Evaluating Angle and Amplitude Encoding Strategies for Variational Quantum Machine Learning: Their Impact on Model's Accuracy

Advanced Quantum Technologies

ADVANCED QUANTUM TECHNOLOGIES
www.advquantumtech.com

RESEARCH ARTICLE OPEN ACCESS

Evaluating Angle and Amplitude Encoding Strategies for Variational Quantum Machine Learning: Their Impact on Model's Accuracy

Antonio Tudisco[1] | Andrea Marchesin[2] | Maurizio Zamboni[1] | Mariagrazia Graziano[3] | Giovanna Turvani[1]

[1]Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy | [2]Department of Electronics and Nanoengineering, Aalto University, Espoo, Finland | [3]Department of Applied Science and Technology, Politecnico di Torino, Turin, Italy

Correspondence: Antonio Tudisco (antonio.tudisco@polito.it)

Received: 1 August 2025 | **Revised:** 30 December 2025 | **Accepted:** 19 January 2026

Keywords: quantum embedding | quantum machine learning | variational quantum algorithms

ABSTRACT

Recent advancements in Quantum Computing and Machine Learning have increased attention to Quantum Machine Learning (QML), which aims to develop machine learning models by exploiting the quantum computing paradigm. One of the widely used models in this area is the Variational Quantum Circuit (VQC), a hybrid model where the quantum circuit handles data inference while classical optimization adjusts the parameters of the circuit. The quantum circuit consists of an encoding layer, which loads data into the circuit, and a template circuit, known as the ansatz, responsible for processing the data. This work involves performing an analysis by considering both Amplitude- and Angle-encoding models, and examining how the type of rotational gate applied affects the classification performance of the model. This comparison is carried out by training the different models on two datasets, Wine and Diabetes, and evaluating their performance. The study demonstrates that, under identical model topologies, the difference in accuracy between the best and worst models ranges from 10% to 30%, with differences reaching up to 41%. Moreover, the results highlight how the choice of rotational gates used in encoding can significantly impact the model's classification performance. The findings confirm that the embedding represents a hyperparameter for VQC models.

1 | Introduction

In the contemporary era, information technology has become ubiquitous with profound implications for society. The advent of new electronic devices and connected services has enabled the collection of significant amounts of data, which can be further leveraged to improve the quality of our daily lives. However, interpreting data from various sources is a complex task, and the extraction of useful information requires sophisticated algorithms and technologies. Artificial Intelligence (AI), specifically within the branch of Machine Learning (ML) [1], offers a promising solution to automate this elaboration process.

Over the past three decades, many algorithms have been developed to accomplish precise tasks, with widespread applications in technology and science domains, including autonomous vehicle control, robotics, natural language processing, and computer vision [2]. However, these solutions are computationally expensive, and there is a need to find new efficient strategies to perform the related tasks.

In parallel with AI research, Quantum Computing (QC) [3, 4] has gained significant attention from the scientific community due to its potential to solve complex problems with algorithms that theoretically prove superior to classical known approaches (e.g., Shor's factorization algorithm [5] and Grover's quantum search

algorithm [6]). This potential, coupled with recent technological advancements that have led to the development of the first quantum computers, has prompted research to investigate opportunities for developing new QC solutions to address real-world complex problems.

The objective is to expand the computational capabilities of modern processing systems beyond limits previously considered unimaginable. Today, researchers are seeking to initiate another revolution based on this new paradigm, similar to the introduction of the first computers in the 1970s, to find applications with a quantum advantage.

Given the computational requirements of current ML algorithms and the prospect of a quantum advantage in the field, Quantum Machine Learning (QML) [7–11] was born. Several models have been proposed in the last few years to address a variety of tasks, ranging from solving high-energy physics problems [12]helping with the detection of drowsiness through EEG signals [13], to improving applications in chemical engineering such as reaction kinetics and optimization [14], to discovering new drugs [15], defining new convolutional neural networks [16], and even UAV path planning using a quantum-inspired experience replay in a Deep Reinforcement Learning framework [17]. The research in this new field is still in its infancy, with models demonstrating limited capabilities; thus, comparisons with well-established classical ML techniques are not yet entirely fair, highlighting potential topics for further investigation [18, 19]. Among these, when considering input classical data, it is necessary to express them through quantum formalism in a process known as embedding. Strategies such as quantum feature maps [20] and data re-uploading [21] have been adopted; these approaches directly influence both the computational resource requirements (e.g., number of qubits and circuit depth) and the expressive power of the final model [22].

This work presents an in-depth analysis of how the available embedding techniques influence the results achievable in performing the classification task using a Variational Quantum Algorithm (VQA) [23] of reference. The study addresses the embedding of input classical data through quantum formalism, and its influence in the classification performance of the model VQA.

In this work, two main categories of embedding strategies are analyzed, Angle and Amplitude (described also in Ref. [24]). The Amplitude encoding strategy consists of embedding the input data vector as the probability amplitude of the state vector. On the other hand, for the Angle encoding strategy the data are passed as the angle parameters for the rotation gates. Further details related to these encoding strategies are described in Section 2.2.1. The Angle encoding mechanism has been subjected to several algorithmic approaches to understand the possibility of a better strategy for performing the embedding task. Beyond Angle and Amplitude encoding strategies, more expressive approaches have recently been proposed, including Hamiltonian-based encoding methods [25–28], where classical data are mapped through data-dependent Hamiltonian evolutions. While a systematic benchmark of such approaches is beyond the scope of this work, preliminary experimental results are reported in Appendix A for completeness.

The study has been conducted on two established real datasets, i.e., Wine [29] and Diabetes [30], while keeping a constant number of qubits to implement the data classification model. This latter choice has posed a constraint on the definition of the subsequent model of Variational Quantum Circuit (VQC), thereby ensuring that the results are independent of it and focus solely on the efficacy of the embedding. The analysis seeks to identify the best embedding strategy for a given input dataset. To this end, a set of multiple random input transformations is considered to understand if one best performs independently from the input data. The experiments have been executed on a local computer using a simulator provided within the Pennylane Library [31], which can be considered representative of any kind of quantum computer's ideal behavior.

The results obtained in this work demonstrate that the embedding circuit has a significant impact on the classification performance of the model. Consequently, it can be regarded as a hyper-parameter and optimized through a benchmarking process specific to the dataset of interest. Indeed, when evaluating performance on both datasets, the differences in various metrics (accuracy, balanced accuracy, recall, precision and F1-score) between the best and the worst models varies from 10% to 30% on average, considering the same kind of architecture (the employing of the re-uploading technique, which is also described in Section 2.2.3, and the number of layers of the circuits).

In the literature, a few other works have explored the topic of quantum embedding in classification problems and how it influences the models' performance. In particular, they generally focus on a limited set of analyses, which do not provide a comprehensive understanding of the effects on real-world case applications. Fauzi R. et al. [32] have conducted tests on the effects of four different encoding techniques, namely Angle, Amplitude, Instantaneous Quantum Polynomial (IQP), and Complex Entangled, on the Iris dataset. In this case, the author suggests that the encoding strategy influences the results, but differently on our work, they do not consider all the possible combinations of the Angle encoding strategy, limiting the analysis to a single example of Angle encoding, and not taking into account the impact of the choice of the rotational gates on classification performance.

On the other hand, Sierra–Sosa et al., in two different works [33, 34], performed similar analyses but on a synthetic dataset. In the former case, the Authors compared the effects of applying Amplitude and Angle encoding strategies, while in the latter, Amplitude, IQP, and Second-order Pauli-Z evolution have been studied. In these cases, the Amplitude encoding strategies have obtained better results with respect the Angle encoding counterpart. A. Matic et al. [35] propose and analyze Quantum Convolutional Neural Networks (QCNNs) by varying both the encoding methods—focusing on high-order encoding and threshold encoding (embedding strategies that are also described in their article)—and the ansatz, including Basic and Strongly Entangling layers [11]. Their study compares the performance of quantum models against classical counterparts, showing that the QCNN using high-order encoding and Basic Entangling layers achieved performance comparable to classical models.

Similarly, in Ref. [36], an exploration of Angle encoding models using RY and RX gates, Amplitude encoding models, and High-

order models is presented. The study tested these models on two datasets, concluding that the encoding circuit influences the classification performance of quantum models. This work focuses exclusively on RX and RY Angle encoding models, excluding other combinations that are discussed and implemented in this article.

In all these works, the different encoding strategies have been applied without considering limitations on the number of qubits, which can significantly impact the resources required by the VQA. The outcome depends not only on the type of embedding technique used but also on the details of the related Quantum Machine Learning (QML) models, which vary in size and number of parameters. It is important to note that comparing results obtained from synthetic datasets with those already present in the state-of-the-art can be challenging. Additionally, when implementing the embedding, the default versions provided by two of the foremost open-source libraries (i.e., Qiskit and Pennylane) are adopted. This approach excludes the possibility of customizing the transformations applied to identify the most suitable ones for the classification task, especially in the case of the Angle encoding.

In summary, this paper proposes the following contributions:

- Propose the evaluation of different Angle encoding strategies by tuning the rotational gates applied.
- Evaluate these Angle encoding strategies based on their trajectories on the Bloch sphere.
- Compare Angle encoding models with Amplitude encoding models using the same number of qubits (and thus the same number of parameters).
- Introduce a methodology for selecting the optimal VQC model.

In the following sections, the background topics of ML and QC will be presented, together with their intersection, QML. Then, the methodology behind the test conducted will be discussed in detail before providing the analysis results (Table 1).

List of Abbreviations

TABLE 1 | List of Abbreviations.

QC	Quantum Computing
QML	Quantum Machine Learning
ML	Machine Learning
VQA	Variational Quantum Algorithm
VQC	Variational Quantum Circuit

2 | Theoretical Foundation

QML is an emerging research field based on developing models for ML by exploiting quantum technology to accelerate the training process and be capable of obtaining better results. The following sections will provide a detailed description of the two main topics, ML and QML, to ensure an accessible discussion to people with different backgrounds. Further elements can be deepened respectively in Refs. [37] and [38].

2.1 | Machine Learning

ML is a branch of Artificial Intelligence that focuses on creating algorithmic models that can adapt their behavior based on the input data and a specific target. In a sense, these models learn from the input information during a training procedure to take actions coherently with expected outcomes. ML algorithms can be distinguished based on different learning mechanisms, with supervised, unsupervised, and reinforcement learning being the three primary categories.

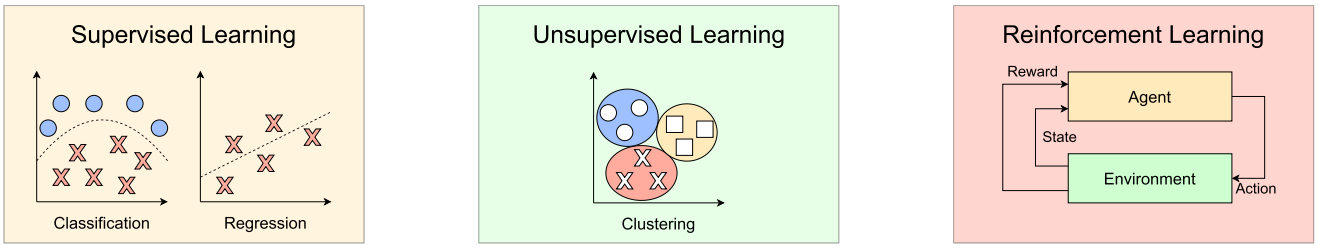
Supervised learning algorithms are trained on structured input data, with explicit and labeled features provided by humans to identify these characteristics in new data. Unsupervised learning techniques consider unlabelled data to explore their characteristics and identify useful patterns for grouping purposes. Finally, reinforcement learning algorithms consider dynamic unlabelled data and aim to train agents to make efficient decisions autonomously in specific environments. Examples of possible applications for ML algorithms belonging to the three categories proposed are reported in Figure 1.

The current study aims to evaluate the influence of embedding techniques on implementing a supervised learning mechanism. Specifically, this analysis will address the data classification task, and the subsequent discussion will focus on its foundational principles.

2.1.1 | Classification Task

In the domain of Supervised ML problems, a classification algorithm endeavors to establish a model that can assign input data to one or more predefined classes, identified by specific labels, based on a set of features. Broadly speaking, there are three distinct types of classification algorithms: *binary*, *multi-class*, and *multi-label*, which vary in terms of the number and kind of output labels that can be assigned to the input data. The objective of binary classification is to distinguish data into one of the two mutually exclusive classes. On the other hand, multi-class and multi-label classifications are characterized by more than two possible classes to which the data can be assigned. However, in the former case, these classes are always mutually exclusive, whereas in the latter, this condition is less stringent.

Focusing on the implementation details of such algorithms, it is always necessary to formulate a model composed of a parametric function $f(x, \theta)$ which can map input data x to available output classes y. Once defined, the model can be trained using a loss function to evaluate the discrepancy between the actual output of the function $\hat{y}$, and its expected value, y. Additionally, an optimizer is employed to adjust the θ parameters of the classification model to reduce the gap between $\hat{y}$ and y, effectively minimizing the classification error. A scheme of the overall process is presented in Figure 2.



(a) (b) (c)

FIGURE 1 | Graphical representation of the typical subdivision of ML mechanisms, and application examples. (a) Supervised learning algorithms, which are typically considered to perform *data classification* or *regression*. (b) Unsupervised learning, where *clustering* is the most common implementation. (c) Reinforcement learning, whose main application is *autonomous driving*.

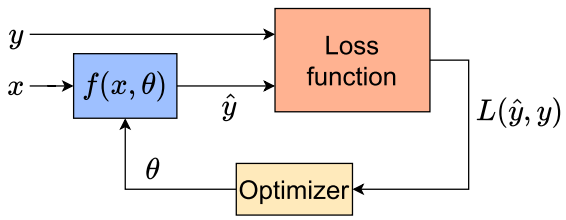


FIGURE 2 | Graphical scheme of the training processing adopted for classification purposes. The model (i.e., the blue rectangle) receives the input data x and estimates the output class they belong to, $\hat{y}$. The loss function then uses this information to evaluate the difference with respect to the correct target y, generating the magnitude of the error made by the model, $L(\hat{y}, y)$. Finally, the optimizer closes the loop by considering the actual error and adjusting the parameters θ of the model to increase its prediction performance during the next iteration.

Several metrics can be utilized to assess the quality of a trained model on a specific dataset. One of the most important is *accuracy*, which represents the percentage of correctly predicted samples over the entire dataset. For instance, in binary classification, where there are only two mutually exclusive classes (denoted as +1 and −1), the accuracy is calculated as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

where:

TP (True Positive) = Number of elements correctly predicted as positive

TN (True Negative) = Number of elements correctly predicted as negative

FP (False Positive)= Number of elements miss predicted as positive

FN (False Negative)= Number of elements miss predicted as negative

Other metrics, such as *precision*, *recall*, *F1-score*, and *Balanced accuracy* [39] can be defined using the same notation. These are particularly useful for investigating the model performance on unbalanced datasets, i.e., where the distribution of the associated elements to the available classes is not uniform. In particular, the precision represents the percentage of samples correctly predicted as true positive over all the data classified as positive:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

The recall is the percentage of data correctly predicted as positive over all the positive data:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

The F1-score is evaluated as the harmonic mean between precision and recall metrics, and it is defined as:

$$\text{F1 score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

The Balanced Accuracy is expressed as:

$$\text{Balanced accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2} \tag{5}$$

where the *Sensitivity* is the recall, while the *Specificity* is the true negative rate, which is:

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{6}$$

2.2 | Quantum Machine Learning

As introduced in the previous discussion, QML places itself in conjunction with two topics of great interest today: QC and ML. Among the approaches that can be found in the literature, the present work takes into consideration, for the related analyses, the particular model of VQC [40, 41].

This quantum algorithm, commonly referred to as the ansatz [42], is composed of a series of parametric rotational gates whose values must be properly tuned during the training phase. Here, the evaluation of the best angles to be set is performed by a classical optimizer. This hybrid quantum-classical approach simplifies the implementation complexity on quantum machines, thus contributing to the model's success on the limited current quantum hardware. A general scheme of the VQC and its training procedure is shown in Figure 3, while a description of how the quantum circuit is constituted is discussed in the following sections.

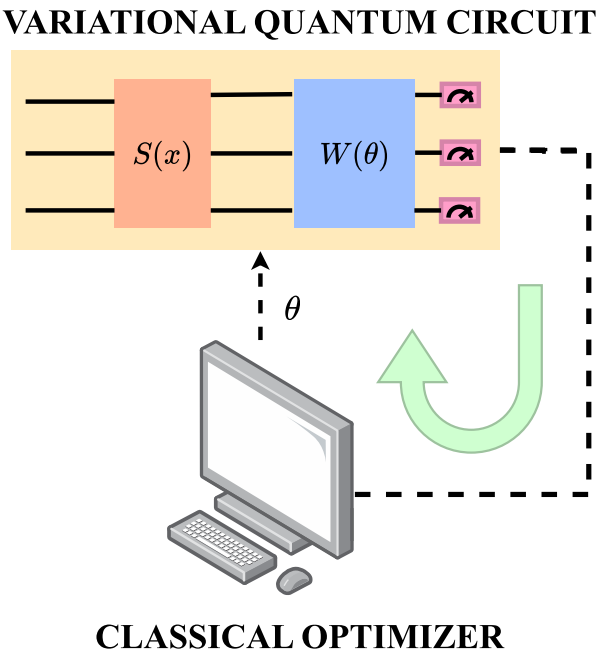


FIGURE 3 | Representation of a training process for a VQC, composed of an encoding circuit, $S(x)$, and the ansatz, $W(\theta)$, used for the ML task. The output of the overall quantum circuit is then passed to a classical optimizer to tailor the angle parameters θ to reduce the classification error.

2.2.1 | Encoding

The encoding circuit is a fundamental circuit of the VQC that permits the representation of classical data following quantum formalism. The input data provided to the VQC are represented by vectors whose elements are associated with the characteristics, called features, of the object considered. Different approaches to implementing this circuit, and some of them are addressed by the present work. In particular, the Basis, Amplitude, and Angle encoding. The details of these strategies are provided below, while for a wider perspective on the available embedding mechanisms, the reader is suggested to consider [24], by Maria Schuld and Francesco Petruccione, in the chapter *"Information Encoding"*.

2.2.1.1 | Basis Encoding. It is an embedding strategy that maps a features vector with N elements, each expressed with M bits, on a quantum circuit by placing the different elements in superposition with each other in the state vector $|\psi\rangle$:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |x_i\rangle \, |i\rangle \tag{7}$$

This strategy can be particularly effective if the number of features to be represented in the quantum circuit is larger than one because the number of qubit scales as in Equation 8:

$$\#\text{qubit} = M + \log_2(N) \tag{8}$$

Figure 4 shows an example of the application of the Basis Encoding to a feature vector to embed the elements $[011_{00}, 001_{01}, 101_{10}, 010_{11}]$.

2.2.1.2 | Amplitude Encoding. In this case, the value of each element of the input data vector is transformed into an amplitude probability of the state vector associated with the quantum circuit. At first, to implement this technique, it is necessary to normalize the input vector x by dividing each feature by the norm-2 of the same vector. Then, it is possible to map the vector as a state vector for the quantum circuit,

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} x_i \, |i\rangle \tag{9}$$

In particular, the Amplitude encoding strategy can be implemented by exploiting the Mottonen State Preparation [43], which is composed of a sequence of two main blocks: a circuit made of controlled RY gates to modify the module of the amplitude probabilities of the state vector and a circuit composed of controlled RZ gates to change the phases of the belonging qubits. Figure 5 shows an example of an application.

Considering the Amplitude encoding requires a number of qubits which scales with the logarithm to the base 2 of the number of elements belonging to the data vector,

$$\#\text{qubit} = \log_2(N) \tag{10}$$

2.2.1.3 | Angle Encoding. It consists of encoding data as angles for rotational gates. In this case, each feature is encoded on a different qubit, and the final state following these transformations will be

$$|\psi\rangle = (\otimes_{i=0}^{N-1} R(x_i)) \, |00\ldots0\rangle \tag{11}$$

An example of using the RX gate to embed the data into the quantum circuit is shown in Figure 6.

By employing the Angle encoding strategy, the number of the qubits needed scales as the number of features of the input data vector,

$$\text{qubit} = N \tag{12}$$

It is important to highlight that the most commonly employed strategy in the state-of-the-art involves using the RY gate for encoding data [44–46].

2.2.2 | Ansatz

The ansatz is the core of the VQC, which allows data processing to implement the classification task. It generally comprises two main components: a rotational circuit, constituted by rotational gates with parametric angles, and an entanglement circuit implemented with controlled gates. Since the choice of ansatz circuit topology directly influences the model's classification performance, it represents as a key hyperparameter in defining the QML model.

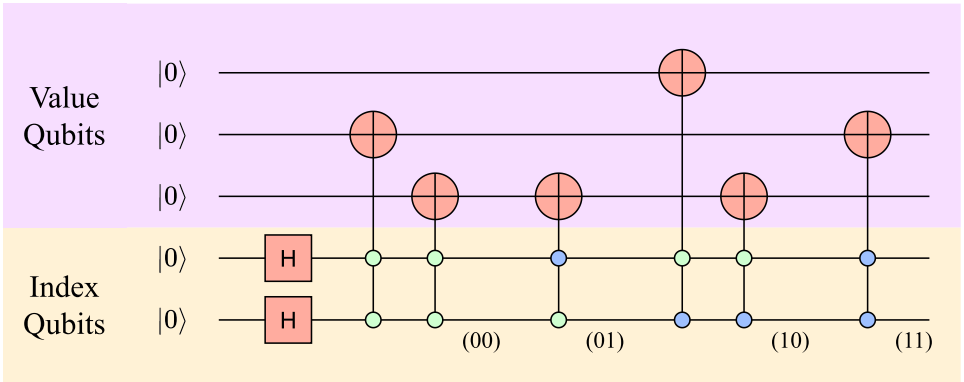


FIGURE 4 | Representation of the Basis Encoding strategy in which the feature vector [011, 001, 101, 010] is embedded into the quantum circuit. Each feature is encoded using CNOT gates, whose control qubits are referenced to the feature index (blue circles when the control is active with input $|1\rangle$, green when it is $|0\rangle$).

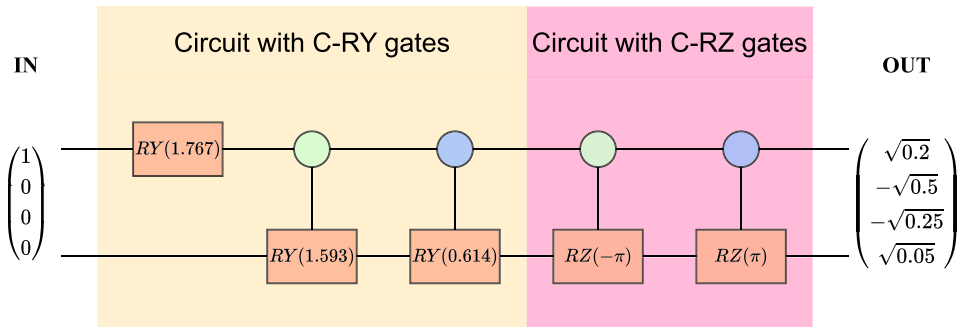


FIGURE 5 | Representation of a Mottonen State Preparation circuit that encodes the state vector $[\sqrt{0.2}, -\sqrt{0.5}, -\sqrt{0.25}, \sqrt{0.05}]$. The controlled gates, whose circles are colored in green, are active when the control qubit is in state $|0\rangle$; otherwise, they are active when the control is in state $|1\rangle$.

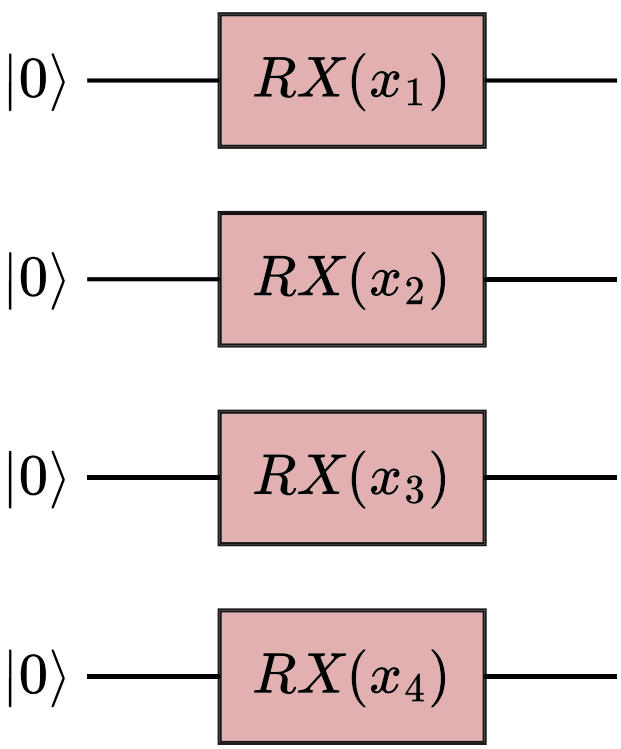


FIGURE 6 | Representation of an Angle encoding strategy using RX as the rotational gate. Each feature of the input vector is embedded in a qubit of the quantum circuit.

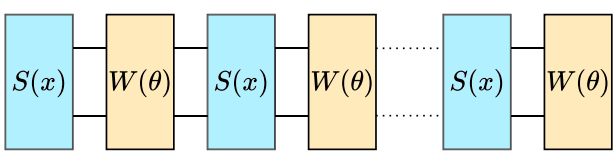


FIGURE 7 | Example of exploiting the re-uploading technique. The blue rectangles represent the embedding circuits, while the yellow ones are the ansatze.

2.2.3 | Re-Uploading Technique

The re-uploading technique [47], illustrated in Figure 7, is intended to improve the QML model's accuracy by embedding the input data multiple times in the quantum circuit, increasing thereby the model's expressive strength, and ability to describe complicated data patterns.

2.2.4 | Measurement

Once the quantum circuit has been defined, the measurement operation is performed to determine the class to which the data belongs. In particular, two different strategies are defined depending on whether the classification is binary or multi-class. For the former, the probability of a single qubit being in state $|1\rangle$ is measured. If it exceeds a threshold, typically 50%, the input data is assigned to class 1; otherwise, it is labeled as 0. For the latter,

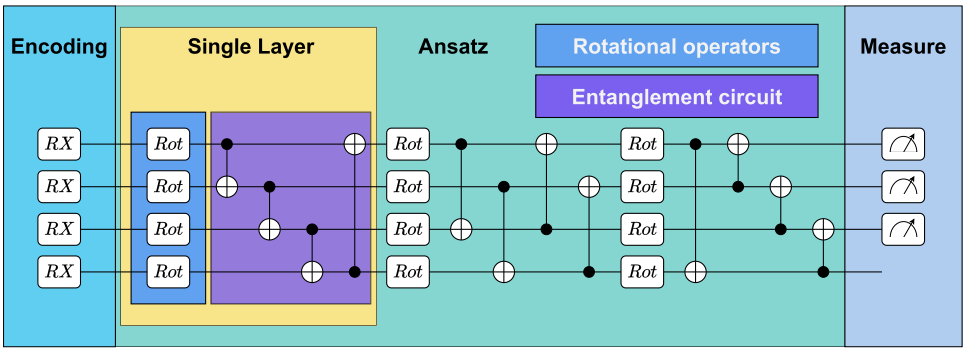


FIGURE 8 | Representation of an example of a VQC model. The encoding circuit is realized using an Angle encoding technique with RX gates, while the Ansatz is the Strongly Entangling layer circuit [11], realized with rotational (ROT) and CNOT gates.

where each input data belongs to one of the possible M classes, the probability of being in the $|1\rangle$ state is measured on the associated M qubits. In this case, after the measurement operation, classical post-processing is necessary, and the Softmax function is applied to the resulting probabilities z_i of each i-th qubit:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{M} e^{z_j}} \tag{13}$$

This function acts as a normalization operation, ensuring that the sum of the components of the output state vector equals one. The input data is then assigned to the class associated with the highest probability qubit.

3 | Implementation

The proposed work focuses on the VQC and its main parameters, described in details in the QML Section 2.2. Specifically, an analysis is conducted on the *encoding mechanism*, the *number of layers of the variational part*, and the application of the *re-uploading strategy*. The study measures the accuracy achieved by modifying these parameters to identify the optimal QML model for the classification task under consideration.

An example of an implemented model is shown in Figure 8, where the most significant blocks that compound the quantum circuit, such as the encoding and the variational circuit, are highlighted.

This section discusses the actual implementation and the methodology used to test the models' performance, providing the necessary details for the reader to replicate the experiments.

3.1 | Encoding Circuit

As previously discussed in the background section, the encoding is a fundamental component of a quantum circuit that enables the representation of data in the quantum domain for processing. In the experiments conducted in this work, Angle and Amplitude encoding strategies have been implemented. These strategies are suitable for near-term application solutions as they can be applied to a limited number of qubits.

Before encoding data into the quantum circuit, a pre-processing step is necessary to normalize the input data vector in the range of variation of the quantities involved by the specific embedding mechanism, i.e., the phase or the amplitude probability of the qubits. To achieve this, a min-max scaling of the features is applied, which allows mapping the data from the interval $[x_{min}, x_{max}]$ (including also negative values) to $[0, \pi]$ for both the Angle and Amplitude encoding strategies. The minimum and maximum values of each feature are taken from the training set. An additional normalization is applied only to the latter by dividing the state vector by its norm-two.

Regarding the implementation of the Amplitude encoding strategy, the Mottonen State Preparation, discussed in Section 2.2.1, has been adopted.

By contrast, Angle encoding offers a high degree of implementation flexibility: one may employ various combinations of rotational gates with the option of prepending a layer of Hadamard gates to prepare the input qubits in a uniform superposition. Therefore, a set of possible combinations of embedding circuits for the Angle encoding strategy can be identified, and the methodology for choosing them is discussed in the following.

3.1.1 | Angle Encoding Circuits

An initial set of feasible permutations has been defined, consisting of groups of sizes 1, 2, 3, and 4 of rotational quantum gates *(i.e., R_X, R_Y, and R_Z)* preceded, if possible, by Hadamard transformations. Then, the equivalent circuits have been identified, and only one instance between them has been considered within the set. For example, the circuit's behavior when applying RZ-RX rotational gates or the RX gate alone is identical. In fact, applying the encoding circuits on a single qubit initially in the $|0\rangle$ state, the resulting state vectors are reported in Equation 14 and Equation 15 for respectively the RX and RZ-RX strategies.

$$RX(x)\,|0\rangle = \begin{pmatrix} \cos(\frac{x}{2}) \\ -i\sin(\frac{x}{2}) \end{pmatrix} \tag{14}$$

$$RX(x)RZ(x)\,|0\rangle = e^{-i\frac{x}{2}} \begin{pmatrix} \cos(\frac{x}{2}) \\ -i\sin(\frac{x}{2}) \end{pmatrix} \tag{15}$$

TABLE 2 | Angle encoding strategies grouped per number of transformations required.

1-gate	2-gates	3-gates	4-gates
RX	RX-RY	RX-RY-RZ	H-RY-RX-RZ
RY	RX-RZ	RX-RZ-RY	H-RY-RZ-RX
	RY-RX	RY-RX-RZ	H-RZ-RX-RY
	RY-RZ	RY-RZ-RX	H-RZ-RY-RX
	H-RY	H-RY-RX	
	H-RZ	H-RY-RZ	
		H-RZ-RX	
		H-RZ-RY	

Twenty possible Angle encoding strategies have been selected, as shown in Table 2. Unlike the majority of works presented in the state-of-the-art, as reported in the Introduction, where the RY strategy is predominantly employed, this paper explores the entire solution space of Angle encoding techniques to observe how they influence the quality of the model.

Other considerations can be made by analyzing the trajectory of the embedding circuits as the input feature varies in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$, as shown in Figure 9, Figure 10, Figure 11, and Figure 12.

As observed, the complexity of the trajectory increases with the number of applied rotational gates. In the single-rotation case, the trajectory is simply a rotation around the axis associated with that gate. With two rotational gates, the trajectory becomes elliptical, and with three gates, it becomes even more complex.

Adding a Hadamard gate, on the other hand, only alters the starting and ending points of the trajectory. Notably, when using a single-rotation encoding, these points are distant to each other, while in more complex encoding circuits (e.g., those with three rotational gates), they coincide. Consequently, if the features only assume discrete values (e.g. binary features), a single-rotation encoding is preferable. Conversely, for continuous features, by using additional rotational gates may be more suitable.

3.2 | Variational Circuit

In the QML domain, the Ansatz refers to the parametric section of the circuit that allows for the processing of input data. The present study utilizes the Strongly Entangling layer, as proposed by Maria Schuld in Ref. [11] and implemented as a template in a Pennylane class [48]. This circuit comprises two main sections, namely, the one composed by the rotational operators and the other with the entanglement circuit, accomplished via CNOT gates. Figure 8 depicts an example of an ansatz of three layers.

The rotational operators involve the use of ROT gates, which are defined by the parameters ϕ, θ, and γ, and their transformation matrix is given by Equation 16. It is important to highlight that a generic unitary transformation can be accomplished by employing such gates.

$$Rot(\phi,\theta,\gamma) = RZ(\phi)RY(\theta)RZ(\gamma) = \\ = \begin{pmatrix} e^{-i(\phi+\gamma)/2}\cos(\theta/2) & -e^{i(\phi-\gamma)/2}\sin(\theta/2) \\ e^{-i(\phi-\gamma)/2}\sin(\theta/2) & e^{i(\phi+\gamma)/2}\cos(\theta/2) \end{pmatrix} \tag{16}$$

The main goal of the Entanglement Circuit shown is to create significant entanglement between qubits. The placement of CNOT gates guarantees that the qubits are highly correlated, allowing a larger portion of the Hilbert space to be explored efficiently, making the quantum algorithm more powerful. Furthermore, this entanglement circuit has demonstrated greater expressibility, as evidenced in Ref. [42].

One of the circuit parameters is the number of layers that constitute the ansatz. To reduce computational costs associated with finding the optimal number, models were tested on a limited set of layers, progressing in multiples of 2 (2, 4, 6, 8, 10).

3.3 | Re-Uploading Technique

The present work studies the effects of an additional technique introduced in Section 2.2.3, the "re-uploading", which involves embedding data into the quantum circuit each time a new ansatz is inserted. This method is utilized by certain models employed in the study, specifically those that leverage Angle or Amplitude encoding. The performance of these models is compared to that of models not implementing this technique.

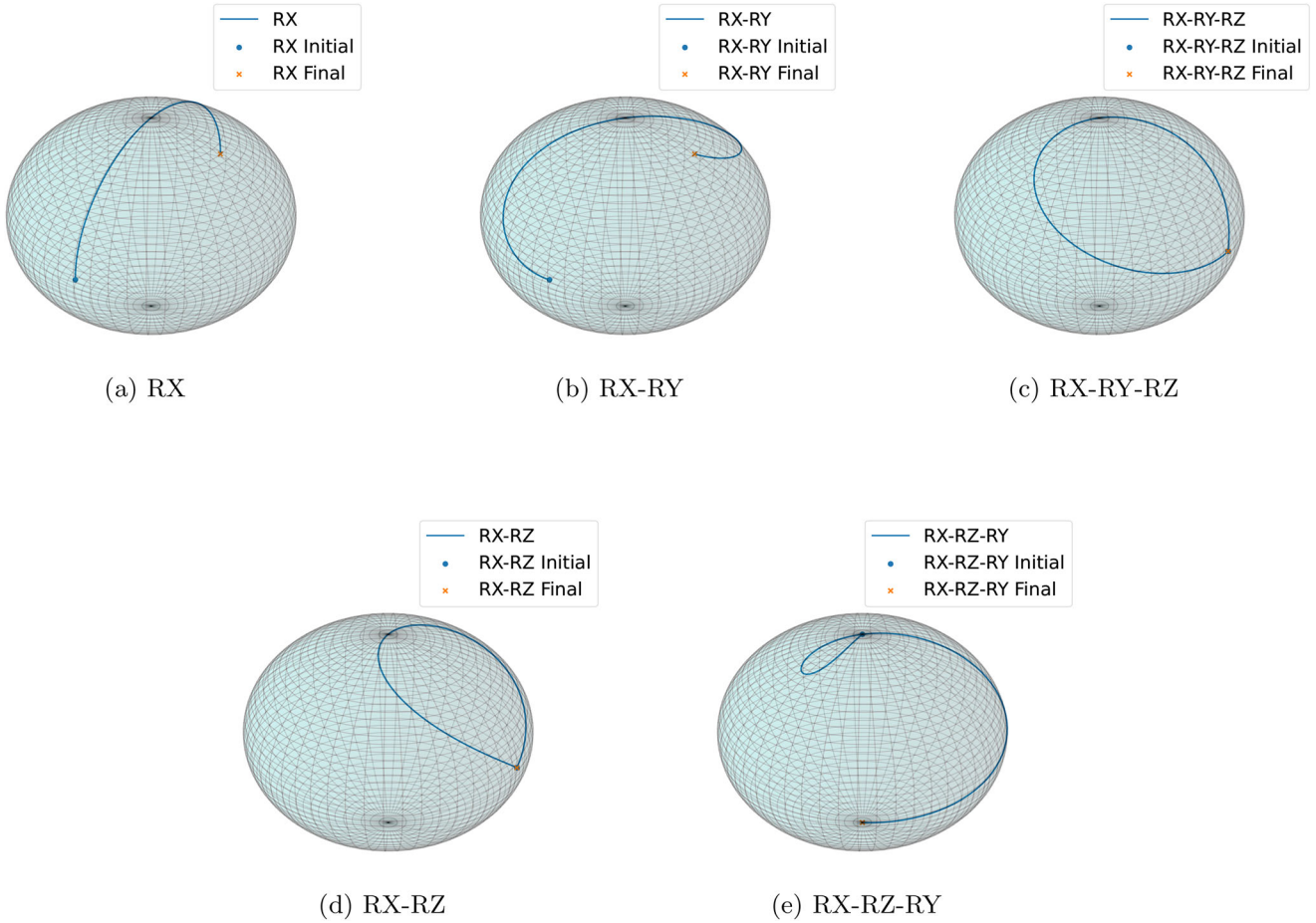
3.4 | Testing Methodology

Once the model has been defined, it is essential to consider the entire training and testing phase, beginning with data preparation. The data must be first normalized between 0 and π. Then, to ensure that both angle-encoded and amplitude-encoded models have the same number of qubits, Principal Component Analysis (PCA) [49] is applied to the angle-encoded models, reducing the number of features from N to $\log_2 N$. This reduction is required for angle encoding strategies whose number of qubits scales linearly with the number of input features. In this case, PCA is not introduced for enhancing the performance of the model, but as a constraint imposed by the angle encoding strategy.

This work employs real datasets to benchmark different encoding strategies. They are divided into training and test sets, which comprise 80% and 20% of the initial dataset, respectively.

Following dataset preparation, the training phase can begin. Each parameter is initialized at a random value and then optimized by exploiting the Adam algorithm [50]. This is a classical stochastic gradient descent optimization mechanism with an adaptive learning rate. In particular, the initial learning rate value has been set to 0.01, which represents a trade-off between a too-large value for which the optimizer can diverge from the optimal solution and a too-small one for which a higher number of steps is required.

The training process has been repeated for 30 epochs, i.e., considering 30 times the overall training dataset. Furthermore,



(a) RX (b) RX-RY (c) RX-RY-RZ

(d) RX-RZ (e) RX-RZ-RY

FIGURE 9 | Representations of the trajectories of applying the transformations RX, RX-RY, RX-RY-RZ, RX-RZ, and RX-RZ-RY varying the angle parameter in the interval $[-\pi/2, \pi/2]$.

the mini-batch strategy has been applied, where the dataset has been divided into subsets of 10 elements, and the parameters have been updated only when the evaluation of the subset has been completed. The training process was stopped at this limited number of epochs because we observed that the variations in both loss and accuracy during the final epochs were minimal.

Each model has been trained and tested ten times to ensure a fair comparison. Therefore, the results shown in the successive sections represent mean values.

3.5 | Settings

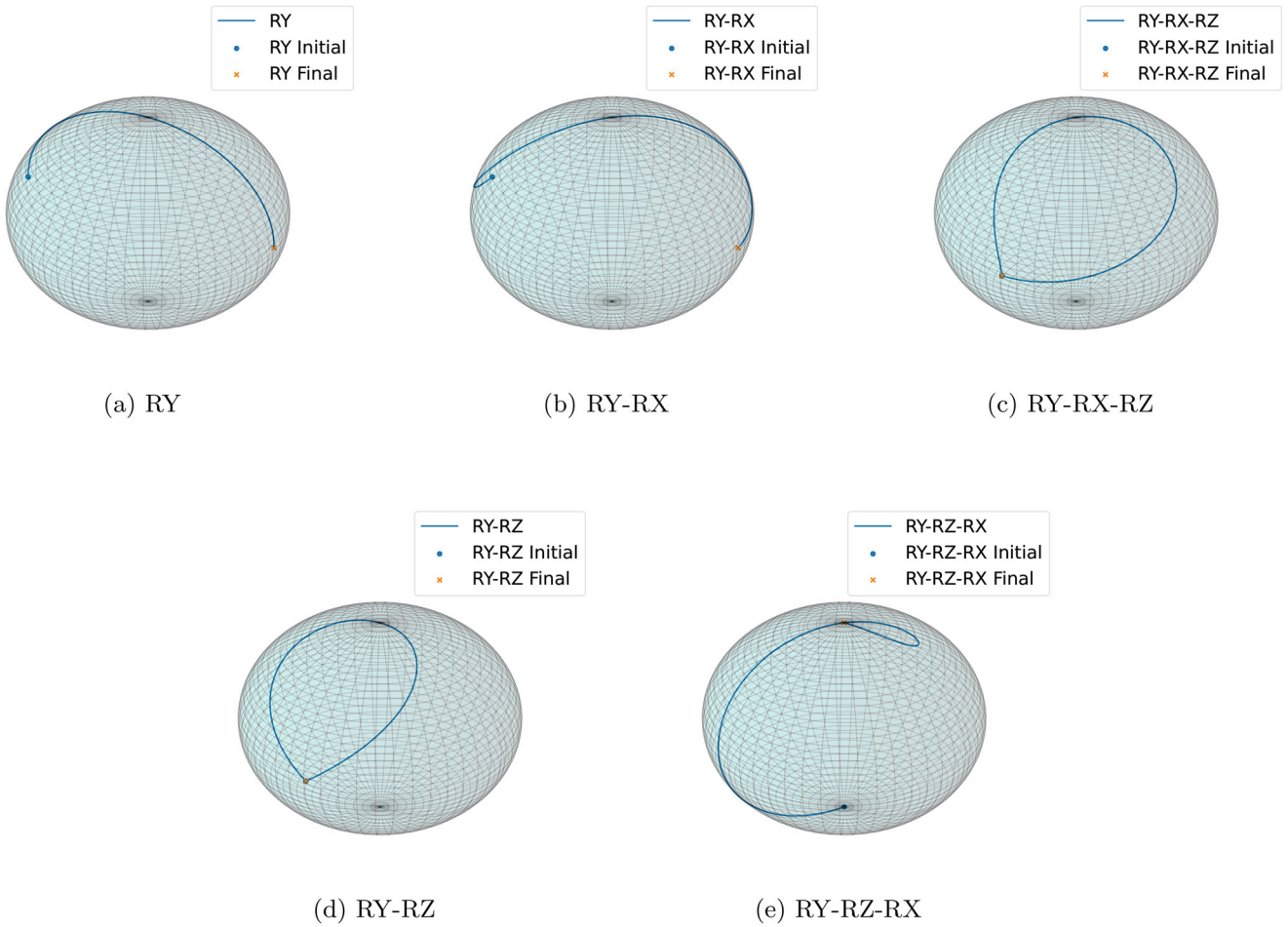
All the quantum circuits considered in this study have been developed using the Pennylane [31] library *(version 0.33.0)*, an open-source library for defining quantum machine learning, quantum chemistry, and quantum computing applications. The circuits have been simulated using the standard Pennylane qubit-based device, called *"default.qubit"*. This is an ideal simulator for testing the functionality of QML models. For the device, it's important to specify the desired number of qubits, which is provided as a parameter in the class definition. In our case, the number of qubits is set to four for the models applied to the Wine dataset, and three for those used with the Diabetes dataset.

In addition, the circuits have been interfaced with the PyTorch [51] library for the training phase to enhance the model's performance further. This integration has enabled classical parameter back-propagation. This approach has been preferred over the parameter-shift rule to reduce the implementation time. Because of its integration with PyTorch, we used PennyLane rather than Qiskit for implementing our quantum models.

4 | Results

The models were tested on two distinct datasets: the Wine [29] dataset and the Diabetes [30] dataset. The Wine dataset consists of 13 features and 768 samples, each classified into one of three classes. It's important to note that this dataset is unbalanced, with 59 samples in the first category, 71 in the second, and 48 in the third. On the other hand, the Diabetes dataset represents a binary classification problem with 768 items. Each one is characterized by eight features, including Pregnancy, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, and Age. Each sample is classified as either class 0 (non-diabetic) or class 1 (diabetic). Also this dataset is unbalanced, with 65% of the data belonging to class 0 and the remaining 35% to class 1.

For the Wine dataset, model performance is evaluated using accuracy as the primary metric. However, given the stronger



(a) RY (b) RY-RX (c) RY-RX-RZ

(d) RY-RZ (e) RY-RZ-RX

FIGURE 10 | Representations of the trajectories of applying the transformations RY, RY-RX, RY-RX-RZ, RY-RZ, and RY-RZ-RX varying the angle parameter in the interval $[-\pi/2, \pi/2]$.

imbalance in the Diabetes dataset, additional metrics are needed to evaluate the model's effectiveness.

Regarding the Wine dataset, models using Amplitude encoding methods have, on average, obtained better accuracy compared to those using Angle encoding, as illustrated in Figure 13. Despite this, the highest-performing model in this task employed Angle encoding with an RY gate for embedding, with 10 Strongly Entangling layers and no re-uploading. The encoding circuit was found to have a substantial impact on accuracy. In fact, the difference in accuracy between the best and worst model, holding the number of layers constant and using re-uploading, averaged 33%, with a maximum difference of 41.1%. The results are shown in Table 3, reporting the accuracy percentage for both the best and the worst model, as well as the associated standard deviation. In the same table, the accuracy difference between the best and worst models and the associated standard deviation ($\sqrt{STD^2_{best} + STD^2_{worst}}$) are also calculated.

For the Diabetes classification task, similar observations were made regarding the influence of the encoding strategy on model performance. When comparing various performance metrics, a significant disparity emerged between the best and worst models. These results were analyzed using accuracy (reported in Table 4), balanced accuracy (Table 5), recall (Table 6), precision (Table 7), and F1-score (Table 8). For each of the tables considered, not only the results of the metrics are provided, but also their associated standard deviation values. In addition, the difference between the best and worst model for each of the metrics considered and the associated standard deviation are calculated as well.

In terms of accuracy, the average difference between the best and worst models with identical architectures (same re-uploading process and number of layers) was approximately 8%, with the maximum difference reaching 10.3%. This gap became even more pronounced when considering balanced accuracy, where the average difference increased to 12%, with a maximum of 19%. Additionally, differences in precision, recall, and F1-score between the best and worst models remained significant, emphasizing the importance of encoding method optimization. However, these metrics exhibited greater uncertainties, which can depend on the intrinsic characteristics of the dataset or of the model. Further investigation on these results can be done to understand on what depends this uncertainty.

Another analysis addressed, models were compared based on the encoding strategy they employed—grouped by Angle and Amplitude encoding—and the use of the re-uploading technique

TABLE 3 | Comparison between the best and the worst encoding models grouped by the number of parametric layers and use of the re-uploading (RU) technique for classification on the Wine dataset. On average, the variation in terms of test accuracy is around 33%.

		Best results			Worst results				
Layers	***RU***	***Encoding***	***Accuracy (%)***	***STD (%)***	***Encoding***	***Accuracy (%)***	***STD (%)***	$\Delta_{Acc.}(\%)$	Δ_{Acc} ***STD (%)***
2	False	H-RZ	90.000000	6.029193	H-RY-RZ-RX	54.722222	9.170874	35.277778	10.975250
2	True	RX	86.944444	6.421682	RX-RY-RZ	62.222222	13.993434	24.722222	15.396564
4	False	RX	95.000000	4.498133	RX-RY-RZ	51.111111	6.441677	43.888889	7.856742
4	True	H-RY	93.611111	3.220838	H-RZ-RY-RX	71.111111	6.307180	22.500000	7.081972
6	False	RY	97.222222	3.207501	RX-RY-RZ	56.388889	3.939268	40.833333	5.079951
6	True	RY	93.888889	5.204989	RX-RZ-RY	64.722222	8.788983	29.166667	10.214604
8	False	H-RZ	96.944444	2.762303	RX-RY-RZ	59.166667	9.537487	37.777778	9.929449
8	True	H-RZ	91.388889	3.574122	RY-RX-RZ	63.611111	8.924504	27.777778	9.613591
10	False	RY	97.500000	2.432208	RX-RY-RZ	56.388889	5.406963	41.111111	5.928819
10	True	H-RZ	92.500000	4.545532	H-RY-RX-RZ	61.388889	9.925131	31.111111	10.916506

TABLE 4 | Comparison between the best and the worst encoding models in terms of accuracy grouped by the number of parametric layers and use of the re-uploading (RU) technique for classification on the Diabetes dataset. On average, the variation in terms of test accuracy is around 8%.

		Best results			Worst results				
RU	***Layers***	***Encoding***	***Accuracy (%)***	***STD (%)***	***Encoding***	***Accuracy (%)***	***STD (%)***	$\Delta_{Acc.}(\%)$	Δ_{Acc} ***STD (%)***
2	False	H-RZ-RY-RX	72.402597	3.064892	Amplitude	63.636364	1.960040	8.766234	3.638038
2	True	RX	72.987013	2.689563	Amplitude	66.493506	3.184100	6.493506	4.168002
4	False	H-RY-RX	73.961039	3.246032	Amplitude	64.480519	2.855995	9.480519	4.323590
4	True	H-RY-RZ	74.610390	3.387290	Amplitude	67.662338	4.023879	6.948052	5.259785
6	False	RX-RZ	74.350649	3.286196	H-RY-RZ-RX	65.194805	0.698031	9.155844	3.359514
6	True	H-RY	73.961039	2.379968	RY-RX-RZ	67.857143	3.982331	6.103896	4.639311
8	False	RX-RZ-RY	74.285714	3.425115	Amplitude	64.870130	2.237932	9.415584	4.091424
8	True	RY-RX	74.155844	2.785404	H-RY-RX-RZ	67.727273	3.225762	6.428571	4.261926
10	False	RX-RZ-RY	74.935065	2.471706	Amplitude	64.545455	2.582932	10.389610	3.575034
10	True	RY	74.025974	2.505592	H-RZ-RY-RX	67.662338	2.575667	6.363636	3.593334

TABLE 5 | Comparison between the best and the worst encoding models in terms of balanced accuracy grouped by the number of parametric layers and use of the re-uploading (RU) technique for classification on the Diabetes dataset. On average, the variation in terms of test accuracy is around 12%.

		Best results			Worst results				
Layers	***RU***	***Encoding***	***Bal. Acc. (%)***	***STD (%)***	***Encoding***	***Bal. Acc. (%)***	***STD (%)***	$\Delta_{B.Acc.}(\%)$	$\Delta_{Bal.Acc.}$ ***STD (%)***
2	False	H-RZ-RY-RX	66.611111	3.737306	H-RY-RZ-RX	51.088889	1.669129	15.522222	4.093098
2	True	H-RZ-RX	67.700000	3.004275	Amplitude	56.800000	4.229137	10.900000	5.187607
4	False	H-RY-RX-RZ	68.051852	4.476554	RX-RY-RZ	50.866667	1.585466	17.185185	4.749025
4	True	H-RZ-RX-RY	69.616667	2.845378	Amplitude	60.948148	3.178764	8.668519	4.266230
6	False	RX-RZ-RY	69.205556	2.348089	H-RY-RZ-RX	51.009259	1.576511	18.196296	2.828234
6	True	H-RY	69.259259	3.192737	RY-RX-RZ	63.025926	4.719782	6.233333	5.698238
8	False	RX-RZ-RY	69.850000	3.590465	RX-RY-RZ	51.835185	3.097246	18.014815	4.741769
8	True	RY	69.472222	2.662523	Amplitude	62.650000	5.052472	6.822222	5.711086
10	False	RX-RZ-RY	70.222222	2.472220	H-RY-RZ-RX	50.896296	1.281699	19.325926	2.784713
10	True	H-RY-RX	69.512963	2.352719	Amplitude	62.305556	5.110338	7.207407	5.625908

TABLE 6 | Comparison between the best and the worst encoding models in terms of recall grouped by the number of parametric layers and use of the re-uploading (RU) technique for classification on the Diabetes dataset. On average, the variation in terms of test recall is around 32%.

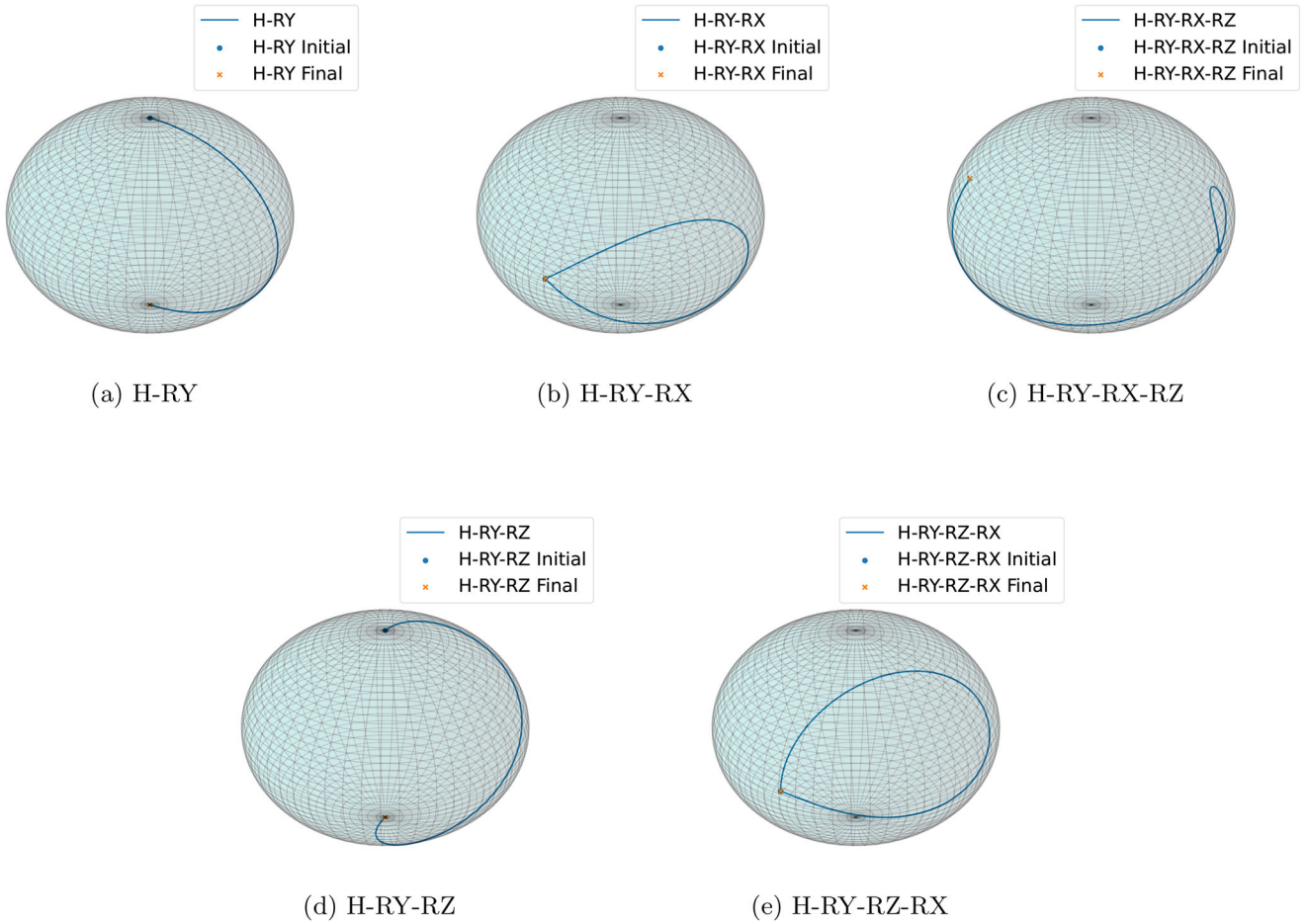
		Best results			Worst results				
Layers	***RU***	***Encoding***	***Recall (%)***	***STD (%)***	***Encoding***	***Recall (%)***	***STD (%)***	$\Delta_{Rec}(\%)$	Δ_{Rec} ***STD (%)***
2	False	H-RY-RX-RZ	55.000000	9.157881	H-RY-RZ-RX	2.777778	4.024199	52.222222	10.003048
2	True	H-RZ-RX	50.000000	5.790637	RX-RY-RZ	24.629630	10.549959	25.370370	12.034663
4	False	H-RY-RX-RZ	53.703704	7.759139	RX-RY-RZ	3.333333	4.765496	50.370370	9.105723
4	True	H-RZ	53.518519	7.974678	Amplitude	41.296296	6.593679	12.222222	10.347565
6	False	RX-RZ-RY	56.111111	4.623453	H-RY-RZ-RX	3.518519	5.754331	52.592593	7.381642
6	True	RY-RZ	54.074074	7.943559	RX-RZ-RY	45.185185	9.288021	8.888889	12.221599
8	False	RX-RZ-RY	55.000000	5.923997	RX-RY-RZ	5.370370	8.210109	49.629630	10.124210
8	True	X	54.629630	6.000686	Amplitude	45.000000	11.249143	9.629630	12.749566
10	False	H-RY-RX-RZ	55.740741	8.302412	H-RY-RZ-RX	2.592593	3.825169	53.148148	9.141223
10	True	RY-RX	55.925926	2.868877	RY-RX-RZ	44.814815	6.462347	11.111111	7.070529

TABLE 7 | Comparison between the best and the worst encoding models in terms of precision grouped by the number of parametric layers and use of the re-uploading (RU) technique for classification on the Diabetes dataset. On average, the variation in terms of test precision is around 29%.

		Best results			Worst results				
Layers	***RU***	***Encoding***	***Prec. (%)***	***STD (%)***	***Encoding***	***Prec. (%)***	***STD (%)***	$\Delta_{Prec}(\%)$	Δ_{Prec} ***STD (%)***
2	False	H-RZ-RX-RY	78.315941	8.552728	Amplitude	48.217717	7.688982	30.098223	11.500852
2	True	H-RZ-RX-RY	71.446353	8.296052	Amplitude	55.572861	11.536338	15.873492	14.209560
4	False	H-RY-RZ-RX	73.762838	25.861825	Amplitude	46.398719	12.635906	27.364119	28.783678
4	True	H-RY-RZ	68.419916	7.095356	Amplitude	54.447523	7.264074	13.972393	10.154352
6	False	RY-RX	74.367797	10.112072	Amplitude	55.664197	9.544638	18.703599	13.905183
6	True	RX-RY-RZ	67.077164	5.693694	RY-RX-RZ	54.814896	6.863735	12.262268	8.917903
8	False	RX-RY	74.514852	4.434600	Amplitude	49.007182	5.624648	25.507670	7.162565
8	True	RY-RX	66.513368	5.280583	H-RY-RX-RZ	54.740224	5.148878	11.773144	7.375330
10	False	RX-RY-RZ	80.769231	40.044354	Amplitude	50.247071	11.931222	30.522160	41.784020
10	True	RY	67.616456	6.318357	Amplitude	53.764180	7.723257	13.852276	9.978494

TABLE 8 | Comparison between the best and the worst encoding models in terms of f1-score grouped by the number of parametric layers and use of the re-uploading (RU) technique for classification on the Diabetes dataset. On average, the variation in terms of test F1-score is around 30%.

		Best results			Worst results				
Layers	***RU***	***Encoding***	***F1 (%)***	***STD (%)***	***Encoding***	***F1 (%)***	***STD (%)***	$\Delta_{Rec}(\%)$	Δ_{Rec} ***STD (%)***
2	False	H-RY-RX-RZ	55.757982	6.366353	H-RY-RZ-RX	12.543964	5.196771	43.214017	8.218082
2	True	H-RZ-RX	56.406678	4.732984	Amplitude	32.955614	10.863263	23.451064	11.849541
4	False	H-RY-RX-RZ	57.549603	6.046461	RX-RY-RZ	11.462068	7.475947	46.087536	9.615065
4	True	H-RZ-RX-RY	59.314808	4.275003	Amplitude	46.488617	4.853552	12.826191	6.467814
6	False	RX-RZ-RY	59.365833	3.365689	H-RY-RZ-RX	14.478836	9.028686	44.886996	9.635613
6	True	H-RY	58.807912	5.260160	RX-RZ-RY	50.280807	7.327488	8.527105	9.020053
8	False	RX-RZ-RY	59.973117	5.146937	H-RY-RZ-RX	15.768978	12.828721	44.204139	13.822700
8	True	RY	59.438495	3.681073	Amplitude	49.040808	8.843486	10.397687	9.579016
10	False	RX-RZ-RY	60.369697	3.583194	H-RY-RZ-RX	9.261734	6.626950	51.107964	7.533641
10	True	RY-RX	59.772130	3.119151	Amplitude	49.453722	7.659514	10.318408	8.270264



(a) H-RY (b) H-RY-RX (c) H-RY-RX-RZ

(d) H-RY-RZ (e) H-RY-RZ-RX

FIGURE 11 | Representations of the trajectories of applying the transformations H-RY, H-RY-RX, H-RY-RX-RZ, H-RY-RZ, and H-RY-RZ-RX varying the angle parameter in the interval $[-\pi/2, \pi/2]$.

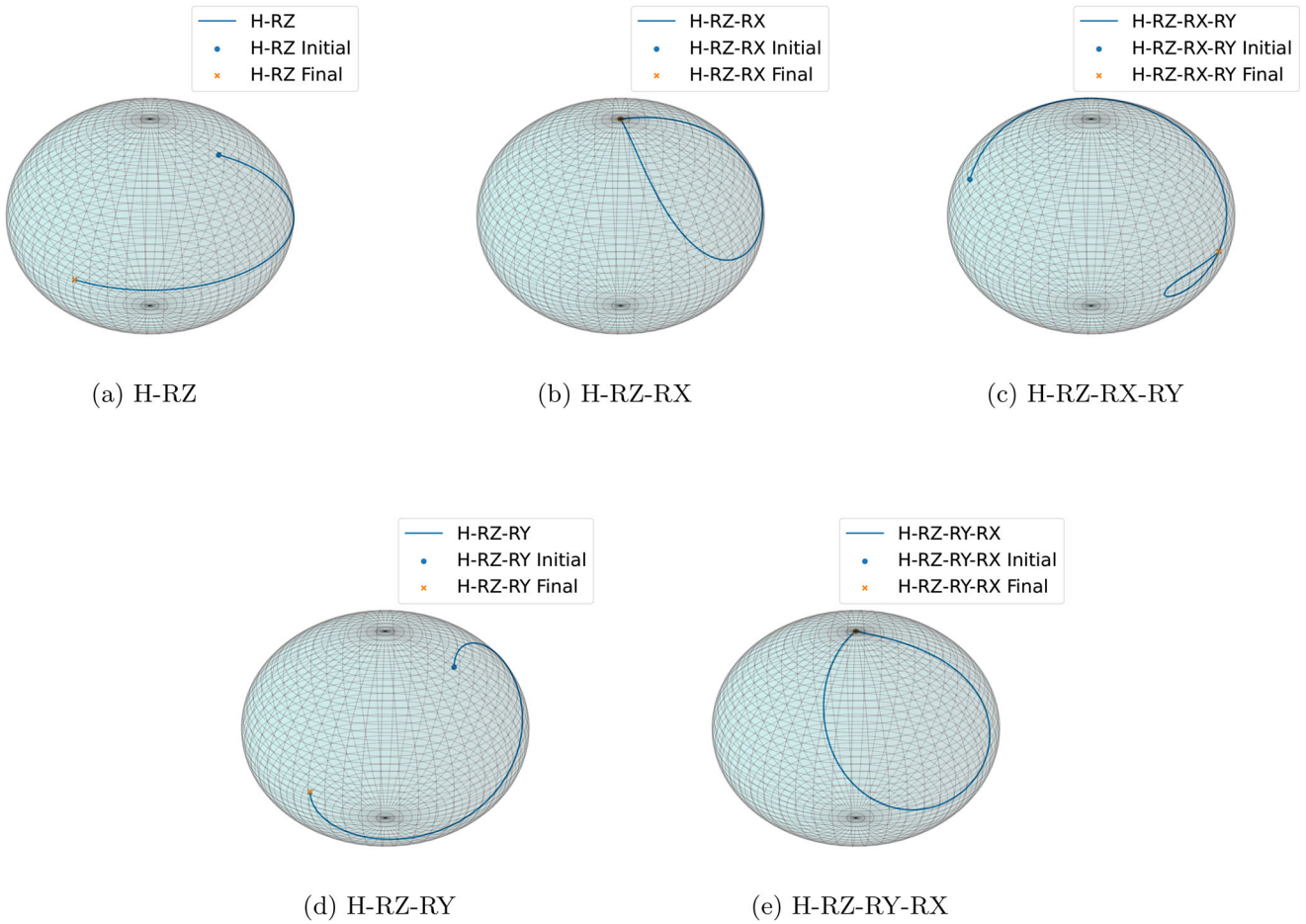
across all considered metrics (Accuracy, Balanced Accuracy, Recall, Precision, and F1-score). The best-performing models that exploit Angle encoding overall outperformed their counterparts with Amplitude encoding on all metrics. Additionally, models that employed the re-uploading strategy showed an average improvement in classification performance, particularly on Balanced Accuracy, Recall, and F1-score, compared to models that did not use it. The selection of the best model depends on the specific classification goal. If balanced accuracy and F1-score are prioritized, the best model is the RX-RZ-RY with 10 layers and no re-uploading. If precision is the primary focus, the RX-RY-RZ with 10 layers and no re-uploading is the optimal choice. For maximizing recall, the RX-RZ-RY with 6 layers and no re-uploading performs best. Low recall results are a consequence of training models on an unbalanced dataset without applying any balancing techniques. In this setting, the optimizer minimizes overall loss by biasing the majority class. As a result, positive predictions are fewer but more accurate, ensuring high precision at the expense of low recall.

All the collected results highlight the critical impact of encoding methods on model classification performance, reinforcing the idea that encoding should be treated as a key hyper-parameter that must be carefully optimized to enhance both model accuracy and efficiency. Refining the encoding process can lead to significant improvements in the model's overall predictive capabilities.

Moreover, the study examines the effect of encoding strategies on model simulation times for both the Wine and Diabetes datasets, as shown in Figure 15. The results were obtained through simulation in Ref. [52], with the number of threads constrained to 24. These results depend on system parameters, such as the number of CPU cores, CPU frequency, and the number of threads, among others. Additionally, since the simulations were not conducted on a real device, it is not possible to guarantee that the measured times directly correlate with the times required on actual hardware. Training and inference times for both the training and test sets are reported. On average, models employing Amplitude encoding (using the Mottonen circuit) took significantly longer to train and infer than those using Angle encoding. As expected, models utilizing the re-uploading strategy also required more time for both training and inference.

5 | Proposed Future Methodology

Given the trajectory analysis and the results obtained, we propose a structured methodology for designing and optimizing Variational Quantum Circuit models. The procedure initiates



(a) H-RZ (b) H-RZ-RX (c) H-RZ-RX-RY

(d) H-RZ-RY (e) H-RZ-RY-RX

FIGURE 12 | Representations of the trajectories of applying the transformations H-RZ, H-RZ-RX, H-RZ-RX-RY, H-RZ-RY, and H-RZ-RY-RX varying the angle parameter in the interval $[-\pi/2, \pi/2]$.

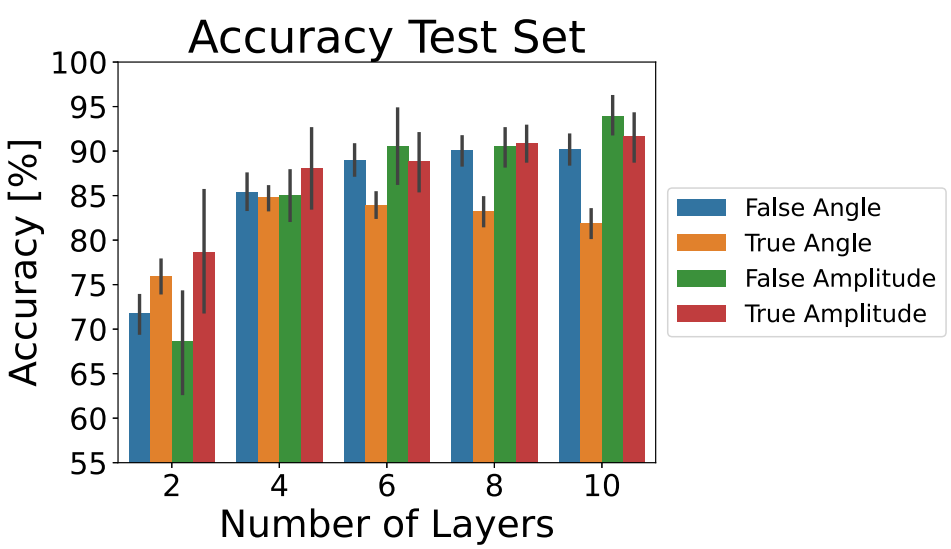


FIGURE 13 | Comparison of models utilizing and not utilizing the re-uploading technique, along with Angle and Amplitude encoding methods, in terms of accuracy on the Diabetes dataset. The results show that the Amplitude encoding models have obtained higher performance on average for the VQC with more than 2 layers. Moreover, the re-uploading strategy does not guarantee higher accuracy results in this case.

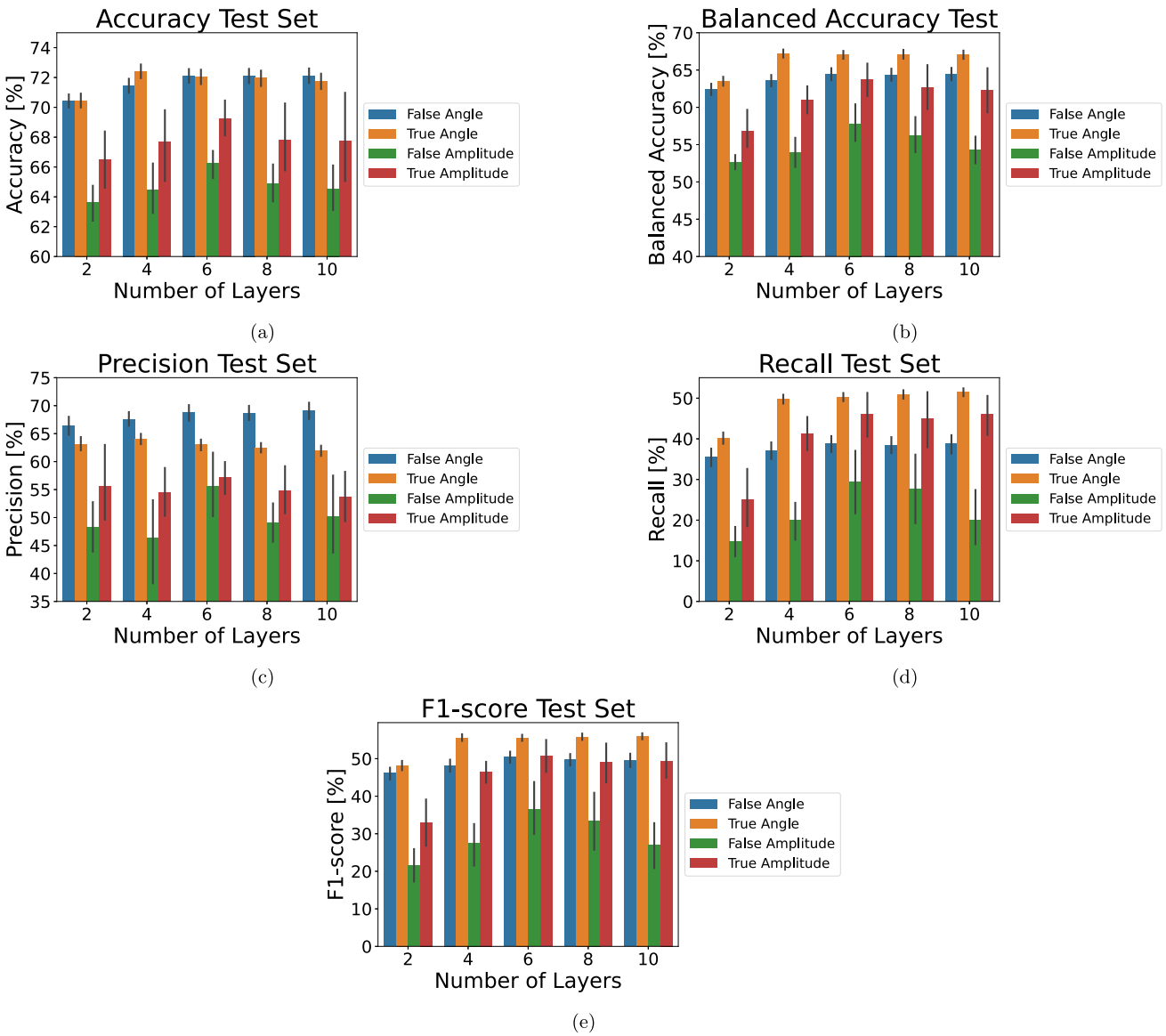


FIGURE 14 | Comparison of models utilizing and not utilizing the re-uploading technique, along with Angle and Amplitude encoding methods, in terms of Accuracy (Figure 14a), Balanced Accuracy (Figure 14b), Precision (Figure 14c), Recall (Figure 14d), and F1-score (Figure 14e). The results indicate that Angle encoding models generally outperform others across all the metrics evaluated. Furthermore, models utilizing the re-uploading technique demonstrate superior performance in terms of Balanced Accuracy, Recall, and F1-score for both Angle and Amplitude encoding. However, for Accuracy and Precision, the re-uploading technique provides improvements exclusively for Amplitude encoding models.

with a simple implementation and progressively incorporates more sophisticated techniques to improve performance. The first stage involves the application of a **single rotational gate Angle encoding** strategy. By this, continuous variable features are encoded into the quantum circuit using a single rotational gate (e.g., RX, RY, or RZ) for each feature. This strategy serves as a baseline implementation whose primary goal is to evaluate the model's performance using this simple embedding technique and establish a starting point to enhance the classification performance.

In cases where the single rotational gate strategy yields suboptimal results, more sophisticated Angle encoding approaches are exploited. The idea is to increase progressively the number of rotational gates to encode features along different axes of the qubit state to enhance the model's ability to capture complex patterns in the data, thereby improving classification performance.

This methodology provides a systematic and scalable approach to designing Variational Quantum Circuits models. It guarantees that the complexity of the model does not increase too much, leading to overfit.

6 | Conclusion

QML is an innovative and rapidly developing field with significant potential for advancement. At this moment in history, due to the limited number of qubits in actual quantum devices and the non-idealities affecting them, it is necessary to consider

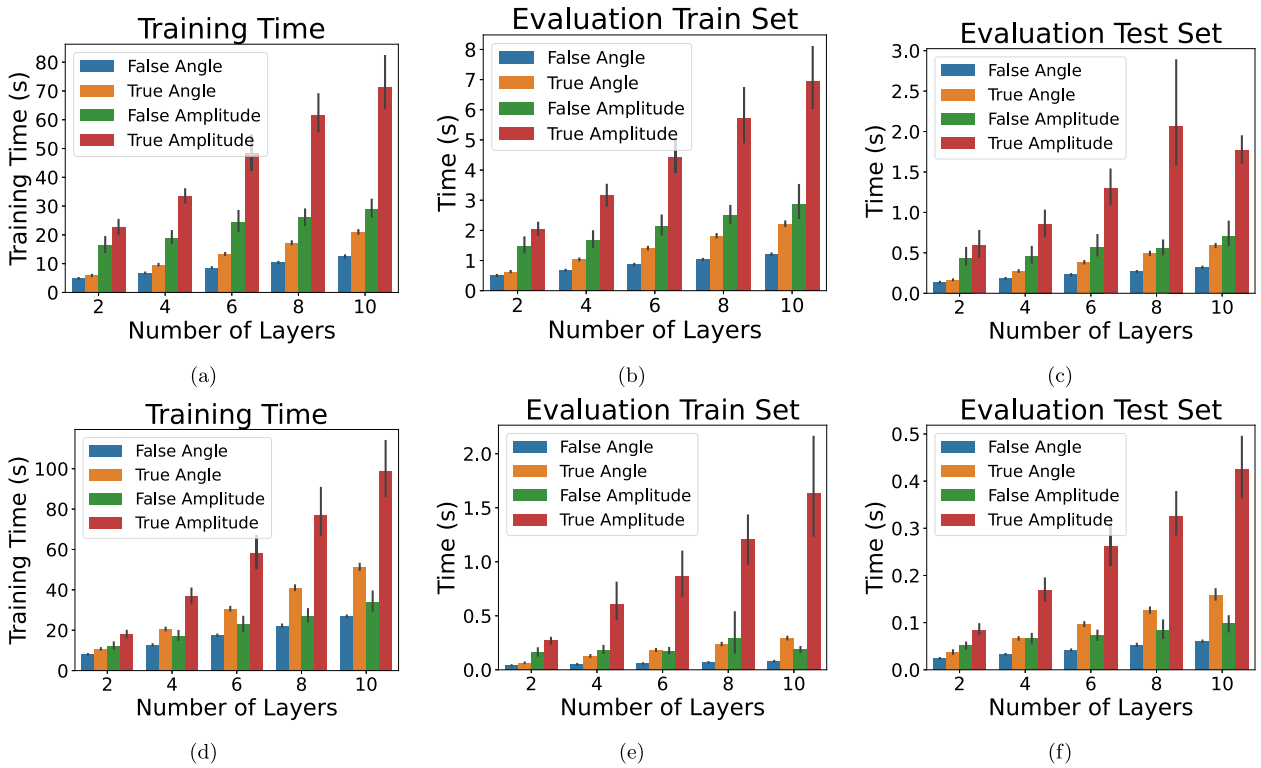


FIGURE 15 | Illustration of the time required to train the model (Figure 15a), to evaluate the training set (Figure 15b), and to evaluate the test set (Figure 15c) on the Wine dataset and on the Diabetes dataset (respectively Figure 15d, Figure 15e, Figure 15f). The four bars displayed represent: models utilizing Angle encoding without reuploading, models employing the reuploading technique with Angle encoding, models using Amplitude encoding without reuploading, and models with both Amplitude encoding and reuploading. Results indicate on both analyses that the Amplitude encoding circuit is the slowest. Furthermore, as the number of layers increases, so does the time needed for model training and inference. Finally, the application of the reuploading technique involves additional training and inference time.

hybrid solutions in which quantum and classical computing work together. In this regard, new models, such as VQCs are being explored, where the quantum computer implements the ML application and the classical computer reveals useful during the training procedure to update its parameters.

This work aims to highlight the impact of quantum embedding on the accuracy of quantum models, with a particular focus on Angle and Amplitude encoding strategies. Specifically, for Angle encoding, a broad and systematic analysis of possible encoding mechanisms is benchmarked. Special attention has been given to the comparability of results, to keep the topology of the variational part constant in every comparison between the adopted encoding strategies. To do so, the number of qubits made available to the models has been kept constant, and PCA has been applied to the considered datasets to reduce the number of features to be processed. While this classical preprocessing step may partially obscure the intrinsic behavior and scaling properties of the quantum models, it has been employed for comparing under fixed quantum resource constraints. This choice has intrinsically disadvantaged the performance of the overall models but guarantees fairness in comparisons.

As observed from the results, the accuracy differences between the best and the worst models, considering the same number of layers for the ansatz and the application or non-application of the re-uploading technique, can have an impact of tens of percentage points. Therefore, the encoding strategy should be considered as a hyper-parameter for defining the models, influencing their accuracy performance. Moreover, it is worth noting that no single encoding technique guarantees the best classification, as it depends on the dataset considered. This adds another complexity element in the definition of QML applications, as its development is not strictly related to the formulation of an effective ansatz but opens up the need to investigate the best encoding strategy for a given dataset.

For the two datasets considered, the best Angle encoding model achieves higher classification performance with respect to the Amplitude encoding model. Notably, across all tested configurations, Amplitude encoding does not emerge as the best-performing strategy, although it provides competitive baseline performance. However, this outcome is not intrinsic to the encoding paradigm itself, but rather results from a refinement in the choice of the rotational gates used for data embedding, as also demonstrated by the figures grouping the results according to the encoding type and the use of the re-uploading technique.

In the future, further exploration will be addressed to investigate possible correlations between the encoding techniques and generic input data to identify a strategy to find the best encoding through a-priori analysis on the input dataset, avoiding a time-consuming benchmark procedure today's required. To achieve this, larger datasets should be explored and tested, which has

not been possible at the moment due to limited computational resources. Indeed, training and inference of models on larger datasets would take significantly longer. This is also evident from the graphs (Figure 15), where models tested on the Wine dataset (with circuits on 4 qubits) take four times as long as those tested on the Diabetes dataset (three qubits).

In addition, given the low results in terms of the model's classifications performance, it is considered appropriate to do further investigation on finding the ideal model that is best able to classify by varying all possible hyper-parameters that constitute it. For instance, one could evaluate alternative ansatz topologies constituted by also different entangling gates and the optimal number of layers for each dataset (considering also the application of the reuploading strategy). Once found the optimal hyper-parameters of the model, it could be possible to compare the QML models with the classical ones.

Finally, to account for the impact of noise, one could evaluate the classification performance on a noisy simulator by training models in both ideal and noisy simulation environments.

Acknowledgements

Open access publishing facilitated by Politecnico di Torino, as part of the Wiley - CRUI-CARE agreement.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are openly available in [VQC-Encoding] at [https://github.com/antotu/VQC-Encoding], reference number [1022002703].

References

1. S. Russell and P. Norvig, *Artificial Intelligence A Modern Approach, Global Edition*.

2. M. I. Jordan and T. M. Mitchell, "Machine Learning: Trends, Perspectives, and Prospects," *Science* 349, no. 6245 (2015): 255–260.

3. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).

4. S. Lloyd, "Universal Quantum Simulators," *Science* 273, no. 5278 (1996): 1073–1078.

5. P. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994): 124–134.

6. L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96 (New York, NY, USA: Association for Computing Machinery, 1996), 212–219, https://doi.org/10.1145/237814.237866.

7. M. Schuld, I. Sinayskiy, and F. Petruccione, "An Introduction to Quantum Machine Learning," *Contemporary Physics* 56, no. 2 (Apr. 2015): 172–185.

8. J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum Machine Learning," *Nature* 549, no. 7671 (Sept. 2017): 195–202.

9. C. Ciliberto, M. Herbster, A. D. Ialongo, et al., "Quantum Machine Learning: A Classical Perspective," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474, no. 2209 (2018): 20170551.

10. M. Schuld and N. Killoran, "Quantum Machine Learning in Feature Hilbert Spaces," *Physical review letters* 122 (Feb. 2019): 040504.

11. M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, "Circuit-Centric Quantum Classifiers," *Physical Review A* 101 (Mar. 2020): 032308.

12. W. Guan, G. Perdue, A. Pesah, et al., "Quantum Machine Learning in High Energy Physics," *Machine Learning: Science and Technology* 2, no. 1 (Mar. 2021): 011003.

13. I. D. Lins, L. M. M. Araújo, C. B. S. Maior, et al., "Quantum Machine Learning for Drowsiness Detection With EEG Signals," *Process Safety and Environmental Protection* 186 (2024): 1197–1213, https://www.sciencedirect.com/science/article/pii/S0957582024003847.

14. D. E. Bernal, A. Ajagekar, S. M. Harwood, S. T. Stober, D. Trenev, and F. You, "Perspectives of Quantum Computing for Chemical Engineering," *AIChE Journal* 68, no. 6 (2022): e17651, https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.17651.

15. S. Mensa, E. Sahin, F. Tacchino, P. K. Barkoutsos, and I. Tavernelli, "Quantum Machine Learning Framework for Virtual Screening in Drug Discovery: A Prospective Quantum Advantage," *Machine Learning: Science and Technology* 4, no. 1 (Feb. 2023): 015023.

16. I. Cong, S. Choi, and M. D. Lukin, "Quantum Convolutional Neural Networks," *Nature Physics* 15, no. 12 (Dec. 2019): 1273–1278.

17. Y. Li, A. H. Aghvami, and D. Dong, "Path Planning for Cellular-Connected UAV: A DRL Solution With Quantum-Inspired Experience Replay," *IEEE Transactions on Wireless Communications* 21, no. 10 (2022): 7897–7912.

18. M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles, "Challenges and Opportunities in Quantum Machine Learning," *Nature Computational Science* 2, no. 9 (Sept. 2022): 567–576.

19. A. Perdomo-Ortiz, M. Benedetti, J. Realpe-Gómez, and R. Biswas, "Opportunities and Challenges for Quantum-Assisted Machine Learning in Near-Term Quantum Computers," *Quantum Science and Technology* 3, no. 3 (Jun. 2018): 030502.

20. S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, "Quantum Embeddings for Machine Learning," (2020).

21. A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, "Data Re-Uploading for a Universal Quantum Classifier," *Quantum* 4 (2020): 226.

22. M. Schuld, R. Sweke, and J. J. Meyer, "Effect of Data Encoding on the Expressive Power of Variational Quantum-Machine-Learning Models," *Physical Review A* 103 (Mar. 2021): 032430.

23. M. Cerezo, A. Arrasmith, R. Babbush, et al., "Variational Quantum Algorithms," *Nature Reviews Physics* 3, no. 9 (Sept. 2021): 625–644.

24. M. Schuld and F. Petruccione, *Supervised Learning With Quantum Computers* vol. 17 (Springer, 2018).

25. J. Schnabel and M. Roth, "Quantum Kernel Methods Under Scrutiny: A Benchmarking Study," *Quantum Machine Intelligence* 7, no. 1 (Apr. 2025): 58, https://doi.org/10.1007/s42484-025-00273-5.

26. B. Jaderberg, A. A. Gentile, Y. A. Berrada, E. Shishenina, and V. E. Elfving, "Let Quantum Neural Networks Choose Their Own Frequencies," *Physical Review A* 109 (Apr. 2024): 042421, https://link.aps.org/doi/10.1103/PhysRevA.109.042421.

27. P. Wang, C. R. Myers, L. C. L. Hollenberg, and U. Parampalli, "Quantum Hamiltonian Embedding of Images for Data Reuploading Classifiers," *Quantum Machine Intelligence* 7, no. 1 (Mar. 2025): 35, https://doi.org/10.1007/s42484-025-00247-7.

28. G. Maragkopoulos, N. Stefanakos, A. Mandilara, and D. Syvridis, "Applications of Hybrid Machine Learning Methods to Large Datasets:

A Case Study," in *2025 International Conference on Quantum Communications, Networking, and Computing (QCNC)* (IEEE, 2025): 683–689.

29. S. Aeberhard and M. Forina, "Wine, UCI Machine Learning Repository," 1991, https://doi.org/10.24432/C5PC7.

30. "Diabetes Dataset," https://www.kaggle.com/datasets/mathchi/diabetes-data-set, accessed: (March 2024).

31. V. Bergholm, J. Izaac, M. Schuld, et al., "Pennylane: Automatic Differentiation of Hybrid Quantum-Classical Computations," (2022).

32. R. Fauzi, M. Zarlis, H. Mawengkang, and P. Sihombing, "Analysis of Several Quantum Encoding Methods Implemented on a Quantum Circuit Architecture to Improve Classification Accuracy," in *2022 6th International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM)* (2022): 152–154.

33. D. Sierra-Sosa, M. Telahun, and A. Elmaghraby, "Tensorflow Quantum: Impacts of Quantum State Preparation on Quantum Machine Learning Performance," *IEEE Access* 8 (2020): 215246–215255.

34. D. Sierra-Sosa, S. Pal, and M. Telahun, "Data Rotation and Its Influence on Quantum Encoding," *Quantum Information Processing* 22, no. 1 (Jan. 2023): 89.

35. A. Matic, M. Monnet, J. Lorenz, B. Schachtner, and T. Messerer, "Quantum-Classical Convolutional Neural Networks in Radiological Image Classification," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)* (Los Alamitos, CA, USA: IEEE Computer Society, sep 2022), 56–66, https://doi.ieeecomputersociety.org/10.1109/QCE53715.2022.00024.

36. M. Monnet, N. Chaabani, T.-A. Dragan, B. Schachtner, and J. M. Lorenz, "Understanding the Effects of Data Encoding on Quantum-Classical Convolutional Neural Networks," (2024), https://arxiv.org/abs/2405.03027.

37. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016).

38. E. F. Combarro, A. D. M. S. González-Castillo, and A. Di Meglio, *A Practical Guide to Quantum Machine Learning and Quantum Optimization* (Packt Publishing, 2023).

39. K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The Balanced Accuracy and Its Posterior Distribution," in *2010 20th International Conference on Pattern Recognition* (2010): 3121–3124.

40. E. Farhi and H. Neven, "Classification with Quantum Neural Networks on Near Term Processors," *arXiv: Quantum Physics* (2018), https://api.semanticscholar.org/CorpusID:119037649.

41. Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, "Expressive Power of Parametrized Quantum Circuits," *Physical Review Research* 2 (Jul. 2020): 033125, https://link.aps.org/doi/10.1103/PhysRevResearch.2.033125.

42. S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms," *Advanced Quantum Technologies* 2, no. 12 (2019): 1900070, https://onlinelibrary.wiley.com/doi/abs/10.1002/qute.201900070.

43. M. Mottonen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, "Transformation of Quantum States Using Uniformly Controlled Rotations," (2004).

44. T. Hur, L. Kim, and D. K. Park, "Quantum Convolutional Neural Network for Classical Data Classification," *Quantum Machine Intelligence* 4, no. 1 (Feb. 2022) 3, https://doi.org/10.1007/s42484-021-00061-x.

45. L.-H. Gong, J.-J. Pei, T.-F. Zhang, and N.-R. Zhou, "Quantum Convolutional Neural Network Based on Variational Quantum Circuits," *Optics Communications* 550 (2024): 129993, https://www.sciencedirect.com/science/article/pii/S0030401823007411.

46. M. Weigold, J. Barzen, F. Leymann, and M. Salm, "Encoding Patterns for Quantum Algorithms," *IET Quantum Communication* 2, no. 4 (2021): 141–152, https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/qtc2.12032.

47. A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, "Data Re-Uploading for a Universal Quantum Classifier," *Quantum* 4 (2020): 226.

48. "Strongly Entangling Layers," https://docs.pennylane.ai/en/stable/code/api/pennylane.StronglyEntanglingLayers.html accessed: (March 2024).

49. S. Wold, K. Esbensen, and P. Geladi, "Principal Component Analysis," *Chemometrics and Intelligent Laboratory Systems* 2, no. 1–3 (1987): 37–52.

50. D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," (2017).

51. A. Paszke, S. Gross, S. Chintala, et al., "Automatic Differentiation in Pytorch," in *NIPS 2017 Workshop on Autodiff* (2017).

52. "Intel Xeon Gold 6134 Processor - Product Specification," https://ark.intel.com/content/www/us/en/ark/products/120493/intel-xeon-gold-6134-processor-24-75m-cache-3-20-ghz.html, accessed 25-October-2021.

Appendix A: Hamiltonian-Based Encoding (Preliminary Results)

In this appendix, we report preliminary results obtained using a Hamiltonian-based encoding strategy. In this approach, classical input features are embedded into quantum states through data-dependent Hamiltonian evolutions of the form:

$$U_{\text{emb}}(x) = \exp\left(-i\, t\, H(x)\right) \tag{A.1}$$

where t is a fixed evolution time and $H(x)$ is a Hamiltonian whose coefficients depend on the input features.

In our implementation, the Hamiltonian is defined as a sum of commuting Pauli operators,

$$H(x) = \sum_i (\omega_i x_i + b_i) Z_i \; + \; \sum_i \gamma_i \, Z_i Z_{i+1} \tag{A.2}$$

with periodic boundary conditions. All coefficients ω_i, b_i, and γ_i are trainable parameters.

All experimental settings, including datasets, training procedure, optimization strategy, and evaluation metrics, are identical to those described in the main text, ensuring a fair comparison under fixed constraints. The results obtained with Hamiltonian-based encoding for both the datasets are reported in Tables A.1, A.2, A.3, A.4, A.5 and A.6. Under the considered experimental setup, this encoding strategy does not outperform the

TABLE A.1 | Accuracy of Hamiltonian encoding models grouped by the re-uploading (RU) technique for classification on the Wine dataset.

Layers	*RU*	*Encoding*	*Accuracy (%)*	*STD (%)*
2	False	Hamiltonian	76.9444	10.9596
2	True	Hamiltonian	76.9444	10.3148
4	False	Hamiltonian	72.7778	12.6144
4	True	Hamiltonian	75.0000	10.2272
6	False	Hamiltonian	86.3889	9.0201
6	True	Hamiltonian	70.5556	8.8036
8	False	Hamiltonian	85.2778	8.3897
8	True	Hamiltonian	71.3889	7.2966
10	False	Hamiltonian	87.5000	14.5373
10	True	Hamiltonian	63.8889	9.5330

TABLE A.2 | Accuracy of Hamiltonian encoding models grouped by the re-uploading (RU) technique for classification on the Diabetes dataset.

Layers	*RU*	*Encoding*	*Accuracy (%)*	*STD (%)*
2	False	Hamiltonian	0.7149	0.0001
2	True	Hamiltonian	0.6955	0.0017
4	False	Hamiltonian	0.7123	0.0002
4	True	Hamiltonian	0.7260	0.0003
6	False	Hamiltonian	0.7162	0.0003
6	True	Hamiltonian	0.7175	0.0006
8	False	Hamiltonian	0.7175	0.0002
8	True	Hamiltonian	0.7117	0.0004
10	False	Hamiltonian	0.7149	0.0002
10	True	Hamiltonian	0.7130	0.0005

TABLE A.3 | Balanced Accuracy of Hamiltonian encoding models grouped by the re-uploading (RU) technique for classification on the Diabetes dataset.

Layers	*RU*	*Encoding*	*Accuracy (%)*	*STD (%)*
2	False	Hamiltonian	0.6429	0.0006
2	True	Hamiltonian	0.6305	0.0029
4	False	Hamiltonian	0.6554	0.0004
4	True	Hamiltonian	0.6723	0.0005
6	False	Hamiltonian	0.6593	0.0007
6	True	Hamiltonian	0.6658	0.0006
8	False	Hamiltonian	0.6530	0.0004
8	True	Hamiltonian	0.6587	0.0005
10	False	Hamiltonian	0.6561	0.0006
10	True	Hamiltonian	0.6559	0.0006

TABLE A.4 | Precision of Hamiltonian encoding models grouped by the re-uploading (RU) technique for classification on the Diabetes dataset.

Layers	*RU*	*Encoding*	*Precision (%)*	*STD (%)*
2	False	Hamiltonian	0.6572	0.0010
2	True	Hamiltonian	0.5890	0.0062
4	False	Hamiltonian	0.6226	0.0009
4	True	Hamiltonian	0.6429	0.0011
6	False	Hamiltonian	0.6283	0.0008
6	True	Hamiltonian	0.6253	0.0023
8	False	Hamiltonian	0.6430	0.0004
8	True	Hamiltonian	0.6138	0.0013
10	False	Hamiltonian	0.6312	0.0012
10	True	Hamiltonian	0.6228	0.0021

TABLE A.5 | Recall of Hamiltonian encoding models grouped by the re-uploading (RU) technique for classification on the Diabetes dataset.

Layers	*RU*	*Encoding*	*Recall (%)*	*STD (%)*
2	False	Hamiltonian	0.4019	0.0067
2	True	Hamiltonian	0.4130	0.0111
4	False	Hamiltonian	0.4648	0.0038
4	True	Hamiltonian	0.4926	0.0020
6	False	Hamiltonian	0.4685	0.0043
6	True	Hamiltonian	0.4926	0.0010
8	False	Hamiltonian	0.4370	0.0025
8	True	Hamiltonian	0.4815	0.0014
10	False	Hamiltonian	0.4593	0.0053
10	True	Hamiltonian	0.4648	0.0026

TABLE A.6 | F1-score of Hamiltonian encoding models grouped by the re-uploading (RU) technique for classification on the Diabetes dataset.

Layers	*RU*	*Encoding*	*Recall (%)*	*STD (%)*
2	False	Hamiltonian	0.4926	0.0034
2	True	Hamiltonian	0.4821	0.0099
4	False	Hamiltonian	0.5291	0.0017
4	True	Hamiltonian	0.5569	0.0013
6	False	Hamiltonian	0.5344	0.0022
6	True	Hamiltonian	0.5503	0.0011
8	False	Hamiltonian	0.5191	0.0014
8	True	Hamiltonian	0.5391	0.0011
10	False	Hamiltonian	0.5277	0.0023
10	True	Hamiltonian	0.5308	0.0018

best-performing Angle encoding configurations, although it achieves competitive performance in several settings. We emphasize that these results are preliminary and were not obtained through extensive hyperparameter optimization. A systematic and fully optimized investigation of Hamiltonian-based encoding strategies is left for future work.