\alpha(F) = \theta_{\mathcal{U}}^\alpha(F) - \theta_D^\alpha$ and $\delta_R^\alpha(F) = \theta_{-F}^\alpha - \theta_D^\alpha$, with vector norms $\|\cdot\|_1$ and $\|\cdot\|_2$ denoting the sum of absolute components and Euclidean norm. Assume $p_\alpha(\theta, x)$ is twice continuously differentiable on the segments from θ_D^α to the updated and retrained parameters. For direction v, let $\nabla_\theta^2 p_\alpha(\theta, x)[v, v]$ have component $v^\top[\nabla_\theta^2 p_\alpha(\theta, x)_z]v$, where $\nabla_\theta^2 p_\alpha(\theta, x)_z$ is the outcome probability Hessian. Assume $L_{\alpha,F} \geq 0$ satisfies, for all v, both segments, and $(x, y) \in D_{\mathrm{eval}}$,

$$\left\|\nabla_\theta^2 p_\alpha(\theta, x)[v, v]\right\|_1 \leq L_{\alpha,F}\|v\|_2^2.$$

For either displacement $u \in \{\delta_{\mathcal{U}}^\alpha(F), \delta_R^\alpha(F)\}$, the Taylor remainder satisfies

$$\left\|p_\alpha(\theta_D^\alpha + u, x) - p_\alpha(\theta_D^\alpha, x) - J_\alpha^{\mathrm{Born}}(x)u\right\|_1 \leq \frac{L_{\alpha,F}}{2}\|u\|_2^2.$$

The triangle inequality then implies

$$\begin{aligned}\mathrm{TV}\left(p_\alpha(\theta_{\mathcal{U}}^\alpha(F), x), p_\alpha(\theta_{-F}^\alpha, x)\right) \leq \frac{1}{2}\left\|J_\alpha^{\mathrm{Born}}(x)(\delta_{\mathcal{U}}^\alpha(F) - \delta_R^\alpha(F))\right\|_1 \\ + \frac{L_{\alpha,F}}{4}\left(\|\delta_{\mathcal{U}}^\alpha(F)\|_2^2 + \|\delta_R^\alpha(F)\|_2^2\right).\end{aligned} \tag{15}$$

Averaging Eq. (15) over D_{eval} bounds $\Delta_{\mathrm{obs}}^{\mathrm{eval}}(\alpha, F)$. For a masked update, let S_α be the allowed parameter coordinates, $J_{\alpha,S_\alpha}^{\mathrm{Born}}(x)$ the corresponding Jacobian columns, and $\delta_{\mathcal{U},S_\alpha}^\alpha(F)$ the restricted displacement. The first order term then compares $J_{\alpha,S_\alpha}^{\mathrm{Born}}(x)\delta_{\mathcal{U},S_\alpha}^\alpha(F)$ with $J_\alpha^{\mathrm{Born}}(x)\delta_R^\alpha(F)$. Architecture enters through this sensitivity and displacement alignment, while the second order term captures local curvature. UQAS measures their combined effect through the Born distributions.

Figure 5 illustrates two possible update geometries in conceptual retained-data loss landscapes. Under the same unlearning operator, the updated parameters remain separated from the retrained parameters on the left and approach them on the right. For a fixed architecture α and request F, the figure uses θ_D, $\theta_{\mathcal{U}}$, and θ_{-F} to denote θ_D^α, $\theta_{\mathcal{U}}^\alpha(F)$, and θ_{-F}^α, respectively.

Properties of unlearning discrepancy. At a fixed architecture, request, and input, let p, q be the updated and retrained Born distributions. For an outcome subset $S \subseteq \{0,1\}^n$, define $p(S) = \sum_{z\in S} p_z$, and define $q(S)$ analogously. Then (Gibbs & Su, 2002)

$$\mathrm{TV}(p, q) = \max_{S\subseteq\{0,1\}^n} |p(S) - q(S)|.$$

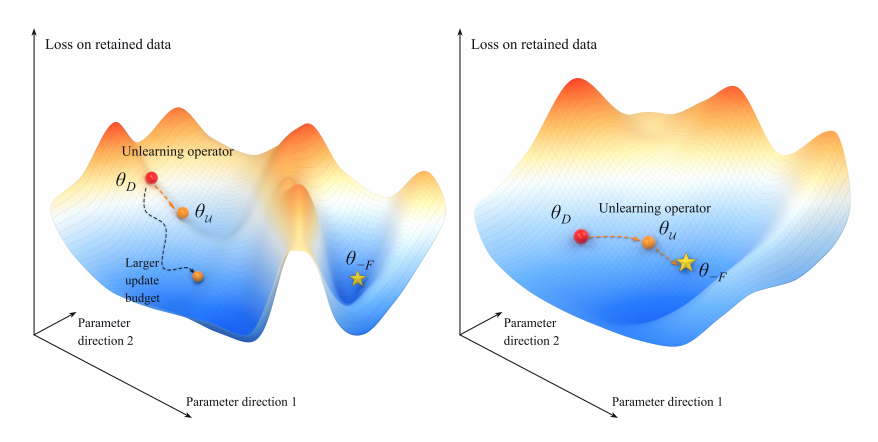


Figure 5: **Unlearning geometry across architectures.**

For known, equally likely distributions, the optimal probability $P^{\star}_{\text{succ}}$ of identifying the source of one outcome is (Watrous, 2018)

$$P^{\star}_{\text{succ}} = \frac{1}{2}\sum_{z} \max\{p_z, q_z\} = \frac{1 + \text{TV}(p, q)}{2}.$$

For the binary tasks in Figure 1, grouping bit strings into classes without postselection cannot increase TV.

Let $\rho_{\mathcal{U}}, \rho_R$ be the corresponding quantum output states and $\text{diag}(p)$ the diagonal matrix with entries p_z. For matrices, $\|\cdot\|_1$ is the trace norm, the sum of singular values. By contractivity of trace distance under measurement (Watrous, 2018),

$$\text{TV}(p, q) = \frac{1}{2}\|\,\text{diag}(p) - \text{diag}(q)\|_1 \le \frac{1}{2}\|\rho_{\mathcal{U}} - \rho_R\|_1.$$

The right-hand side is quantum trace distance. The orthogonal states $|\pm\rangle = (|0\rangle \pm |1\rangle)/\sqrt{2}$, for example, have trace distance one but TV zero in the computational basis. Small measured TV therefore describes agreement under this measurement, not full state closeness or certified removal. Certification concerns the distributions of models induced by randomized training and unlearning (Guo et al., 2019, Ullah et al., 2021).

B Implementation and Experimental Protocol

B.1 Data Encoding and Search Space

We use D for training, D_{val} for validation, D_{eval} for discrepancy evaluation, and D_{new} for testing. The test set D_{new} remains held out until selection is complete, with discrepancy denoted by $\Delta^{\text{new}}_{\text{obs}}(\alpha, F)$. We load Digits classes 3 and 5, Fashion-MNIST classes 0 and 9, and WDBC through scikit-learn (Pedregosa et al., 2011). Preprocessing is fitted on the original training split and fixed thereafter. Inputs are standardized, projected onto n principal components (Jolliffe & Cadima, 2016), then scaled and clipped to $[-1, 1]$. For preprocessed coordinate $[x]_\ell$, the encoded state is

$$|\psi(x)\rangle = \bigotimes_{\ell=1}^{n} \left[R_y\left(\frac{\pi}{2}([x]_\ell + 1)\right)|0\rangle \right]. \tag{16}$$

All candidates use this encoding, which fixes the feature representation across architectures (Havlíček et al., 2019, Schuld et al., 2021, Pérez-Salinas et al., 2020).

For phase classification (Carrasquilla & Melko, 2017), inputs are ground states of the open TFIM chain (Sachdev, 1999) with Hamiltonian $H_n(h) = -\sum_{i=1}^{n-1} Z_i Z_{i+1} - h\sum_{i=1}^{n} X_i$, where X_i, Z_i are Pauli operators at site i and h is the transverse field. Main experiments use $n = 6$. We choose

either $[0, 0.7]$ or $[1.3, 2.5]$ with equal probability, sample h uniformly within it, and directly encode the ground state as $\rho_x = |g_n(h)\rangle\langle g_n(h)|$.

For C classes, readout uses $k = \lceil \log_2 C \rceil$ bits. Let z_i be bit i of outcome z and $b_k(z) = \sum_{i=1}^{k} 2^{k-i} z_i$ its readout value. Retaining groups $b_k(z) < C$ gives the predicted probability of class c,

$$q_\alpha(c \mid x, \theta) = \frac{\sum_{z:\, b_k(z)=c} p_\alpha(\theta, x)_z}{\sum_{z:\, b_k(z)<C} p_\alpha(\theta, x)_z}, \qquad c = 0, \ldots, C-1.$$

The implementation adds 10^{-12} to the denominator for numerical stability. Quantum discrepancy uses the full computational basis distribution before this readout.

Search space. The search space used by all methods contains single-qubit R_x, R_y, R_z rotations, trainable two-qubit R_{ZZ} rotations, and skip blocks, with ring, brick, or sparse controlled-NOT (CNOT) entanglement. Blocks can be inserted or removed. All candidates, including extracted DQAS architectures, are trained independently before validation and unlearning scoring.

B.2 Requests, Optimization, and Aggregation

Request construction. Random requests independently sample 10% of the training data for selection and post-deployment evaluation. Whole class evaluation instead removes all examples with label $C-1$, reusing the architecture selected with synthetic random requests.

Poison instance j corrupts the labels of a sampled 10% subset F_j, producing $\widetilde{D}_j$. Binary labels are flipped and multiclass labels shifted by nonzero offsets modulo C, using identical poisoned indices and labels for all architectures. Each architecture is trained on $\widetilde{D}_j$ to obtain $\theta^\alpha_{\widetilde{D}_j}$, then updated and retrained on the clean retained set $R_j = \widetilde{D}_j \setminus F_j$. Conditioning on the instance gives

$$\theta^\alpha_{\mathcal{U}}(F_j \mid \widetilde{D}_j) = \mathcal{U}(\theta^\alpha_{\widetilde{D}_j}, F_j, R_j, \alpha), \qquad \theta^\alpha_{R_j} = \mathrm{Train}(R_j, \alpha).$$

The instance discrepancy on an evaluation set $\mathcal{B}$ is

$$\Delta_{\mathcal{U}}(\alpha, F_j, \mathcal{B} \mid \widetilde{D}_j) = \frac{1}{|\mathcal{B}|} \sum_{(x,y)\in\mathcal{B}} \mathrm{TV}\Big(p_\alpha(\theta^\alpha_{\mathcal{U}}(F_j \mid \widetilde{D}_j), x), p_\alpha(\theta^\alpha_{R_j}, x)\Big).$$

Using D_{eval} or D_{new} as $\mathcal{B}$ gives $\Delta^{\mathrm{eval}}_{\mathrm{obs}}(\alpha, F_j \mid \widetilde{D}_j)$ or $\Delta^{\mathrm{new}}_{\mathrm{obs}}(\alpha, F_j \mid \widetilde{D}_j)$. For poison requests, Eq. (7) averages these conditional discrepancies over the synthetic requests. Evaluation and Design confirmation use separately sampled poison requests. Pool generation and validation accuracy use models trained on the original clean data D.

Optimization. Within each comparison, we use the same training procedure, unlearning operator, and update budget across architectures. The frozen prefix of the parameter vector contains half its coordinates, rounded down. Training and retraining are deterministic per seed and use the same architecture and initialization, with a common seed schedule across paired comparisons. The architecture observation study, cost measurements, and finite shot pool evaluation use 160 training, 90 validation, and 60 discrepancy evaluation examples, with 30 training and retraining epochs and four masked update epochs. The observation and finite shot pool studies use learning rates 0.1 for training and retraining and 0.02 for updates. Cost measurements additionally use 50 test examples. Request counts and search budgets are specified with each comparison.

Aggregation. Validation accuracy uses models trained on clean data. We first average requests within each seed and then weight seeds equally. Percentage reductions use unrounded means, and 95% intervals are obtained from $10{,}000$ paired bootstrap resamples of seeds (Efron, 1992). Let $\tilde{\alpha}_{\mathrm{Select}} = \alpha_{\mathrm{Select}}$ and $\tilde{\alpha}_{\mathrm{Design}} = \hat{\alpha}_{\mathrm{Design}}$. After selection, we independently sample M_{req} post-deployment requests $F^{\mathrm{req}}_j \sim \mathcal{P}_{\mathrm{req}}$, separate from synthetic and confirmation requests. The paired

gain is

$$\widehat{G}^{v}_{\mathrm{req}} = \frac{1}{M_{\mathrm{req}}} \sum_{j=1}^{M_{\mathrm{req}}} \left[\Delta^{\mathrm{eval}}_{\mathrm{obs}}(\alpha_{\mathrm{acc}}, F^{\mathrm{req}}_{j}) - \Delta^{\mathrm{eval}}_{\mathrm{obs}}(\tilde{\alpha}_{v}, F^{\mathrm{req}}_{j}) \right], \tag{17}$$

where positive values indicate improvement over the accuracy baseline. For poison, both architectures are evaluated on the same independently generated poisoned instance using $\Delta^{\mathrm{eval}}_{\mathrm{obs}}(\alpha, F^{\mathrm{req}}_{j} \mid \widetilde{D}^{\mathrm{req}}_{j})$.

Let κ index a paired comparison for a fixed dataset, search method, and seed, with baseline $\alpha_{\mathrm{acc},\kappa}$ and selected architecture $\tilde{\alpha}_{v,\kappa}$. Select is compared with its pool's accuracy baseline, while Design is compared with its paired accuracy search. For direct comparisons, Select is applied to the candidate pool from Design's paired accuracy search so that both methods use the same baseline and feasibility threshold. The pool oracle minimizes mean discrepancy on the evaluation requests over the feasible pool and serves only as an evaluation reference. On the same post-deployment requests,

$$\bar{\Delta}_{\kappa}(\alpha) = \frac{1}{M_{\mathrm{req}}} \sum_{j=1}^{M_{\mathrm{req}}} \Delta^{\mathrm{eval}}_{\mathrm{obs}}(\alpha, F^{\mathrm{req}}_{\kappa,j}). \tag{18}$$

For weights $w_{\kappa} \geq 0$ with $\sum_{\kappa} w_{\kappa} = 1$,

$$\overline{\Delta}^{\mathrm{acc}} = \sum_{\kappa} w_{\kappa} \bar{\Delta}_{\kappa}(\alpha_{\mathrm{acc},\kappa}), \qquad \overline{\Delta}^{v} = \sum_{\kappa} w_{\kappa} \bar{\Delta}_{\kappa}(\tilde{\alpha}_{v,\kappa}), \qquad r^{v} = 1 - \frac{\overline{\Delta}^{v}}{\overline{\Delta}^{\mathrm{acc}}}. \tag{19}$$

The reported percentage reduction is $100r^{v}$. Final summaries include baseline fallbacks; before confirmation, $\alpha_{\mathrm{Design},\kappa}$ is used and weights are renormalized over feasible provisional comparisons. For poison, $\bar{\Delta}_{\kappa}$ averages the corresponding conditional discrepancies over poisoned evaluation instances.

Table 2 and Figure 7 use the 219 of 240 pools containing at least three feasible architectures. If $\mathcal{C}_t$ denotes eligible pools in setting t, Section 6.2 weights the sixteen setting means equally, with $w_{\kappa} = 1/(16|\mathcal{C}_t|)$, whereas Figure 7 weights the 219 pools equally, with $w_{\kappa} = 1/219$. Table 3 uses 15 independent Design seeds per setting. Its discrepancy, validation accuracy, and gain before confirmation average only feasible provisional choices, whereas final gain includes all seeds and assigns zero gain to baseline fallbacks. Table 4 uses the same feasible comparisons for random requests, with Design evaluated before confirmation.

B.3 Computational Cost Measurement

Cost measurements for Figure 3 cover each binary task at four, six, eight, and ten qubits, with TFIM spin count matching circuit width. They use the split sizes and epoch counts in Appendix B.2 and 10% random requests. Select receives an existing pool of 12 trained candidates from Evolution with 10 initial candidates and 25 search steps, then scores feasible candidates on three synthetic requests. Design's cost includes both the accuracy-based search and the discrepancy-guided search, each starting with eight candidates and adding 16. The latter uses two synthetic requests and, when feasible, four independent confirmation requests. Baseline fallbacks remain included.

Each configuration is timed once in NumPy on an Intel Xeon Platinum 8352V with an NVIDIA GeForce RTX 4090 GPU, using eight workers and one Basic Linear Algebra Subprograms (BLAS) thread per worker. Timers include loading, worker startup, scheduling, computation, and shutdown, but exclude data preparation, evaluation on new requests, interpreter initialization, and serialization. Cost function evaluations count loss calls, including finite difference gradients, whose arithmetic cost varies with architecture.

FLOPs count real additions and multiplications in state vector gate application and local fusion during optimization, validation, and discrepancy evaluation. An invocation on B states with n qubits, L fused local rotations, and Z native R_{ZZ} gates costs $B2^{n}(14L + 6Z) + 112L$ FLOPs. CNOT permutations contribute memory traffic but no arithmetic. This estimate excludes scalar gate preparation, probability readout, loss and discrepancy reductions, optimizer arithmetic, classical control, and memory traffic.

B.4 Hardware Implementation and Resource Usage

Comparisons across devices. Table 6 compares each method with its own accuracy baseline, using 2048 shots on Digits and Fashion-MNIST and 1024 shots on WDBC and TFIM. Select uses five seeds per dataset, with three requests and six inputs per request on Digits, and two requests and two inputs per request on the other tasks. Design uses 13 seeds, four requests, and four inputs per request on Digits, and 15 seeds, four requests, and three inputs per request on Fashion-MNIST. Its WDBC and TFIM evaluations each use five seeds, two requests, and two inputs per request. The Design entries report feasible provisional choices before confirmation. Discrepancies are averaged within each seed and then across seeds.

Measurement protocol. Hardware execution measures updated and retrained circuits prepared in simulation. Paired methods use the same device, physical qubits, inputs, and shot count. For the common-baseline comparison in Table 7, the first five seeds per dataset from the paired Evolution study provide candidate pools distinct from Table 6. Select reranks each pool using the same two synthetic random requests as Design, while Design retains its confirmation and fallback decisions. The common baseline is the architecture with the highest validation accuracy in the pool. Each block is evaluated on two new random requests removing 16 of 160 training examples.

Digits and Fashion-MNIST use four qubits mapped to $[88, 89, 90, 91]$ on `ibm_kingston`, while WDBC and TFIM use six mapped to $[87, 88, 89, 90, 91, 98]$. Routing and mapping are fixed across methods (Li et al., 2019, Murali et al., 2019), and TFIM uses full ground-state preparation (Iten et al., 2016). Circuits use 2048 shots, compilation level one, and a fixed transpiler seed, with dynamical decoupling and twirling disabled.

Each block initially measures discrepancy on four label-balanced inputs. The Digits extension adds four inputs and repeats one original input per request. Table 7 uses this additional measurement set. Combining both sets and counting repeated inputs only once gives discrepancies of 0.136, 0.061, and 0.045 for the baseline, Select, and Design, respectively. For WDBC, the corresponding simulation values are 0.188, 0.218, and 0.042, indicating that the baseline–Select ordering on hardware is not solely due to hardware noise.

Hardware resource usage. Archived usage records account for at least 6,912 QPU seconds, corresponding to a reference cost of $11,059.20 at $1.60 per QPU second. This total includes exploratory runs, preliminary tests, internal measurements, and reported hardware evaluations, with each recorded job counted once.

C Supporting Experiments

C.1 Architecture Variation at Comparable Accuracy

We examine whether the observed variation in unlearning outcomes persists among architectures with comparable predictive accuracy. Figure 6 shows the full population of 256 architectures for each dataset, relating mean validation accuracy to mean TV distance after unlearning. Figure 1 presents the corresponding Digits result in the main text, while Figure 6 reports all four datasets.

For each dataset, all architectures are evaluated using the same six 10% random requests and common optimization settings described in Appendix B.2. Digits and Fashion-MNIST use four qubits, while WDBC and TFIM use six qubits. Mean TV distance is averaged over evaluation inputs, requests, and initializations, while validation accuracy is averaged over the same initializations. These architecture populations are constructed independently of the QAS pools in Table 2 and are not filtered by either validation accuracy or unlearning outcome.

To determine whether this variation remains when predictive accuracy is comparable, Table 8 restricts the comparison to architectures whose mean validation accuracy is within 0.03 of the best architecture on each dataset. This yields 12, 48, 23, and 123 comparable-accuracy candidates for Digits, Fashion-MNIST, WDBC, and TFIM, respectively. Despite the restricted accuracy ranges, substantial variation in mean TV distance remains across all four datasets. The resulting TV-distance ranges are 0.0101–0.3708 for Digits, 0.0081–0.3894 for Fashion-MNIST, 0.0224–0.4508 for WDBC, and 0.0023–0.3443 for TFIM. Thus, architectures with comparable predictive accuracy can still exhibit substantially different agreement with retraining.

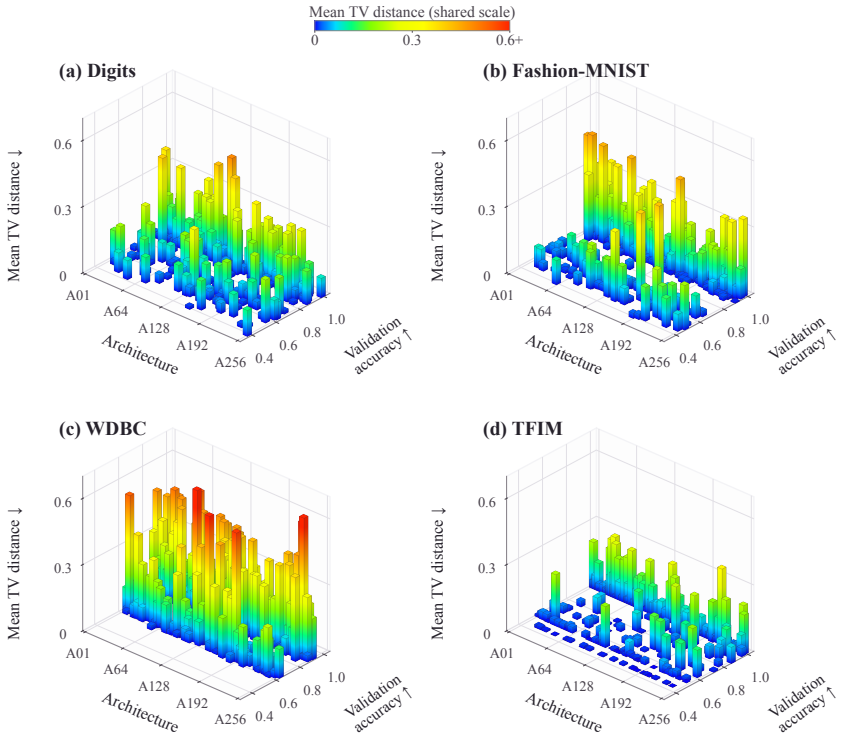


Figure 6: **Architecture-level variation in validation accuracy and mean TV distance on Digits, Fashion-MNIST, WDBC, and TFIM.** Each panel shows the full population of 256 architectures for one dataset. Bar height and color indicate mean TV distance, while the second horizontal axis gives mean validation accuracy.

Table 8: **Architecture variation at comparable validation accuracy.**

	All architectures	Candidates with comparable accuracy		
Task	Mean TV distance range	Validation accuracy range	Count	Mean TV distance range
Digits	0.0016–0.4861	0.9667–0.9963	12	0.0101–0.3708
Fashion-MNIST	0.0039–0.4630	0.9704–1.0000	48	0.0081–0.3894
WDBC	0.0070–0.6826	0.9185–0.9481	23	0.0224–0.4508
TFIM	0.0002–0.3443	0.9741–1.0000	123	0.0023–0.3443

C.2 Evaluation of Alternative Scoring Rules

To test whether generic architecture scores can replace direct unlearning evaluation, we compare UCE with alternative scores on fixed feasible pools (Ying et al., 2019, Dong & Yang, 2020). Parameter sensitivity analyses (Haug et al., 2021, Larocca et al., 2023) motivate the class probability Jacobian J_α^{class}, with i indexing 16 evaluation inputs and j trainable parameters,

$$J_{\alpha,ij}^{\text{class}} = \left.\frac{\partial q_\alpha(1 \mid x_i, \theta)}{\partial \theta_j}\right|_{\theta=\theta_D^\alpha}.$$

The Jacobian norm score is $\|J_\alpha^{\text{class}}\|_F^2/16$, where $\|J\|_F^2$ sums squared entries. Effective rank (eRank) is the exponential entropy of normalized singular values (Roy & Vetterli, 2007). Other scores are trainable parameter count (Params) (Krishnakumar et al., 2022), the Euclidean norm of the initial training loss gradient (GradNorm) (Abdelfattah et al., 2021), and validation accuracy standard deviation across three models trained on bootstrap resamples (Val. accuracy std.) (Bousquet & Elisseeff, 2002). Uniform selection gives feasible candidates equal probability. Alternative score directions use the signs of their Pearson correlations with post-deployment discrepancy over all 2,880 candidates.

Figure 7 shows that UQAS-Select lowers mean discrepancy from 0.140 to 0.028 on independent 10% random requests. The strongest alternative score, eRank, achieves 0.073. The lower discrepancy of UCE shows the benefit of incorporating request outcomes into architecture selection beyond architecture sensitivity alone. Error bars are obtained from 5,000 bootstrap resamples over eligible pools.

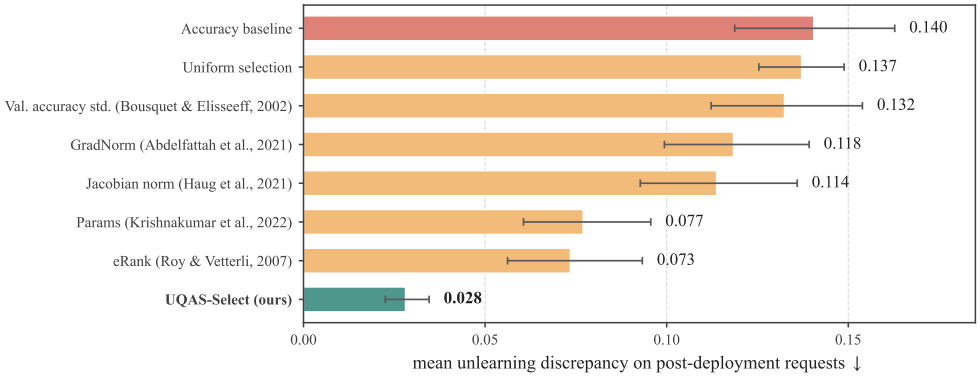


Figure 7: **UCE selects architectures with lower unlearning discrepancy.**

C.3 UQAS-Design and Search Comparisons

On random requests, final Design choices, including fallbacks, reduce mean discrepancy from 0.1097 to 0.0405 on Digits, 0.1242 to 0.0171 on Fashion-MNIST, 0.2560 to 0.0628 on WDBC, and 0.0420 to 0.0088 on TFIM.

Table 9 extends the direct comparison to other request types and Digits classes 0, 1, and 2, using the binary split sizes and multiclass readout above. Select reranks the candidate pool produced by Design's paired accuracy search, and Design is evaluated before confirmation.

Table 9: **Unlearning discrepancy across unlearning request types.**

		Mean unlearning discrepancy ↓			
Request type	Task	Baseline	UQAS-Select	Pool oracle	UQAS-Design
Random	Digits (C=3)	0.111	0.053	0.050	0.063
Poison	Digits	0.159	0.092	0.084	0.118
	Fashion-MNIST	0.143	0.084	0.045	0.034
	WDBC	0.287	0.109	0.102	0.086
	TFIM	0.133	0.042	0.029	0.037
	Digits (C=3)	0.158	0.131	0.109	0.139
Whole class	Digits	0.562	0.542	0.462	0.519
	Fashion-MNIST	0.431	0.304	0.152	0.282
	WDBC	0.648	0.598	0.526	0.552
	TFIM	0.378	0.286	0.218	0.275
	Digits (C=3)	0.473	0.377	0.330	0.397

The request and task extensions show that Design is not uniformly better than Select or the pool oracle. For example, Select has lower discrepancy on three-class Digits for all three request types, whereas Design improves on Select for binary Fashion-MNIST under poison and whole class requests. These comparisons identify where searching beyond the pool adds value.

Expanded validation accuracy search. To test whether additional accuracy search alone explains Design's gains, we extend the Evolution accuracy search from 24 to 72 distinct trained candidates for each of 15 seeds per task. The expanded search contains the original candidate pool and selects

architectures solely by validation accuracy. Both choices are evaluated on the same four 10% random requests. In Table 10, the first column gives the original baseline discrepancy minus the expanded baseline discrepancy over all seeds. The second gives the expanded baseline discrepancy minus provisional Design discrepancy over seeds with a feasible Design choice. This triples the number of trained accuracy candidates, not an exactly matched runtime or measurement budget.

Table 10: **Discrepancy gains with expanded validation accuracy search.**

	Mean gain ↑	
Task	Expanded accuracy search	UQAS-Design
Digits	-0.040	$+0.105$
Fashion-MNIST	$+0.002$	$+0.115$
WDBC	-0.069	$+0.291$
TFIM	$+0.000$	$+0.033$
Digits (C=3)	-0.040	$+0.097$

Tripling the accuracy search does not consistently lower unlearning discrepancy, with mean gains between -0.069 and $+0.002$ in Table 10. Design improves on the expanded baseline by 0.033 to 0.291, supporting the role of unlearning evaluation beyond additional search alone.

C.4 Generalization and Predictive Performance

Evaluation on D_{new} tests whether the selected architectures retain their unlearning advantage on inputs and requests held out from selection. Both variants reduce discrepancy on all four datasets in Table 11. Here G_{new} is the reduction in $\Delta_{\text{obs}}^{\text{new}}$ from the accuracy baseline, and ΔA_{new} is the difference in test accuracy before unlearning, UQAS minus baseline. Mean gains are 0.108 for Select and 0.093 for Design, with an accuracy difference of -0.003 for both.

Table 11: **Evaluation on data and requests held out from architecture selection.**

	Mean gain G_{new} ↑	
Task	UQAS-Select	UQAS-Design
Digits	$+0.072$	$+0.085$
Fashion-MNIST	$+0.125$	$+0.095$
WDBC	$+0.177$	$+0.158$
TFIM	$+0.059$	$+0.033$
Overall	$+0.108$	$+0.093$

Task accuracy after unlearning. To assess predictive performance after the update, Table 12 reuses these architectures and requests, covering 219 eligible Select pools and all final Design choices, including fallbacks. Each comparison uses four 10% requests, 144 retained examples, and 50 test examples. Select weights QAS methods equally within datasets, and both variants weight datasets equally. Entries give updated / retrained test accuracy on $[0, 1]$. Updated UQAS accuracy differs from retraining by $+0.001$ for Select and $+0.002$ for Design, and from the updated accuracy baseline by -0.001 and -0.003.

Table 12: **Test accuracy after unlearning.**

	Mean test accuracy ↑			
	UQAS-Select		UQAS-Design	
Task	Baseline	Ours	Baseline	Ours
Digits	0.962/0.960	0.961/0.959	0.957/0.956	0.957/0.955
Fashion-MNIST	0.987/0.987	0.988/0.988	0.984/0.986	0.981/0.981
WDBC	0.883/0.880	0.878/0.875	0.887/0.861	0.880/0.875
TFIM	0.998/1.000	1.000/1.000	1.000/0.995	1.000/1.000
Overall	0.957/0.957	0.957/0.955	0.957/0.950	0.954/0.953

C.5 Sensitivity to the Unlearning Operator

We evaluate the sensitivity of UQAS to the choice of unlearning operator using masked fine tuning, full fine tuning, NegGrad+, and diagonal influence. All four operators start from θ_D^α and use the same retrained models and evaluation inputs. Masked fine tuning uses partial fine tuning as in CF-k (Goel et al., 2022), freezing the first half of the parameter vector and updating the remainder on retained data; full fine tuning updates all parameters (Jia et al., 2023). Both perform four updates with a learning rate of 0.02. NegGrad+ follows the same update schedule, with the forget-loss ascent term weighted by 0.5 (Kurmanji et al., 2023). For our diagonal influence approximation, we take a single step along the forget gradient, scaled by $|F|/|D|$ and normalized elementwise by the squared retained-loss gradients with damping 10^{-2} (Koh & Liang, 2017).

For UQAS-Select, we rerank each trained pool under each operator and average the gains over 153 eligible pools across the sixteen dataset–QAS settings. For UQAS-Design, we keep the architectures selected using masked fine tuning fixed and evaluate them under all four operators before confirmation. We weight the four binary tasks and the three-class Digits task equally. Both variants are evaluated using independently sampled 10% random requests. Gains are measured as the mean reduction in TV discrepancy relative to the corresponding architecture selected by validation accuracy.

Table 13: **Sensitivity to the unlearning operator.**

	Mean gain ↑	
Operator	UQAS-Select	UQAS-Design
Masked fine tuning (Goel et al., 2022)	$+0.106$	$+0.099$
Full fine tuning (Jia et al., 2023)	$+0.100$	$+0.095$
NegGrad+ (Kurmanji et al., 2023)	$+0.104$	$+0.099$
Diagonal influence (adapted) (Koh & Liang, 2017)	$+0.106$	$+0.073$

Table 13 shows positive mean gains for both UQAS variants under all four unlearning operators. UQAS-Select's mean gain remains stable across operators, ranging from 0.100 to 0.106. UQAS-Design achieves similar gains under masked fine tuning, full fine tuning, and NegGrad+ (0.095–0.099), while the gain decreases to 0.073 under diagonal influence.

C.6 Sensitivity to Unlearning Requests and Discrepancy Measures

We vary synthetic and evaluation request fractions together from 5% to 50%, sampling them independently at each fraction. For poison, this is also the fraction poisoned before training. Random selection is uniform over feasible candidates, and bands show 95% intervals from $5{,}000$ bootstrap resamples of pools. In Evolution pools, Select lowers mean discrepancy at every poison fraction and most random request fractions, as shown in Figure 8.

For random requests, Figure 9 shows positive mean gains across nearly all QAS methods and unlearning fractions. The only exception is Digits with Evolution at 25%, where the mean gain is slightly negative and the paired interval includes zero.

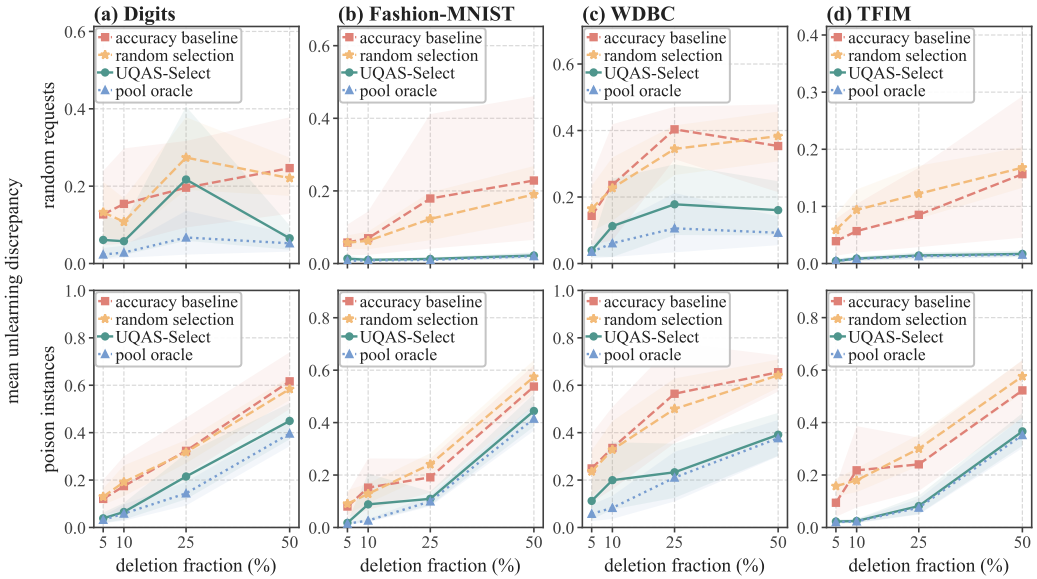


Figure 8: **UQAS-Select under varying unlearning fractions.** Mean unlearning discrepancy for random requests (top) and poison instances (bottom) across four datasets.

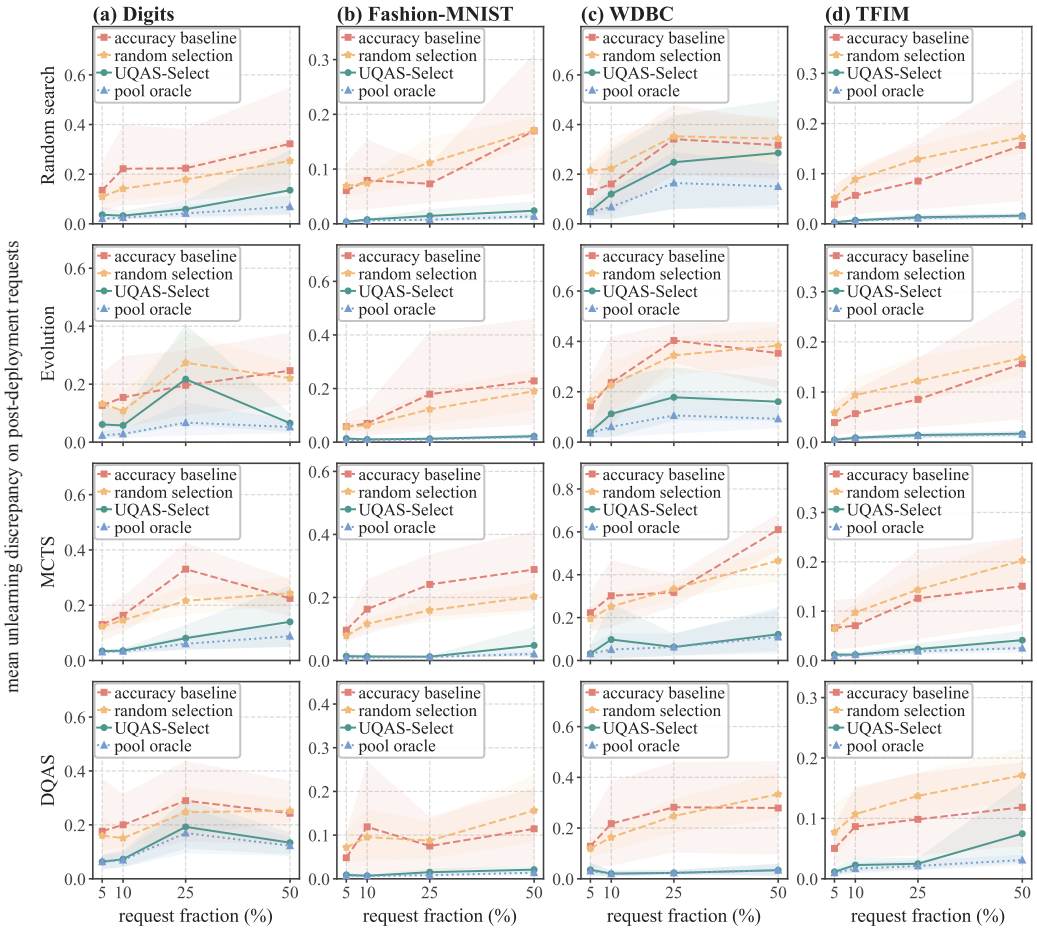


Figure 9: **UQAS-Select across random request fractions and QAS methods.**

For poison instances, Figure 10 shows lower mean discrepancy with UQAS-Select for every dataset and evaluated unlearning fraction when averaged across the four QAS methods.

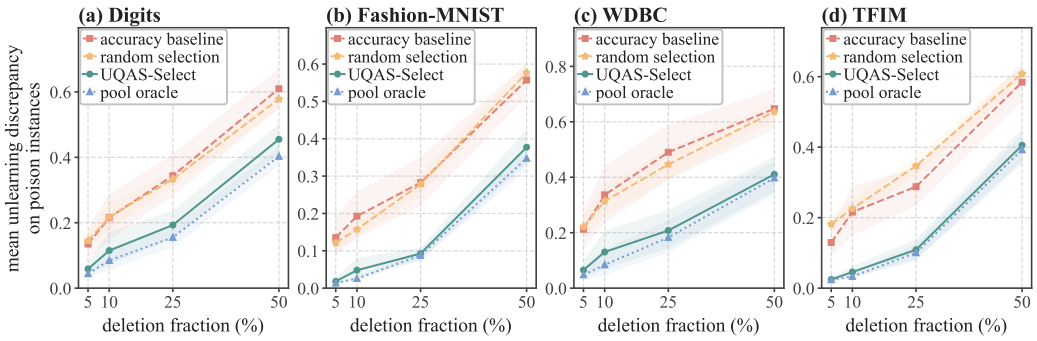


Figure 10: **UQAS-Select across poison fractions, pooled over QAS methods.**

Alternative discrepancy measures. We compare TV with Jensen–Shannon (JS) and Kullback–Leibler (KL) divergences (Lin, 1991) on separate Evolution pools, using the same model pairs and six random requests per pool. The first three requests determine selection and the last three evaluate it. With accuracy tolerance 0.03 and at least three feasible candidates, eight pools remain for each dataset except WDBC, which has six. These pools differ from Figure 1 and Table 2.

For updated and retrained distributions p, q, KL uses $D_{\mathrm{KL}}(q\|p)$ and JS averages their KL divergences to the equal mixture. Both use natural logarithms, with 10^{-12} added to probabilities before renormalization. Unlike JS, KL can diverge without smoothing when p assigns zero probability to an outcome supported by q. TV needs no smoothing.

Table 14 evaluates each selector using its own measure on held-out requests. Gains over the accuracy baseline are positive for every dataset and measure, but less consistent across pools on WDBC and for KL on Digits. JS and KL match TV's choice in $26/30$ and $25/30$ pools, respectively. Thus improvements extend beyond TV, although rankings differ and gain magnitudes are not directly comparable.

Table 14: **UQAS-Select gains with alternative discrepancy measures.**

	Mean gain ↑		
Task	TV	JS	KL
Digits	+0.070	+0.018	+0.052
Fashion-MNIST	+0.117	+0.048	+0.318
WDBC	+0.078	+0.037	+0.203
TFIM	+0.054	+0.009	+0.034

C.7 Finite Measurement Shots

UQAS-Select across QAS methods. Finite shot pool evaluation uses the same 219 eligible pools as Table 2, drawn from 240 pools with 15 per dataset and QAS method. The first three of six historical 10% random requests score candidates, and the remaining three evaluate choices after selection. These independently drawn requests may overlap in individual examples. The experiment replays historical data and requests, with no previously untouched evaluation set used.

We reuse the trained parameters and common optimization settings in Appendix B.2. Each score averages empirical TV over 60 inputs, three requests, and four independent measurement repetitions. The shot count $s \in \{128, 256, 512, 1024, 2048, 4096\}$ applies to each measurement of an updated or retrained Born distribution, not the total scoring budget. Final discrepancy uses exact probabilities on the three evaluation requests. The exact-probability variant selects on the same scoring requests without using evaluation outcomes.

Table 15 reports all sixteen combinations. At 128 shots, 140 of the 219 pools improve, 29 tie, and 50 worsen relative to their accuracy baselines. Random on Digits is the only combination with a higher mean discrepancy, increasing from 0.0617 to 0.0919. At 1024, 2048, and 4096 shots, all sixteen combination means improve, although individual pools can still worsen. Including all 240 pools yields the same counts of improving combination means at every shot count. Mean discrepancy need not decrease monotonically with the shot count because finite shot scores and evaluation requests can favor different architectures.

At 128 shots, averaging baseline and Select discrepancies separately with equal weight for the four QAS methods within each dataset gives reductions of 40.6%, 76.7%, 48.3%, and 47.1% for Digits, Fashion-MNIST, WDBC, and TFIM. These dataset aggregates define the range reported in Section 6.2, not a range over all sixteen combinations. With equal weight for the sixteen combination means, validation accuracy changes from 96.75% to 95.94%, and accuracy after unlearning on the discrepancy evaluation inputs changes from 95.56% to 95.28%. Every selected architecture satisfies its pool's validation accuracy threshold.

Table 15: **UQAS-Select discrepancy on separate evaluation requests.**

		Mean unlearning discrepancy ↓						
			Shots per Born distribution					
QAS	Task	Baseline	128	256	512	1024	2048	4096
Random	Digits	0.0617	0.0919	0.0855	0.0812	0.0313	0.0312	0.0307
	Fashion-MNIST	0.1490	0.0201	0.0160	0.0156	0.0149	0.0145	0.0129
	WDBC	0.2319	0.0747	0.0744	0.0676	0.0358	0.0340	0.0340
	TFIM	0.0709	0.0443	0.0469	0.0477	0.0333	0.0164	0.0131
Evolution	Digits	0.0837	0.0798	0.0786	0.0553	0.0572	0.0571	0.0557
	Fashion-MNIST	0.1196	0.0444	0.0449	0.0196	0.0175	0.0168	0.0161
	WDBC	0.2212	0.1601	0.1519	0.1519	0.1131	0.1131	0.1143
	TFIM	0.0709	0.0489	0.0302	0.0419	0.0227	0.0175	0.0149
MCTS	Digits	0.1565	0.0595	0.0515	0.0394	0.0383	0.0328	0.0366
	Fashion-MNIST	0.1844	0.0168	0.0169	0.0151	0.0156	0.0143	0.0140
	WDBC	0.3521	0.1112	0.1032	0.1032	0.0797	0.1285	0.1250
	TFIM	0.1262	0.0515	0.0419	0.0435	0.0504	0.0414	0.0388
DQAS	Digits	0.1820	0.0561	0.0560	0.0531	0.0522	0.0522	0.0543
	Fashion-MNIST	0.1043	0.0484	0.0476	0.0264	0.0251	0.0200	0.0191
	WDBC	0.1821	0.1641	0.0993	0.0711	0.0711	0.0538	0.0495
	TFIM	0.0816	0.0402	0.0337	0.0307	0.0263	0.0224	0.0168

Direct comparison between Select and Design. The direct comparison in Table 5 uses three prespecified paired blocks per dataset, rather than the feasible comparisons summarized in Table 4. Select receives the 24 candidates from Design's paired accuracy search and uses the same accuracy threshold and two synthetic random requests as Design. Select minimizes its finite-shot score directly, whereas Design retains its archived confirmation decision based on four independent requests, including fallback. The shot count controls UCE scoring, while final discrepancy is evaluated with exact probabilities.

Final choices are evaluated with exact probabilities on the same four independent 10% post-deployment requests and 60 inputs, using 160 training examples. With the common baselines in Table 5, Design has lower discrepancy on all four datasets at 2048 shots, but higher discrepancy on Digits at 128–1024 shots. Table 16 retains all six shot counts.

Table 16: **Final mean unlearning discrepancy across shot counts.**

		Mean unlearning discrepancy ↓					
		Shots per Born distribution					
Method	Task	128	256	512	1024	2048	4096
UQAS-Select	Digits	0.0424	0.0424	0.0369	0.0369	0.0369	0.0369
	Fashion-MNIST	0.1106	0.1106	0.1106	0.1073	0.1008	0.1008
	WDBC	0.2792	0.2786	0.2786	0.2786	0.2786	0.2786
	TFIM	0.0285	0.0285	0.0193	0.0168	0.0168	0.0112
UQAS-Design	Digits	0.0999	0.0879	0.0879	0.0879	0.0134	0.0134
	Fashion-MNIST	0.0268	0.0251	0.0251	0.0181	0.0104	0.0104
	WDBC	0.0775	0.0775	0.0429	0.0418	0.0418	0.0341
	TFIM	0.0161	0.0161	0.0155	0.0155	0.0108	0.0107

Measurement totals cover both distributions, all scoring requests and inputs, four repetitions, and confirmation of the Design candidate against the baseline when feasible. At 2048 shots per distribution, Select and Design require 24.25 and 52.43 million shots on Digits, 30.80 and 55.05 million on Fashion-MNIST, 7.86 and 52.43 million on WDBC, and 28.84 and 55.05 million on TFIM, respectively. Equal shots per distribution therefore do not imply equal total budgets. These simulated workloads measure neither QPU usage nor runtime.

Ranking and gain sensitivity. The sensitivity analysis in Figure 11 examines how finite-shot estimation affects candidate ranking and the resulting unlearning gain. Select uses eight eligible Evolution pools per dataset, except for WDBC, which has six, and evaluates the same three random requests under finite-shot and exact scoring. Scores are averaged over four independent measurement repetitions. Design uses 15 seeds per dataset with separate requests for finite-shot search, confirmation, and exact evaluation. Ranking agreement with exact simulation is measured by Spearman correlation (Spearman, 1961), and gain retention is normalized by the corresponding exact-simulation gain. Design also reports the final relative reduction including baseline fallbacks. Because the two methods use different request and evaluation protocols, their finite-shot results are interpreted separately.

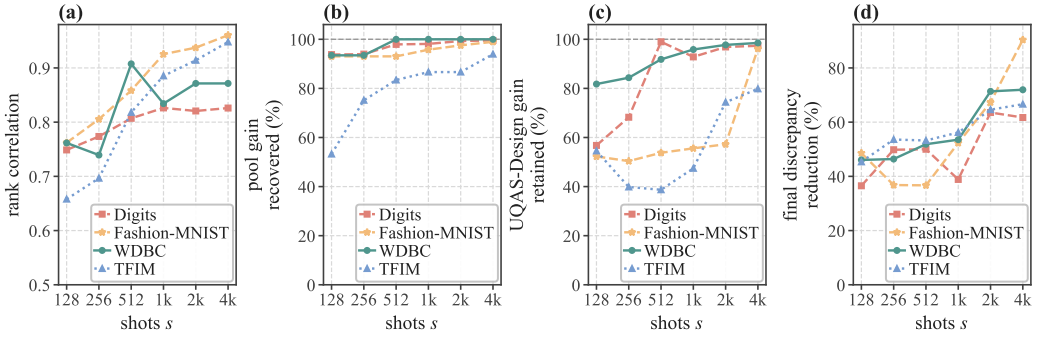


Figure 11: **Robustness to finite-shot measurements.** (a) Select rank correlation with exact simulation. (b) Percentage of exact-simulation gain retained by Select. (c) Percentage of exact-simulation gain retained by Design before confirmation. (d) Final discrepancy reduction of Design.