

Review

Quantum-Accelerated Artificial Intelligence for Edge Devices: A Review of Encodings, Models, Hybrid Architectures, and NISQ-Era Realities

Rita Singh ^{1,*} and Angel Deborah Suseelan ²¹ School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA² Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai 603110, India; angeldeborahs@ssn.edu.in

* Correspondence: rsingh@cs.cmu.edu

Abstract

Edge artificial intelligence (Edge AI) requires real-time inference under stringent constraints on computation, memory, energy, and connectivity. Although training can be offloaded to servers, efficient, high-capacity inference and rapid on-device adaptation remain central challenges. Cloud-based inference offers substantial computational power but depends on connectivity, latency, privacy, and reliability conditions that edge deployments cannot always guarantee. Classical model-compression methods—including quantization, pruning, distillation, and neural architecture search—have extended the feasibility of on-device inference, yet they leave largely unchanged the fundamental cost of the linear-algebraic, sampling, and optimization primitives that dominate modern deep learning. Quantum computing has therefore been proposed as a complementary accelerator for selected AI workloads, with theoretical advantages in linear systems, singular value decomposition, sampling, kernel evaluation, and optimization. This review surveys the emerging field of quantum-accelerated AI for edge systems under a hybrid architectural premise: edge devices remain classical, while quantum processors operate as remote, cloud, MEC, or near-edge accelerators. We synthesize advances across quantum learning models, hybrid optimization methods, hardware and deployment architectures, and quantum-inspired approaches suitable for constrained devices. We also assess the practical barriers that currently separate asymptotic quantum advantage from deployable edge intelligence, including data loading, measurement overhead, noise, latency, and benchmarking gaps. Finally, we outline a staged research roadmap from near-term hybrid workflows to fault-tolerant and integrated quantum-edge architectures.

Keywords: quantum computing; edge AI; quantum machine learning; quantum neural networks; variational quantum circuits; quantum data encodings; hybrid quantum–classical algorithms; noisy intermediate-scale quantum (NISQ); quantum federated learning; quantum-inspired classical algorithms; TinyML; edge computing



Academic Editors: Hubert Zarzycki and Lukasz Jelen

Received: 28 April 2026

Revised: 11 June 2026

Accepted: 23 June 2026

Published: 29 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Edge devices are computing systems that operate close to where data are generated, rather than relying entirely on distant cloud servers. Smartphones, wearables, smart cameras, industrial sensors, autonomous vehicles, and other Internet of Things (IoT) systems increasingly perform recognition, prediction, and decision making locally [1–3]. Running artificial intelligence (AI) at the edge can reduce latency, bandwidth use, cloud dependence,

and privacy exposure, but it also imposes stringent limits on computation, memory, energy, and connectivity [4,5].

Classical edge-AI techniques have made substantial progress under these constraints. Mobile SoCs, embedded GPUs, FPGAs, and microcontrollers can now execute quantized, pruned, and otherwise compressed neural networks within tight power budgets [6–11]. These methods have expanded the range of models that can run on-device, but they do not remove the fundamental cost of the operations that dominate modern deep learning: matrix products, singular value decompositions, attention inner products, softmax normalization, sampling, and gradient-based optimization. In practice, compression often trades accuracy, capacity, robustness, or design complexity for efficiency, while on-device training and continual adaptation remain difficult on the most constrained platforms [8,9].

Quantum computing offers a distinct, complementary route to acceleration. Quantum algorithms use superposition, entanglement, and interference to reduce the asymptotic cost of several primitives relevant to machine learning, including linear-system solving [12], singular value transformation [13], inner-product estimation [14], and sampling from complex distributions [15]. Since these primitives underlie many learning, inference, and optimization tasks, quantum subroutines may eventually improve selected AI workloads associated with edge systems [16,17]. However, present quantum hardware remains in the noisy intermediate-scale quantum (NISQ) regime: available devices contain limited numbers of noisy qubits, have short coherence times, and lack full error correction [18,19]. Accordingly, near-term quantum machine learning (QML) is expected to operate primarily through hybrid quantum–classical workflows, in which quantum circuits or quantum models are embedded within classical optimization and deployment pipelines [20–23].

Within this hybrid view, quantum computing does not replace edge computing; it modifies a few specific stages of the edge-AI pipeline that are currently performed in the cloud. Concretely, we expect quantum and quantum-inspired methods to influence edge AI in four ways. First, they can change how off-device training and adaptation are performed, by accelerating selected linear-algebraic and sampling primitives that dominate retraining and personalization loops [12,13,20]. Second, they can change how features and kernels are constructed for compact on-device models, through variational feature maps and quantum kernels whose outputs are deployed as classical artifacts [24–26]. Third, they can change how distributed and federated learning is coordinated, by introducing quantum-assisted aggregation, blind delegated computation, and post-quantum-secure communication across many edge clients [27–30]. Fourth, they can change what runs on the device itself, but only through quantum-inspired classical methods such as tensor-network compression, low-rank factorization, and shadow-style monitoring, which import structural ideas from quantum information into ordinary classical hardware [31–34]. We use the term “quantum-accelerated” edge AI in this broad sense throughout the paper; “quantum-assisted” and “quantum-inspired” are used as more specific labels when only certain stages are offloaded or when no quantum hardware is involved at all.

The combination of quantum computing and edge computing also introduces distinct challenges that classical edge-AI engineering has not previously had to absorb. These include the cost of preparing classical sensor data as quantum states, the latency and queuing overhead of cloud or MEC-mediated quantum calls, the measurement-shot budget required to estimate expectations or sample from a target distribution, the bias introduced by gate, decoherence, and readout errors, the trainability obstacles posed by barren plateaus in deep parameterized circuits, the difficulty of constructing fair end-to-end benchmarks that include both classical and quantum costs, and the security implications of future cryptanalytically relevant quantum computers for long-lived IoT deployments [29,35–39]. Each of these issues is examined later in the paper. We list them here to flag, at the outset,

that the practical viability of a quantum-assisted edge workflow must be judged against a system-level accounting of these costs, not against the asymptotic complexity of a quantum subroutine in isolation.

A central point of clarification, which determines the scope of this review, is that quantum-accelerated edge AI does not require quantum processors to be embedded inside edge devices. This review adopts a hybrid architectural premise: edge devices remain classical, while quantum processors act as specialized accelerators accessed through cloud services, multi-access edge computing (MEC) nodes, near-edge gateways, or, in longer-term scenarios, co-located photonic or spin-based modules [19,40]. In this model, edge devices continue to perform real-time sensing, preprocessing, inference, and control locally. Quantum resources are instead assigned to tasks that can tolerate communication delay or batch execution, such as training, model compression, kernel construction, optimization, calibration, sampling, and periodic model updates. Their outputs return to the edge as classical artifacts, including weights, thresholds, codebooks, policies, or compressed representations. Figure 1 illustrates this system-level organization.

Under this premise, the relevant question is not whether an edge device can execute a quantum circuit. Rather, it is whether a hybrid workflow that invokes a quantum subroutine can reduce total system cost relative to a purely classical baseline. The relevant costs include latency, energy, memory, model size, communication overhead, sample complexity, and accuracy under deployment constraints. For latency-tolerant or intermittently connected workloads, this question is technically meaningful. For hard real-time control loops, remote quantum acceleration is generally unsuitable.

This framing also determines the scope of the review. We focus on quantum methods that could support edge-AI workflows through acceleration, compression, optimization, training, adaptation, or model construction. This includes quantum data encodings, quantum neural models, variational quantum circuits, quantum kernel methods, quantum-assisted optimization and sampling, and quantum-inspired classical algorithms that can run directly on edge hardware. We also consider the systems layer required to connect these methods to practical edge deployments, including cloud quantum services, MEC-based architectures, near-edge accelerators, federated and split learning, and hardware platforms such as superconducting, trapped-ion, photonic, spin, and neutral-atom devices. Concrete edge-AI tasks in which quantum or quantum-inspired methods may matter include anomaly detection in industrial and biomedical sensing, compressed sensing and feature extraction for low-bandwidth links, route and policy optimization for autonomous platforms, federated personalization of mobile and wearable models, and quantum-assisted training or compression of TinyML networks. We return to these use cases in Section 8.

At the same time, we treat claimed quantum advantages critically. Many proposed speedups are asymptotic and depend on assumptions that are difficult to satisfy in edge settings, including efficient state preparation, low measurement cost, sufficient circuit depth, low noise, and favorable communication latency. Moreover, the field lacks standardized benchmarks that jointly measure quantum runtime, classical preprocessing, data-loading overhead, communication cost, energy use, and edge-level task performance. A major goal of this review is therefore to distinguish plausible hybrid opportunities from claims that are unlikely to survive full system-level accounting.

Table 1 consolidates representative prior work that explicitly considers quantum computing in the context of edge, IoT, or distributed-learning settings. The table is intended as an orientation aid: it groups studies by the layer at which quantum or quantum-inspired methods enter the edge-AI workflow, summarizes their main contribution, and indicates whether the contribution is primarily algorithmic, architectural, or quantum-inspired-classical. The remainder of the paper expands on each of these threads with a unified, edge-

aware perspective. We note that this review focuses on quantum computation in the qubit, photonic, and spin-based senses, and does not survey related but distinct nano-electronics paradigms such as quantum-dot cellular automata, which have been proposed for low-power IoT logic and signal processing at the device level rather than for AI workload acceleration [41].

Table 1. Representative prior works that consider quantum computing in edge, IoT, or distributed-learning settings, organized by the workflow layer at which the quantum or quantum-inspired component enters the system.

Workflow Layer	Representative Works	Quantum/Quantum-Inspired Role	Type
Quantum kernels and feature maps	[24–26]	Embed classical features in a quantum Hilbert space; train a classical model on quantum kernel matrices	Algorithmic
Variational quantum models	[20,22,23,42]	Parameterized circuits as trainable modules within hybrid training loops	Algorithmic
Quantum CNN/RNN/attention	[43–47]	Quantum analogues of structured neural layers for feature construction and compression	Algorithmic
Quantum federated and split learning	[27,28,30,48,49]	Quantum-assisted aggregation, blind delegation, or quantum-capable clients/servers for distributed edge learning	Architectural
QML for IoT and edge applications	[40,50,51]	Surveys and case studies that map quantum methods onto IoT anomaly detection, sensing, and lightweight inference	Architectural
Tensor-network compression	[33,52,53]	Matrix-product-state and tensor-train decompositions used as classically deployable compression primitives	Quantum-inspired
Dequantization/sampling	[32,34]	Classical algorithms that match QRAM-based quantum speedups under sampling-access models, deployable on edge hardware	Quantum-inspired
Randomized/shadow-style estimation	[31]	Few-measurement state and feature estimation usable as compressed monitoring on constrained devices	Quantum-inspired
Post-quantum security for edge/IoT	[29,54–56]	Migration of edge cryptography, firmware signing, and federated communication to post-quantum-secure primitives	Architectural
MEC and edge–cloud orchestration	[1–5,57]	Classical MEC, edge–cloud coordination, and resource allocation frameworks that hybrid quantum–classical edge AI inherits and must extend	Architectural

Contributions and Organization

This paper reviews the emerging field of quantum-accelerated AI for edge systems. Its contributions are threefold. First, it consolidates the quantum primitives, data encodings, learning models, and training methods most relevant to edge AI, including their cost profiles and theoretical limitations. Second, it develops a systems-level view of hybrid quantum–edge architectures, covering cloud quantum services, MEC nodes, near-edge modules, quantum federated and split learning, and quantum-inspired classical methods suitable for constrained devices. Third, it critically assesses the gap between asymptotic quantum speedup claims and NISQ-era deployment realities, with emphasis on state-preparation bottlenecks, readout cost, noise, barren plateaus, communication latency, benchmarking gaps, and energy accounting. Throughout, the review separates theoretical or asymptotic quantum advantages from near-term practical ones, and treats the transition from small NISQ demonstrations to industrial-scale edge deployments as a problem of quantum many-body physics, hardware fidelity, and end-to-end systems engineering rather than as routine architectural scaling.

The remainder of the paper is organized as follows. Section 2 provides background on edge computing and edge AI, summarizing the local execution model and the classical compression techniques on which on-device inference depends. Section 3 provides

background on quantum computing, hybrid quantum–classical execution, and quantum machine learning, and presents a working taxonomy of quantum-native, quantum-assisted, and quantum-inspired methods used throughout the paper. Section 4 presents a representative edge device and its hybrid interaction with quantum resources. Section 5 surveys quantum data encodings and their cost profiles, with emphasis on edge-relevant constraints. Section 6 reviews quantum algorithms for neural-network acceleration, including quantum perceptrons and multilayer networks, quantum analogues of convolutional, recurrent, and attention architectures, and training, sampling, and optimization subroutines. Section 7 addresses expressivity, trainability, and generalization theory for quantum models. Section 8 discusses hardware platforms, cloud quantum services, hybrid architectures, and use cases across IoT, mobile, wearables, drones, and autonomous systems, including quantum federated and split learning and quantum-inspired classical methods. Section 9 examines open challenges, including hardware immaturity, data-loading bottlenecks, integration overheads, security implications, benchmarking gaps, and energy accounting. It also outlines near-, mid-, and long-term research directions. Section 10 concludes.

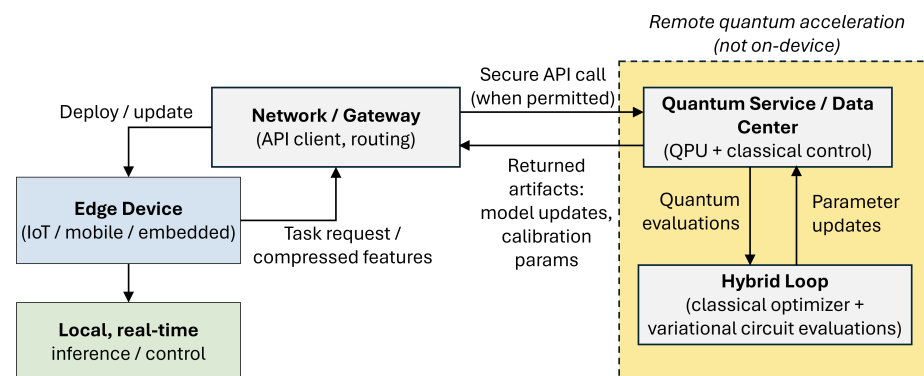


Figure 1. System-level view of quantum-accelerated edge AI. The edge device remains classical and performs real-time inference and control locally, while quantum resources are accessed remotely as specialized accelerators within a hybrid quantum–classical workflow. Forward requests carry compressed features to the gateway and onward to the quantum service, where a hybrid loop alternates variational circuit evaluations with classical optimizer updates. Returned artifacts such as model updates, compressed models, and calibration parameters are deployed back to the edge device.

2. Background on Edge Computing and Edge AI

This section summarizes the edge-AI setting and the classical techniques used to deploy neural networks on constrained devices. Section 3 will then introduce the quantum-computing model, the hybrid quantum–classical execution paradigm, and the quantum machine learning methods relevant to edge systems. These two background sections are deliberately kept separate, since the two communities have largely independent assumptions about data, models, and deployment.

2.1. Edge Computing and Edge AI

Edge computing moves data processing closer to the source of data generation instead of relying exclusively on centralized cloud servers. Edge devices include IoT sensors, smartphones, wearables, embedded controllers, smart cameras, industrial systems, and autonomous platforms. Local processing reduces latency, bandwidth use, cloud dependence, and privacy exposure, and it improves operation when connectivity is intermittent or unavailable [1–4].

Edge AI refers to the execution of AI workloads on or near these devices. In practice, most edge-AI workloads use compact deep neural networks for tasks such as keyword spotting, visual detection, anomaly monitoring, predictive maintenance, biomedical sensing,

and local decision support [5,11]. These workloads run on heterogeneous platforms, including microcontrollers, mobile system-on-chips (SoCs), embedded GPUs, neural processing units (NPU), and FPGAs.

The central constraint is that edge devices must deliver useful inference under tight limits on computation, memory, energy, and latency. Microcontroller-class systems may have only tens to hundreds of kilobytes of SRAM and operate under milliwatt-level power budgets. Mobile and embedded platforms provide more capacity, but they still face thermal, battery, and real-time constraints. These limits shape the models that can be deployed, the precision they can use, and the extent to which training or continual adaptation can occur locally.

2.2. Classical Edge-AI Acceleration

Classical edge-AI acceleration relies mainly on compression, efficient model design, and hardware-aware execution. Quantization reduces the numerical precision of weights and activations from floating point to INT8, INT4, binary, or other low-precision formats. Pruning removes redundant weights, channels, or structures. Knowledge distillation trains a compact student model to imitate a larger teacher. Efficient architectures such as MobileNets, EfficientNets, MicroNet, and MobileViT reduce computation by design. TinyML runtimes such as TensorFlow Lite Micro and CMSIS-NN combine these methods with kernels optimized for constrained hardware [4,8,10,58–65].

These techniques have made edge inference practical across many applications. They can reduce model size, memory traffic, and inference energy by large constant factors. However, they do not change the primitive operations that dominate modern deep learning [66]. Matrix products, attention inner products, softmax normalization, sampling, singular value decompositions, and gradient-based optimization remain costly. Compression improves implementation efficiency, but it often trades accuracy, capacity, robustness, or design complexity for lower resource use. This is why quantum acceleration is of interest in this review. It does not merely offer another compression method. It targets selected linear-algebraic, sampling, and optimization primitives that underlie learning and inference.

Table 2 summarizes the main classical techniques and their limitations.

Table 2. Classical techniques for edge-AI deployment and their main limitations.

Technique	Purpose	Limitation
Quantization	Lowers numerical precision to reduce memory, bandwidth, and arithmetic cost.	May reduce accuracy or robustness, especially at very low precision.
Pruning	Removes redundant weights, channels, or structures.	Requires sparse or structured execution support to realize speedups.
Knowledge distillation	Transfers behavior from a large teacher to a compact student.	Depends on teacher quality and may lose capacity on difficult tasks.
Efficient architectures	Designs models with fewer operations and better hardware fit.	Often requires task-specific search or architecture engineering.
TinyML runtimes	Provide optimized kernels and deployment tools for small devices.	Improve execution efficiency but do not alter algorithmic complexity.

2.3. Recent Advances in Edge and MEC Systems

Beyond compression at the model level, recent work on edge and mobile-edge computing (MEC) has expanded the system-level toolbox on which a future quantum-assisted

edge architecture would build. Comprehensive surveys document how on-device AI is converging with MEC, federated learning, and energy-aware scheduling [3–5,10], and they highlight the importance of coordinated resource allocation and task offloading between many edge clients and a shared infrastructure. Game-theoretic and multi-agent allocation frameworks, for example the Stackelberg-game-based scheme of Zhang et al. for resource allocation and task offloading in MEC-enabled connected intelligent transportation systems [57], are directly relevant here: the same kind of mechanism is needed to decide, in a hybrid quantum–classical edge system, when a quantum subroutine is worth invoking, how to allocate classical preprocessing and queueing among devices, and how to amortize the fixed costs of a remote quantum service across many lightweight clients. We treat these MEC and orchestration techniques as the classical substrate on which quantum-assisted edge AI will sit; the role of quantum computing is then limited to specific, latency-tolerant subroutines invoked through this substrate, rather than to a redesign of the edge layer itself.

3. Background on Quantum Computing and Quantum Machine Learning

Building on the edge-AI background above, this section summarizes the quantum-computing concepts used in the rest of the paper: the basic qubit/circuit model, the hybrid quantum–classical execution paradigm, and the main QML method families. We close with a working taxonomy that separates quantum-native, quantum-assisted, and quantum-inspired approaches; this taxonomy is used throughout the paper to keep the discussion honest about which techniques actually need quantum hardware and which do not.

3.1. Quantum Computing Basics

Quantum computing represents information using quantum states. The basic unit is the qubit, whose pure state is

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1, \quad (1)$$

where $\alpha, \beta \in \mathbb{C}$. An n -qubit register occupies a 2^n -dimensional complex Hilbert space. This exponential state space does not mean that a quantum computer simply performs 2^n classical computations in parallel. A measurement returns a classical outcome sampled from the final probability distribution. Quantum advantage arises when a circuit uses superposition, entanglement, and interference so that useful outcomes become more likely and uninformative outcomes cancel [67,68].

Entanglement allows multi-qubit states to contain correlations that cannot be written as products of single-qubit states [69]. Interference allows amplitudes to combine constructively or destructively. Measurement converts the quantum state into classical information, but it also limits what can be extracted from the state. These features make quantum computing powerful for some structured problems, but they also make data loading, measurement, and circuit design central bottlenecks.

Current devices operate in the noisy intermediate-scale quantum (NISQ) regime [18]. They contain limited numbers of noisy physical qubits, have finite coherence times, and cannot yet support large-scale fault-tolerant computation. Gate errors, readout errors, decoherence, and restricted circuit depth limit the size and reliability of present quantum algorithms [19,70]. As a result, near-term quantum computing is best understood as a specialized accelerator model rather than a standalone replacement for classical computing. The depth, fidelity, and coherence properties of present hardware are not engineering details that will be removed by routine scaling; they reflect open problems in quantum many-body physics, materials science, and quantum error correction [19,70,71]. For this reason, we treat the gap between toy demonstrations and industrial-scale quantum-assisted edge AI as physics-bounded rather than purely software- or architecture-bounded.

3.2. Hybrid Quantum–Classical Computing

Hybrid quantum–classical computing is the dominant near-term execution model. A classical processor prepares inputs, selects circuit parameters, submits circuits to a quantum processor, collects measurement outcomes, and updates the parameters or downstream model. The quantum device is used only for subroutines that may benefit from quantum state evolution or quantum sampling [19,20].

This model is especially relevant to edge AI because it is compatible with the architectural premise of this review. The edge device remains classical and handles real-time sensing, inference, and control. Quantum resources are invoked remotely or near the edge for tasks that can tolerate communication delay, such as training, optimization, kernel evaluation, compression, calibration, sampling, or periodic model updates. The result returned to the edge is classical, such as a set of weights, parameters, thresholds, codebooks, or model updates.

Important hybrid algorithms include the Variational Quantum Eigensolver (VQE), the Quantum Approximate Optimization Algorithm (QAOA), variational quantum classifiers, and quantum neural networks. VQE minimizes the expectation value of a Hamiltonian over circuit parameters [72]. QAOA alternates problem and mixing operations to approximate solutions to combinatorial optimization problems [73]. Variational quantum classifiers and quantum neural networks use parameterized circuits as trainable models whose losses are estimated from measurements and optimized by a classical routine [21–23]. These methods are shallow enough to be plausible on NISQ hardware, but their usefulness depends on noise, data encoding cost, measurement overhead, optimization difficulty, and the full communication cost of the hybrid loop.

3.3. Quantum Machine Learning Overview

Quantum machine learning applies quantum computation to learning, inference, optimization, and data representation [51,74]. The methods most relevant to edge AI fall into four broad groups. Quantum neural networks use parameterized quantum circuits as trainable models. Classical data are embedded into quantum states through feature maps, and trainable gates define the model. Gradients can be estimated by parameter-shift rules or related techniques [22,23,42].

Quantum kernel methods map classical data into a quantum Hilbert space and estimate inner products between quantum feature states. The resulting kernel matrix can then be used by classical methods such as support vector machines [24–26]. These methods are attractive because they separate quantum feature evaluation from classical learning, but they still face data-loading, measurement, and scaling costs.

Quantum-enhanced classical training uses quantum subroutines inside otherwise classical pipelines. Examples include linear-system solving, singular value transformation, sampling, gradient estimation, and optimization [12,13,17]. This family is particularly aligned with the focus of this review because it treats quantum computers as accelerators for selected primitives rather than as replacements for complete edge models.

Quantum generative models use quantum circuits to represent or sample from complex distributions. Examples include quantum circuit Born machines, quantum Boltzmann machines, and quantum generative adversarial networks [75,76]. These models may be relevant to edge settings that require compact generative representations, uncertainty estimation, synthetic data, or anomaly detection, although most demonstrations remain small. Recent work continues to improve the flexibility and controllability of such models; for example, QuanFlex-GAN combines parameterized quantum circuits for noise and label information with a lightweight classical network to synthesize images, and reports improved image quality and training robustness over classical CGANs and earlier QGANs [77].

Hybrid generators of this kind are of interest at the edge mainly as quantum-assisted offline tools—for example for on-device data augmentation or the density component of anomaly detection under resource constraints—while the artifact deployed on the device itself remains classical.

Overall, QML is still an emerging field. Most results are theoretical, simulated, or demonstrated on small hardware instances. Its edge relevance should therefore be judged at the workflow level rather than by asymptotic speedups alone. A useful quantum-assisted edge-AI method must improve an end-to-end metric such as latency, energy, memory, model size, sample efficiency, accuracy, privacy, or adaptation cost after accounting for data preparation, measurement, noise, communication, and classical preprocessing.

3.4. Taxonomy: Quantum-Native, Quantum-Assisted, and Quantum-Inspired Methods

To keep the rest of the paper precise, we distinguish three classes of methods. The distinction is not merely terminological; it determines which hardware is actually required, which workflow layer is affected, and which “quantum advantage” claims are even meaningful for edge deployment.

Quantum-native methods require execution on a quantum processor and have no efficient classical equivalent under standard assumptions. Examples include Shor’s factoring algorithm [35], Grover-style search [78], the HHL linear-system algorithm under its access assumptions [12], and quantum-kernel separations based on intrinsically quantum feature maps [24,26]. For edge AI, these methods are mainly theoretical references: they delineate where a real advantage could exist if data loading, state preparation, depth, and readout could all be paid for at the assumed rates. Most known speedups in this class are highly specialized and depend on specific mathematical structures [39,79].

Quantum-assisted methods use a quantum processor as a subroutine inside an otherwise classical pipeline. The classical system handles data ingest, preprocessing, optimizer state, and deployment; the quantum device is called only for parameterized circuit evaluations, expectation estimation, or sampling. Variational quantum eigensolvers, the quantum approximate optimization algorithm, variational quantum classifiers, quantum convolutional and recurrent modules, and quantum federated learning with a quantum-capable server all sit in this category [22,27,30,43,45,72,73]. This is the class with the most plausible near-term edge relevance, because the quantum component returns a compact classical artifact that can be deployed on a classical edge device.

Quantum-inspired methods use no quantum hardware at all. They borrow structural ideas from quantum information, such as tensor-network factorizations, amplitude-style sampling access, or randomized measurement sketches, and implement them entirely on classical processors [31–34,52,53]. For edge AI, these methods are the most immediately deployable, because they require neither quantum hardware nor a network connection to a quantum service. Their limitation is also clear: once a method has been reduced to classical operations, the achievable speedup is governed by classical complexity, and no NISQ-specific benefit remains.

This taxonomy is used throughout the remainder of the paper. When we discuss algorithms or systems, we identify which class they fall into and which edge-AI workflow stage they affect.

4. Edge-Quantum System Architecture: A Representative Device and Its Hybrid Workflow

This section grounds the system-level premise of the paper in a representative edge device. The goal is not to prescribe a single hardware design, but to make clear how quantum resources would enter an edge-AI workflow in practice. The edge device remains

a classical system. It performs sensing, preprocessing, real-time inference, and control locally. Quantum computation appears only as a remote or near-edge accelerator for selected tasks that can tolerate communication delay.

4.1. Representative Classical Edge Node

A useful reference case is an AI-enabled IoT sensor node used in industrial monitoring, smart infrastructure, healthcare wearables, or mobile sensing. Such devices operate close to the data source and must often run continuously under strict limits on energy, memory, computation, and latency [3,11]. Figure 2 illustrates this class of system and its interaction with external quantum resources.

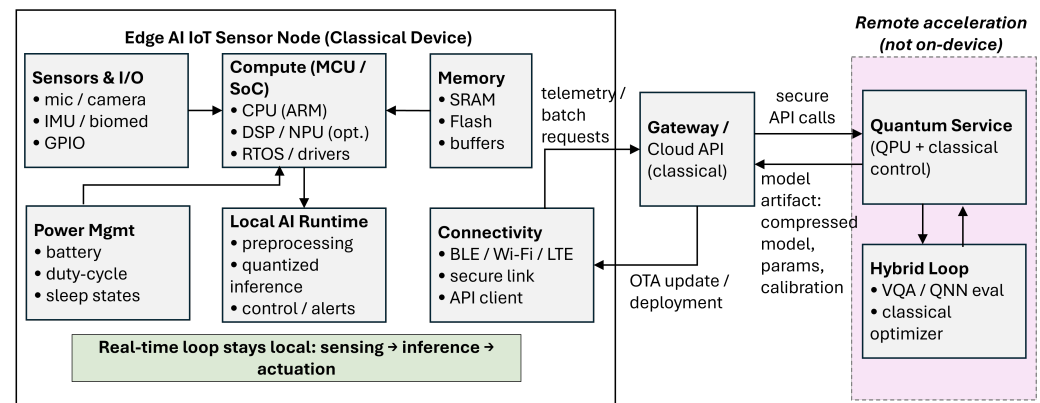


Figure 2. Hybrid workflow for quantum-accelerated edge AI. The edge device remains classical and performs real-time inference and control locally. Latency-tolerant tasks are routed through a gateway or MEC node to remote quantum resources, and the resulting classical artifacts are returned for deployment on the edge device.

A typical node contains sensors, a low-power processor, limited memory, a connectivity module, and power-management circuitry. The processor may be a microcontroller or mobile-class system-on-chip, sometimes assisted by a small DSP or neural processing unit. The memory subsystem usually includes constrained SRAM for runtime buffers and flash memory for firmware and model storage. The sensor interface may connect to microphones, cameras, inertial sensors, industrial probes, or biomedical sensors. Connectivity may be provided through Bluetooth Low Energy, Wi-Fi, LTE, or a gateway link. Power is managed through battery operation, energy harvesting, duty cycling, and sleep states.

The local AI workload is entirely classical. Sensor signals are filtered, normalized, segmented, and passed to compact models such as quantized CNNs, small recurrent models, or other TinyML networks. The device then produces a local decision, alert, control signal, or compressed representation. This local loop is essential because many edge applications cannot depend on round-trip communication with a cloud service. Autonomous control, medical monitoring, safety-critical vision, and low-latency audio processing require predictable timing and local reliability.

The same constraints that make local inference valuable also limit what the device can do. The node can execute a compact model, but it usually cannot support large-scale retraining, expensive hyperparameter search, high-dimensional kernel evaluation, large matrix decompositions, or complex optimization. These tasks are therefore performed off-device when the application can tolerate delay or intermittent connectivity [5]. This division between local real-time execution and off-device adaptation is the opening through which quantum acceleration may enter the system.

4.2. Hybrid Interaction with Quantum Resources

Quantum computing enters this architecture at the workflow level rather than the device level. The edge node communicates with a classical gateway, mobile edge computing node, or cloud backend through secure APIs. That backend prepares the job, manages classical preprocessing, submits circuits or subroutines to a quantum service, collects measurement results, and returns a classical artifact to the edge system. The artifact may be a compressed model, updated weights, optimized thresholds, calibration parameters, a control policy, a codebook, or a feature representation.

This interaction matches the dominant NISQ-era execution model. Present quantum processors are not general replacements for edge hardware. They are specialized accelerators that may be useful for selected subroutines such as optimization, sampling, kernel evaluation, or parts of training and compression [18,19,40]. For example, a hybrid quantum algorithm could be used periodically to tune anomaly-detection thresholds, optimize parameters of a control policy, evaluate a quantum kernel for feature construction, or assist in compressing a model that is later deployed on the edge device. In each case, the deployed model or parameter set remains classical.

The crucial boundary is latency. Real-time sensing, inference, and actuation remain local. Quantum resources are invoked only for tasks whose value exceeds their communication, queuing, data-loading, measurement, and postprocessing costs. This means that quantum acceleration is not relevant to every edge-AI task. It is most plausible for periodic, batch, latency-tolerant, or connectivity-aware workloads where improved optimization, representation, sampling, or compression could reduce total system cost.

This framing also clarifies the practical implication of quantum-accelerated edge AI. The edge device does not need quantum hardware, cryogenics, or quantum control electronics. Its hardware and real-time operating behavior remain classical. The research question is instead whether a hybrid system that includes remote quantum subroutines can improve end-to-end metrics such as accuracy, model size, energy use, memory footprint, sample efficiency, adaptation cost, or deployment latency when compared with a fully classical edge-cloud baseline.

5. Quantum Data Encodings for Edge AI: Cost Profiles and Trade-Offs

Every quantum machine-learning pipeline begins with an encoding map $x \mapsto |\phi(x)\rangle$ that embeds classical data into a quantum state. This choice is not a minor implementation detail. It determines the number of qubits, circuit depth, preparation cost, noise sensitivity, measurement burden, and class of functions that the downstream model can represent [80–82]. For edge-AI workflows in particular, the encoding step is also the boundary at which classical sensor data leave the edge device and become a quantum-side cost: it determines how much classical data must be transmitted to the gateway or quantum service, how the resulting state-preparation circuit interacts with NISQ noise, and how aggressively the device-side feature extractor must compress before transmission. For edge AI, where input dimensionality, latency, bandwidth, and energy are binding constraints, the encoding can dominate the total cost of the hybrid workflow. This section reviews the main encoding families relevant to edge-oriented QML and emphasizes their practical trade-offs.

5.1. Encoding as the Cost Driver

The appeal of quantum machine learning often comes from the size of the quantum state space. An n -qubit register lives in a 2^n -dimensional Hilbert space, so some encodings can represent high-dimensional vectors using few qubits. However, this compact representation is useful only if the state can be prepared, used, and measured efficiently. For edge

applications, this condition is especially important because the full workflow includes classical preprocessing, network transmission, quantum execution, readout, postprocessing, and deployment of a classical artifact back to the device.

Thus, the relevant cost is not only the complexity of the quantum algorithm after the data have been loaded. It is the end-to-end cost of obtaining a useful output from classical edge data. Encodings that are shallow, robust, and compatible with NISQ hardware may be more valuable in practice than encodings that are asymptotically compact but expensive to prepare. This distinction is central to the edge setting.

5.2. Near-Term Encodings for NISQ Workflows

The most practical near-term encodings are those that can be implemented with shallow circuits and modest qubit counts. Basis, unary, angle, phase, Hamiltonian, and IQP encodings fall into this broad category, although they serve different purposes.

In basis encoding, a binary vector $x \in \{0, 1\}^n$ is stored directly in the computational basis:

$$|\phi(x)\rangle = |x_1 x_2 \cdots x_n\rangle. \quad (2)$$

The preparation cost is $O(n)$, since the circuit applies X gates to the qubits for which $x_i = 1$. This encoding is simple and robust, but its expressiveness is limited. Distinct inputs are mapped to orthogonal basis states, and simple readouts recover only limited Boolean structure [81]. Unary, or one-hot, encoding maps a categorical value $i \in \{1, \dots, N\}$ to a state with a single active qubit. It is natural for small categorical variables, routing decisions, token indices, or attention-head selection, but it uses N qubits for a variable with N categories and is therefore costly for large vocabularies.

Angle encoding is often the default NISQ feature map for real-valued data. A vector $x \in \mathbb{R}^n$ is loaded through single-qubit rotations:

$$|\phi(x)\rangle = \bigotimes_{i=1}^n R_y(x_i) |0\rangle = \bigotimes_{i=1}^n [\cos(x_i/2) |0\rangle + \sin(x_i/2) |1\rangle]. \quad (3)$$

This uses n qubits for n features and can be prepared in depth one. Phase encoding uses rotations such as $R_z(x_i)$, often after a Hadamard layer:

$$|\phi(x)\rangle = \bigotimes_{i=1}^n R_z(x_i) H |0\rangle. \quad (4)$$

Angle and phase encodings are periodic, so classical inputs are usually rescaled to a bounded interval such as $[-\pi, \pi]$. Their main advantage is that they are shallow, hardware-friendly, and compatible with repeated data injection. In the data re-uploading construction, the same input is encoded multiple times between trainable layers. A circuit with L such repetitions can represent a truncated Fourier series whose order grows with L [80,83]. This makes angle and phase encodings among the most important near-term tools for expressive QML models on noisy hardware.

Hamiltonian encoding loads data through the parameters of a Hamiltonian $H(x)$ and prepares an evolution of the form $e^{-iH(x)t} |0\rangle$. This encoding is natural when the problem already has physical or combinatorial structure, such as molecular property prediction, control, or optimization. It also appears in VQE-style workflows and in quantum kernels constructed through trotterized dynamics. Its cost depends on the structure of $H(x)$, the simulation method, and the required evolution time.

IQP encodings provide another important NISQ-compatible family. They use Hadamard layers together with diagonal entangling gates to produce feature maps of the form

$$|\phi(x)\rangle = U_{\text{IQP}}(x) H^{\otimes n} |0\rangle, \quad U_{\text{IQP}}(x) = \exp\left(i \sum_S \phi_S(x) \prod_{k \in S} Z_k\right). \quad (5)$$

These circuits are used in quantum-kernel methods and are connected to sampling problems that are conjectured to be classically hard in certain regimes [24]. They also underlie separation results for specific data distributions [26]. For edge-AI workflows, their appeal is that they provide structured nonlinear feature maps with relatively shallow circuits. Their limitation is that the depths required for strong theoretical separations may be sensitive to noise.

5.3. Compact Encodings and the Data-Loading Bottleneck

Amplitude encoding is the most compact standard encoding in qubit count. It stores a normalized vector $x \in \mathbb{R}^N$ in the amplitudes of a $\log_2 N$ -qubit state:

$$|\phi(x)\rangle = \sum_{i=1}^N x_i |i\rangle, \quad \|x\|_2 = 1. \quad (6)$$

This representation is exponentially compact and is assumed by many quantum linear-algebra algorithms, including HHL, quantum PCA, and related methods based on singular value transformation. It is therefore central to many claimed exponential QML speedups.

The main caveat is state preparation. Without exploitable structure or quantum random access, exact amplitude encoding requires $\Theta(N / \log N)$ gates in the worst case [36,81]. This cost can erase the apparent logarithmic advantage of downstream quantum linear algebra. In edge-AI settings, where the original data are classical and often produced by sensors, the loading problem cannot be ignored. Any claimed advantage must count the cost of preparing the amplitude-encoded state, transmitting the relevant data, and extracting useful classical information from the result [36,39].

Quantum random access memory (QRAM) is often proposed as the mechanism that would make amplitude encoding practical at scale. In the ideal model, QRAM maps a superposition of addresses to a superposition of stored data:

$$\sum_i \alpha_i |i\rangle |0\rangle \mapsto \sum_i \alpha_i |i\rangle |x_i\rangle. \quad (7)$$

After an $O(N)$ loading phase, this access is assumed to take $O(\log N)$ time [84]. If such a device were available at scale, it would support many of the strongest quantum linear-algebra speedups. At present, however, large-scale fault-tolerant QRAM has not been realized, and proposed architectures require substantial ancillary hardware and are sensitive to noise [19,36]. For edge AI, QRAM should therefore be treated as an enabling assumption for long-term or asymptotic analysis rather than as a near-term deployment resource.

5.4. Structured and Non-Qubit Encodings

Some encodings are important because they exploit structure in the data or because they point beyond standard qubit-based NISQ workflows. Tensor-network encodings represent classical data through matrix-product states, tree tensor networks, or related low-entanglement structures [33,52]. When the relevant data have limited correlations, a matrix-product-state representation with bond dimension χ can often be prepared with depth scaling as $O(n \log \chi)$. These encodings are especially relevant to edge AI because they connect quantum models to quantum-inspired classical algorithms. If the required bond dimension is small, the same tensor-network structure can often be simulated efficiently on classical hardware, including edge hardware. In that case, the practical benefit

may come not from executing a quantum circuit, but from importing quantum-inspired representations into deployable classical models.

Continuous-variable encodings arise primarily in photonic quantum hardware. Instead of qubits, these systems use modes with continuous quadratures. Classical data may be loaded as displacement amplitudes of coherent states,

$$|\alpha(x)\rangle = D(\alpha(x))|0\rangle, \quad (8)$$

or through squeezing parameters and interferometer phases [85]. Continuous-variable encodings are attractive for signal processing, optical sensing, and possible near-edge photonic accelerators because photonic systems can operate at room temperature and integrate naturally with fiber and optical communication infrastructure. Their algorithmic model is different from the qubit model, since measurements are typically homodyne or heterodyne rather than projective, and the operators are unbounded. For this reason, continuous-variable QML should be viewed as a distinct hardware and algorithmic path rather than as a direct substitute for qubit encodings.

5.5. Cost Summary and Edge Relevance

Table 3 summarizes the main encoding choices and their edge relevance. The practical implication is that encoding must be treated as part of the system cost rather than as a formal preprocessing step. Amplitude encoding is qubit-efficient, but its state-preparation cost often removes the advantage of downstream quantum linear algebra unless QRAM or exploitable data structure is available. Angle, phase, and IQP-style encodings are more realistic near-term choices because they use shallow circuits and integrate naturally with NISQ hardware. Tensor-network and continuous-variable encodings provide alternative paths that may be especially relevant to edge systems, either through quantum-inspired classical deployment or through photonic hardware closer to the network edge.

Table 3. Quantum data encodings and their suitability for edge-AI workflows. n denotes the number of features, $N = 2^n$ the dimension of the classical feature space, and χ the tensor-network bond dimension.

Encoding	Qubits or Modes	Preparation Cost	Edge Role
Basis/unary	n or N	$O(n)$ gates	Categorical features, binary inputs, and small vocabularies
Angle/phase	n	$O(n)$ gates, depth 1	Default NISQ feature maps; compatible with data re-uploading
Amplitude	$\log_2 N$	$O(N/\log N)$ without QRAM	Useful mainly with structured data or strong loading assumptions
QRAM-based	$\log_2 N$ plus auxiliaries	$O(\log N)$ amortized after loading	Long-term enabling assumption; not NISQ-ready
Hamiltonian	Problem-dependent	Depends on $H(x)$ and simulation depth	Physics-structured data, control, and VQE-like workflows
IQP	n	Polynomial depth, often shallow in practice	Quantum-kernel methods and structured nonlinear feature maps
Tensor network/MPS	n or $\log_2 N$	Often $O(n \log \chi)$	Bridge to quantum-inspired classical deployment
Continuous-variable/displacement	Modes	Hardware- and circuit-dependent	Photonic near-edge hardware and signal-processing workflows

6. Quantum Techniques for Neural-Network Acceleration in Edge AI

The relevant question for edge AI is not whether quantum models can replace classical neural networks. Rather, it is whether quantum subroutines can improve selected stages of a learning pipeline whose outputs are returned to the edge as compact classical artifacts. These stages include training, compression, optimization, sampling, kernel evaluation,

calibration, and model adaptation. This framing is essential because real-time inference and control must remain local, while quantum resources are useful only when the task can tolerate communication delay and hybrid execution.

Quantum acceleration is therefore best understood at the level of computational primitives. Modern neural networks rely heavily on linear algebra, similarity estimation, sampling, normalization, and optimization. Dense layers, convolutional operations, attention mechanisms, recurrent updates, implicit layers, and training routines all reduce in part to these operations. Quantum algorithms have been proposed for several of them, including linear-system solving, singular value transformation, inner-product estimation, sampling, and expectation estimation. Their relevance to edge AI depends not only on asymptotic complexity but also on data-loading cost, readout cost, noise, communication overhead, and whether the quantum output can be used without extracting a full classical vector.

Throughout this section we distinguish two regimes for each candidate technique. *Theoretical (asymptotic) advantages* are speedups that hold under idealized assumptions such as efficient state preparation, QRAM access, low noise, sufficient circuit depth, and acceptable readout cost. They describe where a real benefit could exist if these assumptions could be paid for. *Near-term practical advantages*, by contrast, are improvements that remain meaningful when classical preprocessing, data loading, queueing, shot budgets, error mitigation, and communication are explicitly costed. The near-term picture is far more conservative than the asymptotic one: many widely cited quantum speedups do not currently translate into near-term practical edge-AI gains, and the goal of the present section is to identify the subset that plausibly does [36,39,79].

6.1. Primitive-Level Acceleration

Linear algebra is the most developed point of contact between quantum computing and neural-network acceleration. The Harrow–Hassidim–Lloyd (HHL) algorithm solves a linear system

$$A\mathbf{x} = \mathbf{b} \quad (9)$$

by preparing a quantum state proportional to the solution,

$$|x\rangle \propto A^{-1} |b\rangle, \quad (10)$$

under assumptions on sparsity, conditioning, precision, and efficient state preparation [12]. In ideal settings, the dependence on the matrix dimension can be polylogarithmic. This makes HHL conceptually important for inverse problems, least-squares regression, implicit layers, kernel methods, and second-order optimization. It does not, however, directly produce a full classical solution vector. Extracting all entries would remove the advantage. HHL is therefore most relevant when the downstream task requires expectation values, overlaps, classifications, or other low-dimensional summaries of $|x\rangle$ rather than a complete classical reconstruction [16].

A broader modern framework is provided by block-encoding and quantum singular value transformation (QSVT). Given a matrix A , block-encoding embeds A into a larger unitary U_A such that

$$A = \alpha \langle 0|U_A|0\rangle, \quad (11)$$

for a normalization factor α . QSVT then applies polynomial transformations to the singular values of A [13,86]. This framework unifies many quantum linear-algebra routines, including matrix inversion, spectral filtering, low-rank approximation, and matrix-function evaluation [87]. In neural networks, these operations are relevant to model compression, regularization, low-rank attention, kernel methods, and optimization. Their practical

usefulness depends on whether the matrix can be accessed or block-encoded efficiently. For classical edge data, this access assumption is often the bottleneck.

Similarity estimation is another recurring primitive. Attention layers, kernel methods, clustering, nearest-neighbor search, and metric learning all depend on inner products or distances. Quantum circuits can estimate overlaps between states, for example through swap-test-style procedures or related inner-product estimation methods [14,88]. In principle, this can reduce the cost of computing similarities in high-dimensional spaces. In practice, the benefit depends on how the vectors are encoded, how many shots are required, and whether approximate similarities are sufficient. For edge workflows, the most plausible role is not direct on-device attention computation, but remote construction or compression of similarity structures that are later deployed as classical models.

Sampling is equally important. Many learning systems require sampling from probability distributions, including Bayesian models, generative models, stochastic policies, uncertainty estimators, and approximate attention mechanisms. Quantum circuits naturally produce samples from measurement distributions, and quantum generative models such as circuit Born machines, quantum Boltzmann machines, and quantum generative adversarial networks exploit this property [75,76,89]. These methods may be useful when the target distribution is difficult to sample classically or when a compact generative representation can improve an edge model. The challenge is that quantum samples must still be validated, trained, and converted into a form that improves the deployed system.

6.2. Quantum Neural Architectures

Beyond primitive-level subroutines, several quantum models have been proposed as analogues of classical neural architectures [90,91]. These models are best viewed as hybrid components rather than replacements for complete neural pipelines.

The quantum perceptron is the simplest example. Classical perceptrons compute a weighted sum followed by a nonlinear activation [92],

$$y = \sigma(\mathbf{w}^\top \mathbf{x} + b). \quad (12)$$

Quantum versions differ in how they represent the inner product and the activation [93,94]. Phase-kickback perceptrons encode binary inputs and weights through sign-flip operations and estimate an overlap through an ancilla measurement [95]. Unitary perceptrons treat trainable circuit parameters as weights and realize the output by measurement in a chosen basis [96]. Continuous-variable perceptrons use photonic linear transformations and non-Gaussian operations to approximate the linear and nonlinear stages of a neuron [85]. These constructions show that quantum circuits can reproduce neuron-like behavior, but they also expose a central difficulty: nonlinearity usually enters through measurement, and measurement disturbs the quantum state.

As a result, multilayer quantum networks require special composition mechanisms. One approach measures intermediate outputs and re-encodes them into later layers. Another uses ancilla systems and partial tracing to induce nonlinear effective behavior. A third and more practical approach combines small quantum blocks with classical layers, producing a hybrid quantum–classical model whose parameters are trained end to end [42,83,97]. For edge AI, this hybrid pattern is the most plausible. Quantum circuits act as feature maps, kernels, or compact trainable modules, while the classical system provides depth, regularization, deployment tooling, and real-time execution.

Quantum convolutional neural networks extend this idea to spatial or structured data. A classical convolution computes local weighted sums, for example

$$y_{i,j} = \sum_{m,n} w_{m,n} x_{i+m,j+n}. \quad (13)$$

A quantum convolutional layer instead applies a parameterized circuit to local patches encoded as quantum states,

$$|\psi\rangle = U_{\text{conv}}(\theta) |\phi(x)\rangle. \quad (14)$$

Pooling can be implemented through measurement, tracing out qubits, or retaining a subset of the register [43,44]. QCNNs may provide expressive local feature maps and parameter-efficient representations, but current hardware limits them to small inputs and shallow circuits. Their near-term edge relevance therefore lies mainly in offline feature construction, model compression, or hybrid training rather than direct replacement of deployed CNNs.

Quantum recurrent models apply parameterized unitaries to sequential data. A classical recurrent update has the form

$$\mathbf{h}_t = f(W_x \mathbf{x}_t + W_h \mathbf{h}_{t-1}), \quad (15)$$

whereas a quantum recurrent model evolves a quantum hidden state,

$$|h_t\rangle = U(\mathbf{x}_t, \theta) |h_{t-1}\rangle. \quad (16)$$

Such models can represent temporal correlations through quantum state evolution [45,46,98]. Their limitations are the same ones that affect other NISQ models: repeated state preparation, finite coherence, noisy measurements, and unstable training. They may be useful for compact sequence representations, but not for hard real-time sequence processing on edge devices.

Transformer architectures are a particularly important case because they are dominated by attention, similarity estimation, normalization, and large embedding operations [99]. Classical scaled dot-product attention computes

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \quad (17)$$

Quantum methods have been proposed for overlap estimation, low-rank spectral filtering, and sampling-based approximations to attention [47]. For example, if an attention matrix admits a low-rank approximation

$$QK^\top \approx \sum_{i=1}^r \sigma_i u_i v_i^\top, \quad (18)$$

QSVT-style methods could in principle amplify, suppress, or truncate singular components [13]. This aligns with classical efficient attention methods, which already use sparsity, low-rank structure, and random features to reduce complexity. The quantum role would be to assist in producing or optimizing the compressed attention structure, not to run attention directly on a sensor node.

Softmax illustrates both the promise and the caveat. Given logits $\mathbf{z} \in \mathbb{R}^n$,

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}. \quad (19)$$

A quantum sampler could produce outcomes with probabilities

$$\Pr(i) = |\alpha_i|^2 \quad (20)$$

if the amplitudes satisfy $\alpha_i \propto e^{z_i/2}$. This would replace explicit normalization by measurement sampling. However, preparing such a state from classical logits generally re-

quires knowledge of the normalization constant, which is precisely the quantity softmax computes [36,81]. A quantum advantage is therefore plausible only when the amplitudes arise naturally from a quantum subroutine, or when approximate preparation, QRAM, or structured access is available. For edge AI, the more realistic use is remote approximation, sparsification, or compression of attention distributions whose reduced form is then deployed classically.

6.3. Training, Optimization, and Gradients

Training remains one of the most plausible places for quantum resources to enter an edge-AI workflow, because training and adaptation are often performed off-device. A learning model minimizes a loss of the form

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \ell(f_{\theta}(x_i), y_i). \quad (21)$$

Variational quantum algorithms use parameterized circuits to minimize measured expectation values,

$$\min_{\theta} \langle 0 | U^{\dagger}(\theta) H U(\theta) | 0 \rangle, \quad (22)$$

with a classical optimizer updating the parameters [20,72]. This template includes the Variational Quantum Eigensolver, the Quantum Approximate Optimization Algorithm, variational quantum classifiers, and many quantum neural-network models.

Gradients of circuit expectations can often be estimated using the parameter-shift rule:

$$\frac{\partial}{\partial \theta_i} \langle O \rangle = \frac{1}{2} \left[\langle O \rangle_{\theta_i + \frac{\pi}{2}} - \langle O \rangle_{\theta_i - \frac{\pi}{2}} \right]. \quad (23)$$

This enables hybrid backpropagation-like training through quantum modules [42]. Other quantum or quantum-inspired optimization methods, including QAOA, quantum natural-gradient methods, and quantum sampling procedures, may support hyperparameter search, combinatorial model selection, threshold tuning, or policy optimization.

For edge systems, the key point is that the trained output is classical. A remote hybrid loop may optimize weights, thresholds, codebooks, feature maps, or compressed model structures. These artifacts can then be deployed to a classical edge device. This fits the edge constraint that real-time inference must remain local while expensive adaptation can occur asynchronously.

6.4. Complexity, Noise, and Edge Relevance

The apparent advantage of quantum acceleration must be assessed through the full system cost. Idealized asymptotic complexity is informative, but it is not sufficient for edge AI. Data must be loaded, circuits must be executed, results must be measured, classical postprocessing must be performed, and the output must be transmitted back to the edge system. A useful hybrid workflow must satisfy a practical inequality of the form

$$T_{\text{quantum}} + T_{\text{communication}} + T_{\text{overhead}} < T_{\text{classical}}, \quad (24)$$

for the target workload and accuracy requirement.

Table 4 summarizes the main neural-network bottlenecks, candidate quantum techniques, intended benefits, and primary caveats. The table emphasizes that quantum methods target modules within a classical learning pipeline rather than entire edge models.

Table 4. Representative neural-network bottlenecks and candidate quantum-assisted techniques, classified by whether the advantage is currently theoretical/asymptotic (T) or plausibly near-term practical (NT) under NISQ constraints.

Target	Quantum Technique	Potential Benefit	Horizon	Main Caveat
Linear solves and implicit layers	HHL, block-encoding, QSVT	Structured linear-algebra acceleration	T	State preparation, conditioning, readout
Low-rank compression	SVE, QSVT, quantum PCA	Spectral filtering and model compression	T	Efficient matrix access required
Similarity and kernels	Swap test, quantum kernels	Nonlinear feature maps and kernel evaluation	NT	Shot cost and kernel concentration
Attention and softmax	Quantum sampling, low-rank quantum methods	Approximate attention or sparsification	T	Normalization and loading costs
Training and optimization	VQA, QAOA, quantum gradients	Hybrid optimization and parameter search	NT	Barren plateaus and noise
Probabilistic inference	Quantum sampling, Born machines	Sampling from complex distributions	NT	Verification and hardware scale
Tensor-network compression (on-device)	MPS/tensor-train factorization (quantum-inspired)	Classical model compression and structured weight representations	NT (classical)	Depends on data-structure low-rankness

Noise is the main near-term limitation. NISQ devices suffer from gate errors, readout errors, decoherence, restricted connectivity, and limited circuit depth. Variational algorithms also face noise-induced bias and barren plateaus, where gradients become too small to estimate reliably [37,100]. Error-mitigation methods such as zero-noise extrapolation, probabilistic error cancellation, and measurement-error mitigation can improve estimates, but they increase sampling overhead [101,102]. This overhead is acceptable only when the quantum routine is used sparingly, asynchronously, or offline.

These constraints sharpen the edge-AI design rule. Quantum acceleration is most plausible for coarse-grained, latency-tolerant tasks such as periodic retraining, calibration, compression, kernel construction, feature selection, model adaptation, and distributed optimization. It is not plausible for fine-grained real-time inference loops that require deterministic local response. Even in a fault-tolerant future, quantum processors are more likely to become components of heterogeneous cloud, MEC, or near-edge infrastructure than components of small edge devices themselves.

The conclusion is therefore modular and conditional. Quantum techniques may reduce training cost, improve compression, support richer feature maps, or enhance sampling and optimization for models that are ultimately deployed on classical edge hardware. They are unlikely to replace classical edge-AI pipelines directly. Their value must be demonstrated through end-to-end improvements in latency, energy, memory, model size, sample efficiency, robustness, or adaptation cost after all loading, communication, measurement, and postprocessing overheads are included.

7. Expressivity, Trainability, and Generalization

The preceding section identified quantum techniques that may accelerate selected parts of a neural-network pipeline. Whether these techniques can improve an edge-AI system depends on more than circuit complexity. Like any learning model, a quantum model must represent useful functions, be trainable with finite resources, and generalize from finite data. These three properties define the theoretical filter through which proposed quantum advantages must pass.

7.1. Expressivity

Expressivity concerns the class of functions a model can represent. In QML, it is not enough to say that an n -qubit state occupies a 2^n -dimensional Hilbert space. Measure-

ment returns limited classical information, and useful expressivity depends on the input encoding, trainable circuit, measurement rule, and observable being estimated [80,103].

For variational circuits with angle encoding and intermediate trainable layers, data re-uploading provides a precise expressivity result. Repeating the encoding layer L times allows the circuit to represent functions that can be written as truncated Fourier series,

$$f_{\theta}(x) = \sum_{|\omega| \leq L} c_{\omega}(\theta) e^{i\omega x}. \quad (25)$$

The accessible frequencies are determined by the spectrum of the encoding unitary, while the coefficients are controlled by the trainable circuit parameters [80,83]. This gives a concrete link between circuit architecture and function class. It also explains why repeated shallow encodings can be more useful in NISQ workflows than a single deep expressive circuit.

Kernel-based QML provides a second expressivity route. A quantum feature map $|\phi(x)\rangle$ defines a kernel through overlaps such as $|\langle \phi(x) | \phi(x') \rangle|^2$, which can then be used by classical kernel methods [25]. Havlíček et al. demonstrated such a circuit-based kernel on IBM hardware [24], and Liu, Arunachalam, and Temme proved a polynomial separation for a discrete-log-based learning problem under standard cryptographic assumptions [26]. Such results are important because they provide rare end-to-end evidence that a quantum model can define a classically hard feature space. They also show the limitation: the strongest theoretical advantages usually require specially structured data distributions, not arbitrary classical sensor streams.

7.2. Trainability

A circuit that is expressive may still be impossible to train. The central obstacle is the barren plateau phenomenon. For sufficiently random or deep parameterized circuits, the variance of the loss gradient can decay exponentially with the number of qubits,

$$\text{Var}[\partial_{\theta} \mathcal{L}] \sim 2^{-\alpha n}, \quad (26)$$

for some $\alpha > 0$ [37]. In such regimes, gradient-based optimization becomes infeasible because the true gradient is hidden beneath measurement noise. Later work identified several sources of barren plateaus, including excessive entanglement, global cost functions, noise, and overly expressive random ansätze [38,100].

The barren plateau phenomenon has important implications for the “small-models-to-industrial-scale” transition that motivates much of the optimism around QML. The scaling is not a routine engineering hurdle. The exponential suppression of gradient variance is rooted in the concentration of measure on the unitary group and is therefore tied to the physics of how high-dimensional quantum states distribute their amplitude [37,38,100]. As the number of qubits grows and the ansatz becomes more expressive, the signal that drives classical optimization vanishes faster than measurement budgets can compensate. Combined with the additional noise-induced variant of the plateau [100] and with the requirement that the readout precision ϵ scales the measurement budget as $O(1/\epsilon^2)$ [104], this means that simply increasing qubit count and circuit depth does not move a small NISQ demonstration toward industrial-scale deployment. The credible path is the opposite one: keep circuits shallow, restrict the ansatz to problem-informed structure, and reserve quantum modules for subroutines whose useful output is low-dimensional. Any roadmap that assumes barren plateaus will be “scaled away” by hardware improvements alone is physically over-optimistic.

Trainability can be improved by restricting the circuit and cost structure. Shallow, problem-informed ansätze often have more favorable gradient scaling than random deep circuits. Local cost functions can avoid the exponential suppression associated with global observables. Layerwise training and warm starts reduce the effective dimension of the optimization problem during early training. Tensor-network initializations can place the circuit in a low-entanglement region where gradients remain tractable [52]. These strategies are particularly relevant to edge AI because edge-oriented workflows favor small models, compact outputs, and limited training budgets. They also support the broader design principle of this paper: quantum modules should be shallow, structured, and tied to specific subroutines rather than used as large unstructured replacements for classical networks.

Noise further complicates trainability. Gate errors, decoherence, and readout errors bias loss estimates and increase the number of measurements required for reliable gradients. Error mitigation can reduce this bias, but at the cost of additional circuit executions. Consequently, a quantum-assisted edge workflow should avoid designs whose benefit depends on deep circuits, large numbers of parameters, or highly precise gradient estimates. The most plausible designs are small hybrid models, quantum kernels, structured ansätze, and quantum-assisted optimization routines whose outputs remain useful despite moderate noise.

7.3. Generalization

Generalization asks whether a trained quantum model performs well on unseen data. In classical machine learning, generalization depends on data size, parameter count, hypothesis-class capacity, and regularization. Similar considerations apply to quantum models. Caro et al. showed that QNNs with T parameterized gates can generalize from $O(T/\epsilon^2)$ training examples to error ϵ , which is broadly comparable to classical models of similar parameter count [105]. This suggests that quantum models do not automatically escape standard sample-complexity constraints.

Other work has proposed quantum-specific capacity measures. Abbas et al. introduced an effective dimension based on the Fisher information of the model distribution and observed that some QNNs can have higher effective dimension than classical networks with comparable parameter counts [103]. This may correlate with improved data efficiency when the encoding and circuit are well matched to the task. However, greater effective dimension is not automatically beneficial. If the model becomes too expressive relative to the data, it may overfit or become harder to train.

A further limitation comes from classical learnability. Huang et al. argued that for many QML tasks based on classical data, a classical learner can often use the available data to match the predictions of the quantum model [79]. In this view, quantum advantage is most plausible when the data are generated by an intrinsically quantum process, or when the feature map induces structure that is classically hard to simulate, as in certain cryptographic kernel separations [26]. For edge AI, where inputs are usually classical sensor measurements, this is an important caution. Quantum advantage is more likely to appear as a system-level gain in training, optimization, compression, or sampling than as a universal improvement in predictive accuracy.

7.4. Implications for Edge AI

The theoretical picture reinforces the engineering argument of the previous sections. Quantum models are most promising for edge workflows when they are small, shallow, structured, and connected to a specific computational bottleneck. Angle, phase, IQP, and kernel-based encodings are plausible near-term choices when they provide useful

representations without excessive data-loading cost. Deep unstructured circuits are less plausible because trainability and noise will dominate before any expressive advantage can be realized.

Theory also clarifies when quantum-inspired classical methods may be preferable. If the useful structure of a quantum model can be captured by a tensor network, low-rank representation, or other classically simulable approximation, then the best edge deployment may be entirely classical. This is not a failure of the quantum approach. It is a practical route by which quantum theory can inform models that run on constrained hardware.

For this review, expressivity, trainability, and generalization serve as a filter. A quantum acceleration proposal is credible for edge AI only if it represents the required function class, can be trained under realistic noise and sampling budgets, and improves an end-to-end deployment metric after accounting for communication, data preparation, readout, and classical postprocessing. This filter motivates the systems-level analysis in Section 8, where hardware platforms, cloud quantum services, hybrid architectures, and edge-relevant use cases are considered together.

8. Integration with Edge Platforms

The preceding sections identified quantum techniques that may accelerate selected learning, optimization, sampling, and compression tasks. Their relevance to edge AI, however, depends on system integration. In edge settings, quantum computing is useful only when the quantum processor can be treated as a remote or near-edge accelerator whose output is compact, classical, and valuable enough to justify communication, queueing, measurement, and postprocessing overhead. This section examines the hardware, access models, deployment patterns, quantum-inspired alternatives, and benchmarking criteria that determine whether such integration is credible.

8.1. Hardware and Access Models

Quantum hardware is not expected to reside on ordinary edge devices. Current quantum processors require specialized control, calibration, isolation, and in many cases cryogenic or optical infrastructure. Superconducting qubits offer fast gates and mature tooling, but they require millikelvin cooling and are therefore confined to data-center or cloud environments [18,106,107]. Trapped-ion systems provide long coherence times and high-fidelity operations, but they depend on complex optical control and are also unsuitable for direct edge deployment [108]. Neutral-atom devices offer large arrays and analog-programmable dynamics, but their integration requirements remain closer to laboratory or data-center settings than to embedded systems [71,109].

Photonic and spin-based platforms are more compatible with edge-proximate deployment in the long term. Photonic systems can operate at or near room temperature and integrate naturally with optical communication and sensing infrastructure [110,111]. Spin-based and solid-state approaches, including nitrogen-vacancy centers and silicon spin qubits, may eventually support compact quantum modules, although they remain experimental for the purposes considered here [112]. Even for these platforms, the more realistic near- and mid-term location is a gateway, mobile edge computing (MEC) node, on-premise server, or specialized network facility rather than a sensor, wearable, drone, or microcontroller-class device.

The practical access path today is therefore cloud or cloud-like execution. Quantum providers expose hardware through classical APIs and software development kits, while hybrid frameworks such as Qiskit, PennyLane, Strawberry Fields, TKET, Braket, and related tools allow circuits to be embedded in classical workflows [113–115]. These frameworks are important because they make the quantum processor appear as a callable accelerator

inside a larger classical pipeline. For edge-AI workflows, the relevant distinction is not the provider name, but the access model, latency regime, and kind of workload that can tolerate it.

Three access patterns deserve to be considered separately because they have qualitatively different latency, privacy, and amortization profiles. First, *cloud quantum access* routes a hybrid job from an edge device through the public internet to a quantum service operated by a provider. The round-trip time is dominated by wide-area network latency and provider queueing; it is suitable for retraining, kernel construction, and model compression performed at hour-to-day timescales, but unsuitable for any task that must complete within a single edge inference interval. Second, *MEC-mediated access* places a classical coordinator inside the operator network, close to the edge, and routes the heavy classical preprocessing, batching, and orchestration there; the quantum service is still remote, but the MEC node can aggregate updates from many edge clients, schedule quantum calls efficiently, and cache returned artifacts. This pattern fits federated aggregation, personalization, and anomaly-threshold tuning for fleets of edge devices. Third, *near-edge quantum acceleration* would place a small photonic or spin-based quantum module at a gateway, on-premise server, or specialized network facility. This pattern would reduce round-trip latency relative to cloud access and could in principle support more frequent quantum calls in privacy-sensitive deployments, but it depends on hardware [110–112] that is still maturing. Figures 1 and 2 should be read with this distinction in mind: the “Quantum Service” block in those figures may represent any of the three patterns above.

Table 5 summarizes the main access models. Cloud access is suitable for tasks whose timescale is seconds, minutes, or hours, such as training, calibration, model compression, or kernel construction. Reserved capacity can reduce queueing effects but does not remove the need for communication and orchestration. MEC-mediated access is a natural systems direction because it places the classical coordinator closer to the edge and can amortize quantum calls across many devices. Near-edge quantum modules remain speculative, but photonic and spin-based hardware make this pathway more plausible than direct quantum deployment on ultra-constrained devices.

Table 5. Access models for quantum-assisted edge AI.

Access Model	Typical Location	Suitable Tasks	Main Limitation
Cloud quantum service	Remote data center	Training, compression, kernel construction, calibration	Queueing, wide-area latency, and limited scheduling control
Reserved cloud capacity	Remote data center	Repeated hybrid optimization and parameter search	Cost, availability, and provider dependence
MEC-mediated access	Gateway, edge server, or cellular MEC node	Federated aggregation, near-edge adaptation, workload orchestration	Experimental integration and uncertain latency guarantees
Near-edge photonic or spin module	Gateway, on-premise server, or specialized network facility	Sampling, optical feature processing, signal-processing workflows	Hardware immaturity and limited algorithmic maturity
On-device quantum hardware	Sensor, wearable, mobile device, or embedded controller	Not realistic for near-term edge AI	Power, cooling, control, and reliability overhead

8.2. Hybrid Edge–Quantum Architectures

A practical edge–quantum architecture separates real-time execution from latency-tolerant acceleration. The edge device performs sensing, preprocessing, inference, and control locally. A gateway, MEC node, or cloud backend aggregates data or model statistics, prepares quantum jobs, submits them to a quantum service, processes the measurement results, and returns a classical artifact to the edge system. This artifact may be a com-

pressed model, updated weights, optimized thresholds, feature maps, kernel parameters, calibration constants, control-policy parameters, or codebooks.

The feasibility condition is system-level rather than purely algorithmic. A hybrid workflow is useful only if

$$T_{\text{uplink}} + T_{\text{quantum}} + T_{\text{downlink}} + T_{\text{overhead}} < T_{\text{classical}}, \quad (27)$$

where T_{overhead} includes queueing, state preparation, measurement, error mitigation, post-processing, and orchestration. This inequality is rarely satisfied for real-time inference because the local device must meet strict timing requirements. It is more plausible for asynchronous workloads such as retraining, hyperparameter search, model compression, kernel evaluation, federated aggregation, anomaly-threshold tuning, simulation-based policy optimization, and periodic calibration.

Noise also enters the architecture through this inequality. If $\hat{O}$ is an observable estimated by a quantum circuit, a noisy device produces

$$\langle \hat{O} \rangle_{\text{noisy}} = \langle \hat{O} \rangle_{\text{ideal}} + \epsilon_{\text{noise}}, \quad (28)$$

where ϵ_{noise} depends on circuit depth, gate errors, decoherence, connectivity, and readout error. In hybrid learning, this noise can bias gradients, destabilize optimization, and reduce the reliability of sampling-based approximations. Error mitigation methods such as zero-noise extrapolation, probabilistic error cancellation, measurement-error mitigation, and shallow problem-informed ansätze can improve reliability [101,102,116,117]. They also increase sampling and execution cost. For this reason, error mitigation is compatible with edge integration only when the quantum routine is used sparingly, asynchronously, or in batch mode.

The resulting design principle is simple. Quantum processors should not sit in the real-time loop. They should support coarse-grained tasks whose outputs are small enough to transmit and stable enough to deploy. This division of labor is compatible with NISQ hardware and with existing edge-cloud systems, but it requires careful accounting of communication, measurement, and postprocessing costs.

8.3. Edge Workloads and Deployment Patterns

Different edge domains expose different opportunities for quantum-assisted workflows [50]. Ultra-constrained IoT sensors and embedded devices benefit mainly indirectly. They cannot host quantum circuits and usually cannot send raw high-dimensional data continuously. A backend may instead use aggregated statistics or representative samples to optimize anomaly-detection thresholds, compress a model, construct a kernel, or periodically update a lightweight classifier. The deployed model remains classical and small enough for the local memory and energy budget.

Mobile and wearable devices provide more compute and better connectivity, so they can support periodic personalization, privacy-aware adaptation, and federated learning. Quantum resources, if useful, would operate at the server or MEC layer. Their cost would be amortized across many clients, while each device receives only compact model updates. This is a natural fit for health monitoring, activity recognition, keyboard prediction, and other workloads where local data are private but periodic global adaptation is valuable.

Autonomous systems such as drones, vehicles, and robotic platforms impose the strictest real-time constraints. Perception, control, collision avoidance, and safety-critical decision making must remain entirely local. Quantum acceleration is relevant only outside the control loop, for example in mission planning, simulation-based training, route optimization, policy search, or post-mission model refinement. In these settings, the benefit

is not lower per-frame inference latency, but better models or policies produced during non-real-time phases.

Federated and split learning provide especially natural deployment patterns because they already separate local execution from server-side coordination. Federated learning trains a shared model without centralizing raw data. Each client computes a local update from private data, and a server aggregates the updates into a global model:

$$\theta^{(k+1)} = \sum_{i=1}^N w_i \theta_i^{(k+1)}, \quad \sum_i w_i = 1. \quad (29)$$

Quantum federated learning has been proposed in several forms, including quantum clients with variational models, blind delegated quantum computation, decentralized quantum-capable clients, and classical clients with a quantum-assisted server [27,28,30,48,49,118]. For the edge premise of this review, the most relevant form is the last one. Edge devices remain classical, and quantum processing is invoked at the aggregation or optimization step, where the cost can be shared across many devices and where inter-round timescales can tolerate latency.

Split learning offers a related pattern. A client runs the early layers of a model on private data and transmits an intermediate activation to a server. The server completes the remaining layers and returns gradients or predictions [119]. A quantum version can place a quantum feature map, kernel evaluator, or variational block on the server side of the cut. The edge device transmits a compact activation vector rather than raw data, while the server uses quantum resources to enrich the representation or perform the final classification. This pattern is most plausible for workloads with small feature vectors, moderate latency tolerance, and strong privacy or bandwidth constraints.

MEC testbeds are the systems bridge between these ideas and practical deployment. A MEC node can aggregate data from many local devices, decide when a quantum call is worth making, cache quantum-derived artifacts, and schedule hybrid jobs across classical and quantum resources. Such testbeds remain experimental, but they are important because they expose the real trade-offs among latency, queueing, accuracy, model size, privacy, and resource use.

8.4. Concrete Edge-AI Use Cases for Quantum-Assisted and Quantum-Inspired Methods

To make the preceding discussion concrete, we list a small set of edge-AI tasks in which quantum-assisted or quantum-inspired methods are plausibly useful. These tasks share three features: their training, optimization, or representation step is not bound to the real-time inference loop; they admit compact classical artifacts that can be deployed back to the device; and they benefit from improvements in linear algebra, sampling, kernel construction, or compression. We give the candidate method family in parentheses and refer to the relevant references already in the bibliography.

Industrial and biomedical anomaly detection. Threshold tuning, density estimation, and one-class learning over telemetry or biomedical sensor streams can be offloaded to a hybrid loop that uses variational classifiers, quantum kernels, or quantum-Born-machine-style generative models for the density component; the deployed model is a small classical threshold or classifier (quantum-assisted) [22,24,50,75–77].

Compressed sensing and feature extraction for low-bandwidth links. When raw sensor data are too large to transmit, randomized projections and shadow-style sketches inspired by quantum information can be used directly on the edge device to produce compact summaries, and tensor-network compression can shrink the model that consumes them (quantum-inspired) [31,33,53].

Route, schedule, and policy optimization for autonomous systems. Drones, robots, and vehicles often perform mission planning, routing, or scheduling outside the real-time control loop. QAOA, variational optimization, and quantum sampling have been proposed for related combinatorial and policy problems, and their outputs are compact classical policies or schedules that can be deployed locally (quantum-assisted) [20,73,88].

Federated personalization for mobile and wearable models. Health monitoring, activity recognition, and on-device keyboard or speech models benefit from periodic personalization with privacy guarantees. Quantum federated learning with a quantum-assisted server and post-quantum-secure communication is a natural fit, because the edge clients remain classical and the quantum work is amortized across many devices (quantum-assisted, with quantum-cryptographic security) [27–30,57].

TinyML training, compression, and architecture search. Hyperparameter search, architecture selection, and quantization-aware training can be reformulated as combinatorial or variational optimization problems suitable for QAOA, hybrid VQE-like loops, or quantum-sampling search; the resulting compact classical model is then deployed on a microcontroller-class device (quantum-assisted) [10,20,22,65,73].

In each case, the edge device itself is unchanged: it remains a classical sensor, mobile, or embedded system running compact inference. The contribution of quantum or quantum-inspired methods is restricted to the off-device or device-side classical parts of the workflow that prepare, compress, or update that model.

8.5. Quantum-Inspired Methods on Edge Hardware

Quantum-inspired classical methods are the most immediate way in which quantum ideas can affect real edge deployments. They borrow mathematical structures from quantum information, such as tensor networks, low-rank factorizations, amplitude-inspired sampling, and randomized measurement ideas, but implement them entirely on classical hardware [32,34]. This makes them compatible with disconnected, private, battery-powered, and microcontroller-class environments.

Tensor-network compression is the clearest example. Matrix product states, tree tensor networks, and tensor-train decompositions can represent neural weights or activations with fewer parameters when the relevant correlations are structured or low-rank. These methods originated in quantum many-body physics and have been adapted to supervised learning and neural-network compression [33,52,53]. For edge AI, their value is practical: they can reduce memory footprint and multiply-accumulate cost without requiring quantum hardware, quantum networking, or NISQ error mitigation.

Quantum-inspired sampling provides a second route. Dequantization results, beginning with Tang's work on recommendation systems, showed that some QRAM-based quantum speedups can be matched by classical algorithms given analogous sampling access [32]. For edge workloads, the lesson is constructive rather than negative. Importance-sampled approximations to SVD, regression, kernel evaluation, and low-rank updates may be implementable on ordinary edge hardware and may capture much of the practical value of the quantum formulation when the data structure is favorable.

A third route is randomized projection and shadow-style diagnostics. Ideas related to classical shadows and randomized measurement [31] can be adapted to estimate feature statistics, activation summaries, drift indicators, or model health metrics from small sketches. Such techniques are useful for edge monitoring because they reduce memory and communication while preserving enough information for periodic calibration or anomaly detection.

These methods should not be viewed as a fallback. They are a realistic path by which quantum theory can influence deployed edge systems before quantum hardware is

mature. Their limitation is also clear. Once the method is reduced to classical operations, the achievable speedup is governed by classical complexity. If the problem lacks exploitable low-rank, sparse, tensor-network, or sampling structure, quantum-inspired methods will not create an advantage.

8.6. Benchmarking and Design Principles

Because quantum-assisted edge AI is a system-level claim, it must be benchmarked with system-level metrics. Isolated circuit runtime or asymptotic quantum complexity is not enough. A meaningful benchmark should include classical preprocessing, data movement, quantum queueing, state preparation, circuit execution, measurement, error mitigation, postprocessing, downlink transmission, and the performance of the final deployed edge model. The relevant metrics include end-to-end latency, energy, memory footprint, model size, accuracy under fixed resource budgets, robustness, privacy exposure, communication cost, and adaptation cost.

Simulation remains essential because current quantum hardware is limited. Classical simulators can evaluate algorithmic scaling, sensitivity to noise, and integration overhead for small and moderate circuit sizes. In general, exact state-vector simulation scales as

$$C_{\text{sim}}(n) = O(2^n), \quad (30)$$

which limits the number of qubits that can be studied directly. Nevertheless, simulation is valuable for screening designs before hardware execution. A credible workflow should first show an advantage over strong classical baselines in simulation, then test noise sensitivity, and only then justify execution on a quantum backend.

Several design principles follow. Real-time and safety-critical inference should remain local on classical edge hardware. Quantum resources should be invoked for offline, periodic, asynchronous, or aggregated tasks. Quantum algorithms should be selected when they return compact classical outputs rather than full high-dimensional states or parameter tensors. Gateways, MEC nodes, and federated servers should be used to amortize quantum execution across many devices. Circuits should be shallow, problem-informed, and robust to noise. Benchmarks should compare against optimized classical and quantum-inspired baselines, not against naive classical implementations.

The overall conclusion is that integration, not algorithmic novelty alone, determines the value of quantum acceleration for edge AI. Direct quantum hardware on edge devices is not required and is not realistic in the near term. Cloud, MEC, and near-edge quantum resources may support edge AI indirectly by improving training, compression, adaptation, sampling, or optimization. In the near term, quantum-inspired classical methods may provide the most immediate on-device gains. The viability of any proposed system should therefore be judged by whether it improves an end-to-end edge metric after all communication, noise, energy, and orchestration costs are included.

9. Challenges, Limitations, and Research Roadmap

The preceding sections show that quantum acceleration for edge AI is best understood as a hybrid systems problem rather than as a direct replacement for classical edge inference. This framing also clarifies the main limitations. Practical progress depends on quantum hardware maturity, efficient data loading, manageable integration overhead, secure deployment, and credible benchmarking. This section summarizes these challenges and uses them to define a staged research roadmap.

9.1. Hardware Maturity

The most immediate limitation is the state of present quantum hardware. Contemporary processors operate in the noisy intermediate-scale quantum (NISQ) regime, with limited numbers of physical qubits, imperfect gates, finite coherence times, restricted connectivity, and no large-scale fault tolerance [18,19]. These constraints directly limit the depth, width, and reliability of quantum circuits that can be used inside hybrid learning workflows.

Many quantum algorithms proposed for neural-network acceleration assume logical qubit counts, circuit depths, and error rates that are not yet available. Even when a device contains many physical qubits, connectivity restrictions can increase compilation overhead because nonlocal interactions must be decomposed into additional gates. This increases circuit depth and amplifies noise. The maximum usable circuit depth is bounded by the ratio between coherence time and gate duration,

$$D_{\max} \lesssim \frac{T_{\text{coh}}}{T_{\text{gate}}}, \quad (31)$$

but in practice the usable depth is also shaped by readout error, calibration drift, crosstalk, and the sampling cost required to estimate expectations.

For edge AI, this means that near-term quantum routines must be shallow, structured, and tolerant of noisy estimates. Deep unstructured quantum neural networks are unlikely to be useful in the NISQ era. More plausible candidates are small quantum kernels, low-depth variational circuits, sampling routines, and optimization subroutines whose outputs are compact classical artifacts.

9.2. Data Loading and Readout Bottlenecks

A second major limitation is data movement between classical and quantum representations. Many quantum algorithms assume that classical data can be efficiently prepared as quantum states. For example, amplitude encoding maps a normalized vector $\mathbf{x} \in \mathbb{R}^N$ into

$$|x\rangle = \sum_{i=1}^N x_i |i\rangle. \quad (32)$$

Without structure or quantum random access memory, preparing such a state can require $O(N)$ operations [36]. This cost can remove the advantage of algorithms that become polylogarithmic only after the state has already been prepared.

Quantum random access memory is often invoked to address this issue, but scalable, fault-tolerant QRAM remains theoretical. It would require substantial additional hardware, control, and error management. For edge AI, where data are often classical, high-dimensional, streaming, and noisy, the loading problem is especially severe. Near-term workflows should therefore avoid sending raw sensor streams to a quantum backend. They should operate instead on reduced representations, aggregated statistics, compact embeddings, model parameters, or already compressed data.

Readout creates a complementary bottleneck. Quantum algorithms usually return a state or a distribution, not a full classical vector. Extracting detailed classical information requires repeated measurements, and estimating an expectation value to precision ϵ typically requires sampling overhead that scales as $O(1/\epsilon^2)$ without amplitude-estimation-type speedups [104]. This reinforces the design principle used throughout this review: quantum subroutines should produce low-dimensional summaries, rankings, parameters, kernels, thresholds, or compressed models rather than full high-dimensional outputs.

9.3. Hybrid Integration Overheads

Even if a quantum subroutine is algorithmically attractive, it must improve the end-to-end system. In an edge-quantum workflow, total cost includes local edge processing, communication, queueing, quantum execution, measurement, error mitigation, postprocessing, and deployment of the returned artifact. A simplified latency model is

$$T_{\text{total}} = T_{\text{edge}} + T_{\text{comm}} + T_{\text{quantum}} + T_{\text{overhead}}. \quad (33)$$

The quantum component is useful only when the reduction in computation or model cost exceeds the additional communication and orchestration burden.

This condition is rarely satisfied for latency-critical inference. In autonomous systems, medical monitoring, industrial control, and interactive mobile applications, the real-time loop must remain local. Hybrid quantum execution is more credible for periodic retraining, calibration, kernel construction, compression, federated aggregation, simulation-based policy search, and other asynchronous tasks. These tasks can tolerate delay and can amortize quantum execution across many devices or many future inferences.

Hybrid orchestration also introduces reliability risks. A production edge-AI system must handle network outages, cloud queue delays, quantum-service unavailability, noisy results, and versioning of returned model artifacts. These issues are not secondary. In safety-critical edge systems, added complexity can outweigh any theoretical computational gain. Robust fallback paths and classical baselines are therefore necessary components of any realistic deployment.

Latency under realistic network conditions deserves explicit attention. Edge applications typically require per-inference budgets in the millisecond range for control, perception, or interactive workloads; mobile and IoT applications routinely demand sub-second total response. Wide-area access to a cloud quantum service, with its provider queueing, scheduling, and shot aggregation overhead, currently operates at second-to-minute timescales in the best case and at hour-to-day timescales when significant compute is requested [20,114,115]. Even MEC-mediated access cannot bring round-trip times into the per-inference latency budget of a real-time controller. The honest implication is that quantum-assisted edge AI cannot satisfy hard real-time inference constraints in the foreseeable future, regardless of underlying algorithmic speedup. Its realistic operating regime is the periodic, batch, or asynchronous one: retraining cycles measured in hours or days, federated rounds measured in minutes, calibration and threshold updates measured in tens of seconds. Privacy arguments must be similarly conservative. Hybrid execution that sends features, gradients, or kernel queries through a public network can leak information about local data, and the privacy budget of the resulting workflow must be modeled explicitly. Blind quantum computation, federated aggregation, and post-quantum secure transport reduce but do not eliminate this risk [27–29,54].

9.4. Security and Privacy

Quantum technologies affect edge-AI security in two directions. On the positive side, quantum key distribution can support secure key exchange over appropriate optical links, quantum random number generators can strengthen cryptographic entropy sources, and blind quantum computation can allow a client to delegate certain computations without revealing inputs or outputs [28,54,55,120–122]. These capabilities may be useful in distributed and adversarial edge environments, although QKD is currently more realistic for backhaul or infrastructure links than for leaf-edge devices.

On the negative side, scalable quantum computers threaten the public-key cryptography used in today's edge systems. Shor's algorithm breaks RSA, Diffie–Hellman, and elliptic-curve cryptography in polynomial time [35], while Grover's algorithm gives a

quadratic speedup against symmetric primitives [78]. Firmware signing, secure boot, TLS or DTLS handshakes, model-update authentication, and over-the-air updates all depend on cryptographic mechanisms that must be made post-quantum-secure. The harvest-now, decrypt-later threat is especially relevant for long-lived IoT deployments.

The practical response is migration to post-quantum cryptography. NIST has standardized lattice-based, hash-based, and code-based schemes through its PQC process [29,56]. For edge AI, this means that model updates, firmware updates, federated-learning communication, and any quantum-assisted training loop should use post-quantum-secure authentication and key exchange. The cost of these primitives must be included in edge deployment budgets, especially for microcontroller-class devices.

9.5. Benchmarking and Evaluation

A major barrier to progress is the absence of standardized benchmarks for quantum-assisted edge AI. Classical edge AI has mature reference points such as MLPerf Tiny, which provides workloads for keyword spotting, visual wake words, anomaly detection, and image classification with latency, accuracy, and energy accounting on constrained hardware [123]. Toolchains such as TensorFlow Lite Micro and TinyML frameworks provide reproducible baselines [65,124]. No comparable benchmark suite exists for quantum-enhanced edge workflows.

Current QML evaluations vary widely in dataset choice, circuit family, simulator precision, noise model, hardware backend, and classical baseline. Some studies use small MNIST subsets, synthetic Gaussian data, or physics-inspired tasks, while others report simulator-only results or small hardware demonstrations. This makes it difficult to determine whether an observed improvement comes from the quantum component, the feature map, the optimizer, the data split, or an unfair classical comparison.

For edge AI, benchmarking must go beyond circuit runtime. A meaningful evaluation should measure end-to-end task completion time, energy, communication cost, memory footprint, model size, accuracy, robustness to sensor noise and drift, reliability under quantum-service delays, and the performance of the final model on the target edge platform. Quantum execution should be treated as a remote accelerator whose costs include uplink, queueing, state preparation, execution, measurement, error mitigation, downlink, and postprocessing. Without this accounting, claims of quantum advantage are not actionable for edge deployment.

To make this concrete, we sketch a fair-benchmarking protocol that we recommend for quantum-assisted edge-AI studies. (i) *Fix the edge target.* Specify the device class (microcontroller, mobile SoC, embedded GPU), memory and energy budget, and the deployed task and metric of interest. (ii) *Use a strong classical baseline.* Compare against an optimized quantized, distilled, or compressed classical model trained with current TinyML best practices and reported on the same hardware [10,64,65,123]. (iii) *Also report a quantum-inspired classical baseline.* If the quantum workflow uses kernels, tensor structure, or sampling, report what a classical algorithm using analogous structure or dequantized access achieves on the same data and device [32–34]. (iv) *Account for the full hybrid loop.* Report uplink and downlink latency, MEC or cloud queueing time, state preparation, shot count, error mitigation overhead, and classical postprocessing alongside the quantum runtime. (v) *Use realistic noise and access models.* Distinguish simulator-only results from hardware results, and from cloud-hardware results under realistic queue delays; explicitly state the noise model used in simulation. (vi) *Report robustness under degraded conditions.* Include results when the quantum service is briefly unavailable, when network latency spikes, and when noise rates drift; report whether a classical fallback path is invoked. (vii) *Disclose privacy and security overhead.* If the workflow relies on federated, split, or delegated computation,

report the cost of post-quantum-secure communication and any privacy leakage from the quantum-side computation. A claim of quantum advantage for edge AI is only actionable when these elements are all visible to the reader.

The same protocol clarifies the limitations of current evaluations. Many published QML results either compare a small quantum model to an unoptimized classical baseline, omit the cost of cloud access or state preparation, or report only simulator results that exclude noise and queueing. Under the protocol above, the realistic question for edge AI is not whether the quantum algorithm has a favorable asymptotic complexity, but whether the full hybrid pipeline delivers a measurable improvement in latency, energy, memory, model size, or accuracy under deployment constraints. Until this is demonstrated, quantum-assisted edge AI should be reported as a research direction with specific candidate use cases, not as a ready-to-deploy technology.

9.6. Research Roadmap

The limitations above suggest a staged roadmap. It is helpful to be explicit about which kind of progress we expect in each horizon, because the current evidence base for QML on edge tasks consists almost entirely of small-scale proof-of-concept demonstrations on few-qubit hardware or on simulators with synthetic or low-resolution datasets [22,24,27,30,43,95]. Generalizing from these demonstrations to fleets of industrial edge devices is not a routine engineering exercise; it depends on improvements in hardware fidelity, coherence, connectivity, and error correction that are themselves bounded by quantum many-body physics [19,70,71]. The roadmap below therefore reflects that constraint: near-term gains rely primarily on quantum-inspired and small hybrid methods, while quantum-native edge integration is positioned as a long-term goal whose viability ultimately depends on physical breakthroughs in qubit hardware and control.

Short-term milestones (next several years; NISQ era). Progress will come primarily from hybrid and quantum-inspired methods that respect NISQ constraints. Quantum-inspired classical algorithms are especially important because they can run directly on edge hardware. Tensor networks, low-rank factorizations, randomized projections, and sampling-based approximations can improve compression, sparsification, monitoring, and approximate inference without requiring quantum data loading or remote quantum execution [32–34]. Small-scale QNN and quantum-kernel demonstrations also remain useful, but their role should be to test integration patterns, noise sensitivity, and end-to-end workflows rather than to claim immediate superiority over optimized classical baselines [22]. Concrete short-term milestones include reproducible quantum-inspired compression of TinyML models on commodity edge hardware, fair end-to-end benchmarks for at least one canonical edge-AI task, and the demonstration that a hybrid retraining pipeline using cloud-accessible NISQ devices can match or exceed a strong classical baseline on a non-toy problem under the protocol of Section 9.5.

Figure 3 summarizes this staged trajectory, emphasizing that near-term progress is likely to come from quantum-inspired and hybrid workflows, while deeper integration depends on improved hardware, lower-latency access models, and system-level benchmarking.

Medium-term milestones (post-NISQ, error-mitigated or early fault-tolerant era). In the mid term, the key opportunity is tighter integration of quantum modules into the edge–cloud continuum. Fault-tolerant or error-mitigated quantum processors are more likely to appear first in cloud data centers, mobile edge computing nodes, or on-premise industrial servers than on edge devices themselves. In this regime, quantum routines could support more reliable linear algebra, optimization, sampling, and probabilistic inference for fleets of edge devices. Local inference would remain classical, but model training,

compression, personalization, and federated aggregation could become more tightly coupled to quantum accelerators. Concrete medium-term milestones include the first MEC testbeds that route federated rounds through an error-mitigated quantum service, standardized benchmark suites that include both classical and quantum-assisted edge baselines, and demonstrated end-to-end gains on at least one production-scale edge workload after all loading, communication, and security costs are included.

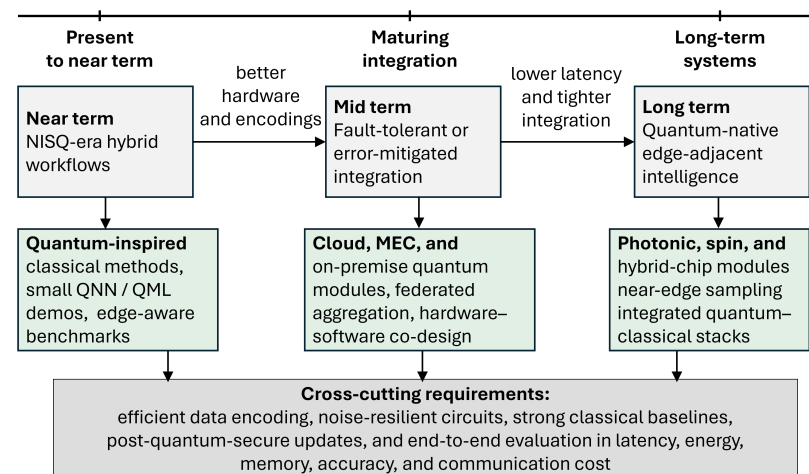


Figure 3. Staged research roadmap for quantum-accelerated edge AI. Near-term progress is expected from quantum-inspired classical methods, small hybrid QML demonstrations, and edge-aware benchmarks. Mid-term progress depends on more reliable quantum modules integrated through cloud, MEC, or on-premise infrastructure. Long-term progress may come from photonic, spin-based, and hybrid-chip modules deployed near the edge rather than inside ordinary edge devices. Across all stages, progress requires efficient data encoding, noise-resilient circuits, strong classical baselines, secure update paths, and end-to-end evaluation under edge-system constraints.

Hardware–software co-design will be essential at this stage. Encoding schemes must minimize data movement. Circuits must be shallow or error-corrected enough to produce reliable outputs. Classical preprocessing must reduce raw edge data to compact representations suitable for quantum subroutines. Edge software stacks must support versioned model artifacts, fallback paths, secure update channels, and fair comparison against classical and quantum-inspired methods.

Long-term milestones (fault-tolerant era). In the long term, photonic, spin-based, and hybrid quantum–classical chips may enable quantum acceleration closer to the network edge. Photonic platforms are especially relevant because they can operate at or near room temperature, integrate with optical communication infrastructure, and naturally support sampling and kernel-style operations [110]. This does not imply that every sensor or wearable will contain a quantum processor. A more credible long-term vision is quantum-native edge-adjacent intelligence, where specialized quantum modules at gateways, MEC nodes, or local servers support large populations of classical edge devices. Even at this horizon, the practical viability of quantum-native edge integration depends on improvements in physical qubit fidelity, error-corrected logical-qubit overhead, room- and fiber-temperature compatibility, and integration with optical and electronic infrastructure [19,70,71,111]. These are physics-level prerequisites rather than software ones.

Across all horizons, four priorities stand out. First, data encoding must become more efficient, especially for streaming and incremental edge data. Second, noise-resilient circuits and training strategies must be designed around realistic hardware rather than idealized models. Third, benchmarks must measure complete edge workflows and include strong classical baselines. Fourth, application design must be concrete. The most

useful studies will focus on specific edge tasks such as anomaly detection, distributed sensing, adaptive control, biomedical monitoring, privacy-preserving personalization, and simulation-based planning.

The roadmap is therefore cautious but constructive. Quantum acceleration is not ready to transform edge inference today, and direct quantum hardware on edge devices is not a near-term requirement. The plausible path runs through quantum-inspired classical methods, hybrid quantum–classical workflows, standardized system-level benchmarks, and eventually quantum modules integrated into cloud, MEC, or edge-adjacent infrastructure.

10. Conclusions

This review has examined quantum-accelerated AI for edge systems from an architectural, algorithmic, and systems perspective. The central conclusion is that quantum computing is not a near-term replacement for classical edge AI, nor does it require quantum processors to be embedded in edge devices. The credible model is hybrid. Edge devices remain classical and continue to perform sensing, preprocessing, real-time inference, and control locally. Quantum processors, when useful, act as remote or near-edge accelerators for selected tasks whose outputs return to the edge as compact classical artifacts, such as weights, kernels, thresholds, calibration parameters, codebooks, or compressed models. We use “quantum-accelerated” as a general label for this class of hybrid workflows, but caution that for most near-term tasks “quantum-assisted” or “quantum-inspired” more accurately describes the role of the quantum component, since the quantum subroutine sits inside a classical pipeline (or, in the inspired case, runs entirely classically).

Under this premise, the relevant question is not whether an edge device can execute a quantum circuit. It is whether a hybrid quantum–classical workflow can improve an end-to-end edge-AI metric relative to a strong classical baseline. That metric may be latency, energy, memory footprint, model size, sample efficiency, adaptation cost, accuracy under resource constraints, or robustness. A quantum subroutine is useful only if its benefit survives the full cost of data preparation, communication, queueing, circuit execution, measurement, error mitigation, postprocessing, and secure deployment.

The current evidence supports a cautious but constructive view. Quantum data encodings define much of the practical cost. Amplitude encoding provides the most attractive asymptotic story, but without efficient QRAM or exploitable structure, state preparation can erase the downstream advantage. Angle, phase, IQP, and data re-uploading schemes are more realistic near-term choices because they are shallow, expressive, and compatible with NISQ hardware. Quantum neural architectures, including quantum perceptrons, variational quantum circuits, quantum convolutional models, recurrent models, and quantum attention constructions, are best understood as modular components rather than full replacements for classical networks. Their most plausible role is in feature construction, kernel evaluation, sampling, optimization, compression, and periodic adaptation.

Theory further narrows the field of credible proposals. Structured variational circuits can define useful hypothesis classes, and quantum kernels provide some of the clearest cases of provable separation. At the same time, barren plateaus, noise, readout cost, and classical learnability results limit the range of classical sensor-data tasks for which quantum models should be expected to outperform well-designed classical ones. For edge AI, the strongest case for quantum acceleration is therefore not universal predictive superiority. It is system-level improvement in workflows where training, compression, aggregation, or optimization can be offloaded and amortized.

The systems analysis leads to the same conclusion. Cloud quantum services, mobile edge computing nodes, and future near-edge modules may support latency-tolerant

workloads such as retraining, calibration, kernel construction, federated aggregation, and simulation-based policy search. They are not suitable for hard real-time inference loops. Quantum federated and split learning are promising precisely because they preserve classical edge execution while allowing quantum resources to operate at synchronization points or server-side model partitions. In the near term, quantum-inspired classical methods may be the most immediately deployable outcome of this field. Tensor-network compression, low-rank approximations, randomized projections, and sampling-inspired algorithms can run directly on edge hardware and provide a practical route from quantum theory to constrained-device deployment.

Several barriers remain decisive. NISQ hardware limits circuit size, depth, and reliability. Data loading and readout often dominate the cost of algorithms with favorable asymptotic complexity. Hybrid integration introduces communication, orchestration, scheduling, and failure-mode overheads. Security must be handled in both directions: quantum technologies may strengthen some protocols, while future quantum adversaries require migration of edge-AI infrastructure toward post-quantum cryptography. Finally, the field lacks standardized benchmarks that evaluate complete edge workflows rather than isolated quantum circuits.

Progress will depend on research that is both algorithmically rigorous and systems-aware. The most important priorities are efficient encodings for classical and streaming edge data, shallow and noise-resilient ansätze, quantum-inspired methods that deliver immediate on-device gains, reproducible benchmarks with strong classical baselines, and hardware–software co-design for cloud, MEC, photonic, spin-based, and hybrid quantum–classical platforms. Claims of quantum advantage should be evaluated in wall-clock time, joules per task, memory footprint, communication cost, and deployed-model performance.

The long-term promise of quantum-accelerated edge AI is real, but it is conditional. It will not be realized by placing quantum processors inside every edge device or by replacing classical inference pipelines wholesale. It will be realized, if at all, through carefully engineered hybrid systems in which quantum resources improve specific bottlenecks and classical edge devices retain the real-time responsibilities for which they are best suited. This systems-level view provides the most credible path from the theory of quantum machine learning to practical edge intelligence.

Funding: This research received no external funding.

Data Availability Statement: No numerical data were generated or utilized for this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [[CrossRef](#)]
2. Satyanarayanan, M. The emergence of edge computing. *Computer* **2017**, *50*, 30–39. [[CrossRef](#)]
3. Wang, X.; Tang, Z.; Guo, J.; Meng, T.; Wang, C.; Wang, T.; Jia, W. Empowering Edge Intelligence: A Comprehensive Survey on On-Device AI Models. *arXiv* **2025**, arXiv:2503.06027.
4. Wang, X.; Li, Q.; Jia, W. Cognitive Edge Computing: A Comprehensive Survey on Optimizing Large Models and AI Agents for Pervasive Deployment. *arXiv* **2025**, arXiv:2501.03265.
5. Cordova-Cardenas, R.; Amor, D.; Gutiérrez, Á. Edge AI in Practice: A Survey and Deployment Framework for Neural Networks on Embedded Systems. *Electronics* **2025**, *14*, 4877. [[CrossRef](#)]
6. Lane, N.D.; Bhattacharya, S.; Georgiev, P.; Forlivesi, C.; Kawsar, F. An early resource characterization of deep learning on wearables, smartphones and Internet-of-Things devices. In *Proceedings of the International Workshop on Internet of Things towards Applications (IoT-App '15)*, Seoul, Republic of Korea, 1 November 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 7–12.
7. Chen, Y.-H.; Krishna, T.; Emer, J.S.; Sze, V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J. Solid-State Circuits* **2017**, *52*, 127–138. [[CrossRef](#)]

8. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In Proceedings of the 4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016.
9. Li, E.; Zeng, L.; Zhou, Z.; Chen, X. Edge AI: On-demand accelerating deep neural network inference via edge computing. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 447–457. [\[CrossRef\]](#)
10. Somvanshi, S.; Islam, M.M.; Chhetri, G.; Chakraborty, R.; Mimi, M.S.; Shuvo, S.A.; Islam, K.S.; Javed, S.A.; Rafat, S.A.; Dutta, A.; Das, S. From Tiny Machine Learning to Tiny Deep Learning: A Survey. *arXiv* **2025**, arXiv:2506.18927.
11. Dutta, L.; Bharali, S. TinyML Meets IoT: A Comprehensive Survey. *Internet Things* **2021**, *16*, 100461. [\[CrossRef\]](#)
12. Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **2009**, *103*, 150502. [\[CrossRef\]](#) [\[PubMed\]](#)
13. Gilyén, A.; Su, Y.; Low, G.H.; Wiebe, N. Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, Phoenix, AZ, USA, 23–26 June 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 193–204.
14. Buhrman, H.; Cleve, R.; Watrous, J.; de Wolf, R. Quantum fingerprinting. *Phys. Rev. Lett.* **2001**, *87*, 167902. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Aaronson, S.; Arkhipov, A. The computational complexity of linear optics. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, San Jose, CA, USA, 6–8 June 2011; Association for Computing Machinery: New York, NY, USA, 2011; pp. 333–342.
16. Schuld, M.; Sinayskiy, I.; Petruccione, F. An introduction to quantum machine learning. *Contemp. Phys.* **2015**, *56*, 172–185.
17. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum machine learning. *Nature* **2017**, *549*, 195–202. [\[CrossRef\]](#) [\[PubMed\]](#)
18. Preskill, J. Quantum computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [\[CrossRef\]](#)
19. Bharti, K.; Cervera-Lierta, A.; Kyaw, T.H.; Haug, T.; Alperin-Lea, S.; Anand, A.; Degroote, M.; Heimonen, H.; Kottmann, J.S.; Menke, T.; et al. Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.* **2022**, *94*, 015004. [\[CrossRef\]](#)
20. Cerezo, M.; Arrasmith, A.; Babbush, R.; Benjamin, S.C.; Endo, S.; Fujii, K.; McClean, J.R.; Mitarai, K.; Yuan, X.; Cincio, L.; et al. Variational quantum algorithms. *Nat. Rev. Phys.* **2021**, *3*, 625–644. [\[CrossRef\]](#)
21. Farhi, E.; Neven, H. Classification with quantum neural networks on near term processors. *arXiv* **2018**, arXiv:1802.06002.
22. Schuld, M.; Bocharov, A.; Svore, K.M.; Wiebe, N. Circuit-centric quantum classifiers. *Phys. Rev. A* **2020**, *101*, 032308. [\[CrossRef\]](#)
23. Benedetti, M.; Lloyd, E.; Sack, S.; Fiorentini, M. Parameterized quantum circuits as machine learning models. *Quantum Sci. Technol.* **2019**, *4*, 043001. [\[CrossRef\]](#)
24. Havlíček, V.; Córcoles, A.D.; Temme, K.; Harrow, A.W.; Kandala, A.; Chow, J.M.; Gambetta, J.M. Supervised learning with quantum-enhanced feature spaces. *Nature* **2019**, *567*, 209–212. [\[CrossRef\]](#) [\[PubMed\]](#)
25. Schuld, M.; Killoran, N. Quantum machine learning in feature Hilbert spaces. *Phys. Rev. Lett.* **2019**, *122*, 040504. [\[CrossRef\]](#) [\[PubMed\]](#)
26. Liu, Y.; Arunachalam, S.; Temme, K. A rigorous and robust quantum speed-up in supervised machine learning. *Nat. Phys.* **2021**, *17*, 1013–1017. [\[CrossRef\]](#)
27. Chen, S.Y.-C.; Yoo, S. Federated quantum machine learning. *Entropy* **2021**, *23*, 460. [\[CrossRef\]](#) [\[PubMed\]](#)
28. Li, W.; Lu, S.; Deng, D.-L. Quantum federated learning through blind quantum computing. *Sci. China Phys. Mech. Astron.* **2021**, *64*, 100312. [\[CrossRef\]](#)
29. Bernstein, D.J.; Lange, T. Post-quantum cryptography. *Nature* **2017**, *549*, 188–194. [\[CrossRef\]](#) [\[PubMed\]](#)
30. Innan, N.; Khan, M.A.-Z.; Marchisio, A.; Shafique, M.; Bennai, M. FedQNN: Federated learning using quantum neural networks. In *Proceedings of the 2024 International Joint Conference on Neural Networks (IJCNN)*, Yokohama, Japan, 30 June–5 July 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 1–9.
31. Huang, H.-Y.; Kueng, R.; Preskill, J. Predicting many properties of a quantum system from very few measurements. *Nat. Phys.* **2020**, *16*, 1050–1057. [\[CrossRef\]](#)
32. Tang, E. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, Phoenix, AZ, USA, 23–26 June 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 217–228.
33. Stoudenmire, E.; Schwab, D.J. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems (NeurIPS)*; NIPS: Red Hook, NY, USA, 2016; Volume 29.
34. Arrazola, J.M.; Delgado, A.; Bardhan, B.R.; Lloyd, S. Quantum-inspired algorithms in practice. *Quantum* **2020**, *4*, 307. [\[CrossRef\]](#)
35. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [\[CrossRef\]](#)
36. Aaronson, S. Read the fine print. *Nat. Phys.* **2015**, *11*, 291–293. [\[CrossRef\]](#)

37. McClean, J.R.; Boixo, S.; Smelyanskiy, V.N.; Babbush, R.; Neven, H. Barren plateaus in quantum neural network training landscapes. *Nat. Commun.* **2018**, *9*, 4812. [\[CrossRef\]](#) [\[PubMed\]](#)
38. Holmes, Z.; Sharma, K.; Cerezo, M.; Coles, P.J. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum* **2022**, *3*, 010313. [\[CrossRef\]](#)
39. Schuld, M.; Killoran, N. Is quantum advantage the right goal for quantum machine learning? *PRX Quantum* **2022**, *3*, 030101. [\[CrossRef\]](#)
40. Zeguendry, A.; Jarir, Z.; Quafafou, M. Quantum Machine Learning: A Review and Case Studies. *Entropy* **2023**, *25*, 287. [\[CrossRef\]](#) [\[PubMed\]](#)
41. Taherkhani, E.; Moaiyeri, M.H.; Angizi, S. A nano-design of a quantum-dot-based arithmetic and logic unit for enhancing the efficiency of future IoT applications. *Nano Commun. Netw.* **2024**, *39*, 100482.
42. Schuld, M.; Bergholm, V.; Gogolin, C.; Izaac, J.; Killoran, N. Evaluating analytic gradients on quantum hardware. *Phys. Rev. A* **2019**, *99*, 032331. [\[CrossRef\]](#)
43. Cong, I.; Choi, S.; Lukin, M.D. Quantum convolutional neural networks. *Nat. Phys.* **2019**, *15*, 1273–1278. [\[CrossRef\]](#)
44. Henderson, M.; Shakyia, S.; Pradhan, S.; Cook, T. Quantum convolutional neural networks: Powering image recognition with quantum circuits. *Quantum Mach. Intell.* **2020**, *2*, 2. [\[CrossRef\]](#)
45. Bausch, J. Recurrent quantum neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*; NIPS: Red Hook, NY, USA, 2020; Volume 33, pp. 1368–1379.
46. Chen, S.Y.-C.; Yoo, S.; Fang, Y.-L.L. Quantum long short-term memory. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Singapore, 22–27 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 8622–8626.
47. Cherrat, E.A.; Kerenidis, I.; Mathur, N.; Landman, J.; Strahm, M.; Li, Y.Y. Quantum vision transformers. *Quantum* **2024**, *8*, 1265. [\[CrossRef\]](#)
48. Song, Y.; Wu, Y.; Wu, S.; Li, D.; Wen, Q.; Qin, S.; Gao, F. A quantum federated learning framework for classical clients. *Sci. China Phys. Mech. Astron.* **2024**, *67*, 250311. [\[CrossRef\]](#)
49. Huang, R.; Tan, X.; Xu, Q. Quantum federated learning with decentralized data. *IEEE J. Sel. Top. Quantum Electron.* **2022**, *28*, 6500110. [\[CrossRef\]](#)
50. Langote, M.S.; Borhade, D.; Gopale, P.; Tanpure, R. Quantum Machine Learning for IoT Data Analysis: Challenges and Opportunities. In *Integration of Artificial Intelligence in IoT: Opportunities and Challenges*; Lin, F., Kesswani, N., Patel, A., Bordoloi, S., Koley, C., Eds.; Lecture Notes in Networks and Systems; Springer: Singapore, 2025; Volume 1361, pp. 53–65. [\[CrossRef\]](#)
51. Wang, Y.; Liu, J. A comprehensive review of quantum machine learning: From NISQ to fault tolerance. *Rep. Prog. Phys.* **2024**, *87*, 116402. [\[CrossRef\]](#)
52. Huggins, W.; Patil, P.; Mitchell, B.; Whaley, K.B.; Stoudenmire, E.M. Towards quantum machine learning with tensor networks. *Quantum Sci. Technol.* **2019**, *4*, 024001. [\[CrossRef\]](#)
53. Novikov, A.; Podoprikhin, D.; Osokin, A.; Vetrov, D.P. Tensorizing Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*; NIPS: Red Hook, NY, USA, 2015; Volume 28, pp. 442–450.
54. Pirandola, S.; Andersen, U.L.; Banchi, L.; Berta, M.; Bunandar, D.; Colbeck, R.; Englund, D.; Gehring, T.; Lupo, C.; Ottaviani, C.; et al. Advances in quantum cryptography. *Adv. Opt. Photonics* **2020**, *12*, 1012–1236. [\[CrossRef\]](#)
55. Herrero-Collantes, M.; Garcia-Escartin, J.C. Quantum random number generators. *Rev. Mod. Phys.* **2017**, *89*, 015004. [\[CrossRef\]](#)
56. Chen, L.; Jordan, S.; Liu, Y.-K.; Moody, D.; Peralta, R.; Perlner, R.; Smith-Tone, D. *Report on Post-Quantum Cryptography*; NIST Internal Report 8105; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2016.
57. Zhang, S.; Tong, X.; Chi, K.; Gao, W.; Chen, X.; Shi, Z. Stackelberg Game-Based Multi-Agent Algorithm for Resource Allocation and Task Offloading in MEC-Enabled C-ITS. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 17940–17951.
58. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
59. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Long Beach, CA, USA, 9–15 June 2019; PMLR: New York, NY, USA, 2019; pp. 6105–6114.
60. Li, Y.; Chen, Y.; Dai, X.; Chen, D.; Liu, M.; Yuan, L.; Liu, Z.; Zhang, L.; Vasconcelos, N. MicroNet: Improving Image Recognition with Extremely Low FLOPs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, Canada, 11–17 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 468–477.
61. Mehta, S.; Rastegari, M. MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, Virtual, 25–29 April 2022.
62. Lai, L.; Suda, N.; Chandra, V. CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs. *arXiv* **2018**, arXiv:1801.06601.

63. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.
64. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 2704–2713.
65. David, R.; Duke, J.; Jain, A.; Janapa Reddi, V.; Jeffries, N.; Li, J.; Kreeger, N.; Nappier, I.; Natraj, M.; Wang, T.; et al. TensorFlow Lite Micro: Embedded machine learning on TinyML systems. In *Proceedings of Machine Learning and Systems (MLSys)*; PMLR: New York, NY, USA, 2021; Volume 3, pp. 800–811.
66. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [[CrossRef](#)] [[PubMed](#)]
67. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*; Cambridge University Press: Cambridge, UK, 2010.
68. Aaronson, S. *Quantum Computing Since Democritus*; Cambridge University Press: Cambridge, UK, 2013.
69. Horodecki, R.; Horodecki, P.; Horodecki, M.; Horodecki, K. Quantum entanglement. *Rev. Mod. Phys.* **2009**, *81*, 865–942. [[CrossRef](#)]
70. Google Quantum AI. Suppressing quantum errors by scaling a surface code logical qubit. *Nature* **2023**, *614*, 676–681. [[CrossRef](#)]
71. Bluvstein, D.; Evered, S.J.; Geim, A.A.; Li, S.H.; Zhou, H.; Manovitz, T.; Ebadi, S.; Cain, M.; Kalinowski, M.; Hangleiter, D.; et al. Logical quantum processor based on reconfigurable atom arrays. *Nature* **2024**, *626*, 58–65. [[PubMed](#)]
72. Peruzzo, A.; McClean, J.; Shadbolt, P.; Yung, M.-H.; Zhou, X.-Q.; Love, P.J.; Aspuru-Guzik, A.; O’Brien, J.L. A variational eigenvalue solver on a photonic quantum processor. *Nat. Commun.* **2014**, *5*, 4213. [[CrossRef](#)] [[PubMed](#)]
73. Farhi, E.; Goldstone, J.; Gutmann, S. A Quantum Approximate Optimization Algorithm. *arXiv* **2014**, arXiv:1411.4028.
74. Schuld, M.; Petruccione, F. *Supervised Learning with Quantum Computers*; Springer: Cham, Switzerland, 2018.
75. Lloyd, S.; Weedbrook, C. Quantum generative adversarial learning. *Phys. Rev. Lett.* **2018**, *121*, 040502. [[CrossRef](#)] [[PubMed](#)]
76. Benedetti, M.; Garcia-Pintos, D.; Perdomo, O.; Leyton-Ortega, V.; Nam, Y.; Perdomo-Ortiz, A. A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Inf.* **2019**, *5*, 45. [[CrossRef](#)]
77. Lou, X.; Wang, L. QuanFlex-GAN: A flexible algorithm for image generation via quantum generative modeling. *Quantum Mach. Intell.* **2025**, *7*, 87.
78. Grover, L.K. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC), Philadelphia, PA, USA, 22–24 May 1996*; Association for Computing Machinery: New York, NY, USA, 1996; pp. 212–219.
79. Huang, H.-Y.; Broughton, M.; Mohseni, M.; Babbush, R.; Boixo, S.; Neven, H.; McClean, J.R. Power of data in quantum machine learning. *Nat. Commun.* **2021**, *12*, 2631. [[CrossRef](#)] [[PubMed](#)]
80. Schuld, M.; Sweke, R.; Meyer, J.J. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys. Rev. A* **2021**, *103*, 032430. [[CrossRef](#)]
81. Weigold, M.; Barzen, J.; Leymann, F.; Salm, M. Encoding patterns for quantum algorithms. *IET Quantum Commun.* **2021**, *2*, 141–152. [[CrossRef](#)]
82. LaRose, R.; Coyle, B. Robust data encodings for quantum classifiers. *Phys. Rev. A* **2020**, *102*, 032420. [[CrossRef](#)]
83. Pérez-Salinas, A.; Cervera-Lierta, A.; Gil-Fuster, E.; Latorre, J.I. Data re-uploading for a universal quantum classifier. *Quantum* **2020**, *4*, 226. [[CrossRef](#)]
84. Giovannetti, V.; Lloyd, S.; Maccone, L. Quantum random access memory. *Phys. Rev. Lett.* **2008**, *100*, 160501. [[CrossRef](#)] [[PubMed](#)]
85. Killoran, N.; Bromley, T.R.; Arrazola, J.M.; Schuld, M.; Quesada, N.; Lloyd, S. Continuous-variable quantum neural networks. *Phys. Rev. Res.* **2019**, *1*, 033063. [[CrossRef](#)]
86. Low, G.H.; Chuang, I.L. Optimal Hamiltonian simulation by quantum signal processing. *Phys. Rev. Lett.* **2017**, *118*, 010501. [[CrossRef](#)] [[PubMed](#)]
87. Kerenidis, I.; Prakash, A. Quantum recommendation systems. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS), Berkeley, CA, USA, 9–11 January 2017*.
88. Rebertost, P.; Schuld, M.; Wossnig, L.; Petruccione, F.; Lloyd, S. Quantum gradient descent and Newton’s method for constrained polynomial optimization. *New J. Phys.* **2019**, *21*, 073023. [[CrossRef](#)]
89. Amin, M.H.; Andriyash, E.; Rolfe, J.; Kulchytsky, B.; Melko, R. Quantum Boltzmann machine. *Phys. Rev. X* **2018**, *8*, 021050. [[CrossRef](#)]
90. Mangini, S.; Tacchino, F.; Gerace, D.; Bajoni, D.; Macchiavello, C. Quantum computing models for artificial neural networks. *EPL (Europhys. Lett.)* **2021**, *134*, 10002. [[CrossRef](#)]
91. Massoli, F.V.; Vadicamo, L.; Amato, G.; Falchi, F. A leap among quantum computing and quantum neural networks: A survey. *ACM Comput. Surv.* **2022**, *55*, 98. [[CrossRef](#)]
92. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386–408. [[CrossRef](#)] [[PubMed](#)]

93. Kak, S.C. On quantum neural computing. *Inf. Sci.* **1995**, *83*, 143–160. [\[CrossRef\]](#)
94. Schuld, M.; Sinayskiy, I.; Petruccione, F. The quest for a quantum neural network. *Quantum Inf. Process.* **2014**, *13*, 2567–2586. [\[CrossRef\]](#)
95. Tacchino, F.; Macchiavello, C.; Gerace, D.; Bajoni, D. An artificial neuron implemented on an actual quantum processor. *npj Quantum Inf.* **2019**, *5*, 26. [\[CrossRef\]](#)
96. Torrontegui, E.; García-Ripoll, J.J. Unitary quantum perceptron as efficient universal approximator. *EPL (Europhys. Lett.)* **2019**, *125*, 30004. [\[CrossRef\]](#)
97. Mitarai, K.; Negoro, M.; Kitagawa, M.; Fujii, K. Quantum circuit learning. *Phys. Rev. A* **2018**, *98*, 032309. [\[CrossRef\]](#)
98. Di Sipio, R.; Huang, J.-H.; Chen, S.Y.-C.; Mangini, S.; Worring, M. The dawn of quantum natural language processing. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 22–27 May 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 8612–8616.
99. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*; NIPS: Red Hook, NY, USA, 2017; Volume 30.
100. Cerezo, M.; Sone, A.; Volkoff, T.; Cincio, L.; Coles, P.J. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nat. Commun.* **2021**, *12*, 1791. [\[CrossRef\]](#) [\[PubMed\]](#)
101. Temme, K.; Bravyi, S.; Gambetta, J.M. Error mitigation for short-depth quantum circuits. *Phys. Rev. Lett.* **2017**, *119*, 180509. [\[CrossRef\]](#) [\[PubMed\]](#)
102. Endo, S.; Cai, Z.; Benjamin, S.C.; Yuan, X. Hybrid quantum-classical algorithms and quantum error mitigation. *J. Phys. Soc. Jpn.* **2021**, *90*, 032001. [\[CrossRef\]](#)
103. Abbas, A.; Sutter, D.; Zoufal, C.; Lucchi, A.; Figalli, A.; Woerner, S. The power of quantum neural networks. *Nat. Comput. Sci.* **2021**, *1*, 403–409. [\[CrossRef\]](#) [\[PubMed\]](#)
104. Brassard, G.; Høyer, P.; Mosca, M.; Tapp, A. Quantum amplitude amplification and estimation. *Contemp. Math.* **2002**, *305*, 53–74. [\[CrossRef\]](#)
105. Caro, M.C.; Huang, H.-Y.; Cerezo, M.; Sharma, K.; Sornborger, A.; Cincio, L.; Coles, P.J. Generalization in quantum machine learning from few training data. *Nat. Commun.* **2022**, *13*, 4919. [\[CrossRef\]](#) [\[PubMed\]](#)
106. Arute, F.; Arya, K.; Babbush, R.; Bacon, D.; Bardin, J.C.; Barends, R.; Biswas, R.; Boixo, S.; Brandao, F.G.S.L.; Buell, D.A.; et al. Quantum supremacy using a programmable superconducting processor. *Nature* **2019**, *574*, 505–510. [\[CrossRef\]](#) [\[PubMed\]](#)
107. Krantz, P.; Kjaergaard, M.; Yan, F.; Orlando, T.P.; Gustavsson, S.; Oliver, W.D. A quantum engineer’s guide to superconducting qubits. *Appl. Phys. Rev.* **2019**, *6*, 021318. [\[CrossRef\]](#)
108. Bruzewicz, C.D.; Chiaverini, J.; McConnell, R.; Sage, J.M. Trapped-ion quantum computing: Progress and challenges. *Appl. Phys. Rev.* **2019**, *6*, 021314. [\[CrossRef\]](#)
109. Henriët, L.; Beguin, L.; Signoles, A.; Lahaye, T.; Browaeys, A.; Raymond, G.-O.; Jurczak, C. Quantum computing with neutral atoms. *Quantum* **2020**, *4*, 327. [\[CrossRef\]](#)
110. O’Brien, J.L.; Furusawa, A.; Vučković, J. Photonic quantum technologies. *Nat. Photonics* **2009**, *3*, 687–695. [\[CrossRef\]](#)
111. Madsen, L.S.; Laudenbach, F.; Askarani, M.F.; Rortais, F.; Vincent, T.; Bulmer, J.F.F.; Miatto, F.M.; Neuhaus, L.; Helt, L.G.; Collins, M.J.; et al. Quantum computational advantage with a programmable photonic processor. *Nature* **2022**, *606*, 75–81. [\[CrossRef\]](#)
112. Awschalom, D.D.; Hanson, R.; Wrachtrup, J.; Zhou, B.B. Quantum technologies with optically interfaced solid-state spins. *Nat. Photonics* **2018**, *12*, 516–527. [\[CrossRef\]](#)
113. LaRose, R. Overview and comparison of gate level quantum software platforms. *Quantum* **2019**, *3*, 130. [\[CrossRef\]](#)
114. Bergholm, V.; Izaac, J.; Schuld, M.; Gogolin, C.; Ahmed, S.; Ajith, V.; Alam, M.S.; Alonso-Linaje, G.; AkashNarayanan, B.; Asadi, A.; et al. PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv* **2018**, arXiv:1811.04968.
115. Javadi-Abhari, A.; Treinish, M.; Krsulich, K.; Wood, C.J.; Lishman, J.; Gacon, J.; Martiel, S.; Nation, P.D.; Bishop, L.S.; Cross, A.W.; et al. Quantum computing with Qiskit. *arXiv* **2024**, arXiv:2405.08810.
116. Li, Y.; Benjamin, S.C. Efficient variational quantum simulator incorporating active error minimization. *Phys. Rev. X* **2017**, *7*, 021050. [\[CrossRef\]](#)
117. Kandala, A.; Temme, K.; Córcoles, A.D.; Mezzacapo, A.; Chow, J.M.; Gambetta, J.M. Error mitigation extends the computational reach of a noisy quantum processor. *Nature* **2019**, *567*, 491–495. [\[CrossRef\]](#) [\[PubMed\]](#)
118. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, 20–22 April 2017*; PMLR: New York, NY, USA, 2017; pp. 1273–1282.
119. Vepakomma, P.; Gupta, O.; Swedish, T.; Raskar, R. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv* **2018**, arXiv:1812.00564.
120. Giovannetti, V.; Lloyd, S.; Maccone, L. Advances in quantum metrology. *Nat. Photonics* **2011**, *5*, 222–229. [\[CrossRef\]](#)
121. Giovannetti, V.; Lloyd, S.; Maccone, L. Quantum-enhanced measurements: Beating the standard quantum limit. *Science* **2004**, *306*, 1330–1336. [\[CrossRef\]](#) [\[PubMed\]](#)

122. Bennett, C.H.; Brassard, G. Quantum cryptography: Public key distribution and coin tossing. *Theor. Comput. Sci.* **2014**, *560*, 7–11. [[CrossRef](#)]
123. Banbury, C.; Reddi, V.J.; Torelli, P.; Holleman, J.; Jeffries, N.; Kiraly, C.; Montino, P.; Kanter, D.; Ahmed, S.; Pau, D.; et al. MLPerf Tiny Benchmark. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, Virtual, 6–14 December 2021.
124. Warden, P.; Situnayake, D. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*; O'Reilly Media: Sebastopol, CA, USA, 2019.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.