Quantum Machine Learning for Classification Tasks Using Variational Quantum Circuits and Quantum Neural Networks

G.Jyothi[1], CH.Thanusha[2],CH.Madhu[3],B.Satyanarayana[4],P.Akash[5]
Department of Computer Science & Engineering (AI & ML)
Avanthi Institute of Engineering & Technology, Vizianagaram, India
Jyothi1992@gmail.com[1],thanushachinnari2003@email.com[2],madhuchenchali1@gmail.com[3],
jagasatyanarayanabotsa@gmail.com[4], pailaakash21@gmail.com[5]

Abstract

This paper presents the design, implementation, and evaluation of a Quantum Machine Learning (QML) framework for binary classification tasks using a Variational Quantum Classifier (VQC). The proposed system classifies mushrooms as edible or poisonous by leveraging 22 categorical attributes drawn from the UCI Mushroom dataset. Unlike conventional classification approaches that rely exclusively on classical computational paradigms, this work integrates quantum computational techniques—specifically superposition, entanglement, and variational optimization—to explore enhanced feature representation and hybrid learning strategies. Classical preprocessing transforms categorical attributes into numerical representations, after which Quantum Random Access Code (QRAC)-based compression reduces feature dimensionality to conform with limited qubit resources. Compressed features are subsequently encoded into quantum states via a parameterized ZZFeatureMap, and binary classification is performed through a RealAmplitudes ansatz optimized with a classical optimizer interfacing a Qiskit quantum simulator. The trained model is integrated into a Flask-based web application enabling real-time user prediction, while Docker containerization ensures reproducibility across computing environments. Experimental results demonstrate the feasibility of combining quantum machine learning with classical deployment infrastructure. The system achieves high classification accuracy, validating the practicality of hybrid quantum-classical architectures for structured categorical datasets.

Index Terms—Quantum Machine Learning, Variational Quantum Classifier, Quantum Neural Networks, QRAC Encoding, Hybrid Quantum-Classical Systems, Binary Classification.

I. Introduction

The rapid proliferation of data-intensive applications has intensified demand for classification systems capable of handling high-dimensional, complex feature spaces efficiently. Classical machine learning algorithms—including Support Vector Machines (SVM), Decision Trees, and Artificial Neural Networks—have demonstrated strong predictive performance on structured datasets; however, they face inherent scalability constraints as feature dimensionality grows [1]. The advent of Quantum Computing offers a fundamentally different computational paradigm, exploiting quantum mechanical phenomena such as superposition and entanglement to process information in ways that are classically intractable for specific problem classes [2].

Quantum Machine Learning (QML) has emerged at the intersection of these two domains, proposing algorithms that harness quantum hardware to enhance classical learning tasks [3]. Among the most promising near-term quantum algorithms are Variational Quantum Algorithms (VQAs), which combine parameterized quantum circuits with classical optimization routines, making them well-suited for Noisy Intermediate-Scale Quantum (NISQ) devices currently available [4]. The Variational Quantum Classifier (VQC), a supervised learning instantiation of the VQA framework, can approximate

non-linear decision boundaries in quantum Hilbert space and has been shown to perform comparably to shallow classical neural networks on certain datasets [5].

A significant challenge in applying QML to real-world datasets is the limited qubit count available on current hardware. High-dimensional classical data cannot be directly encoded without substantial compression. Quantum Random Access Coding (QRAC) provides a mechanism to encode multiple classical bits into fewer qubits, thereby bridging the gap between classical data dimensionality and quantum hardware constraints [6].

Despite growing theoretical advances in QML, most existing implementations remain confined to simulation environments and academic demonstrations. There is a notable absence of complete, deployable systems that integrate quantum classification with practical web-based interfaces. This paper addresses that gap by presenting an end-to-end hybrid quantum-classical pipeline that performs real-time mushroom edibility classification through a Flask web application, containerized via Docker for reproducibility.

The primary contributions of this work are: (i) implementation of a VQC using ZZFeatureMap and RealAmplitudes ansatz within the Qiskit framework; (ii) application of QRAC-based dimensionality reduction to fit the 22-feature Mushroom dataset within limited qubit resources; (iii) integration of the trained quantum model into a real-time web application; and (iv) demonstration of reproducible deployment via Docker containerization.

II. Related Work

The theoretical foundation of quantum computation was established by Feynman, who proposed that quantum systems could efficiently simulate physical phenomena beyond classical reach [7]. Shor subsequently demonstrated exponential speedup for integer factorization, confirming that quantum algorithms could surpass classical methods for specific problems [8]. These foundational works catalyzed interest in applying quantum principles to computational learning.

A. Quantum Machine Learning Foundations

Lloyd et al. proposed quantum algorithms for linear systems and supervised learning that provided theoretical speed advantages over their classical counterparts [9]. Schuld and Petruccione extensively explored quantum feature spaces and kernel-based quantum classifiers, demonstrating that quantum states naturally reside in exponentially large Hilbert spaces, enabling implicit high-dimensional feature mapping that may yield more expressive decision boundaries than classical models [1].

Biamonte et al. provided a comprehensive survey of quantum machine learning, identifying key algorithmic primitives including quantum principal component analysis, quantum support vector machines, and quantum neural networks [3]. Their analysis highlighted both the theoretical promise and the practical limitations of near-term QML implementations.

B. Variational Quantum Algorithms

Farhi et al. introduced the Quantum Approximate Optimization Algorithm (QAOA), which established the template for hybrid quantum-classical optimization [10]. Peruzzo et al. demonstrated the Variational Quantum Eigensolver, showing that variational circuits could extract ground-state energies using shallow circuits suited to NISQ hardware [11]. Havlíček et al. applied this framework specifically to supervised classification, proposing quantum feature maps that enhance separability in kernel-based classifiers [5].

C. Quantum Data Encoding

Efficient data encoding remains one of the most critical open challenges in QML. Wootters and Zurek analyzed the no-cloning theorem and its implications for quantum information encoding [12]. Several encoding strategies have been studied in subsequent literature, including basis encoding, amplitude encoding, and angle encoding. QRAC-based methods, introduced by Nayak, allow multiple classical bits to be represented using fewer qubits with bounded error probability, making them particularly relevant for NISQ devices operating under qubit constraints [6].

D. Hybrid Deployment Systems

While simulation-based QML experiments are widespread, full-stack deployable quantum

applications remain rare in the literature. Cerezo et al. reviewed variational quantum algorithms extensively but noted the predominance of simulation-centric evaluations without deployment pipelines [4]. The present work directly addresses this gap by integrating quantum inference with web-based user interaction and containerized deployment.

III. Methodology and System Design

A. System Architecture Overview

The proposed system is structured as a layered hybrid architecture consisting of five interconnected modules: (1) Data Preprocessing, (2) QRAC Feature Compression, (3) Quantum Model (VQC), (4) Training and Optimization, and (5) Web Deployment. Fig. 1 illustrates the high-level data flow through these layers.

UserPreprocessing ModuleLabel Encoding → QRAC Compression22 features → 8 qubitsQuantum Model LayerZZFeatureMap (Feature Encoding)RealAmplitudes AnsatzVariational Quantum Classifier (VQC)Training & OptimizationQiskit Simulator + COBYLA OptimizerHybrid Quantum-Classical LoopWeb Interface (Flask)User Input → Prediction OutputReal-time Classification ResultDeployment LayerDocker Containerization

Fig. 1. High-level system architecture of the hybrid quantum-classical mushroom classification pipeline.

B. Dataset and Preprocessing

The UCI Mushroom dataset comprises 8,124 instances described by 22 categorical attributes including cap shape, odor, gill size, spore print color, and habitat. Since quantum circuits require continuous numerical input, all categorical attributes are first transformed using scikit-learn's LabelEncoder, mapping each category value to a unique integer. The target variable—edibility—is encoded as a binary label (0 = edible, 1 = poisonous). The dataset is partitioned into training and testing subsets using an 80/20 split with a fixed random seed for reproducibility.

C. QRAC-Based Dimensionality Reduction

A central challenge in applying VQCs to the 22-feature Mushroom dataset is the dimensionality mismatch between the classical feature space and the limited qubit count available in current quantum simulators. Quantum Random Access Coding (QRAC) provides a compression scheme in which n classical bits are encoded into m qubits ($m < n$) with a bounded success probability [6]. In this work, groups of three consecutive numerical features are mapped into a single qubit representation, compressing the 22-dimensional input to an 8-dimensional quantum-compatible vector. The compression function is defined as:

$$q_j = (\Sigma_{k=3j}^{3j+2} x_k \bmod 2) \bmod 2 \quad (1)$$

where x_k denotes the k-th encoded feature value and q_j the j-th compressed qubit-compatible feature for $j = 0, 1, \ldots, 7$. This scheme preserves parity-based correlations among grouped features while substantially reducing the input dimensionality.

D. Variational Quantum Classifier Design

The VQC is constructed using two quantum circuit components within the Qiskit Machine Learning framework: a feature map and a parameterized ansatz. The **ZZFeatureMap** with two repetitions encodes the 8-dimensional compressed input vector into quantum amplitudes, introducing entanglement between adjacent qubits to capture pairwise feature correlations. Formally, for an input vector $\mathbf{x}$, the feature map prepares the state:

$$|\varphi(\boldsymbol{x})\rangle = U_\varphi(\boldsymbol{x})\, H^{\otimes n}\, |0\rangle^{\otimes n} \quad (2)$$

where H denotes the Hadamard gate and $U_\varphi(\boldsymbol{x})$ is a diagonal unitary encoding second-order feature interactions. The **RealAmplitudes** ansatz, also with two repetitions, then applies a sequence of parameterized single-qubit rotation gates $R_y(\theta)$ and CNOT entangling gates. The total parameterized unitary is:

$$W(\theta) = \prod_l [CNOT_{layer} \cdot R_y(\theta_l)] \quad (3)$$

Classification predictions are derived from the expectation value of the Pauli-Z operator measured on the first qubit following circuit execution:

$$\hat{y} = sgn(\langle\varphi(\boldsymbol{x})|\, W^\dagger(\theta)\, Z_0\, W(\theta)\, |\varphi(\boldsymbol{x})\rangle) \quad (4)$$

E. Hybrid Training Procedure

The training procedure follows the standard hybrid quantum-classical optimization loop. At each iteration, the parameterized VQC circuit is executed on the Qiskit AerSimulator with 1,024 shots per evaluation. Measurement outcomes are aggregated to estimate expectation values, from which a cross-entropy loss is computed against the true labels. The COBYLA (Constrained Optimization BY Linear Approximations) classical optimizer updates the ansatz parameters $\boldsymbol{\theta}$ iteratively until convergence or a maximum of 200 iterations is reached. The parameter update rule follows the gradient-free COBYLA approach, making no derivative computations on the quantum device. Trained parameters and preprocessing encoders are serialized and stored as model artifacts for subsequent inference.

F. Web Interface and Deployment

The trained VQC is integrated into a Flask web application that exposes a prediction endpoint. Users submit 22 mushroom attribute values through an HTML form; the backend applies the identical label encoding and QRAC compression pipeline used during training before invoking the stored VQC for inference. Classification results (Edible / Poisonous) are rendered dynamically on a result page. The entire application is containerized using Docker, with all Python dependencies specified in a *requirements.txt* manifest. The container exposes the Flask server on port 5000, enabling consistent execution across development and production environments.

$q_0 q_1 q_2 q_3$HHHHZZFeatureMap(Feature Encoding)RφRφRφRφRealAmplitudes(Ansatz W(θ))RyRyRyRyMeasure⟨Z_0⟩MMMMŷEncodingVariationalReadout

Fig. 2. Schematic of the VQC quantum circuit: ZZFeatureMap encoding layer, RealAmplitudes variational ansatz, and measurement readout producing classification label ŷ.

IV. Results and Discussion

A. Experimental Setup

All quantum circuit simulations were performed using the Qiskit AerSimulator statevector backend with 1,024 measurement shots per circuit evaluation. The COBYLA optimizer was configured with a maximum of 200 iterations. Classical preprocessing, QRAC compression, and web interface code were implemented in Python 3.9. Model training was conducted on a standard laptop-class CPU; no specialized quantum hardware was employed.

B. Classification Performance

Table I summarizes the classification performance metrics achieved by the VQC on the Mushroom dataset test partition. The model attained a test accuracy of 87.4%, demonstrating that QRAC-compressed features retain sufficient discriminative information for effective quantum classification. Precision and recall for the poisonous class were notably high (0.91 and 0.85 respectively), reflecting the practical importance of minimizing false-negative predictions in a food-safety application.

TABLE I

CLASSIFICATION PERFORMANCE ON THE MUSHROOM TEST SET

Metric	Edible	Poisonous	Overall
Accuracy	—	—	87.4%
Precision	0.84	0.91	0.88
Recall	0.90	0.85	0.88
F1-Score	0.87	0.88	0.87

C. Training Convergence

Fig. 3 illustrates the training loss as a function of optimizer iteration. The hybrid quantum-classical loop converged within approximately 140 iterations, exhibiting a characteristic non-monotonic descent pattern attributable to shot noise in quantum measurement outcomes. The final training accuracy reached 89.1%, with minimal overfitting observed on the validation partition.

Training LossOptimizer Iteration0.90.70.50.350100150200

Fig. 3. Training loss versus optimizer iteration for VQC training using COBYLA on the Mushroom dataset. Convergence is observed near iteration 140.

D. Comparison with Classical Baselines

To contextualize VQC performance, Table II compares the proposed system against standard classical classifiers trained on the same preprocessed (label-encoded) feature set without QRAC compression. Although classical models operating on the full 22-feature space achieve higher raw accuracy, the VQC demonstrates competitive performance despite operating on only 8 QRAC-compressed features, underscoring the effectiveness of quantum feature representation for structured categorical data.

TABLE II

COMPARISON OF CLASSIFIER ACCURACY ON MUSHROOM DATASET

Classifier	Features	Accuracy (%)
Decision Tree	22 (full)	99.8
Random Forest	22 (full)	99.9
SVM (RBF)	22 (full)	98.6
Neural Network	22 (full)	99.1
VQC (Proposed)	**8 (QRAC)**	**87.4**

The accuracy gap between the VQC and classical models is principally attributable to information loss during QRAC compression and the limited circuit depth imposed by simulation runtime constraints. With deeper circuits, additional QRAC grouping strategies, or access to physical quantum hardware with higher qubit counts, classification accuracy is expected to improve substantially.

E. Web Application Validation

The deployed Flask application was validated across 50 manual test submissions covering both edible and poisonous mushroom profiles drawn from the test set. Predictions generated by the web interface matched offline VQC inference in all 50 cases, confirming pipeline consistency between training and production inference. Average end-to-end response time from form submission to result display was 1.8 seconds on a local Docker deployment, acceptable for interactive use.

V. Conclusion and Future Work

This paper presented a complete hybrid quantum-classical classification system leveraging a Variational Quantum Classifier for mushroom edibility prediction. The system demonstrated the practical integration of QRAC-based feature compression, parameterized quantum circuit training, and real-time web deployment within a reproducible Docker environment. The VQC achieved 87.4% classification accuracy on 8 QRAC-compressed features, validating the feasibility of quantum-enhanced classification for structured categorical datasets without requiring physical quantum hardware.

Although classical models achieve superior raw accuracy on the full feature set, the proposed architecture establishes a deployable proof-of-concept for hybrid quantum-classical intelligent systems—an area that remains underexplored in the existing literature. The system bridges the gap between theoretical QML research and practical software engineering, contributing a repeatable pipeline from data ingestion through quantum inference to user-facing prediction.

Future work will pursue several enhancements. First, deployment on real quantum processors (e.g., IBM Quantum systems) will enable empirical evaluation beyond simulation constraints. Second, the QRAC compression scheme will be replaced with learned quantum feature selection strategies to minimize information loss. Third, the framework will be extended to multi-class classification problems and higher-dimensional biomedical datasets. Fourth, explainability techniques will be integrated to provide insight into how quantum circuit parameters influence classification decisions—a critical requirement for safety-critical domains. Finally, cloud-based orchestration using Kubernetes will scale the web application to support concurrent users in production.

Acknowledgment

The authors express sincere gratitude to Mrs. G. Jyothi, M.Tech., Assistant Professor, and Mr. A. Venkateswara Rao, M.Tech. (Ph.D.), Head of Department, at Avanthi Institute of Engineering & Technology, for their invaluable guidance and mentorship throughout the course of this project.

References

[1] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.

[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[3] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, pp. 195–202, 2017.

[4] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, and P. J. Coles, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[5] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, pp. 209–212, 2019.

[6] A. Nayak, "Optimal lower bounds for quantum automata and random access codes," in *Proc. 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 1999, pp. 369–376.

[7] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, no. 6–7, pp. 467–488, 1982.

[8] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[9] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," *arXiv preprint arXiv:1307.0411*, 2013.

[10] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[11] A. Peruzzo, J. McClean, P. Shadbolt, M. H. Yung, X. Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, p. 4213, 2014.

[12] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, pp. 802–803, 1982.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] Qiskit Team, *Qiskit: An open-source framework for quantum computing*. IBM Quantum, 2023. [Online]. Available: https://qiskit.org/documentation/

[15] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.