A Library for Benchmarking Quantum-Circuit Simulators for Quantum Machine Learning

João Alfredo Bessa[1], **Carlos Eduardo Santos**[1], **Rosiane de Freitas**[1]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Manaus – AM – Brazil

{joao.bessa,carlos.santos,rosiane}@icomp.ufam.edu.br

Abstract. *Quantum machine learning (QML) is one of the main candidate applications for quantum hardware in the short term (NISQ era), and many QML models – notably quantum support vector machines (QSVMs) – are kernel methods whose kernel is the fidelity between the quantum states that encode the data. We present* QBenchFrame*, an open-source Python library that compiles a single circuit description into a backend-agnostic intermediate representation and executes it, ideally and under a configurable depolarization model, on Qiskit, Cirq, PennyLane and the Brazilian Ket library, while also emitting ready-to-run source code in each language. Its central metric is the Uhlmann–Jozsa quantum fidelity computed as a kernel of conjugate transposes,* $F = \langle\psi|\,\rho\,|\psi\rangle$ *– exactly the fidelity quantum kernel that underlies QSVMs.*

Resumo. *O aprendizado de máquina quântico (QML) é uma das principais aplicações candidatas para o* hardware *quântico de curto prazo (era NISQ), e muitos modelos de QML – notadamente as* quantum support vector machines *(QSVMs) – são métodos de* kernel *cujo* kernel *é a fidelidade entre os estados quânticos que codificam os dados. Apresentamos o* QBenchFrame*, uma biblioteca Python de código aberto que compila uma única descrição de circuito em uma representação intermediária agnóstica a* backend *e a executa, de forma ideal e sob um modelo de despolarização configurável, em Qiskit, Cirq, PennyLane e na biblioteca brasileira* Ket*, além de emitir o código-fonte em cada linguagem. Sua métrica central é a fidelidade quântica de Uhlmann–Jozsa calculada como um* kernel das transpostas conjugadas*,* $F = \langle\psi|\,\rho\,|\psi\rangle$ *– exatamente o* kernel *quântico de fidelidade que fundamenta as QSVMs.*

1. Introduction

Quantum machine learning (QML) is among the most actively pursued applications of Noisy Intermediate-Scale Quantum (NISQ) hardware [Huang 2023]. A particularly clean bridge to classical learning theory is the *quantum kernel* method: a datum x is embedded into a state $|\phi(x)\rangle$ by a parameterised feature map, and a classical kernel machine is trained on the resulting matrix $K_{ij} = |\langle\phi(x_i)|\phi(x_j)\rangle|^2$ [Havlíček et al. 2019, Schuld and Killoran 2019]. Supervised quantum models have in fact been shown to be kernel methods with exactly this state-overlap Gram matrix [Schuld 2021], so state fidelity is not a peripheral diagnostic for QML – it is the learning primitive itself.

Two obstacles slow the empirical study of such models. First, the quantum-software ecosystem is fragmented: validating a circuit on Qiskit, Cirq, Penny-

Lane or Ket requires re-encoding it in each SDK and reconciling divergent outputs [Jamadagni et al. 2024, Guo et al. 2025], a slow process and a recurrent source of subtle bugs. Second, the noise metric used in practice is often a *classical* fidelity over sampled histograms; seeing only measurement probabilities, it is blind to relative phases and cannot report how noise erodes a *quantum kernel*, whose entries depend on the full state, coherences included.

We present *QBenchFrame*, an open-source, `pip`-installable Python library that addresses both obstacles. It compiles one circuit description into a backend-agnostic intermediate representation (IR) and runs it – ideally and under a configurable depolarizing model – across four SDKs, while emitting ready-to-run source code for each. Its primary metric is the quantum Uhlmann–Jozsa fidelity, $F = \langle\psi|\,\rho\,|\psi\rangle$, computed as a conjugate-transpose kernel; this is the same fidelity quantum kernel that defines QSVMs. Our contributions are:

- *QBenchFrame*, a tested library with a backend-agnostic IR, a pluggable backend registry (Qiskit, Cirq, PennyLane and *Ket*), a CLI and a multi-language code generator (Sections 4–6);
- a kernel-based quantum fidelity that is exact for a pure reference and coincides with the QSVM kernel (Section 5);
- a multi-backend benchmark, generated by the library's own report module, showing that the classical fidelity overestimates noise robustness on variational circuits (Section 7); and
- a QML case study tracking kernel-target alignment and QSVM accuracy under per-gate depolarizing noise and finite-shot estimation (Section 8).

Scope. Two claims are easy to conflate, so we separate them. The *framework* – parser, IR, backend registry, code generator, noise model and timing harness – is circuit-agnostic: it accepts any circuit over its gate set, and Section 7 exercises it on GHZ and Grover, which are not kernel circuits and serve as controls. The *metric* is deliberately specialised: the fidelity kernel is the object a QSVM consumes, which is why it is the default figure of merit and why the evaluation culminates in a QML case study (Section 8). Claims of general applicability below therefore concern the harness; the claim about the metric is narrower – that it is the right default for QML workloads – and Table 1 delimits where it does and does not differ from the classical alternative.

2. Related Work

Cross-SDK benchmarking. State-vector simulator suites have been compared on raw speed and memory [Jamadagni et al. 2024, Aoki et al. 2024], and application-level studies contrast Qiskit, Cirq and PennyLane on hybrid quantum-classical workloads, repeatedly finding PennyLane's autodiff machinery slower for pure simulation [Rodríguez-Díaz et al. 2024]. Work at the intersection with software testing has catalogued quantum-specific bugs and code-generation issues across SDKs [Guo et al. 2025]. These efforts focus on performance or correctness in isolation; QBenchFrame instead provides a single, scriptable harness that runs the same circuit on several SDKs and scores it with a learning-relevant fidelity.

Noise-oriented benchmarks. Randomised benchmarking and its variants estimate average gate fidelities from Clifford sequences [Knill et al. 2008, Magesan et al. 2011], and cross-entropy benchmarking (XEB) [Arute et al. 2019] extracts a linear cross-entropy score from sampled bitstrings; both are gate- or bitstring-level diagnostics, and neither reports how noise erodes a kernel matrix. Error-mitigation schemes such as zero-noise extrapolation and probabilistic error cancellation [Temme et al. 2017] instead correct noisy expectation values post hoc. QBenchFrame is complementary: it exposes the noise effect directly at the QML-relevant object – the fidelity kernel – rather than at the gate or expectation-value level.

Estimating state overlaps. Exact estimation is expensive: $F = \langle\psi|\rho|\psi\rangle$ needs $O(4^n)$ memory for a dense ρ. Swap tests [Buhrman et al. 2001] estimate $|\langle\psi|\phi\rangle|^2$ from measurement statistics without materialising the state; shadow tomography [Aaronson 2018] and classical shadows [Huang et al. 2020] reconstruct a compressed ρ from polynomially many Pauli measurements; and tensor-network contraction is near-exact up to moderate entanglement [Nguyen et al. 2021]. QBenchFrame computes the exact density-matrix fidelity and falls back to the classical coefficient for larger registers, with the consequences analysed in Section 9.

Quantum kernels. Quantum feature maps induce the fidelity kernel $K_{ij} = |\langle\phi(x_i)|\phi(x_j)\rangle|^2$ and its kernel-target alignment (Section 3); the result that supervised quantum models are kernel methods with precisely this kernel [Schuld 2021, Havlíček et al. 2019, Schuld and Killoran 2019] motivates our choice of metric. Expressibility [Sim et al. 2019] and barren-plateau limits [Cerezo et al. 2021] constrain variational feature maps, and both are visible in the fidelity-decay curves the library generates. Including the Brazilian library Ket [da Rosa and de Santiago 2022] broadens the benchmarked ecosystem beyond the usual three.

LLM-based code generation. The closest work in intent is QuanBench [Guo et al. 2025], which benchmarks *LLM-generated* circuit code across Qiskit, Cirq and PennyLane and finds systematic errors in gate mappings and qubit-ordering conventions. QBenchFrame attacks the same cross-SDK translation gap from the opposite direction: rather than generating code from natural language and judging it, it lowers a single IR into deterministic, rule-based source for each SDK and validates the result through cross-backend agreement of the returned states.

3. Background

A qubit is a unit vector in a two-dimensional Hilbert space; n qubits span a 2^n-dimensional space [Griffiths 2011]. Ideal evolution is unitary and trackable by a state vector at cost $\mathcal{O}(2^n)$; noisy evolution maps pure states to *mixed* states described by a density matrix ρ, whose exact simulation costs $\mathcal{O}(4^n)$ [Li et al. 2022]. Depolarization, the channel used throughout, replaces a qubit's state by the maximally mixed state with probability p.

Simulators. QBenchFrame integrates four SDKs. **Qiskit Aer** (IBM) and **Cirq** (Google) both ship state-vector and density-matrix simulators,

the former with a rich noise API; **PennyLane** (Xanadu) targets differentiable QML through its `default.qubit`/`default.mixed` devices; and **Ket** [da Rosa and de Santiago 2022] is a Brazilian quantum-programming language and runtime (libket with the KBW simulator) whose concise Python embedding exposes exact amplitudes through a `dump` primitive but does not read OpenQASM.

Quantum kernels and QSVMs. A feature map $x \mapsto |\phi(x)\rangle$ induces the *fidelity quantum kernel*

$$K_{ij} = |\langle\phi(x_i)|\phi(x_j)\rangle|^2. \tag{1}$$

Because K is positive-semidefinite it can be handed to any kernel machine, and the resulting QSVM separates data in the quantum feature space [Havlíček et al. 2019, Schuld and Killoran 2019]. Its quality on a labelled set $\{(x_i, y_i)\}$ is summarised by the *kernel-target alignment*

$$A(K, y) = \frac{\langle K, yy^\top\rangle_F}{\|K\|_F \, \|yy^\top\|_F}, \tag{2}$$

a label-correlation proxy for downstream accuracy. Noise threatens both A and accuracy, which is why a phase-aware quantum fidelity is the right benchmarking metric for QML simulators.

4. Library Architecture

QBenchFrame is organised around a backend-agnostic IR (Figure 1). A textual program – a simplified pseudocode or OpenQASM 2.0 – is parsed into a `Circuit` (a list of gates with qubits and parameters). From this single IR the library both (i) drives execution backends and (ii) generates source code.

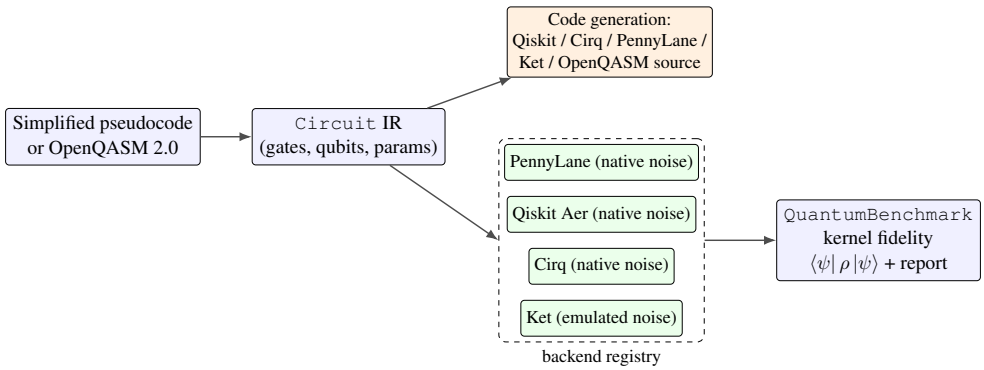


Figure 1. Architecture of QBenchFrame. A single IR decouples the input language from the backends; adding an SDK means writing one adapter, and the same IR drives the multi-language code generator.

4.1. Intermediate representation and parsing

The parser accepts a simplified pseudocode (e.g. `qreg q[3]`, `h q[0]`, `cx q[0], q[1]`, `rx(pi/2) q[0]`) or standard OpenQASM 2.0: comments and optional semicolons are stripped, angle expressions such as `pi/2` are evaluated in a restricted namespace, multi-register indices are flattened with per-register offsets, and aliases (e.g. `cnot`→`cx`) are normalised. The result is a `Circuit` – an ordered list of `Gate`s, each with a canonical name, qubits and parameters – exposing `to_qasm()` for QASM-based backends and `without_measurements()` for state extraction.

4.2. Backend registry and adapters

Each backend subclasses an abstract `Backend` exposing `run_ideal`/`run_noisy` (counts), `statevector` and `density_matrix_noisy`. Heavy SDK imports are *lazy*, so importing the library never requires every SDK: the registry probes availability on demand and runs only what is installed. Qiskit, Cirq and PennyLane consume the IR's OpenQASM 2.0 and return density matrices natively (`AerSimulator(method="density_matrix")`, `DensityMatrixSimulator`, `default.mixed`). Ket, which does not read OpenQASM, is driven gate-by-gate from the IR – rotations map directly and controls use `ctrl` (e.g. `ctrl(q[0], X)(q[1])` for CNOT) – with its exact state read from a `dump` and sampled to produce counts.

4.3. Noise model

The depolarizing model is asymmetric: single-qubit gates carry error probability p and two-qubit gates $2p$, reflecting the higher decoherence of entangling buses. Native-noise backends instantiate this through the SDK's own API; for a backend without a native channel (Ket), the library emulates a *global* depolarizing channel on the ideal state,

$$\rho = (1-\lambda)\,|\psi\rangle\langle\psi| + \lambda\,\frac{\mathbb{I}}{d}, \qquad \lambda = 1 - \prod_{g\in 1\mathrm{q}}(1-p)\ \prod_{g\in 2\mathrm{q}}(1-2p), \tag{3}$$

with $d = 2^n$ and λ accumulated along the circuit, flagged as emulated in the report and analysed in Section 9.

4.4. Code generation: how the IR is lowered, and its limits

Code generation is a deliberately *direct, rule-based* lowering: one emitter per target makes a single linear pass over the IR gate list and translates each `Gate` independently, with no scheduling, routing or optimisation pass. Three cases arise. (i) *Direct mapping* – a lookup table sends the canonical name to the target primitive (`cx` becomes `qc.cx` in Qiskit and `qml.CNOT` in PennyLane). (ii) *Structural re-expression*, when the target has no first-class primitive: Cirq lacks CY, CRX and CP, which are emitted through `.controlled()` wrappers and phases as `cirq.Z**(`θ/π`)`; Ket has no two-qubit controlled gate at all and expresses every control with the higher-order combinator `ctrl(`c`,`U`)(`t`)`; PennyLane has no terminal measurement instruction, so measurement becomes the QNode's return value (`qml.counts()` rather than `qml.state()`); and `barrier` maps to `qc.barrier()` in Qiskit but is dropped for the others, where it has neither an equivalent nor an effect on semantics. (iii) *Gaps* – a gate outside a target's table is emitted as an explicit `# unsupported gate` comment, never silently replaced by an approximate substitute.

This keeps each emitter small and auditable, at the price of three limitations we state explicitly. First, the IR covers a fixed set of 25 gates plus `measure`/`barrier`; an unknown name is *rejected at parse time* with an error, so unsupported input fails fast instead of degrading silently, but the general OpenQASM 2.0 `u2`/`u3` gates, user-defined `gate` declarations and classically conditioned operations fall outside it. Second, coverage is per target and not uniform: SDG and TDG are emitted only for Qiskit and QASM, and Ket lacks SX, ID and CP, so a circuit using them yields complete code for some targets

and a flagged gap for others. Third, generated code is emitted as text and never re-parsed, so equivalence is established empirically – through the agreement of the ideal states the backends return for the same IR (Section 7) – rather than proved.

CLI and testing. A command-line interface exposes `run`, `export` and `list-backends`. The backend-independent core (parser, IR→QASM, fidelity math, noise model, code generation) is covered by 28 unit tests that run with only `numpy`.

5. Kernel-Based Quantum Fidelity

To score noise we use the quantum Uhlmann–Jozsa fidelity. Stacking normalised state vectors as the columns of the matrix Ψ, the Gram/kernel matrix is $K = \Psi^\dagger\Psi$ with $K_{ij} = \langle\psi_i|\psi_j\rangle$, and the fidelity kernel of Eq. (1) is $|K_{ij}|^2$; the conjugate transpose $\Psi^\dagger$ supplies the bras – hence "kernel of the transposes". Since the ideal reference $|\psi\rangle$ is pure, its fidelity against the noisy ρ collapses to one diagonal kernel element,

$$F(|\psi\rangle, \rho) = \left(\operatorname{Tr}\sqrt{\sqrt{\rho_\psi}\,\rho\,\sqrt{\rho_\psi}}\right)^2 = \langle\psi|\,\rho\,|\psi\rangle = \psi^\dagger\rho\,\psi \in [0,1], \tag{4}$$

where $\psi \in \mathbb{C}^d$ is the column vector holding the amplitudes of $|\psi\rangle$, i.e. a single column of Ψ. The expression is exact, numerically stable (no matrix square root) and recovers $|\langle\psi|\phi\rangle|^2$ for pure ρ; the general mixed–mixed case uses the full form via a Hermitian-PSD square root. The classical Bhattacharyya coefficient $F_c = \left(\sum_x \sqrt{p(x)q(x)}\right)^2$ over histograms is retained only as a secondary, phase-blind metric, and as the fallback for circuits too large for a dense ρ (Section 9).

The kernel behind QML. This choice is deliberate: the quantity $|\langle\psi_i|\psi_j\rangle|^2$ that scores noise *is* the fidelity quantum kernel of QSVMs, the Gram matrix $K = \Phi^\dagger\Phi$ handed to a classical SVM [Havlíček et al. 2019, Schuld and Killoran 2019, Schuld 2021]. A low kernel fidelity under noise is therefore a direct prediction that the quantum kernel a QSVM would compute on that device has degraded – a learning-relevant figure of merit, not a generic distance.

6. Usage and Code Export

QBenchFrame offers a one-call API and a CLI. Listing 1 benchmarks a circuit across every available backend; Listing 2 shows source code automatically generated from the IR (`bench.export_code()` / `qbenchframe export`).

Listing 1. Python API.

```
from qbenchframe import QuantumBenchmark, NoiseModel
bench = QuantumBenchmark.from_file("ansatz.qasmtxt", shots=4096,
         noise=NoiseModel(rate=0.02),
         backends=["qiskit", "cirq", "pennylane", "ket"])
bench.run(); print(bench.summary()); bench.report("report.pdf")
```

Listing 2. Generated source (excerpt) for an angle-encoding map.

```
# ===== KET (generated) =====
from ket import Process, RY, ctrl, X, measure
```

```
p = Process(); q = p.alloc(2)
RY(0.7, q[0]); RY(1.2, q[1]); ctrl(q[0], X)(q[1])
# ===== PENNYLANE (generated) =====
import pennylane as qml
@qml.qnode(qml.device("default.qubit", wires=2))
def circuit():
   qml.RY(0.7, wires=0); qml.RY(1.2, wires=1)
   qml.CNOT(wires=[0, 1]); return qml.state()
```

7. Multi-Backend Benchmark Results

We first exercise QBenchFrame as a general harness, on three circuits chosen to span the scope set out in Section 1: GHZ (5 qubits) and Grover (2 qubits), which are ordinary non-QML circuits and act as controls, and a variational ansatz (4 qubits: a rotation layer, circular entanglement and a phase layer), the family QML feature maps belong to. Unless stated otherwise the depolarizing rate is $p = 2\%$, and every figure in this section is produced by the library's own report module.

Figure 2 is the report's metrics panel across the four backends: per-backend latency (left, log scale) and quantum vs. classical fidelity (right). The headline result is the fidelity gap: the classical Bhattacharyya fidelity reads $\approx 98\%$, suggesting a nearly intact circuit, whereas the quantum kernel fidelity is only 76%. Qiskit, Cirq and PennyLane agree on the quantum value to the reported precision – a useful cross-SDK consistency check – and Ket sits 2.0 pp lower (74.0%) because its noise is emulated via Eq. (3) instead of being applied as a per-gate channel. The same emulation explains the latency panel, where Ket's noisy phase costs 0.10 ms against 5.3–7.0 ms for the density-matrix backends: an $O(2^n)$ versus $O(4^n)$ difference rather than a speed advantage. Section 9 examines what this approximation does and does not preserve.

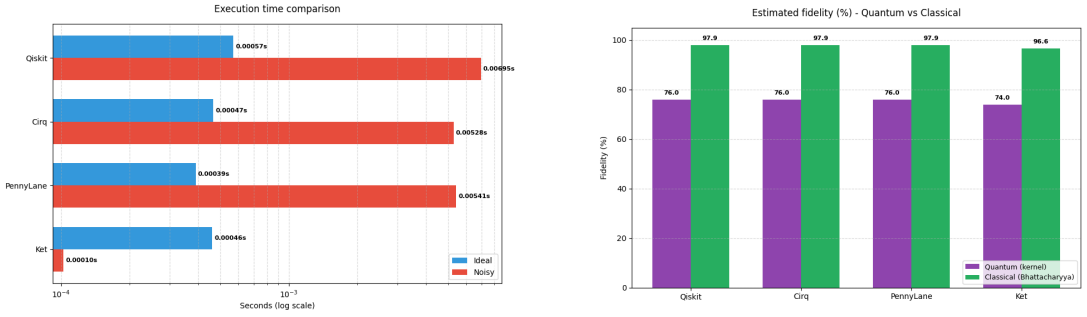


Figure 2. Library report (metrics panel) for the variational ansatz at $p = 2\%$: per-backend latency (left) and quantum vs. classical fidelity (right). The classical metric ($\approx 98\%$) masks the true 76% kernel fidelity; the four backends agree on the quantum value.

The reason for the gap is visible in the per-backend measurement histograms that the report also produces (Figure 3): depolarization only mildly flattens the sampled distribution – enough to keep the histogram-based metric near 98% – while the state's coherences, invisible to sampling, have decayed substantially. The native-noise backend (Qiskit) and the emulated-noise backend (Ket) show qualitatively the same flattening, validating the emulation as a first-order stand-in.

Finally, Figure 4 sweeps the noise rate and plots the quantum kernel fidelity for the three circuit families; the decay is monotone and steeper for circuits with more entangling

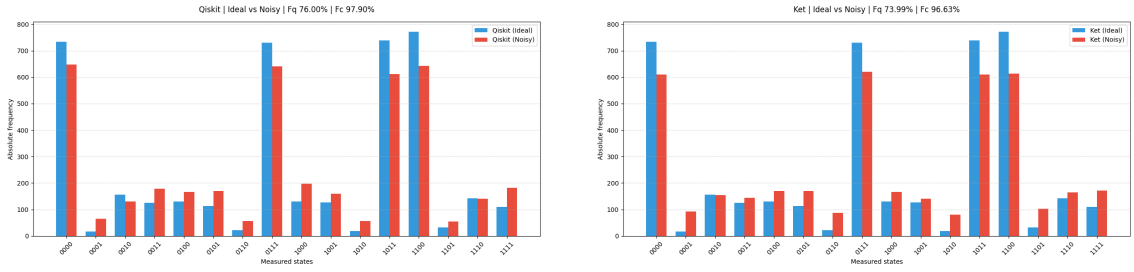


Figure 3. Library report: ideal vs. noisy measurement histograms for the ansatz at $p = 2\%$. Left: Qiskit (native noise). Right: Ket (emulated noise). Both flatten mildly, explaining the optimistic classical fidelity against the true kernel fidelity in the titles.

gates and qubits, as expected physically. Because QBenchFrame computes all of these identically across SDKs, the comparisons are reproducible rather than anecdotal.

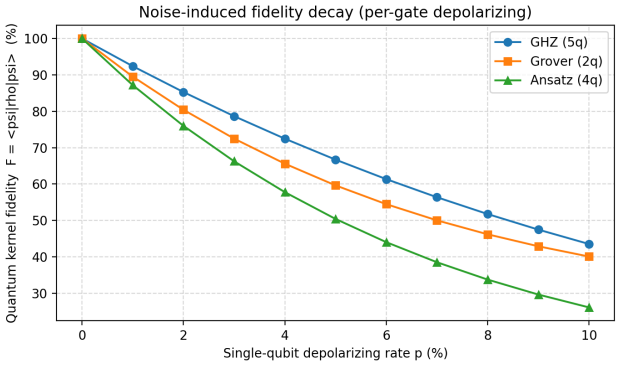


Figure 4. Quantum kernel fidelity vs. single-qubit depolarizing rate p (exact density-matrix simulation), for GHZ, Grover and the variational ansatz.

The gap is circuit-dependent. Table 1 reports both fidelities at $p = 2\%$ for the three bundled circuits. For Grover and GHZ the classical and quantum fidelities nearly coincide (gap ≤ 3.5 points), but for the variational ansatz the classical metric overestimates fidelity by 22.5 points. The discrepancy is not generic: it concentrates on the parameterised, entangling circuits from which QML feature maps are built, exactly where a phase-aware kernel fidelity is needed to compare circuits faithfully.

Table 1. Quantum (kernel) vs. classical (Bhattacharyya) fidelity at $p = 2\%$ (exact density matrix). The gap is large only for the variational ansatz, the family closest to QML feature maps.

Circuit	n	2q gates	F_q	F_c	gap
GHZ	5	4	85.3%	88.8%	3.5 pp
Grover	2	2	80.4%	80.4%	0.0 pp
Ansatz	4	4	76.0%	98.5%	22.5 pp

8. QML Results: a Quantum Kernel under Noise

We now use QBenchFrame for its target use case: a quantum-kernel classifier, the canonical QML model whose kernel is the library's fidelity metric. Data points $x \in \mathbb{R}^2$ are

embedded by a depth-2 angle-encoding feature map (Figure 5), the quantum kernel is $K_{ij} = |\langle\phi(x_i)|\phi(x_j)\rangle|^2$, and a kernel-ridge QSVM is trained on it. We evaluate two binary blobs ($N = 40$) under per-gate depolarizing noise (rate p, with $2p$ on the entangler) plus finite-shot kernel estimation (256 shots), averaging over random splits and shot realisations. Kernel entries come from QBenchFrame's exact density-matrix engine, cross-validated over the four backends in Section 7.

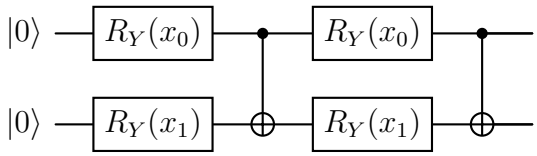


Figure 5. Depth-2 angle-encoding feature map $|\phi(x)\rangle$ used by the quantum kernel. Its output overlaps form the fidelity kernel of Eq. (1).

Figure 6 shows the kernel matrix. Ideally it is sharply block-structured – high within-class, low across-class overlaps – which is what lets the QSVM separate the classes. Under noise ($p = 10\%$, 256 shots) the blocks fade and estimation noise speckles the matrix, directly visualising the degradation of the learning-relevant object.

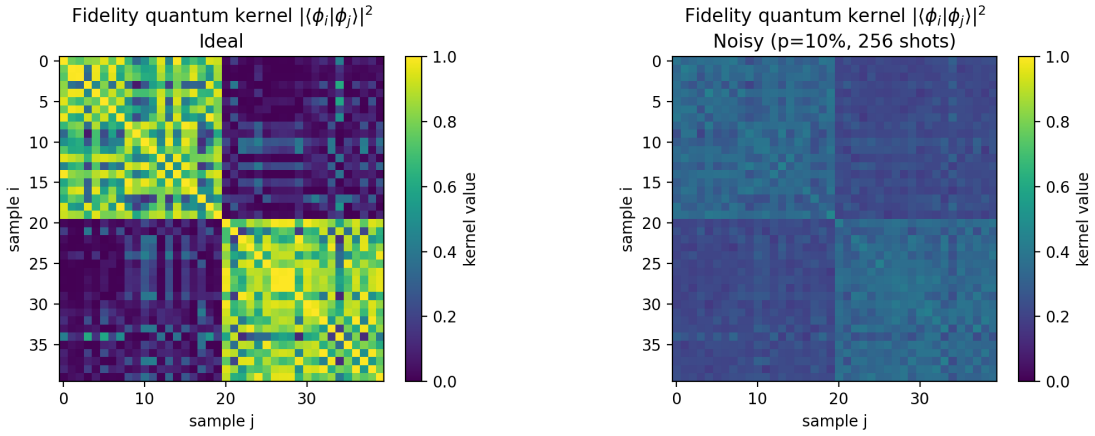


Figure 6. Fidelity quantum kernel $K_{ij} = |\langle\phi_i|\phi_j\rangle|^2$ (samples sorted by class). Left: ideal. Right: per-gate depolarizing $p = 10\%$ with 256 shots. Noise erases the block structure the QSVM relies on.

Figure 7 quantifies the effect, with the exact values listed in Table 2(b): the kernel-target alignment A of Eq. (2) falls monotonically from 0.573 (ideal) to 0.025 at $p = 20\%$, while QSVM test accuracy holds near 80% before collapsing toward the 50% chance level. The alignment – a direct property of the quantum kernel, and hence of the library's fidelity metric – is thus a smoother and earlier indicator of trouble than accuracy alone. That is the practical payoff of benchmarking with the fidelity kernel: it measures exactly the object that determines QSVM performance.

Comparing feature maps. QBenchFrame also compares candidate feature maps. Table 2(a) varies the depth (repetitions) of the angle-encoding map on the same dataset: alignment peaks at depth 2 and falls at depth 3, where the deeper, more entangling map begins to scramble class structure. Accuracy saturates on this easy data, so alignment is the finer discriminator – the kind of circuit-design feedback the library provides for QML.

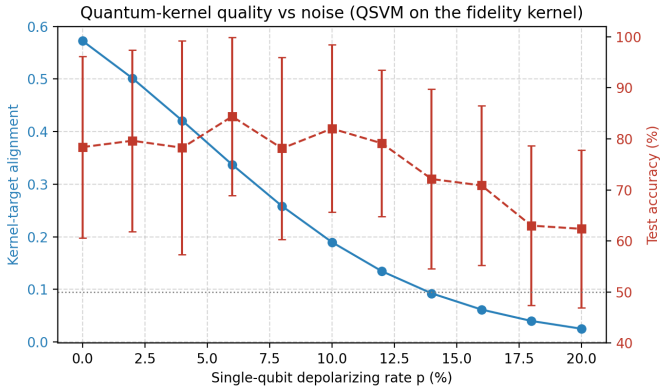


Figure 7. Quantum-kernel quality vs. noise. Kernel-target alignment (left axis) and QSVM accuracy with 256 shots (right axis, error bars over splits/shot realisations) both degrade with the gate-error rate p; the dotted line marks chance.

Table 2. (a) Ideal angle-encoding feature maps of increasing depth on the same dataset; depth 2 maximises alignment. (b) The depth-2 kernel under per-gate depolarizing noise (256 shots, mean over 40 split/shot realisations).

(a) feature-map depth			(b) noise sweep		
Depth	Align.	Accuracy	p	Align.	QSVM accuracy
1	0.421	100.0%	0%	0.573	78.4 ± 17.8
2	**0.573**	99.6%	5%	0.379	82.4 ± 16.5
3	0.468	93.4%	10%	0.190	82.0 ± 16.4
			20%	0.025	62.4 ± 15.4

Validation. The kernel fidelity is checked against the closed form $F = (1 - \lambda) + \lambda/d$ for the emulated channel, and the 28-test suite covers parsing, IR→QASM emission, the noise model and code generation without any quantum SDK.

9. Discussion and Limitations

Emulated noise on Ket. Ket's KBW simulator exposes no depolarizing channel in a version-stable way, so its adapter declares `supports_noise = False` and the orchestrator substitutes the *global* channel of Eq. (3) for the per-gate channels the other three backends apply natively. The two are not equivalent: Eq. (3) accumulates a single scalar λ along the gate list and then mixes the ideal state uniformly, so it is blind to *where* in the circuit an error occurs, to which qubits a gate acts on, and to any correlation between noise and circuit structure; it is exact only when the true channel is itself globally depolarizing. Two consequences follow for cross-backend comparison. On *fidelity* the approximation is close but biased low – 74.0% against the 76.0% the three native-noise backends agree on at $p = 2\%$ (Figure 2), a 2.0 pp gap – and it preserves the ordering of circuits, which is what the harness is used for; the histograms of Figure 3 flatten the same way under both. On *cost* the two are not comparable at all, since mixing a state vector is $O(2^n)$ whereas a per-gate density-matrix simulation is $O(4^n)$. Ket's noisy timings therefore measure a different computation and must not be read as a speed advantage; only its ideal timings are directly comparable. Every emulated result is flagged as such in the report, and native Ket noise is the first item of future work.

The classical fallback, and what it does to conclusions. The quantum fidelity needs a dense ρ, costing $\mathcal{O}(4^n)$ memory and practical to $n \approx 12$–15 on a workstation. Above that – or whenever a backend cannot return a state – the benchmark reports the Bhattacharyya coefficient in place of the kernel fidelity. This fallback is not neutral, and our own data bound its error: it is exactly the phase-blind metric that Table 1 shows to be near-exact on GHZ and Grover (≤ 3.5 pp) but optimistic by 22.5 pp on the variational ansatz – precisely the parameterised, entangling family QML feature maps belong to – with the gap widening as entangling gates are added. Two consequences are practical. First, the fallback value must be read as an *upper bound* on the kernel fidelity, not an estimate: a conclusion of the form "noise has degraded this circuit" holds *a fortiori*, whereas "this circuit is still faithful" does not follow. Second, because the substitution makes the quantum and classical columns numerically identical, its occurrence is visible in the report rather than silent. Above ~ 15 qubits we recommend calibrating the gap on the largest exactly simulable member of the same circuit family; scalable estimators – classical shadows [Huang et al. 2020] and tensor-network contraction [Nguyen et al. 2021] – are the principled fix and are planned.

Other limitations. Qubit-ordering conventions differ across SDKs (Qiskit is little-endian, the others big-endian), so QBenchFrame compares a backend's ideal and noisy states *within* the same backend, making the kernel fidelity convention-invariant. The gate set and code-generation gaps of Section 4.4 bound which circuits can be lowered at all. Finally, QSVM accuracy can stay high on easy data even as alignment falls (Table 2), which is why we report alignment, a more sensitive kernel-level probe, as the primary QML metric.

10. Conclusion and Future Work

QBenchFrame turns cross-simulator comparison for QML workloads into a single, reproducible call, behind a backend-agnostic IR with Qiskit, Cirq, PennyLane and Ket adapters and a rule-based code generator. Because its fidelity is the very quantum kernel that defines QSVMs, benchmarking noise with it measures what degrades quantum-kernel learning. Future work adds native Ket noise, further backends (Qulacs, qsim), scalable fidelity estimators [Huang et al. 2020, Nguyen et al. 2021] and richer noise models.

Availability. A release of the library is available at `https://github.com/TeRaBiTe1024/qbenchframe`.

Acknowledgments. This work was partially supported by the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES-PROEX) - Funding Code 001, the National Council for Scientific and Technological Development (CNPq), and the Amazonas State Research Support Foundation - FAPEAM - through the POSGRAD 2024-2025.

References

Aaronson, S. (2018). Shadow tomography of quantum states. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 325–338.

Aoki, N. et al. (2024). Performance analysis of quantum computer simulators across different environments. In *2024 IEEE/ACIS 22nd Int. Conf. on Software Engineering Research, Management and Applications (SERA)*, pages 115–120.

Arute, F. et al. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510.

Buhrman, H., Cleve, R., Watrous, J., and de Wolf, R. (2001). Quantum fingerprinting. *Physical Review Letters*, 87:167902.

Cerezo, M., Sone, A., Volkoff, T., Cincio, L., and Coles, P. J. (2021). Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12:1791.

da Rosa, E. C. R. and de Santiago, R. (2022). Ket quantum programming. *ACM Journal on Emerging Technologies in Computing Systems*, 18(1):1–25.

Griffiths, D. J. (2011). *Introduction to Quantum Mechanics*. Pearson.

Guo, X., Wang, M., and Zhao, J. (2025). QuanBench: benchmarking quantum code generation with large language models. *arXiv preprint arXiv:2510.16779*.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., and Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature*, 567:209–212.

Huang, H.-L. (2023). Near-term quantum computing techniques: variational quantum algorithms, error mitigation, circuit compilation, benchmarking and classical simulation. *Science China Physics, Mechanics & Astronomy*.

Huang, H.-Y., Kueng, R., and Preskill, J. (2020). Predicting many properties of a quantum system from very few measurements. *Nature Physics*, 16:1050–1057.

Jamadagni, A. et al. (2024). Benchmarking quantum computer simulation software packages: state vector simulators. *arXiv preprint arXiv:2401.09076*.

Knill, E. et al. (2008). Randomized benchmarking of quantum gates. *Physical Review A*, 77:012307.

Li, B. et al. (2022). Pulse-level noisy quantum circuits with QuTiP. *Quantum*, 6:630.

Magesan, E., Gambetta, J. M., and Emerson, J. (2011). Scalable and robust randomized benchmarking of quantum processes. *Physical Review Letters*, 106:180504.

Nguyen, T. et al. (2021). Tensor network quantum virtual machine for simulating quantum circuits at exascale. *arXiv preprint arXiv:2104.10523*.

Rodríguez-Díaz, F. et al. (2024). An experimental comparison of Qiskit and PennyLane for hybrid quantum-classical support vector machines. In *Advances in Artificial Intelligence (CAEPIA 2024)*, pages 121–130.

Schuld, M. (2021). Supervised quantum machine learning models are kernel methods. *arXiv preprint arXiv:2101.11020*.

Schuld, M. and Killoran, N. (2019). Quantum machine learning in feature Hilbert spaces. *Physical Review Letters*, 122:040504.

Sim, S., Johnson, P. D., and Aspuru-Guzik, A. (2019). Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2:1900070.

Temme, K., Bravyi, S., and Gambetta, J. M. (2017). Error mitigation for short-depth quantum circuits. *Physical Review Letters*, 119:180509.