



Svedberg Open  
DISSEMINATION OF KNOWLEDGE

## International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: <https://www.svedbergopen.com/>



Research Paper

Open Access

# Resource-Aware Quantum Big Data Analytics: A Hybrid Quantum-Classical Framework And Scalability Study For Post-Exascale Systems

Shristi Singh<sup>1</sup>, Sandeep Kumar Tiwari<sup>2</sup>, Anand Singh Bisen<sup>3</sup>, Shashank Swami<sup>4</sup>, Anand Kumar Pandey<sup>5</sup>

<sup>1</sup>Research Scholar, Dept. of Computer Science and Engineering, School of Engineering and Technology, Vikrant University, Gwalior, MP, India. Email: [shristis Singh299@gmail.com](mailto:shristis Singh299@gmail.com)

<sup>2</sup>Dept. of Computer Science and Engineering, School of Engineering and Technology, Vikrant University, Gwalior, MP, India. Email: [sandeep72128@gmail.com](mailto:sandeep72128@gmail.com)

<sup>3</sup>Dept. of Computer Science and Engineering, School of Engineering and Technology, Vikrant University, Gwalior, MP, India.

<sup>4</sup>Dept. of Computer Science and Engineering, School of Engineering and Technology, Vikrant University, Gwalior, MP, India.

<sup>5</sup>Dept. of Computer Science and Engineering, School of Engineering and Technology, Vikrant University, Gwalior, MP, India.

### Abstract

Quantum processors are often proposed as accelerators for data-intensive analytics, yet the cost of loading classical data, repeating circuits, and reading out measurements is rarely included when such proposals are assessed. This paper presents Q-BDA, a five-layer framework that places a quantum subroutine inside a hybrid pipeline and judges it with an end-to-end cost model covering encoding, circuit execution, shot sampling, communication, and classical post-processing. The framework is tested with a fidelity quantum kernel built from a two-repetition ZZ feature map and simulated exactly, without noise, on a single CPU core. Over ten random splits, the quantum-kernel support-vector machine reached 94.5%, 98.7% and 92.5% test accuracy on the Moons, Circles and Breast Cancer data, against 97.2%, 99.5% and 94.2% for a tuned RBF kernel. A kernel build needs one circuit per matrix entry, 291,480 circuits at  $N = 800$ , and explicit pairwise simulation was 84 times slower than a statevector shortcut that has no hardware counterpart. When the number of qubits grew from 2 to 12, off-diagonal kernel values fell from 0.39 to 0.0009, test accuracy on Breast Cancer fell from 94% to 73% while the RBF baseline stayed above 94%, and the shots needed to resolve a typical kernel value rose from 1.6 to 1,200. A parametric projection suggests that direct kernel construction for  $10^4$  samples would occupy a superconducting-type device for roughly nine days to five months. The results support treating quantum hardware as a workload-selective accelerator, and they provide a reproducible protocol for deciding when it is worth using.

**Keywords:** Quantum machine learning · Quantum kernels · Big data analytics · Hybrid quantum-classical computing · Post-exascale computing · Resource estimation · Scalability.

## 1 Introduction

Data volumes in science, industry, healthcare, and sensor networks keep growing faster than the machines that analyse them. The move to exascale computing showed that peak arithmetic rate is only one ingredient of useful performance: data movement, memory hierarchy, synchronisation, and energy use increasingly decide how fast a real application runs [1]. Post-exascale systems are expected to be heterogeneous, mixing CPUs, GPUs, and special-purpose accelerators, and quantum processing units are a candidate member of that mix.

Quantum algorithms do offer proven advantages for some problems, including factoring [2] and unstructured search [3], and their reach has been surveyed widely [4]. Whether such advantages carry over to analytics pipelines is a separate question. Classical records must first be mapped into quantum states, circuits must be run many times to estimate expectation values, and the answers must return to classical memory. Several authors have pointed out that these input and output steps can erase an algorithmic speedup [5, 6]. A fair assessment therefore has to look at the whole pipeline rather than at the circuit alone.

Quantum machine learning is the area where this tension is most visible. Fidelity quantum kernels embed data in a Hilbert space and estimate similarities through measurements [7, 8]. Variational circuits are tuned by a classical optimiser [9]. Quantum principal component analysis [10] and quantum support-

vector formulations [11] promise large gains under strong data-access assumptions. Reviews describe a lively field and an unsettled picture of practical benefit [12–14].

This paper asks how a data-analytics team could decide, before spending hardware time, whether a quantum subroutine suits a given workload, and what it would cost. Our contributions are as follows.

1. **Framework.** Q-BDA is a five-layer architecture with a control plane that allocates resources and falls back to classical methods (Section 3).
2. **Cost model.** A single end-to-end model covers encoding, execution, shots, communication, and post-processing, and a short selection procedure uses it to screen candidate quantum methods (Sections 3.2 and 3.4).
3. **Measured scalability study.** Using exact simulation of a ZZ-feature-map kernel, we vary sample size and feature dimension independently and record predictive quality, circuit counts, circuit depth, simulator cost, kernel concentration, and shot noise (Sections 4 and 5).
4. **Reproducible protocol.** Seeds, software versions, raw result files, and the full simulation code accompany the manuscript, and every table below is generated from those files.

All measurements come from a noiseless simulator on a CPU. We do not claim a quantum advantage, and the hardware figures in Section 5.5 are model projections with stated assumptions, not device measurements.

## 2 Related Work

**Quantum machine learning and near-term devices.** Biamonte et al. gave an early overview of how quantum information processing can support learning tasks [15], and Schuld et al. wrote an accessible introduction [16]. Preskill’s description of noisy intermediate-scale quantum (NISQ) devices set the context in which most experiments are now interpreted [17], and Bharti et al. reviewed algorithms designed for that regime [18]. Variational methods such as the variational eigensolver [19] and the quantum approximate optimisation algorithm [20] are typical examples. They are limited by training landscapes that can become flat as circuits grow [21].

**Kernels.** Havlíček et al. ran a supervised classifier on a superconducting processor using a quantum feature map and a classical support-vector machine [7], and Schuld and Killoran described the feature-space view that underlies this approach [8]. Later work showed that the benefit of a kernel depends strongly on data [22] and that a rigorous separation can be constructed for specially designed classification tasks [23]. Two results are central here. First, Thanasi et al. proved that fidelity kernels can concentrate exponentially in the number of qubits, so that estimating them to useful precision needs exponentially many shots [24]. Second, Shaydulin and Wild showed that the bandwidth of the feature map has a large effect on accuracy [25]. Our experiments measure both effects in a single pipeline.

**Quantum linear algebra and data access.** Algorithms for linear systems [26], principal component analysis [10], and support-vector machines [11] can in principle run in time polylogarithmic in the data size, but only if the data are available through quantum random access memory [27] or an equivalent state-preparation routine. Preparing a generic state on  $n$  qubits needs a number of gates that grows exponentially with  $n$  [28]. Classical “dequantised” algorithms that exploit the same sampling access remove the advantage for several of these tasks [29]. Amplitude estimation gives a quadratic gain for estimation problems [30], but only when the underlying state can be prepared cheaply.

**Practicality.** Hoefler et al. argued that practical advantage is most plausible for problems with small data and large compute, since the cost of moving data dominates otherwise [6]. Fault tolerance adds a further multiplier: factoring a 2048-bit integer is estimated to need about twenty million noisy physical qubits [31].

**Gap.** Existing studies treat algorithms, encodings, and hardware separately. What is missing is a compact method that links workload properties, encoding, circuit resources, shot budgets, and measured scaling in one analysis. Q-BDA is intended to fill that gap.

## 3 The Q-BDA Framework

### 3.1 Architecture

Q-BDA consists of five layers linked in sequence and a control plane that spans them (Fig. 1). Layer L1 handles cleaning, scaling, and feature reduction on classical hardware. Layer L2 characterises the workload by sample size  $N$ , feature dimension  $d$ , sparsity, algebraic structure, and accuracy and time budgets. Layer L3 chooses an encoding and a feature map and reports the resulting qubit count. Layer L4 schedules classical and quantum resources, including the number of shots. Layer L5 measures accuracy,

cost, and empirical scaling exponents. The control plane stores the cost model and the resource budget, triggers fallback to a classical baseline when a candidate is infeasible, and records provenance for reproducibility.

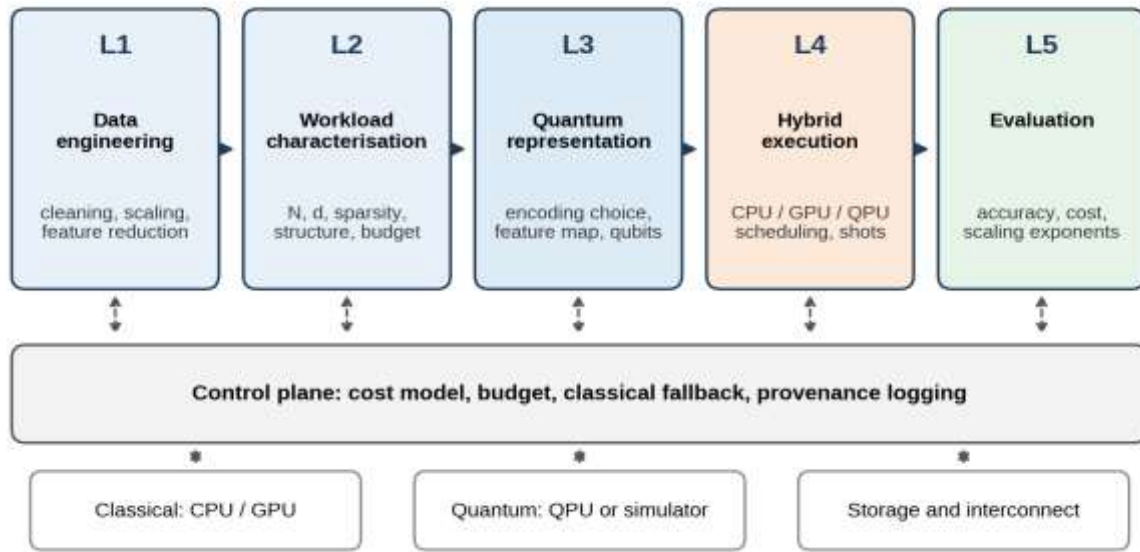


**Fig. 1** The Q-BDA architecture. Layers L1–L5 form the analytical pipeline; a control plane holds the cost model, budget, fallback rules, and provenance log, and mediates access to classical, quantum, and storage resources.

### 3.2 End-to-end cost model

The cost of a hybrid job is the sum of five terms,

$$C_{\text{total}} = C_{\text{cl}} + C_{\text{enc}} + C_{\text{exec}} + C_{\text{comm}} + C_{\text{post}} \quad (1)$$

where  $C_{\text{cl}}$  is classical preprocessing and modelling,  $C_{\text{enc}}$  the conversion of data into circuit parameters or states,  $C_{\text{exec}}$  the quantum execution including measurement,  $C_{\text{comm}}$  the communication and orchestration, and  $C_{\text{post}}$  the classical post-processing. A first-order model for circuit execution is

$$C_{\text{exec}} \approx PS(t_{\text{init}} + Dt_g + t_{\text{ro}}) \quad (2)$$

with  $P$  distinct circuits,  $S$  shots per circuit, circuit depth  $D$ , per-shot initialisation time  $t_{\text{init}}$ , effective gate time  $t_g$ , and readout time  $t_{\text{ro}}$ . For a kernel method with  $n_{\text{tr}}$  training and  $n_{\text{te}}$  test samples, symmetry reduces the number of distinct circuits to

$$P(N) = \frac{n_{\text{tr}}(n_{\text{tr}}+1)}{2} + n_{\text{te}} n_{\text{tr}} = \Theta(N^2). \quad (3)$$

The quadratic growth of  $P$  is the main reason why direct quantum kernels are hard to scale to large data. Because  $P$  multiplies  $S$  in Eq. (2), any growth in the shots needed per entry multiplies the problem further.

### 3.3 Encoding and simulation resources

The qubit count depends on the encoding. Angle or ZZ-type maps use one qubit per feature, amplitude encoding stores  $d$  features in  $\lceil \log_2 d \rceil$  qubits, and a swap-test kernel on amplitude-encoded vectors needs two such registers and an ancilla:

$$n_{\text{angle}} = d, \quad n_{\text{amp}} = \lceil \log_2 d \rceil, \quad n_{\text{swap}} = 2\lceil \log_2 d \rceil + 1. \quad (4)$$

The compact amplitude representation is attractive, but it shifts the cost into state preparation, whose gate count is exponential in the number of qubits for generic data [28]. For simulation on a classical computer, one statevector of  $n$  qubits stored in double-precision complex numbers requires

$$M(n) = 16 \cdot 2^n \text{ bytes}, \quad (5)$$

so memory and time double with every added qubit (Fig. 2). Logical qubit counts from a noiseless simulation should not be read as physical-hardware requirements, since error correction multiplies them substantially [31].

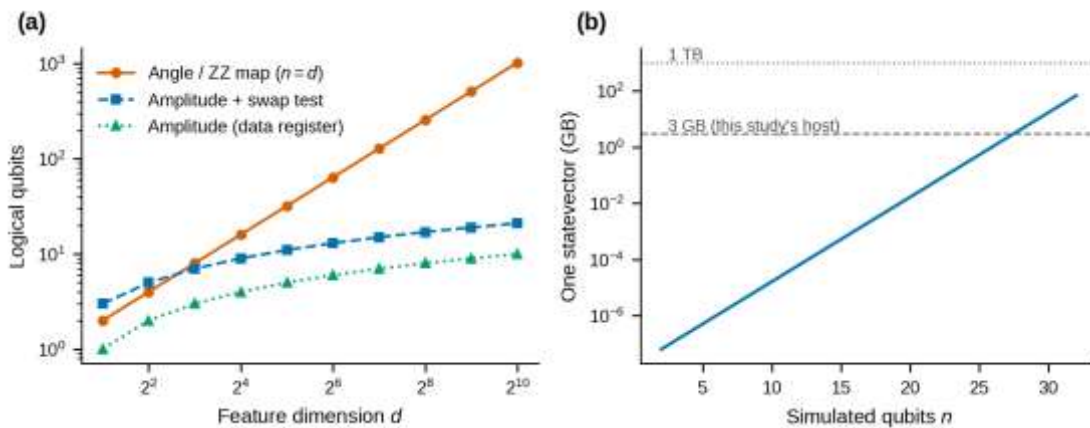


**Fig. 2** (a) Logical qubits required by three encodings as a function of the feature dimension  $d$  (analytic). (b) Memory for one exactly simulated statevector as a function of qubit number  $n$  (analytic, Eq. (5)).

### 3.4 Quantum paradigms and workload selection

Table 1 summarises the candidate paradigms and the cost that dominates each in a data pipeline. Candidates are screened with the following procedure.

#### Algorithm 1. Resource-aware workload selection

**Input:** workload  $W$  (with  $N$ ,  $d$ , sparsity, accuracy target), candidate quantum methods  $Q$ , classical baselines  $B$ , budget  $R$  (qubits, depth, time, shots).

1. Extract  $N$ ,  $d$ , sparsity, data-access pattern, and accuracy target from  $W$  (L1–L2).
2. For each  $q$  in  $Q$ , choose an encoding and estimate qubits  $n(q)$ , depth  $D(q)$ , circuits  $P(q)$ , and shots  $S(q)$  (L3).
3. Estimate  $C_{\text{total}}(q)$  with Eqs. (1) and (2), including encoding and orchestration.
4. Discard every  $q$  whose  $n$ ,  $D$ , or  $C_{\text{total}}$  exceeds  $R$ .
5. If no candidate remains, return the best classical baseline in  $B$ .
6. Otherwise run the surviving candidates and all baselines under identical splits and preprocessing (L4).
7. Record accuracy, runtime, circuit counts, and scaling exponents (L5).
8. Recommend a quantum candidate only if it matches or exceeds the best baseline at acceptable  $C_{\text{total}}$ ; otherwise return the baseline.

**Table 1** Candidate quantum paradigms for analytics and the cost that dominates their end-to-end use

| Paradigm              | Data-access assumption                                        | Dominant end-to-end cost                                      | Outlook (author's assessment)       |
|-----------------------|---------------------------------------------------------------|---------------------------------------------------------------|-------------------------------------|
| Quantum kernel        | Explicit feature map, fidelity estimated by repeated circuits | $N^2$ circuits $\times$ shots; kernel concentration           | Plausible for small $N$ and low $d$ |
| Variational algorithm | Parameterised circuit, classical optimiser                    | Iterations $\times$ gradients $\times$ shots; flat landscapes | Plausible for small models          |
| Quantum PCA           | Density-matrix or qRAM-type access                            | State preparation and precision                               | Unproven for classical data         |
| Amplitude estimation  | Efficiently preparable state                                  | Circuit depth and coherence                                   | Needs fault tolerance               |
| Quantum optimisation  | Problem mapped to a Hamiltonian                               | Embedding, depth, repeated sampling                           | Problem-dependent                   |

## 4 Experimental Design

**Kernel.** Each sample  $\mathbf{x}$  is encoded by a ZZ feature map with  $r = 2$  repetitions and full pairwise entanglement. Each repetition applies a Hadamard layer to all qubits followed by the diagonal operator  $D(\mathbf{x})$ , which acts on the computational basis state with bits  $b_i$  as

$$D(\mathbf{x})|b\rangle = \exp(i \sum_i 2 x_i b_i + i \sum_{i < j} 2 (\pi - x_i)(\pi - x_j) (b_i \oplus b_j)) |b\rangle. \quad (6)$$

The full map is  $U(\mathbf{x}) = [D(\mathbf{x})H^{\otimes n}]^r$ , and the fidelity kernel is

$$K(\mathbf{x}, \mathbf{x}') = |\langle 0^n | U^\dagger(\mathbf{x}') U(\mathbf{x}) | 0^n \rangle|^2. \quad (7)$$

On hardware, an estimate of one entry is the fraction  $k/S$  of shots returning the all-zero string, with  $k \sim \text{Binomial}(S, K)$  and variance  $K(1 - K)/S$ . A bandwidth factor  $\beta$  multiplies the features before encoding. The circuit statistics (depth, two-qubit gates) were obtained by scheduling the gate list of this map as soon as possible on  $n$  qubits.

**Two simulation strategies.** Strategy S1 builds the  $N$  statevectors once and obtains all entries from inner products, a shortcut that is not available on hardware. Strategy S2 simulates one compute–uncompute circuit for every entry, which mirrors the circuit count that a device must execute; it is vectorised along rows. Both strategies produce the same kernel to within  $10^{-15}$ , so they differ only in cost.

**Data and protocol.** Table 2 lists the datasets. Each experiment used a stratified 70/30 train–test split, preprocessing fitted on the training part only (standardisation and principal component analysis for Breast Cancer, then min–max scaling to  $[0, \pi]$  for all datasets), and the same split for all methods. Hyperparameters were chosen by five-fold cross-validation on the training part: the box constraint  $C \in \{0.1, 1, 10, 100\}$  for all support-vector machines,  $\beta \in \{0.1, 0.25, 0.5, 1, 2\}$  for the quantum kernel, and  $\gamma \in \{0.01, 0.03, 0.1, 0.3, 1, 3\}$  for the radial-basis-function (RBF) kernel  $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ . Baselines were an RBF support-vector machine [32] and a linear support-vector machine, both from scikit-learn [33]. Predictive results are means and standard deviations over ten splits (seeds 42–51); timing results use the median of three runs where the run was short and a single run otherwise. All code used NumPy [34] on a single CPU core with 3 GB memory (software versions are given in the repository). Accuracy, precision, recall, and F1 are computed on the test set, with class 1 as the positive class.

**Table 2** Datasets and experimental settings

| Dataset       | Type              | N used       | d (= qubits)       | Notes                                             |
|---------------|-------------------|--------------|--------------------|---------------------------------------------------|
| Moons         | Synthetic         | 200          | 2                  | Noise 0.2                                         |
| Circles       | Synthetic         | 200          | 2                  | Noise 0.1, factor 0.4                             |
| Breast Cancer | Real, 30 features | 200 (of 569) | 4; 2–12 in scaling | Stratified subsample; PCA fitted on training part |
| Synthetic-4D  | Synthetic         | 50–800       | 4                  | Four informative features, two clusters per class |

## 5 Results

### 5.1 Predictive performance

Table 3 and Fig. 3a show that the quantum kernel was competitive with the tuned RBF kernel but did not exceed it on any dataset. The gap was 2.7 points on Moons, 0.8 on Circles, and 1.7 on Breast Cancer. A paired Wilcoxon signed-rank test over the ten splits gave  $p = 0.016$  for Moons,  $p = 0.625$  for Circles, and  $p = 0.141$  for Breast Cancer (uncorrected, with only ten splits, so the values are indicative). The linear baseline failed on the radially separated Circles data (accuracy 54.0%), confirming that the nonlinear kernels were doing useful work. Cross-validation tended to select small bandwidths (modal  $\beta = 0.5, 0.25$ , and 0.1 for Moons, Circles, and Breast Cancer), in line with earlier findings on bandwidth [25].

**Table 3** Test-set performance in percent (mean  $\pm$  standard deviation over ten stratified splits,  $N = 200$  per dataset)

| Dataset       | Method         | Accuracy       | Precision      | Recall          | F1-score       |
|---------------|----------------|----------------|----------------|-----------------|----------------|
| Moons         | Quantum kernel | 94.5 $\pm$ 2.2 | 96.1 $\pm$ 3.7 | 93.0 $\pm$ 4.3  | 94.4 $\pm$ 2.3 |
|               | RBF-SVM        | 97.2 $\pm$ 2.2 | 98.0 $\pm$ 2.3 | 96.3 $\pm$ 4.3  | 97.1 $\pm$ 2.3 |
|               | Linear SVM     | 87.3 $\pm$ 3.9 | 87.3 $\pm$ 4.5 | 87.7 $\pm$ 6.3  | 87.3 $\pm$ 4.1 |
| Circles       | Quantum kernel | 98.7 $\pm$ 2.7 | 97.6 $\pm$ 4.7 | 100.0 $\pm$ 0.0 | 98.7 $\pm$ 2.5 |
|               | RBF-SVM        | 99.5 $\pm$ 0.8 | 99.0 $\pm$ 1.6 | 100.0 $\pm$ 0.0 | 99.5 $\pm$ 0.8 |
|               | Linear SVM     | 54.0 $\pm$ 4.2 | 52.7 $\pm$ 3.0 | 75.7 $\pm$ 9.7  | 62.0 $\pm$ 4.8 |
| Breast Cancer | Quantum kernel | 92.5 $\pm$ 3.7 | 93.4 $\pm$ 2.8 | 94.6 $\pm$ 4.2  | 94.0 $\pm$ 3.1 |
|               | RBF-SVM        | 94.2 $\pm$ 3.9 | 94.7 $\pm$ 2.7 | 96.0 $\pm$ 4.5  | 95.3 $\pm$ 3.2 |
|               | Linear SVM     | 93.7 $\pm$ 4.3 | 93.3 $\pm$ 3.4 | 96.8 $\pm$ 4.4  | 95.0 $\pm$ 3.4 |

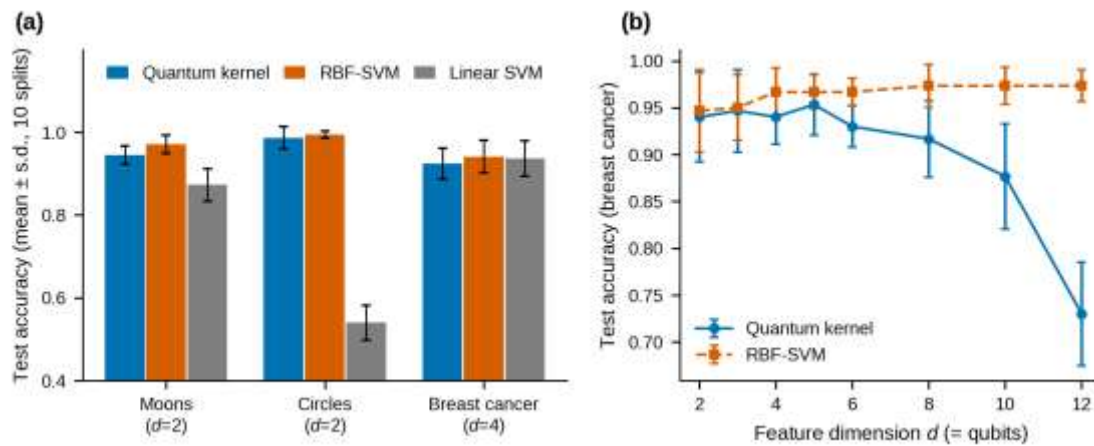


**Fig. 3** (a) Test accuracy of the quantum kernel, RBF, and linear support-vector machines (error bars: one standard deviation over ten splits). (b) Test accuracy on Breast Cancer as the number of principal components, and hence qubits, increases (five splits).

### 5.2 Scaling with sample size

At fixed  $d = 4$ , the number of circuits per kernel build grew exactly quadratically with  $N$  (Table 4). Explicit pairwise simulation (S2) took 0.010 s at  $N = 50$  and 0.81 s at  $N = 800$ . The overall log-log slope was 1.60, below the value of 2 implied by the circuit count, because a fixed cost per row dominates at small  $N$ ; the local exponent between  $N = 400$  and 800 was 1.67. The cached-statevector shortcut (S1) was 84 times faster than S2 at  $N = 800$  and scaled with a slope near 0.99, since its cost is dominated by building the  $N$  states. This difference matters for interpretation: simulator timings obtained with S1 conceal the quadratic circuit count that a real device must pay (Fig. 4). The classical RBF kernel for the same  $N$  took 0.061 s with a plain NumPy implementation, so none of these timings should be read as a statement about hardware speed. Single-split accuracy on this synthetic four-dimensional problem varied with  $N$  in a noisy way and is listed only for completeness.

**Table 4** Sample-size scaling at  $d = 4$  (synthetic four-feature problem, 70/30 split)

| $N$ | Circuits | $N^2$   | S2 (s) | S1 (s) | RBF (s) | Acc. Q / RBF (%) |
|-----|----------|---------|--------|--------|---------|------------------|
| 50  | 1,155    | 2,500   | 0.0099 | 0.0007 | 0.0001  | 46.7 / 66.7      |
| 100 | 4,585    | 10,000  | 0.0247 | 0.0009 | 0.0003  | 70.0 / 73.3      |
| 200 | 18,270   | 40,000  | 0.0695 | 0.0016 | 0.0009  | 80.0 / 76.7      |
| 400 | 72,940   | 160,000 | 0.2524 | 0.0043 | 0.0075  | 90.0 / 95.0      |
| 800 | 291,480  | 640,000 | 0.8055 | 0.0096 | 0.0609  | 87.9 / 85.4      |

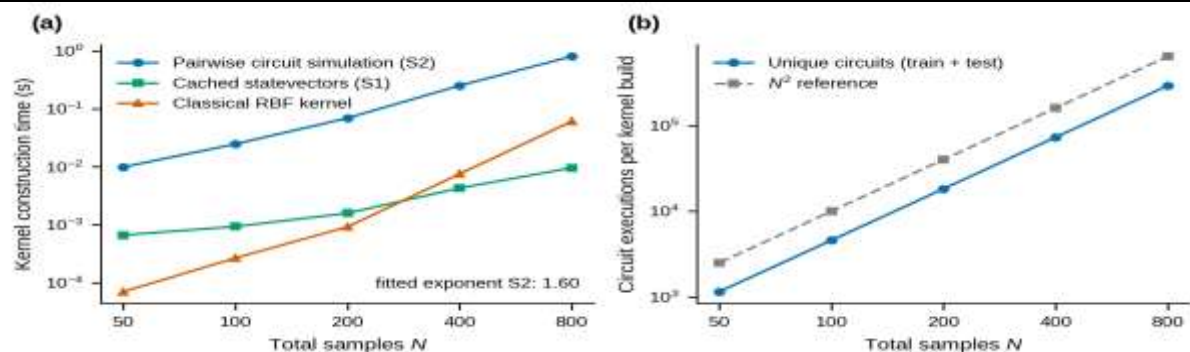


**Fig. 4** (a) Kernel-construction time for pairwise circuit simulation (S2), cached statevectors (S1), and a classical RBF kernel, with the fitted S2 exponent. (b) Number of distinct circuit executions per kernel build (train and test) compared with  $N^2$ .

### 5.3 Scaling with feature dimension and kernel concentration

Table 5 lists the effect of raising  $d$  from 2 to 12 on Breast Cancer with  $N = 200$ . Circuit depth grew roughly linearly with the number of qubits for the kernel's ASAP schedule, while the two-qubit gate count grew as  $2d(d - 1)$  for the two repetitions, reaching 264 for  $d = 12$ . Simulation time grew by a factor of about 2.3 per added qubit for S2 over the larger dimensions (Fig. 5a).

The more consequential effect was kernel concentration. The mean off-diagonal kernel value fell from 0.39 at two qubits to 0.0009 at twelve, a reduction by a factor of about 1.8 per qubit, and the standard deviation across entries fell in step with it (Fig. 5b). If the mean is  $\mu$ , an estimate with standard error equal to  $\mu$  needs about  $(1 - \mu)/\mu$  shots, which rose from 1.6 at  $d = 2$  to 1,200 at  $d = 12$  (Fig. 5c), and this figure applies to every one of the  $P$  entries. Accuracy followed. The quantum kernel dropped from 94% at  $d = 2$  to 73% at  $d = 12$  even in the noiseless case, while the tuned RBF kernel stayed between 94.7% and 97.3% (Fig. 3b). Smaller bandwidths slowed but did not stop the concentration (Fig. 5b), consistent with theory [24].

**Table 5** Feature-dimension scaling on Breast Cancer ( $N = 200$ ; accuracy over five splits)

| d  | Dim. $2^d$ | Depth | 2-qubit gates | S2 (s) | S1 (s) | Mean K | Acc. Q (%)     | Acc. RBF (%)   |
|----|------------|-------|---------------|--------|--------|--------|----------------|----------------|
| 2  | 4          | 10    | 4             | 0.031  | 0.0026 | 0.3915 | 94.0 $\pm$ 4.8 | 94.7 $\pm$ 4.4 |
| 3  | 8          | 22    | 12            | 0.049  | 0.0010 | 0.1465 | 94.7 $\pm$ 4.4 | 95.0 $\pm$ 3.5 |
| 4  | 16         | 31    | 24            | 0.066  | 0.0017 | 0.0731 | 94.0 $\pm$ 2.9 | 96.7 $\pm$ 2.6 |
| 5  | 32         | 40    | 40            | 0.127  | 0.0029 | 0.0368 | 95.3 $\pm$ 3.2 | 96.7 $\pm$ 1.8 |
| 6  | 64         | 49    | 60            | 0.245  | 0.0054 | 0.0188 | 93.0 $\pm$ 2.2 | 96.7 $\pm$ 1.5 |
| 8  | 256        | 67    | 112           | 1.285  | 0.0288 | 0.0054 | 91.7 $\pm$ 4.1 | 97.3 $\pm$ 2.3 |
| 10 | 1024       | 85    | 180           | 8.297  | 0.2399 | 0.0019 | 87.7 $\pm$ 5.6 | 97.3 $\pm$ 2.0 |
| 12 | 4096       | 103   | 264           | 38.170 | 1.0313 | 0.0009 | 73.0 $\pm$ 5.5 | 97.3 $\pm$ 1.7 |

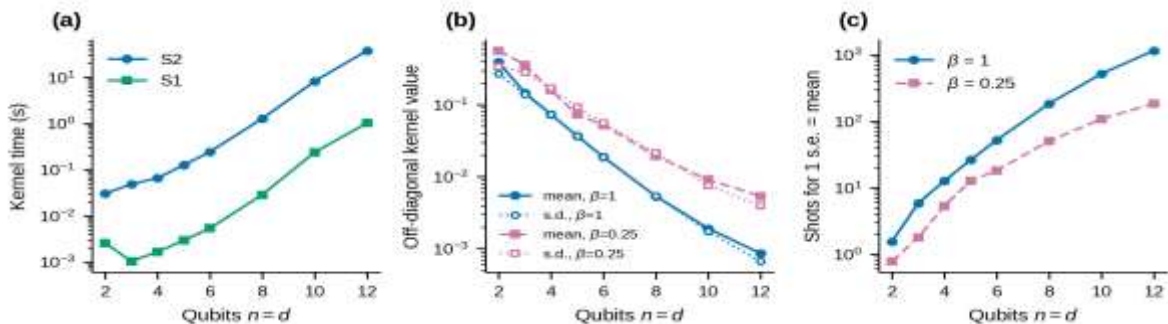


**Fig. 5** (a) Kernel-construction time against qubit number. (b) Mean and standard deviation of off-diagonal kernel values for two bandwidths. (c) Shots for which the binomial standard error equals the mean kernel value,  $(1 - \mu)/\mu$ .

### 5.4 Finite-shot sampling

On a device each entry is estimated from a finite number of shots. Fig. 6a shows the effect at  $d = 2$  (Moons) and  $d = 4$  (Breast Cancer), with the sampled kernel projected onto the nearest positive semidefinite matrix and hyperparameters re-selected for every setting. With 10 shots, accuracy fell to 83% and 82%, compared with 94.5% and 92.5% for the exact kernel. With 1000 shots it recovered to 91.8% and 91.3%. For  $N = 200$  this means 18,270,000 shots in total for a single kernel build at 1000 shots per entry.

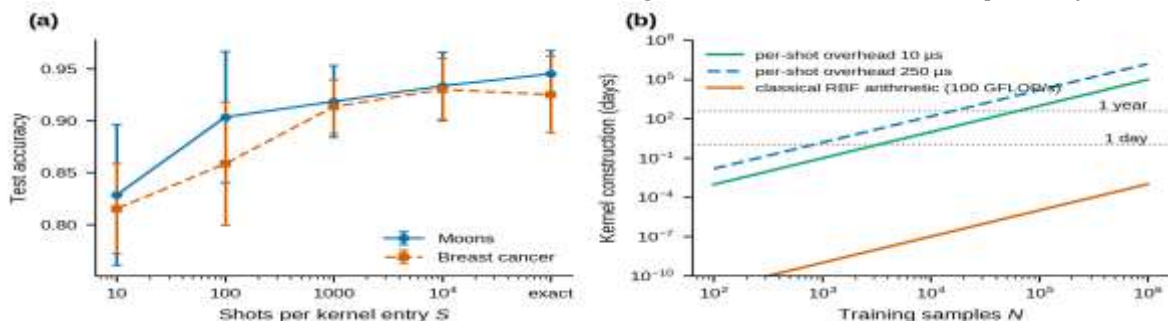


**Fig. 6** (a) Test accuracy of the quantum kernel as a function of shots per kernel entry (ten splits). (b) Projected time for direct kernel construction on a device (Eq. (2),  $S = 1000$ ,  $D = 62$ ,  $t_g = 0.1 \mu$ s) for two assumed per-shot overheads, compared with classical RBF arithmetic at an assumed 100 GFLOP/s. The curves are model projections, not measurements.

### 5.5 Projection to hardware

Using Eq. (2) with  $S = 1000$  shots, the measured depth of the four-qubit kernel circuit ( $D = 62$ ), an effective gate time of  $0.1 \mu\text{s}$ , and per-shot overheads of  $10 \mu\text{s}$  and  $250 \mu\text{s}$ , Table 6 estimates the time to build the training kernel alone. These parameters are assumptions chosen to illustrate the structure of the cost, not measurements of a specific machine. The conclusion is robust to their exact values: because  $P$  grows with  $N^2$  and each entry needs many shots, direct kernel construction moves from hours or days at  $10^3$  samples to centuries at  $10^6$  samples, whereas the arithmetic of the classical RBF kernel remains within minutes.

**Table 6** Projected time to build the training kernel ( $d = 4$ ,  $S = 1000$ ; model estimate under assumed parameters)

| N      | Distinct circuits    | 10 $\mu\text{s}$ overhead | 250 $\mu\text{s}$ overhead | Classical RBF |
|--------|----------------------|---------------------------|----------------------------|---------------|
| $10^3$ | $5.0 \times 10^{05}$ | 2.2 hours                 | 1.5 days                   | 0 s           |
| $10^4$ | $5.0 \times 10^{07}$ | 9.4 days                  | 148.2 days                 | 0 s           |
| $10^5$ | $5.0 \times 10^{09}$ | 2.6 years                 | 40.6 years                 | 1 s           |
| $10^6$ | $5.0 \times 10^{11}$ | 3 centuries               | 41 centuries               | 1.4 minutes   |

## 6 Discussion

Three findings follow from the measurements. First, in the best case the quantum kernel matched, but did not beat, a tuned classical kernel on small problems. This agrees with the view that kernel quality depends on how well the feature map fits the data [22]. Second, the cost driver changes with the regime. For small  $d$ , the quadratic number of circuits and the shots per circuit dominate; for larger  $d$ , concentration dominates, and it affects both accuracy and the shot budget. Third, the way a simulator is used can hide cost. A cached-statevector kernel looks cheap, yet it relies on access to amplitudes that hardware does not provide.

These results translate into practical rules for the workload-selection step of Q-BDA. A direct fidelity kernel suits problems with a modest number of samples, a small feature dimension after reduction, and a nonlinear structure that a classical kernel does not capture well. It is a poor match for datasets with millions of rows unless the training set is reduced or the kernel approximated. Methods that rely on amplitude encoding, qRAM, or quantum linear algebra must show that state preparation is included in their complexity claims, as the data-access assumptions of those algorithms have already been shown to matter [5, 29].

For post-exascale systems, the findings favour a heterogeneous design in which the control plane places the quantum step only on a small, high-value subproblem, keeps data movement local, and retains classical fallbacks. Energy should also be measured, and a natural extension of Eq. (1) is a matching energy sum.

## 7 Limitations

All results come from noiseless exact simulation on a single CPU core, so they exclude gate errors, readout errors, queueing, and calibration drift. Only one feature-map family (ZZ with two repetitions) and one kernel type were studied. The datasets are small by big-data standards and the largest simulated system has twelve qubits. Predictive results use ten splits, and the significance tests are uncorrected. The sample-size study uses one synthetic problem and a single split for accuracy. The hardware projections depend on assumed parameters. Finally, the classical baselines were tuned with a small grid; stronger baselines would only make the comparison harder for the quantum kernel.

## 8 Conclusion

This paper introduced Q-BDA, a five-layer framework with an end-to-end cost model for deciding whether a quantum subroutine belongs in a data-analytics pipeline. Simulation of a ZZ-feature-map kernel showed competitive accuracy on small data, a circuit count that grows as  $N^2$ , a kernel that concentrates by roughly a factor of 1.8 per added qubit, and a shot budget that rises accordingly. The projection in Section 5.5 shows why direct kernels cannot simply be scaled to large datasets. Quantum processors are therefore best treated as workload-selective accelerators, chosen through explicit cost accounting. Future work will add noisy simulation and hardware runs, other feature maps and kernel approximations, more application datasets, energy measurements, and fault-tolerant resource estimates.

### Declarations

**Funding.** No external funding is reported for this work.

**Competing interests.** The author declares no competing interests.

**Ethics approval.** Not applicable. Only public benchmark datasets and synthetic data were used.

**Data availability.** The datasets are available through scikit-learn (Moons, Circles, Breast Cancer [35]). All raw and processed result files accompany this manuscript (results folder).

**Code availability.** The simulation and figure scripts (qbda\_experiments.py, make\_figures.py) accompany this manuscript and should be deposited in a public repository with a persistent identifier before submission [repository DOI to be added].

**Author contributions.** [To be completed by the author.]

## References

1. Reed, D.A., Dongarra, J.: Exascale computing and big data. *Commun. ACM* 58(7), 56–68 (2015). <https://doi.org/10.1145/2699414>
2. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* 41, 303–332 (1999). <https://doi.org/10.1137/S0036144598347011>
3. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pp. 212–219 (1996). <https://doi.org/10.1145/237814.237866>
4. Montanaro, A.: Quantum algorithms: an overview. *npj Quantum Inf.* 2, 15023 (2016). <https://doi.org/10.1038/npjqi.2015.23>
5. Aaronson, S.: Read the fine print. *Nat. Phys.* 11, 291–293 (2015). <https://doi.org/10.1038/nphys3272>
6. Hoeffler, T., Häner, T., Troyer, M.: Disentangling hype from practicality: on realistically achieving quantum advantage. *Commun. ACM* 66(5), 82–87 (2023). [arXiv:2307.00523](https://arxiv.org/abs/2307.00523)
7. Havlíček, V., Córcoles, A.D., Temme, K., et al.: Supervised learning with quantum-enhanced feature spaces. *Nature* 567, 209–212 (2019). <https://doi.org/10.1038/s41586-019-0980-2>
8. Schuld, M., Killoran, N.: Quantum machine learning in feature Hilbert spaces. *Phys. Rev. Lett.* 122, 040504 (2019). <https://doi.org/10.1103/PhysRevLett.122.040504>
9. Cerezo, M., Arrasmith, A., Babbush, R., et al.: Variational quantum algorithms. *Nat. Rev. Phys.* 3, 625–644 (2021). <https://doi.org/10.1038/s42254-021-00348-9>
10. Lloyd, S., Mohseni, M., Rebentrost, P.: Quantum principal component analysis. *Nat. Phys.* 10, 631–633 (2014). <https://doi.org/10.1038/nphys3029>
11. Rebentrost, P., Mohseni, M., Lloyd, S.: Quantum support vector machine for big data classification. *Phys. Rev. Lett.* 113, 130503 (2014). <https://doi.org/10.1103/PhysRevLett.113.130503>
12. Houssein, E.H., Abohashima, Z., Elhoseny, M., Mohamed, W.M.: Machine learning in the quantum realm: the state-of-the-art, challenges, and future vision. *Expert Syst. Appl.* 194, 116512 (2022). <https://doi.org/10.1016/j.eswa.2022.116512>
13. Peral-García, D., Cruz-Benito, J., García-Peñalvo, F.J.: Systematic literature review: quantum machine learning and its applications. *Comput. Sci. Rev.* 51, 100619 (2024). <https://doi.org/10.1016/j.cosrev.2024.100619>
14. Cerezo, M., Verdon, G., Huang, H.-Y., Cincio, L., Coles, P.J.: Challenges and opportunities in quantum machine learning. *Nat. Comput. Sci.* 2, 567–576 (2022). <https://doi.org/10.1038/s43588-022-00311-3>
15. Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S.: Quantum machine learning. *Nature* 549, 195–202 (2017). <https://doi.org/10.1038/nature23474>
16. Schuld, M., Sinayskiy, I., Petruccione, F.: An introduction to quantum machine learning. *Contemp. Phys.* 56, 172–185 (2015). <https://doi.org/10.1080/00107514.2014.964942>
17. Preskill, J.: Quantum computing in the NISQ era and beyond. *Quantum* 2, 79 (2018). <https://doi.org/10.22331/q-2018-08-06-79>
18. Bharti, K., Cervera-Lierta, A., Kyaw, T.H., et al.: Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.* 94, 015004 (2022). <https://doi.org/10.1103/RevModPhys.94.015004>
19. Peruzzo, A., McClean, J., Shadbolt, P., et al.: A variational eigenvalue solver on a photonic quantum processor. *Nat. Commun.* 5, 4213 (2014). <https://doi.org/10.1038/ncomms5213>
20. Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm. [arXiv:1411.4028](https://arxiv.org/abs/1411.4028) (2014)
21. McClean, J.R., Boixo, S., Smelyanskiy, V.N., Babbush, R., Neven, H.: Barren plateaus in quantum neural network training landscapes. *Nat. Commun.* 9, 4812 (2018). <https://doi.org/10.1038/s41467-018-07090-4>
22. Huang, H.-Y., Broughton, M., Mohseni, M., et al.: Power of data in quantum machine learning. *Nat. Commun.* 12, 2631 (2021). <https://doi.org/10.1038/s41467-021-22539-9>

23. Liu, Y., Arunachalam, S., Temme, K.: A rigorous and robust quantum speed-up in supervised machine learning. *Nat. Phys.* 17, 1013–1017 (2021). <https://doi.org/10.1038/s41567-021-01287-z>
24. Thanasilp, S., Wang, S., Cerezo, M., Holmes, Z.: Exponential concentration in quantum kernel methods. *Nat. Commun.* 15, 5200 (2024). <https://doi.org/10.1038/s41467-024-49287-w>
25. Shaydulin, R., Wild, S.M.: Importance of kernel bandwidth in quantum machine learning. *Phys. Rev. A* 106, 042407 (2022). <https://doi.org/10.1103/PhysRevA.106.042407>
26. Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* 103, 150502 (2009). <https://doi.org/10.1103/PhysRevLett.103.150502>
27. Giovannetti, V., Lloyd, S., Maccone, L.: Quantum random access memory. *Phys. Rev. Lett.* 100, 160501 (2008). <https://doi.org/10.1103/PhysRevLett.100.160501>
28. Plesch, M., Brukner, Č.: Quantum-state preparation with universal gate decompositions. *Phys. Rev. A* 83, 032302 (2011). <https://doi.org/10.1103/PhysRevA.83.032302>
29. Tang, E.: A quantum-inspired classical algorithm for recommendation systems. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 217–228 (2019). <https://doi.org/10.1145/3313276.3316310>
30. Brassard, G., Høyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. *Contemp. Math.* 305, 53–74 (2002). <https://doi.org/10.1090/conm/305/05215>
31. Gidney, C., Ekerå, M.: How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum* 5, 433 (2021). <https://doi.org/10.22331/q-2021-04-15-433>
32. Cortes, C., Vapnik, V.: Support-vector networks. *Mach. Learn.* 20, 273–297 (1995). <https://doi.org/10.1007/BF00994018>
33. Pedregosa, F., Varoquaux, G., Gramfort, A., et al.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* 12, 2825–2830 (2011)
34. Harris, C.R., Millman, K.J., van der Walt, S.J., et al.: Array programming with NumPy. *Nature* 585, 357–362 (2020). <https://doi.org/10.1038/s41586-020-2649-2>
35. Street, W.N., Wolberg, W.H., Mangasarian, O.L.: Nuclear feature extraction for breast tumor diagnosis. *Proc. SPIE* 1905, 861–870 (1993). <https://doi.org/10.1117/12.148698>