Quantum State Preparation for Classical Data Encoding

Miguel A. A. Lisboa[1,2], Victor H. F. Brasil[1], João V. H. Duarte[1], Leandro C. Souza[1]

[1]Centro de Informática – Universidade Federal da Paraíba (UFPB)
João Pessoa – PB – Brazil

[2]Centro de Excelência em Computação Quântica – Venturus
Campinas – SP – Brasil

miguel@academico.ufpb.br, victorbrasil@cc.ci.ufpb.br

jvhduarte@gmail.com, leandro@ci.ufpb.br

Abstract. *Quantum state preparation is the process of producing a target quantum state that will be used as the input to a quantum circuit. Many quantum algorithms require an input state where amplitudes, phases, or basis probabilities encode problem data, and the cost of preparing this state can be significant in gate count and circuit depth. This review summarizes common goals, assumptions, and methods for quantum state preparation, with emphasis on preparing states from classical vectors, probability distributions, and feature data used in quantum machine learning. We organize approaches by the information they load and by the resources they require, including gate count, circuit depth, qubit overhead, and classical preprocessing. It also compares exact and approximate preparation procedures, and discusses how precision targets affect cost. The review highlights links between families of methods, typical sources of resource estimates, and criteria that help match a preparation method to a task and hardware constraints.*

1. Introduction

Quantum machine learning uses quantum circuits as computational models for learning tasks such as classification, regression, clustering, and generative modeling [Biamonte et al. 2017]. In most practical settings, the input information is a classical dataset, stored as feature vectors, images, text representations, time series, or samples from empirical distributions. A quantum processor does not accept classical data directly. It operates on quantum states. Therefore, when classical data are used as inputs to a quantum model, a mapping from classical data to quantum states is required [Dunjko and Briegel 2018].

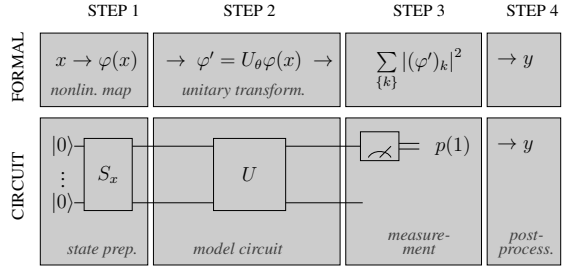

Figure 1. Pipeline of a quantum machine learning model [Schuld et al. 2020].

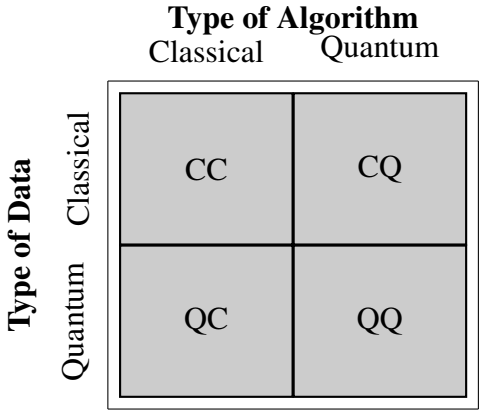

Figure 2. Comparison of classical and quantum data processing with respective algorithms. [Schuld and Petruccione 2021]

Quantum state preparation for classical data encoding is the construction of an input quantum state whose structure represents a chosen classical object. The object can be a bit string, a real or complex vector, a probability distribution, or a set of features. The prepared state can represent this object through basis states, amplitudes, phases, or outcome probabilities [Schuld and Petruccione 2021]. The choice of representation has direct computational consequences. It changes the number of qubits required, the gate count, the circuit depth, and the amount of classical preprocessing. It also changes the error model, because different encodings respond differently to hardware noise and to finite precision in gate synthesis.

The cost of state preparation can be comparable to, or larger than, the cost of the subsequent quantum circuit [Preskill 2018]. This matters in learning workflows because training and evaluation often require repeated executions. In a typical hybrid workflow, a parameterized quantum circuit is executed multiple times to estimate expectation values from measurement outcomes. If each execution requires state preparation from fresh classical data, then preparation cost is multiplied by the number of circuit evaluations [Cerezo et al. 2021]. As a result, state preparation is part of the end to end complexity of a learning method and must be included in resource comparisons.

An encoding is a function that turns a data point into a quantum state. Let $x \in X$ denote an input and let an n-qubit register have state space $\mathcal{H} \cong (\mathbb{C}^2)^{\otimes n}$. An encoding can be written as a map

$$E : X \to \mathcal{D}(\mathcal{H}), \qquad x \mapsto \rho(x), \tag{1}$$

where $\mathcal{D}(\mathcal{H})$ is the set of density operators.

Quantum state preparation is the operational problem of implementing E on hardware. Starting from $|0\rangle^{\otimes n}$ (and optional ancillas), a circuit $U_{\mathrm{enc}}(x)$ produces a physical state that approximates the target,

$$|0\rangle^{\otimes n} \xrightarrow{U_{\mathrm{enc}}(x)} \sigma(x) \approx \rho(x). \tag{2}$$

The resource cost of this step is measured by qubits, circuit depth, and entangling gate count [Nielsen and Chuang 2010]. An accuracy target is also required because hardware

noise and finite gate precision produce preparation error. In learning workflows, this matters because model outputs are obtained from measurements on the encoded state, and any deviation in the prepared state changes the measured statistics.

Different encoding families correspond to different choices of where the data are placed in the quantum state, such as computational basis strings, amplitudes, phases, or output probabilities. These choices trade off implementability on current hardware against compactness and expressivity. In the quantum machine learning setting, the encoding should therefore be treated as part of the model specification, not only as an input formatting step, because it determines both how the input enters the circuit and a significant part of the computational cost [Benedetti et al. 2019].

Surveys of state preparation already exist, both in textbooks [Schuld and Petruccione 2021] and in reviews of embedding techniques [Khan et al. 2024]. We group encodings by the kind of classical object they load, and keep preparation cost in the foreground throughout. The recent literature on exact loading [Sun et al. 2023, Zhang et al. 2022, Gosset et al. 2026] and on structure-exploiting schemes [Rosenkranz et al. 2025, Hur et al. 2022] enters the comparison directly, not as a separate section.

2. Basis Encoding

Basis encoding maps the binary representation of a classical input directly onto the computational basis of a multiqubit register [Schuld and Petruccione 2021]. It is a direct encoding for discrete data because the mapping is defined at the level of bits. Let $x \in \{0,1\}^n$ be a binary string. Basis encoding prepares

$$x \mapsto |x\rangle = |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle \,. \tag{3}$$

The preparation starts from $|0\rangle^{\otimes n}$ and applies Pauli-X gates on the qubits whose bits are equal to 1,

$$|0\rangle^{\otimes n} \xrightarrow{\prod_{i:\, x_i=1} X_i} |x\rangle \,. \tag{4}$$

This preparation has depth 1 if the X gates can be applied in parallel, and it uses no entangling gates. The main resource cost is the number of qubits, which scales linearly with the number of encoded bits.

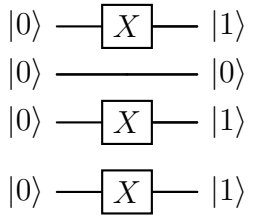

Figure 3. Basis encoding example for $x = 1011$ on $n = 4$ qubits.

In quantum machine learning, basis encoding is used when the input features are discrete, when the data are already binary, or when continuous features are discretized. For example, categorical variables can be represented by binary codes, and quantized real-valued features can be represented by fixed-point bit strings. In these

cases the encoding cost is small and predictable, and state preparation is not the dominant cost. The circuit applied after encoding determines how the input bits are processed [Schuld and Petruccione 2021].

Basis encoding does not compress high-dimensional real vectors, because representing continuous values requires discretization into multiple bits. It is best suited for settings where the relevant structure is combinatorial or where the learning model is designed to operate on bit-level interactions [Nielsen and Chuang 2010]. It is also useful as a reference point for other encodings because the data-loading step is simple and most of the computation is shifted to the model circuit.

3. Amplitude Encoding

Amplitude encoding represents a classical vector by using its entries as the amplitudes of a quantum state. Let $x \in \mathbb{C}^N$ be a vector, with $N = 2^n$ for an n-qubit register. After normalization, amplitude encoding maps x to

$$x \mapsto |\phi(x)\rangle := \sum_{i=0}^{N-1} \alpha_i |i\rangle , \qquad \alpha_i = \frac{x_i}{\|x\|_2}, \tag{5}$$

where $\{|i\rangle\}_{i=0}^{N-1}$ is the computational basis and $\|x\|_2 = \sqrt{\sum_i |x_i|^2}$. If N is not a power of two, a standard choice is to embed x into a larger vector of length 2^n by padding with zeros [Nielsen and Chuang 2010].

This encoding is compact in qubit count. A vector of length N is represented using $n = \log_2 N$ qubits. This is useful in quantum machine learning when feature vectors have large dimension and the available number of qubits is limited. The tradeoff is that the cost moves to state preparation [Shende et al. 2006]. The amplitudes α_i must be created by a circuit, and a generic vector has N independent degrees of freedom. Without additional structure or an explicit data-access model, preparing an arbitrary amplitude-encoded state exactly requires resources that scale with N.

Amplitude encoding also changes how information is accessed by measurement. A single measurement in the computational basis returns an index i with probability $|\alpha_i|^2$. This means that the vector is stored in amplitudes, but direct readout of all entries is not the intended use. Amplitude encoding is most natural when the learning procedure uses aggregate quantities obtained from measurements, rather than attempting to reconstruct the full vector.

For generic inputs, the encoding circuit $U_{\text{enc}}(x)$ that prepares $|\phi(x)\rangle$ from $|0\rangle^{\otimes n}$ uses sequences of controlled rotations arranged in a binary-tree pattern, where rotation angles are computed from partial norms of x. This makes explicit that preparation cost grows with input dimension for unstructured data, and it can translate into substantial circuit depth under hardware connectivity constraints.

With amplitude encoding, the per-evaluation cost in a training loop therefore includes the cost of preparing $|\phi(x)\rangle$ for each input.

Amplitude encoding therefore trades qubit count for preparation cost. It reduces the qubit requirement from N to $\log_2 N$ for a length-N vector, while shifting the main

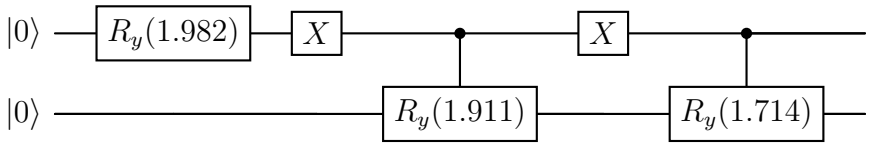

Figure 4. Amplitude encoding example on two qubits. The circuit prepares $|\psi\rangle = \sqrt{0.1}\,|00\rangle + \sqrt{0.2}\,|01\rangle + \sqrt{0.3}\,|10\rangle + \sqrt{0.4}\,|11\rangle$ using a rotation-tree construction.

implementation burden to the preparation routine [Möttönen et al. 2005]. It is most suitable when compact representation is necessary and the available hardware can support the required preparation depth.

4. Angle Encoding

Angle encoding represents classical features by using them as parameters of single-qubit rotations. It is widely used in quantum machine learning because it is simple to implement on gate-based hardware and because the preparation circuit can be shallow. The basic idea is to map a real feature vector $x \in \mathbb{R}^d$ to a quantum state obtained by applying rotations whose angles depend on the components of x [Benedetti et al. 2019].

A common form uses one qubit per feature. For $n = d$, start from $|0\rangle^{\otimes n}$ and apply, for each qubit i, a rotation around a fixed axis,

$$|\phi(x)\rangle := \bigotimes_{i=1}^{n} R_\mu(x_i)\,|0\rangle\,, \tag{6}$$

where $\mu \in \{x, y, z\}$ specifies the rotation axis and

$$R_\mu(\theta) := e^{-i\theta\sigma_\mu/2}. \tag{7}$$

This produces a state in which each feature controls the local state of one qubit. The preparation cost is low because it uses only single-qubit gates, and its depth is typically constant if the rotations are applied in parallel.

The main limitation is dimensionality. If one qubit is used per feature, then d features require d qubits. When the number of features is larger than the available qubits, two strategies are common. The first is feature compression by a classical preprocessing step (for example, dimensionality reduction). The second is feature re-uploading, where the same qubits are used multiple times in a circuit and features are inserted repeatedly as rotation angles [Cerezo et al. 2021]. In this case, the encoding is not only an initial state preparation, but part of the circuit structure.

From a resource perspective, angle encoding is favorable when shallow circuits are required. The preparation uses only single-qubit rotations, and it does not require complex control logic or special memory assumptions. The main costs are the number of qubits needed to represent the feature dimension and the number of times features are re-inserted when feature re-uploading is used. For these reasons, angle encoding is often used as a baseline encoding for near-term quantum machine learning models, especially variational classifiers and regressors that operate on moderate-dimensional classical data [Cerezo et al. 2021].

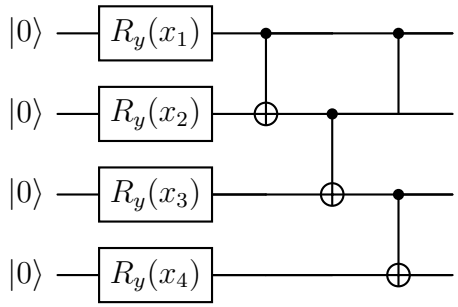

Figure 5. Angle encoding circuit example.

5. Qsample Encoding

Qsample encoding is used when the classical object is a probability distribution. Let p be a distribution over $[N] = \{0, 1, \ldots, N-1\}$ with $N = 2^n$. The qsample state is

$$|\psi_p\rangle := \sum_{i=0}^{N-1} \sqrt{p(i)}\ |i\rangle\,. \tag{8}$$

Measuring $|\psi_p\rangle$ in the computational basis returns outcome i with probability $p(i)$ [Soklakov and Schack 2006]. This is the defining property: the state is prepared so that its measurement statistics match the target distribution.

This encoding is relevant when the learning task is distributional, for example when the goal is to model or generate probability distributions. The practical difficulty is state preparation. Starting from $|0\rangle^{\otimes n}$, a circuit U_p must produce $|\psi_p\rangle$,

$$|0\rangle^{\otimes n} \xrightarrow{U_p} |\psi_p\rangle\,. \tag{9}$$

For a general distribution, exact preparation is costly because $|\psi_p\rangle$ contains N amplitudes. The resource cost therefore depends on what access to p is available. If $p(i)$ can be computed efficiently and has exploitable structure, controlled-rotation constructions can prepare $|\psi_p\rangle$ with manageable overhead. If only classical samples from p are available, coherent preparation typically requires additional assumptions, because classical sampling alone does not specify a circuit that creates the superposition with amplitudes $\sqrt{p(i)}$ [Grover and Rudolph 2002].

In summary, qsample encoding maps a classical distribution to a quantum state that reproduces it under measurement. Its main limitation is that preparation cost depends strongly on the assumed access model for p.

6. QRAM Encoding

QRAM encoding describes a data-loading model where a quantum circuit can query a classical memory coherently. The central idea is that a superposition of addresses can be used to load data values into a register without measuring the address. In its basic form, the QRAM primitive is written as a unitary that acts on an address register and a data register [Giovannetti et al. 2008b]. For a dataset stored as values d_i indexed by

$i \in \{0, \ldots, N-1\}$, the query operation is modeled as

$$\sum_{i=0}^{N-1} c_i \, |i\rangle \, |0\rangle \;\longmapsto\; \sum_{i=0}^{N-1} c_i \, |i\rangle \, |d_i\rangle \, . \tag{10}$$

When the stored object is a feature vector $x^{(i)} \in \mathbb{R}^d$ for each index i, the data register can be taken large enough to hold an encoding of the components, and the query is interpreted as loading the full record $x^{(i)}$.

In quantum machine learning, QRAM is used to express the cost of accessing large classical datasets inside a quantum workflow. It separates two contributions. The first is the cost of the quantum model that consumes the encoded state. The second is the cost of providing coherent access to the data [Giovannetti et al. 2008a]. If QRAM is treated as an available primitive, then certain encodings become simple to describe. For example, it becomes possible to prepare superpositions of training examples,

$$\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \, |x^{(i)}\rangle \, , \tag{11}$$

or to load entries needed by amplitude-type routines through repeated queries. In a feature-map view, QRAM can be used to implement embeddings whose state amplitudes or phases depend on data values that are fetched coherently, which enables overlap-based similarity estimation without reconstructing the data classically from measurement outcomes.

The limiting factor is that QRAM is not a standard capability of current gate-based devices [Preskill 2018]. The model assumes a memory structure that supports coherent routing and low error across many stored cells. This is a strong hardware assumption. Its cost is usually accounted for separately from circuit depth and gate count, because it depends on architectural design, error accumulation, and the physical distance between memory elements. For this reason, QRAM-based encoding is often best interpreted as an input-access assumption that changes the meaning of "efficient data loading" rather than as a specific low-level circuit [Schuld and Petruccione 2021].

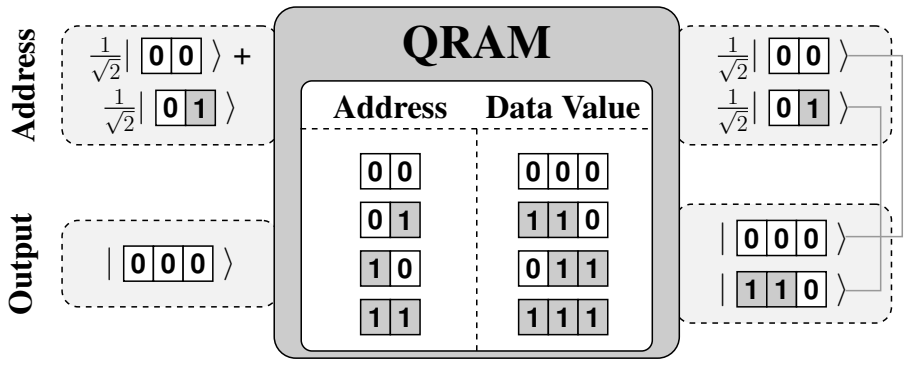

Figure 6. QRAM loading: a superposition of addresses queries a memory and coherently loads the corresponding data values [Khan et al. 2024].

In summary, QRAM encoding formalizes coherent access to classical data inside a quantum computation. In quantum machine learning, it is mainly a way to state what

is assumed about dataset access when analyzing resource scaling for data loading and for overlap-based primitives.

7. Superdense Encoding

Superdense encoding is a communication-based method that uses shared entanglement to transmit classical information efficiently to a quantum processor. It applies when the data owner and the quantum processor are separated, and the relevant cost is the number of communicated qubits [Bennett and Wiesner 1992]. The method starts with a maximally entangled pair shared between sender and receiver, for example the Bell state

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right). \tag{12}$$

Two classical bits $(b_1, b_2) \in \{0,1\}^2$ are encoded by applying a Pauli operator to the sender's qubit,

$$X^{b_1} Z^{b_2} \otimes I \;\; |\Phi^+\rangle\,, \tag{13}$$

and then sending that single qubit to the receiver. The receiver performs a Bell-basis measurement (or an equivalent decoding circuit) to recover (b_1, b_2).

In the context of classical data encoding for quantum machine learning, this mechanism is relevant when classical inputs must be provided to a remote quantum device under communication constraints, and shared entanglement is available as an additional resource. The method does not reduce the local computational cost of preparing an encoding on a single device. It changes the communication budget required to transfer discrete data into the quantum processor. This is mainly applicable to discrete or quantized features, where bits are the basic units of information [Nielsen and Chuang 2010].

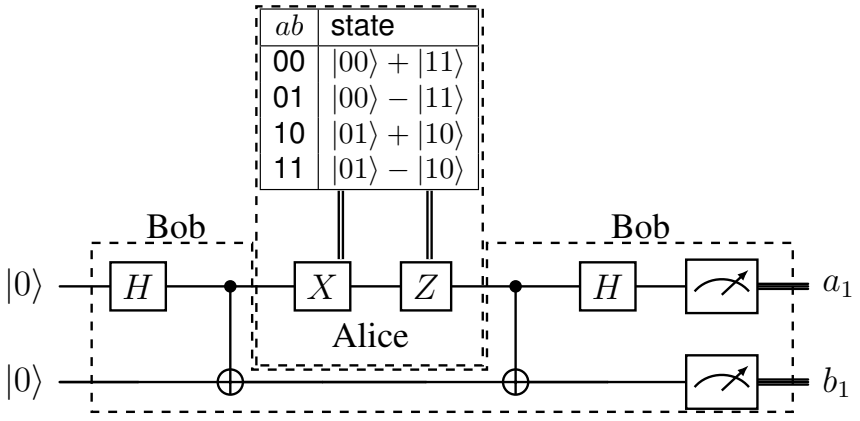

Figure 7. Superdense coding: Alice encodes two classical bits (a, b) by applying $X^a Z^b$ to her half of an EPR pair, and Bob decodes with a Bell-basis measurement [Khan et al. 2024].

In summary, superdense encoding is a state preparation method defined by a distributed setting. It is useful when the objective is to reduce communication required to inject classical bits into a quantum processor, given pre-shared entanglement.

8. Hamiltonian Encoding

Hamiltonian encoding represents classical data by making a Hamiltonian depend on the input and using the resulting dynamics, or its stationary states, as the encoded quantum state. Let $x \in X$ be an input. A Hamiltonian encoding specifies a family of Hamiltonians $H(x)$ acting on an n-qubit register and defines the encoded state through time evolution,

$$|\phi(x)\rangle := e^{-itH(x)} |\psi_0\rangle , \tag{14}$$

for a fixed reference state $|\psi_0\rangle$ and a chosen evolution time t. A common structure is a linear decomposition,

$$H(x) = H_0 + \sum_{j=1}^{d} x_j H_j, \tag{15}$$

where the operators H_j are fixed and the features x_j control their weights. This makes the encoding explicit: the data determine how the register evolves.

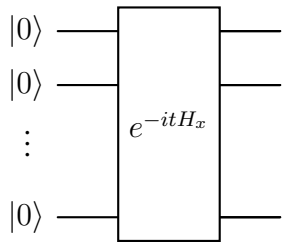

Figure 8. Hamiltonian encoding block: starting from $|0\rangle^{\otimes n}$, apply the data-dependent evolution e^{-itH_x}.

In variational models, data-dependent unitaries can be placed before trainable layers. In this setting, Hamiltonian encoding emphasizes an input-dependent generator $H(x)$, rather than a gate-by-gate specification of the embedding.

The state preparation cost is the cost of implementing $e^{-itH(x)}$ with sufficient accuracy. On digital hardware, this is a Hamiltonian simulation problem. The cost depends on the structure of $H(x)$, the available gate set, the connectivity, and the target precision [Low and Chuang 2017]. On analog or semi-analog hardware, the encoding can be implemented by directly programming couplings so that the physical evolution approximates $H(x)$, which shifts cost from gate depth to calibration and control.

A related variant uses ground-state preparation. In that case, the encoding is defined by the ground state $|g(x)\rangle$ of a Hamiltonian $H(x)$ [Lloyd 1996],

$$H(x) |g(x)\rangle = E_0(x) |g(x)\rangle . \tag{16}$$

This form is relevant when the hardware naturally prepares low-energy states, or when the learning objective is tied to energy landscapes. The practical cost is then the cost of preparing $|g(x)\rangle$ reliably, which depends on spectral gaps, noise, and the chosen preparation method.

In summary, Hamiltonian encoding treats classical data as parameters of a generator of quantum dynamics. It is useful when the data admit a natural operator-based representation or when the hardware supports programmable dynamics efficiently.

9. Comparative Analysis

Classical data encoding affects two things at the same time. It defines how the input enters the quantum state space, and it defines the cost paid before any learning circuit is executed [Preskill 2018]. For quantum machine learning, a comparison between encodings is therefore meaningful only when it addresses both representation and resources under a fixed use case, such as expectation-value prediction or sampling from an output distribution.

Encoding	Qubits	Gate count
Basis	$N\tau$	$\mathcal{O}(N\tau)$
Angle	N	$\mathcal{O}(N)$
Amplitude	$\lceil \log_2 N \rceil$	$\mathcal{O}(N)$
Qsample	$\lceil \log_2 N \rceil$	$\mathcal{O}(N)^{\dagger}$
QRAM	$\lceil \log_2 N \rceil + \tau$	$\mathcal{O}(\log N)$ per query‡
Superdense	$\approx B/2$ (+ ebits)	$\mathcal{O}(B)$
Hamiltonian	n	$\mathcal{O}(\mathrm{Sim}(H(x), t, \varepsilon))$

Table 1. High-level comparison of encoding families. Gate count refers to the total number of elementary gates in the preparation circuit.

The encoding choice also interacts with the learning primitive [Cerezo et al. 2021]. When the learning method is based on expectation values, the encoding controls which functions of x can be expressed efficiently by the chosen circuit family, because the encoding fixes how input dependence enters the unitary that is executed. Encoding also affects which measurements are natural and which classical post-processing is required, because different encodings place information in basis labels, amplitudes, phases, or sampling statistics.

Hardware constraints determine which costs dominate. On noisy devices, depth and two-qubit gates are limiting, which favors encodings whose preparation uses only single-qubit rotations or simple bit flips [Preskill 2018]. In that regime, the practical question is how much expressivity can be obtained from shallow embeddings and shallow entangling layers under shot noise. On fault-tolerant devices, deeper preparation becomes possible, and compact encodings can become attractive, but only when the preparation method does not reintroduce a cost comparable to handling the data classically [Schuld and Petruccione 2021].

Input access assumptions are often the deciding factor in whether an encoding is practical. QRAM-based encoding is best viewed as an access model that allows coherent loading of stored data values. It can simplify the description of data loading and can support preparations that would otherwise be expensive, but it relies on a memory architecture that is not part of standard gate-based devices. Superdense encoding is relevant in a different sense: it reduces the communication required to deliver classical bits to a remote quantum processor when shared entanglement is available. It does not change the local circuit depth needed to represent continuous features, and it is mainly relevant when the bottleneck is communication rather than local state preparation. Hamiltonian encoding shifts the representation into an operator $H(x)$ and produces the encoded state through evolution or stationary-state preparation. Its practicality depends on whether

the hardware can implement the required dynamics efficiently and with controlled error [Low and Chuang 2017].

Finally, accuracy enters through preparation error and finite-shot error. Preparation error changes the input state and therefore changes measured statistics. Shot noise limits how precisely these quantities can be estimated. In training loops, both costs are paid repeatedly, so the relevant unit is the cost per data point per circuit evaluation at a target prediction tolerance. Under this view, an encoding is suitable for quantum machine learning when it provides an input dependence that matches the learning primitive while remaining within the depth, qubit, and shot budgets of the hardware [Benedetti et al. 2019].

10. Conclusion

The preparation of quantum states from classical data constitutes a primary computational bottleneck that often negates subsequent algorithmic acceleration. Quantum Random Access Memory remains physically unfeasible on current hardware, and an arbitrary n-qubit state still has 2^n degrees of freedom whose cost moves between circuit size, depth, ancilla count and T-count [Shende et al. 2006, Sun et al. 2023, Zhang et al. 2022, Gosset et al. 2026] but does not disappear for unstructured inputs. The loading assumption deserves a similar caveat: in several low-rank tasks, classical algorithms with sample-and-query access match the corresponding quantum runtime, yet cheap loading also enables real speedups, as Grover search does for 3SUM in $O(n \log n)$ time. General-purpose loaders should yield to schemes exploiting the input structure [Rosenkranz et al. 2025] or co-designed with the learning model [Hur et al. 2022], and reported with full resource estimates rather than asymptotic bounds alone.

References

Benedetti, M., Lloyd, E., Sack, S., and Fiorentini, M. (2019). Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001.

Bennett, C. H. and Wiesner, S. J. (1992). Communication via one- and two-particle operators on einstein-podolsky-rosen states. *Physical Review Letters*, 69:2881–2884.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. (2017). Quantum machine learning. *Nature*, 549(7671):195–202.

Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S. C., Endo, S., Fujii, K., McClean, J. R., Mitarai, K., Yuan, X., Cincio, L., and Coles, P. J. (2021). Variational quantum algorithms. *Nature Reviews Physics*, 3:625–644.

Dunjko, V. and Briegel, H. J. (2018). Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics*, 81(7):074001.

Giovannetti, V., Lloyd, S., and Maccone, L. (2008a). Architectures for a quantum random access memory. *Physical Review A*, 78:052310.

Giovannetti, V., Lloyd, S., and Maccone, L. (2008b). Quantum random access memory. *Physical Review Letters*, 100:160501.

Gosset, D., Kothari, R., and Wu, K. (2026). Quantum state preparation with optimal T-count. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics.

Grover, L. K. and Rudolph, T. (2002). Creating superpositions that correspond to efficiently integrable probability distributions.

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

Khan, M. A., Aman, M. N., and Sikdar, B. (2024). Beyond bits: A review of quantum embedding techniques for efficient information processing. *IEEE Access*, 12:46118–46137.

Lloyd, S. (1996). Universal quantum simulators. *Science*, 273(5278):1073–1078.

Low, G. H. and Chuang, I. L. (2017). Optimal hamiltonian simulation by quantum signal processing. *Physical Review Letters*, 118:010501.

Möttönen, M., Vartiainen, J. J., Bergholm, V., and Salomaa, M. M. (2005). Transformation of quantum states using uniformly controlled rotations. *Quantum Information & Computation*, 5:467–473.

Nielsen, M. A. and Chuang, I. L. (2010). *Quantum Computation and Quantum Information*. Cambridge University Press.

Preskill, J. (2018). Quantum computing in the nisq era and beyond. *Quantum*, 2:79.

Rosenkranz, M., Brunner, E., Marin-Sanchez, G., Fitzpatrick, N., Dilkes, S., Tang, Y., Kikuchi, Y., and Benedetti, M. (2025). Quantum state preparation for multivariate functions. *Quantum*, 9:1703.

Schuld, M., Bocharov, A., Svore, K. M., and Wiebe, N. (2020). Circuit-centric quantum classifiers. *Physical Review A*, 101:032308.

Schuld, M. and Petruccione, F. (2021). *Machine Learning with Quantum Computers*. Springer.

Shende, V. V., Bullock, S. S., and Markov, I. L. (2006). Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010.

Soklakov, A. N. and Schack, R. (2006). Efficient state preparation for a register of quantum bits. *Physical Review A*, 73:012307.

Sun, X., Tian, G., Yang, S., Yuan, P., and Zhang, S. (2023). Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(10):3301–3314.

Zhang, X.-M., Li, T., and Yuan, X. (2022). Quantum state preparation with optimal circuit depth: Implementations and applications. *Physical Review Letters*, 129(23):230504.