Research Article

# Hybrid Quantum Neural Networks with Amplitude Encoding: Advancing Recovery Rate Predictions

**Ying Chen[1,2], Paul Griffin[3], Paolo Recchia[2], Zhou Lei[2], Hongrui Zhang[1]**

1. Mathematics, National University of Singapore, Singapore; 2. AIDF, National University of Singapore, Singapore; 3. School of Computing and Information Systems, Singapore Management University, Singapore

**Recovery rate prediction plays a pivotal role in bond investment strategies, enhancing risk assessment, optimizing portfolio allocation, improving pricing accuracy, and supporting effective credit risk management. However, forecasting faces challenges like high-dimensional features, small sample sizes, and overfitting. We propose a hybrid Quantum Machine Learning model incorporating Parameterized Quantum Circuits (PQC) within a neural network framework. PQCs inherently preserve unitarity, avoiding computationally costly orthogonality constraints, while amplitude encoding enables exponential data compression, reducing qubit requirements logarithmically. Applied to a global dataset of 1,725 observations (1996–2023), our method achieved superior accuracy (RMSE 0.228) compared to classical neural networks (0.246) and quantum models with angle encoding (0.242), with efficient computation times. This work highlights the potential of hybrid quantum-classical architectures in advancing recovery rate forecasting.**

**Corresponding author:** Paolo Recchia, paolo__re@nus.edu.sg

## 1. Introduction

Recovery rate prediction is a pivotal element of credit risk management, complementing other key metrics such as Exposure at Default (EAD) and Probability of Default (PD)[1]. While EAD and PD assess the likelihood and extent of credit losses, recovery rates uniquely quantify the proportion of funds recoverable following a default. This metric is particularly valuable for risk assessment, portfolio optimization, and pricing strategies. Despite its critical role, recovery rate prediction has not received commensurate attention in practice. A common approach is to assume a constant recovery rate,

 doi.org/10.32388/VDZ7JK 

typically around 40%, even though empirical data often exhibit a wide range from 0% to 100%, frequently bimodal near 10% and 100%[2][3][4][5]. Such oversimplifications can lead to inaccurate risk evaluations, suboptimal investment decisions, and flawed pricing models, especially in distressed or lower-rated bonds. This lack of focus partly stems from the technical challenges associated with accurate recovery rate modeling.

The technical challenges in forecasting recovery rates are substantial. High-dimensional feature spaces combined with limited datasets often lead to overfitting, a common pitfall in predictive modeling. Classical machine learning approaches have attempted to mitigate this issue through techniques such as Orthogonal Neural Networks (OrthNNs) and Unitary Neural Networks (UNNs)[6][7][8][9][10], which constrain weight matrices to orthogonal or unitary forms. These methods have demonstrated improved generalization and stability in training. However, maintaining orthogonality during gradient-based training is computationally expensive, often requiring additional steps to re-orthogonalize weights, with time complexity scaling as $O(N^3)$ for input size $N$. Such limitations underscore the need for novel approaches that can effectively address these challenges.

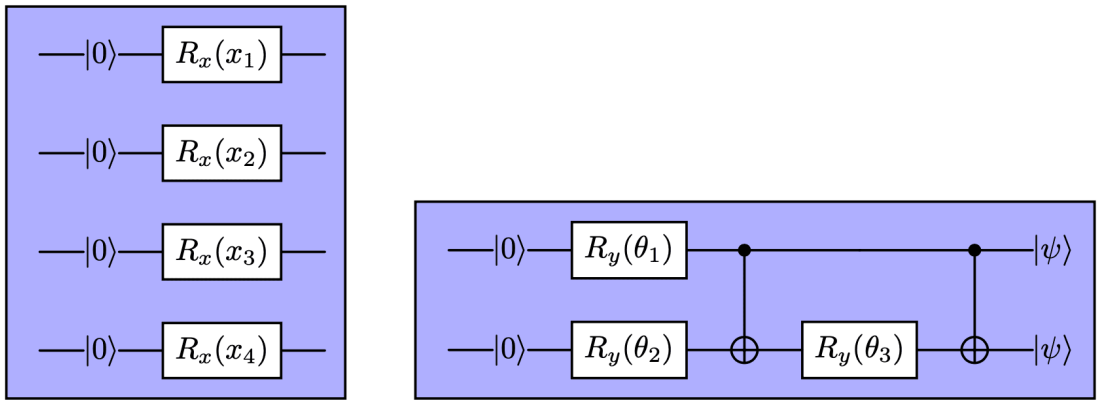
Quantum Machine Learning (QML) presents a promising alternative to addressing complex computational challenges by integrating Parameterized Quantum Circuits (PQC) as quantum nodes within neural network frameworks. PQCs, which are quantum circuits with tunable parameters, inherently preserve unitarity due to the fundamental principles of quantum mechanics. This intrinsic property eliminates the need for computationally intensive constraints required in classical neural networks to maintain orthogonality or unitarity. By embedding PQCs into neural networks, the resulting models benefit from enhanced generalization and stability, particularly for high-dimensional, small-sample-size datasets prone to overfitting[11]. This unitarity-preserving feature not only simplifies model training but also reduces computational burdens, positioning PQCs as a highly efficient and effective alternative to traditional neural network layers—offering significant potential for advancing machine learning applications.

Quantum Neural Networks (QNNs) have become a prominent tool with applications in quantitative finance among QML approaches. For instance, QNNs have been applied to portfolio optimization, where Quantum Circuit Born Machines outperform classical Restricted Boltzmann Machines[12], and to market forecasting, where Quantum Elman Neural Networks have proven effective for sequential data tasks[13]. Additionally, hybrid QNN models have shown advantages in time series forecasting

when implemented on Quantum Processing Units (QPUs)[14][15]. The Quantum Amplitude Estimation (QAE) algorithm[16], known for its quadratic speedup over classical Monte Carlo techniques[17], presents a promising alternative for probabilistic modeling tasks, such as those encountered in option pricing. Building on the foundations of QAE, other QML approaches, including Quantum Generative Adversarial Networks (qGANs)[18], have gained traction for probabilistic modeling, demonstrating their effectiveness in applications like option pricing. Furthermore,[19] introduced Variational Quantum Amplitude Estimation (VQAE), which combines classical variational optimization with QAE. This approach ensures that the circuit depth remains below a desired threshold, highlighting its potential for practical applications in financial pricing tasks. In fraud detection, QNNs have also demonstrated superior performance, achieving better precision and lower false-positive rates compared to classical methods[20][21]. Our choice of QNNs for recovery rate forecasting is motivated by their ability to model complex relationships in high-dimensional datasets and their proven versatility across various financial applications.

A critical aspect of QNN is the encoding of classical data into quantum states. A recent study by[22] utilized a QNN integrated with a classical neural network for credit scoring of Small and Medium Enterprises (SMEs), achieving comparable performance to classical models with fewer training epochs. The model employed Angle Data Encoding, which simplifies data preparation and enables shallow circuits suitable for Noise Intermediate Scale Quantum (NISQ) hardware but scales linearly with the feature space, limiting its efficiency for high-dimensional datasets. Angle encoding maps classical data to rotation angles of single-qubit gates[23][24][25][26], requiring a number of qubits proportional to the input size, namely, we need as many qubits as the input size. While this approach simplifies circuit preparation and is feasible for current noisy quantum computers, it lacks scalability. In addition to the Angle Encoding, there is another encoding technique, Amplitude Encoding[23][27][24][25]. The key advantage of the Amplitude Encoding approach is its exponential data compression compared to classical requirements, as the number of required qubits increases only logarithmically with the input size. Specifically, the number of qubits decreases from $N$ to $\log_2 N$, where $N$ indicates the number of input features. For instance, as illustrated in Figure 1a and 1b, a dataset with four features requires four qubits when using Angle Encoding. In contrast, Amplitude Encoding requires only two qubits to encode the same four features. As the number of features in the dataset increases, the advantage of Amplitude Encoding becomes even more pronounced, enabling

efficient scaling for higher-dimensional datasets. Moreover, fewer qubits lead to fewer trainable parameters in the PQC, enhancing computational efficiency.




(a) Angle Encoding. (b) Amplitude Encoding.

**Figure 1.** Two common approaches to encode an example of four classical data features $X = \{x_1, x_2, x_3, x_4\}$. In Figure 1a, the four classical features are mapped into the rotation angles of the four one-qubit $R_x$ rotation gate (Angle Encoding). In Figure 1b, the features are mapped into the amplitude of the two-qubits state $|\psi\rangle$ (Amplitude Encoding). To prepare this quantum state, one-qubit $R_y$ rotation gates and CNOT gates must be applied[28], where the angles $\theta_1, \theta_2, \theta_3$ depend on the four classical data $X = \{x_1, x_2, x_3, x_4\}$.

In this paper, we propose a hybrid Quantum Machine Learning model that integrates Parameterized Quantum Circuits (PQC) into a neural network framework. PQCs inherently preserve unitarity due to quantum mechanical principles, eliminating the need for computationally intensive orthogonality constraints. Moreover, we leverage amplitude encoding in PQCs for exponential data compression, reducing the number of required qubits logarithmically with input size. Compared to Angle Encoding, this approach minimizes trainable parameters, enhances efficiency, and maintains accuracy in high-dimensional settings.

The proposed method demonstrated superior performance using a global dataset of 1,725 observations with 256 features spanning 576 firms from 1996 to 2023. It achieved a Root Mean Square Error (RMSE) of 0.228, lower than the RMSE of 0.246 for classical Neural Networks and 0.242 for quantum models with Angle Encoding (see the Results section below for more details). Additionally, the QML model with Amplitude Encoding has fewer trainable parameters, leading to a faster training

time of 0.73 seconds per epoch compared to the 0.81 necessary for the QML with Angle Encoding. The lower qubit requirements and reduced computation time underscore the practical applicability of our method for recovery rate forecasting.

The remainder of this paper is organized as follows: Section 2 presents the data and its features. Section 3 details the hybrid QML approach and amplitude encoding. Section 4 discusses the numerical analysis, and Section 5 concludes with insights and directions for future research.

## 2. Data

We consider a dataset comprising 256 features and 1,725 observations covering 576 firms from 1996 to 2023. The data is obtained through the NRF Research Project UP5 of the National University of Singapore. The UP5 data contains Macroeconomic and market-related features obtained from FRED and Refinitiv; financial statement features of firms sourced from Bloomberg (BBG); and bond-level features, as well as firm-level or market-level credit product features provided by the Credit Research Initiative (CRI) of the National University of Singapore.

The recovery amount of each bond in this study is defined as the bond's price 30 days after the default date. This is the most common way used in the literature.[29] uses a price "roughly" 30 days after the default event. Early S&P reports use the average price 30 to 45 days post-default, while more recent S&P reports focus on exactly 30 days afterward.[30] use average prices of the first 30 default days. We follow the literature and use the 30-day period.

Table 1 summarizes the characteristics of recovery rates. It has a mean value of 48%, a median value of 42%, and a standard deviation (STD) of 0.33. Figure 2 displays the histogram of recovery rates. In the Figure, the y-axis represents the frequency of each bar, whereas the x-axis the recovery rate. It reveals a broad range of recovery rates. They are almost all distributed between 0 and 1 and occasionally exceed 1. The histogram also exhibits a bimodal pattern with primary and secondary peaks around 10% and 100% respectively. Although recovery rates tend to cluster around $40\%$[1] they exhibit significant variability, with a large standard deviation. This wide distribution highlights the limitations of assuming a fixed recovery rate (such as $40\%$) in pricing models, which oversimplifies the complex and dynamic nature of actual recovery rates and can be misleading. Assuming a fixed recovery rate leads to inaccurate risk assessments, flawed pricing strategies, and miscalculated credit risk metrics.

| | $N$ Obs. | Mean | STD | Min | 25% | 50% | 75% | Max |
|---|---|---|---|---|---|---|---|---|
| Recovery Rate | 1,725 | 0.4845 | 0.3317 | 0 | 0.1811 | 0.4170 | 0.7896 | 1.0996 |

**Table 1.** Summary Statistics of Recovery Rate

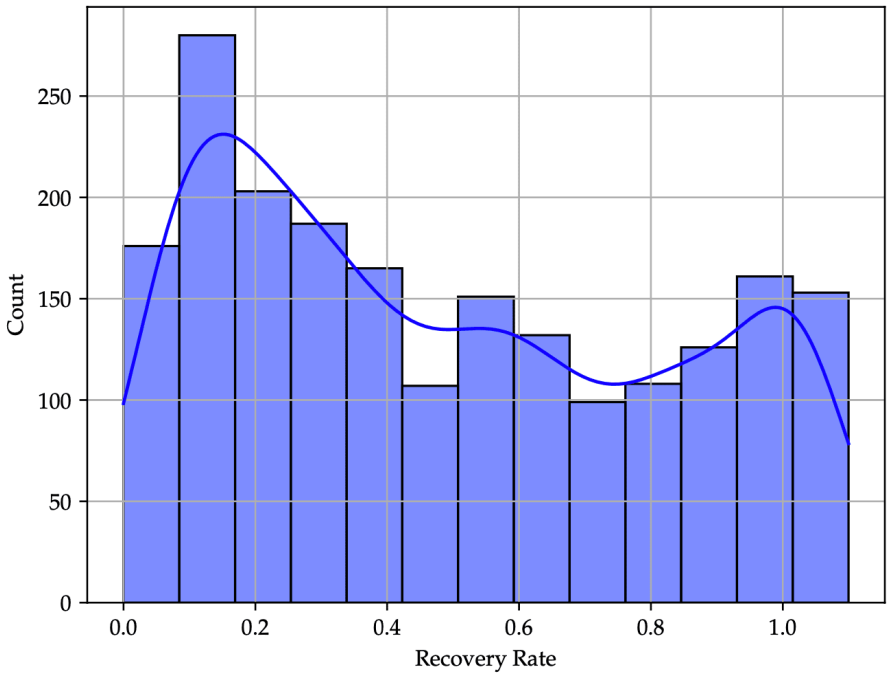



**Figure 2.** This histogram with a kernel density estimate (smooth blue line) shows the distribution of recovery rates ranging from 0 to 1.1 for defaulted bonds.

The relationship between various features and recovery rates of defaulted bonds is highly complex and nonlinear, which makes traditional linear models inadequate for accurate predictions. Previous studies have highlighted the intricate interactions between different factors influencing recovery rates. For example,[31] finds that industry-specific characteristics, such as public utilities and chemicals, influence recovery rates significantly.[32] note that macroeconomic variables like GDP growth and stock market returns have weak correlations with recovery rates, while factors like default rates, seniority, and collateral levels play a more direct role.[33] further documents that recovery rates are lower in distressed industries, emphasizing the importance of industry-specific dynamics.

Additionally, models based on mixtures of Gaussian distributions, as introduced by[34], show superior out-of-time forecasting accuracy compared to traditional parametric models. Similarly, nonparametric approaches like regression trees and support vector machines, as demonstrated by[35] and[36], outperform linear regression in terms of prediction accuracy, especially in out-of-sample scenarios. These studies indicate that nonlinear relationships, including interactions between bond characteristics, market conditions, and macroeconomic factors, are better captured by more flexible machine learning models. Thus, a neural network capable of modeling such complex and nonlinear relationships is well-suited for predicting recovery rates of defaulted bonds.

Our own data also reveals the complex nature of recovery rate prediction. For instance, when we examine the relationship between recovery rate, coupon rate, and maturity, we observe no clear linear relationship. A 3D plot of the recovery rate against these features, as shown in Figure 3, implies that the recovery rate is influenced by multiple factors in non-linear ways, further emphasizing the inadequacy of traditional linear regression models. Additionally, we observe considerable variability in average recovery rates across different years in Figure 4, which reflects the impact of changing market conditions. These patterns suggest that forecasting recovery rates using traditional statistical regression models would be ineffective. Thus, we adopt neural networks, which are better equipped to handle the nonlinearity and complexity of the data.

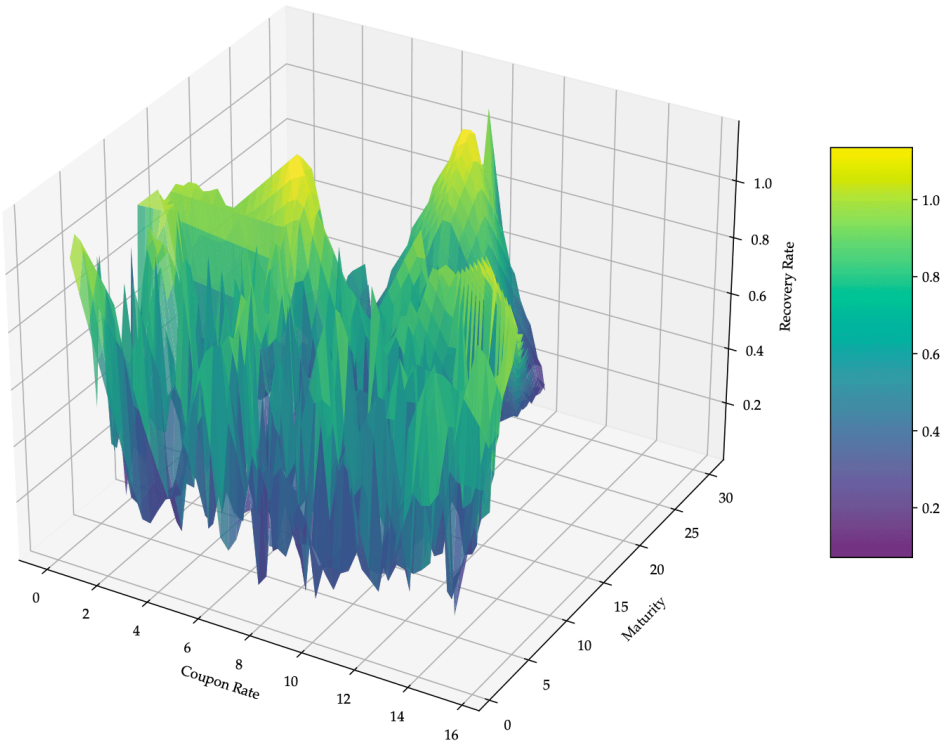



**Figure 3.** The plot visualizes how the recovery rate varies with changes in the coupon rate and maturity. The surface is generated through cubic interpolation to provide a smooth representation of the underlying trend. The color gradient of the surface indicates the magnitude of the recovery rate, with darker shades corresponding to lower recovery rates and lighter shades indicating higher recovery rates.


doi.org/10.32388/VDZ7JK

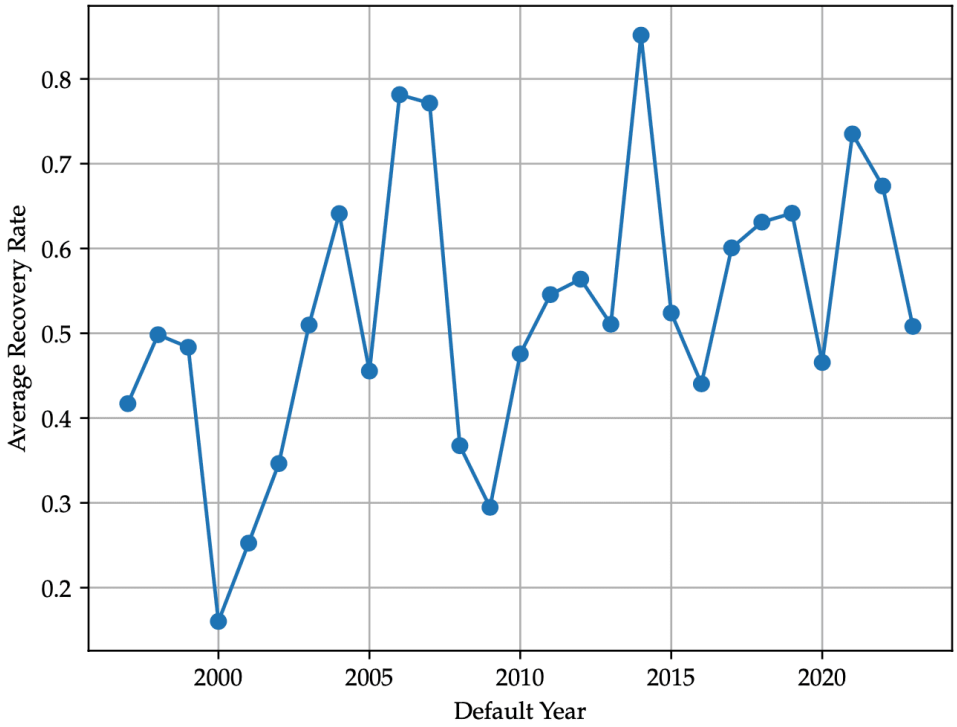



**Figure 4.** The plot illustrates the trend of recovery rates over time, with each data point representing the average recovery rate for a given default year. The x-axis shows the default year, while the y-axis represents the corresponding average recovery rate. This visualization highlights any changes in recovery rates across different years, which can provide insights into how recovery behavior evolves over time.

## 3. Methodology

Our Quantum Machine Learning (QML) model integrates a classical master layer and a Quantum Neural Network (QNN) to predict recovery rates, similar to the QML frameworks in[22]. The classical master layer comprises an input layer and a hidden layer of equal size, using a LeakyReLU activation function. This architecture offers two advantages. First, it extracts meaningful internal representations from high-dimensional, redundant input data. Second, it introduces non-linearity, enabling the model to capture complex relationships. Building on this classical layer, the QNN combines a Parameterized Quantum Circuit (PQC) and quantum data encoding to process the data.

### *3.1. Parameterized Quantum Circuit (PQC)*

The PQC is a quantum circuit with adjustable parameters embedded in the rotation angles of single-qubit gates. To enhance expressiveness—the ability to represent diverse quantum states and span the Hilbert space[37]—entanglement between qubits is introduced using controlled gates, typically CNOT gates. A common PQC consists of:

1. A layer of $n$ single-qubit rotation gates, $G(\alpha, \beta, \gamma)$, applied to each qubit. The angles $\alpha, \beta, \gamma$ are the trainable parameters of the PQC.
2. A layer of $n$ two-qubits controlled gates to introduce entanglement. Specifically, we employ the CNOT gate with a range of one, where the $i$th qubit acts as the control and is connected to its adjacent $(i+1)$th qubit, which serves as the target.

This configuration, referred to as a strongly entangling circuit[11], provides the PQC with entangling power while maintaining a manageable number of parameters. Specifically, it requires $O(3n)$ trainable parameters for $n$ qubits. After the PQC computation, the expectation values of $Z$ Pauli observables are measured and passed to the classical output layer. A classical optimizer minimizes the loss function, Root Mean Squared Error (RMSE) in this case, and updates the network parameters via backpropagation.

### *3.2. Amplitude Encoding*

Before the PQC processes input, classical data is encoded into quantum states, represented as $|\psi\rangle$ (Figure 5). For the high-dimensional feature space in recovery rate prediction, we adopt Amplitude Data Encoding[28][23][11]. This technique maps $2^n$ classical features to the amplitudes of an $n$-qubit quantum state:

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x| \tag{1}$$

where $\alpha_x$ are normalized amplitudes, calculated from the input data, satisfying:

$$\sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1. \tag{2}$$

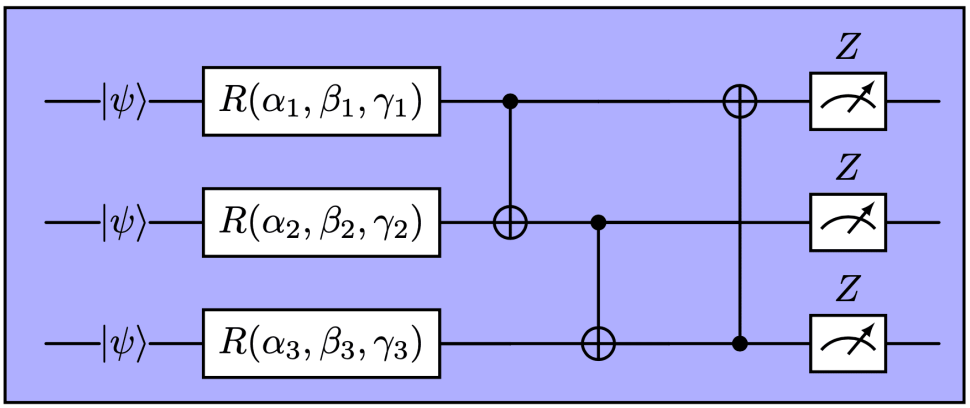



**Figure 5.** The Strongly Entangling Layer PQC (Parameterized Quantum Circuit) begins with the quantum state input, denoted as $|\psi\rangle$. Each rotation gate in the circuit is represented as $R(\alpha_i, \beta_i, \gamma_i)$, where $\alpha_i$, $\beta_i$, and $\gamma_i$ are the rotation angles around the $X$, $Y$, and $Z$ axes of the Bloch sphere, respectively. These angles are the trainable parameters of the PQC.

A key advantage of Amplitude Encoding is its exponential compression of input data. The number of qubits required grows only logarithmically with the number of features, resulting in fewer trainable parameters in the PQC, improving the model's scalability and efficiency. For instance, the number of trainable parameters in the strongly entangling PQC used in this study scales as $O(3 \log_2 N)$, making it well-suited for handling high-dimensional data.

Though some limitations due to noise and decoherence might affect the computation in real quantum hardware, in this work, we train the QML model in a fault-tolerant quantum simulator where decoherence or gate errors are not concerns. The complete PQC structure and data encoding process are illustrated in Figures 5 and 6.


doi.org/10.32388/VDZ7JK

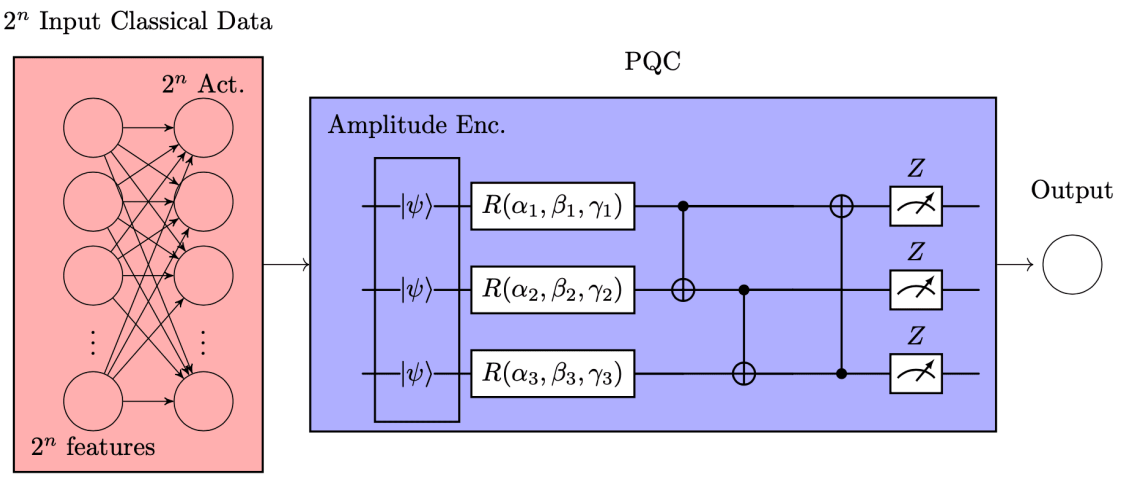



**Figure 6.** The QML model with Amplitude Encoding. We encode a set of $N = 2^n$ classical data into the amplitude of the input quantum state denoted as $|\psi\rangle$. After the application of the Strongly Entangling PQC, a measurement is performed. These measurement results are then sent to the classical output layer and post-processed in the classical optimizer.

### *3.3. Alternative Quantum and Classic Model*

To comprehensively evaluate the performance of our proposed QML model, we compare it against two alternative models: a classical feedforward neural network (FNN) and another QML model utilizing Angle Encoding, inspired by [22].

Angle Encoding[23][24][25][26] maps classical data onto the rotation angles of single-qubit gates. This method requires one qubit per input feature, making it computationally demanding and impractical for high-dimensional datasets like those used in our recovery rate prediction due to the limited availability of logical qubits in current quantum hardware and classical simulator. To address this challenge, we follow the approach outlined in[22], incorporating a classical preprocessing layer (called auxiliary layer hereafter) to reduce the dimensionality of the input data. This auxiliary layer extracts key features and ensures that the number of features aligns with the number of available qubits. The outputs of this classical layer are then encoded into the quantum circuit using Angle Encoding. This architecture, illustrated in Figure 7, enables scalability by allowing flexibility in selecting the number of qubits while maintaining the model's expressiveness. The number of trainable parameters in the additional auxiliary layer and the PQC scales as $O(3Nn)$, where $N$ is the number of input features, and $n$ is the number of selected qubits.


doi.org/10.32388/VDZ7JK

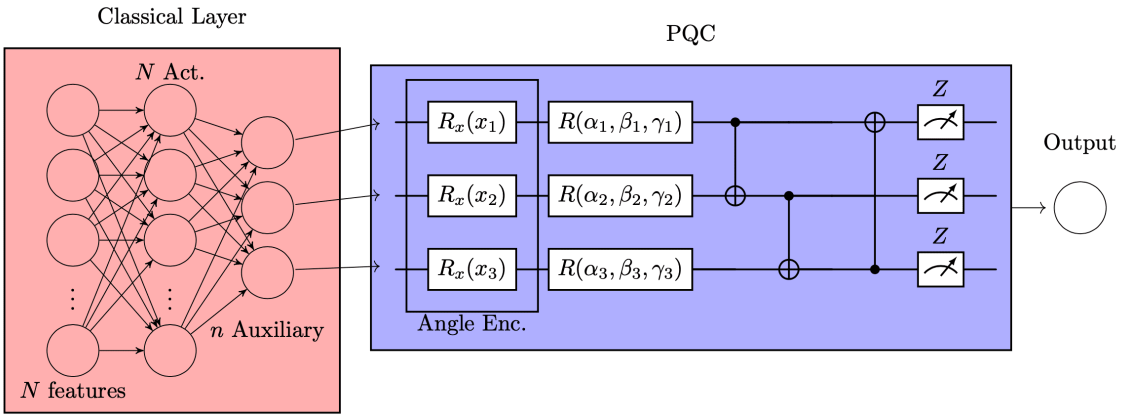



**Figure 7.** The QML model with Angle Encoding. Starting from a set of $N$ input features, we introduce an auxiliary layer to reduce the number of inputs as the amount of $n$ qubits. The outputs of the auxiliary layer are classical data encoded into the angle of the single qubit rotation gate $R_x(x_i)$. After the application of the Strongly Entangling PQC with Angle Encoding, a measurement is performed. These measurement results are then sent to the classical output layer and post-processed in the classical optimizer.

The classical feedforward neural network (FNN) serves as another benchmark for comparison. In the FNN, a hidden layer is appended to the master input layer, with the number of hidden nodes carefully chosen to ensure a comparable number of trainable parameters to the QML models. This setup not only provides a fair basis for comparison but also allows us to assess the effectiveness of the classical model and its susceptibility to overfitting. By tuning the number of hidden nodes, we balance the trade-off between model complexity and predictive performance.

Through these comparisons, we aim to evaluate the strengths and limitations of our QML model relative to classical neural networks and alternative quantum approaches, particularly in handling the intricate, high-dimensional relationships present in recovery rate prediction.

## 4. Results

In this section, we present the results produced by our models, based on the methodologies and parameter settings outlined in Sections 3 and 2. These findings offer insights into the predictive performance of both classical and quantum machine learning approaches on the chosen dataset. The evaluation of the models is conducted using standard metrics, such as RMSE calculated through k-fold cross-validation.

### *4.1. Parameter Settings and Experimental Setup*

In our proposed QML architecture, the number of qubits is fixed, with 256 classical features encoded into 8 qubits. While additional qubits could theoretically be utilized with Amplitude Encoding, doing so introduces added complexities and challenges. These include an increase in trainable parameters, the need to handle padding during initialization to accommodate the extra qubits, deeper circuits for state preparation, and potential redundancy in the input data. In this study, since the simplest configuration with 256 features encoded in 8 qubits yielded satisfactory results, we opted to focus on this straightforward setup, leaving the exploration of more complex configurations for future research.

The parameters used in the regression models are optimized to minimize RMSE on the training data selected in a cross-validation setting. The optimization process utilized the Adam optimizer[38], with specific hyperparameters such as learning rate and batch size detailed in Table 2. These parameters were selected through a systematic grid search to ensure optimal model performance. Table 2 also lists the computational resources used for training and evaluation, including both classical and quantum setups.

Deep learning practitioners commonly utilize neural network models optimized with the Adam optimizer. These models have well-established, high-performing implementations across various frameworks. For our implementation, we employ Python (version 3.12.3) and the PyTorch framework (version 2.4.1) with CUDA (version 12.1) to enable GPU acceleration. All experiments presented in the following sections were conducted on an NVIDIA GeForce RTX™ 4050 Laptop GPU.

The quantum machine learning (QML) models were implemented using PennyLane (version 0.38.0), an open-source framework for quantum programming and differentiable PQCs. All QML models with PQCs were executed on a state-vector quantum simulator. Specifically, the built-in PennyLane device called `default.qubit`[39][40]. The `default.qubit` device, written in Python with Autograd and PyTorch backends, simulates quantum operations and performs measurements on quantum systems using a classical CPU. In particular, we conducted our quantum experiments on the 13th Gen Intel® Core™ i9-13900H CPU.

It is worth noting that alternative quantum devices could be employed[39], including those with GPU acceleration, tensor network implementations, or density matrix simulators for noisy environments. However, our experiments focused on fault-tolerant PQCs with a limited number of qubits (no more

than 14). In this context, the `default.qubit` state-vector simulator proved to be the most efficient and effective choice.

Regarding gradient computation for PQC parameters, all quantum experiments were performed on a classical computer, allowing gradient calculation via automatic differentiation and backpropagation. This was achieved with the built-in functionality of the PennyLane `default.qubit` simulator. It is important to emphasize that backpropagation is not feasible on real quantum hardware, where alternative methods, such as parameter-shift or adjoint differentiation[41][42], must be used.

Table 2 summarizes the devices used for training the classical and quantum layers together with the hyperparameters of the Adam optimizer.

| Layer | Device | Gradient | Optimizer | Learning Rate | Batch size |
|---|---|---|---|---|---|
| Classical | GPU | Backpropagation | Adam | $1\times10^{-3}$ | 64 |
| Quantum (PQC) | `default.qubit` state vector simulator on CPU | Backpropagation | Adam | $1\times10^{-3}$ | 64 |

**Table 2.** Specification of the devices, the gradient calculation methods, the optimizer, and the hyperparameters for the classical and quantum layer.

## *4.2. Model Comparison and Evaluation*

In this section, we evaluate the effectiveness of our proposed QML model using Amplitude Encoding (QML Amp), comparing it against a QML model with Angle Encoding (QML Ang) based on[22] and a classical feedforward neural network (FNN). The specifications of the models' hyperparameter and architecture are reported in Table 3.

| Model | Input Layer | I Hidden Layer | Auxiliary Layer | II Hidden Layer |
|---|---|---|---|---|
| FNN | 256 | 256 LeakyRelu with slope = -0.3 | No | 8 LeakyRelu with slope = -0.3 |
| QML Ang | 256 | 256 LeakyRelu with slope = -0.3 | 8 | 8 Qubits Strongly Entangling PQC |
| QML Amp | 256 | 256 LeakyRelu with slope = -0.3 | No | 8 Qubits Strongly Entangling PQC |

**Table 3.** The specification of the model's architecture.

To benchmark the proposed models, we conduct k-fold cross-validation by dividing the dataset into four folds, each consisting of 75% training data and 25% test data. To ensure a fair comparison, all models and experiments are executed on the same four-fold splits for a fixed number of one hundred epochs. During training, we record the RMSE calculated on the test data of each fold. To achieve comparable trainable parameter counts across models, the number of qubits in the QML Ang the number of hidden nodes in the classical FNN are selected to approximate the parameter count of the QML Amp. A detailed discussion of how the performance of the FNN and the QML Ang evolves as the number of hidden nodes (for the FNN) and qubits (for the QML model) increases is provided in Appendices A and B.

Figure 8 presents the average RMSE (solid line) and the standard deviation of RMSE values (shaded area) achieved by the proposed models. Table 4 reports the best average RMSE on test data achieved during training, the average RMSE standard deviation, alongside the average execution time per epoch.

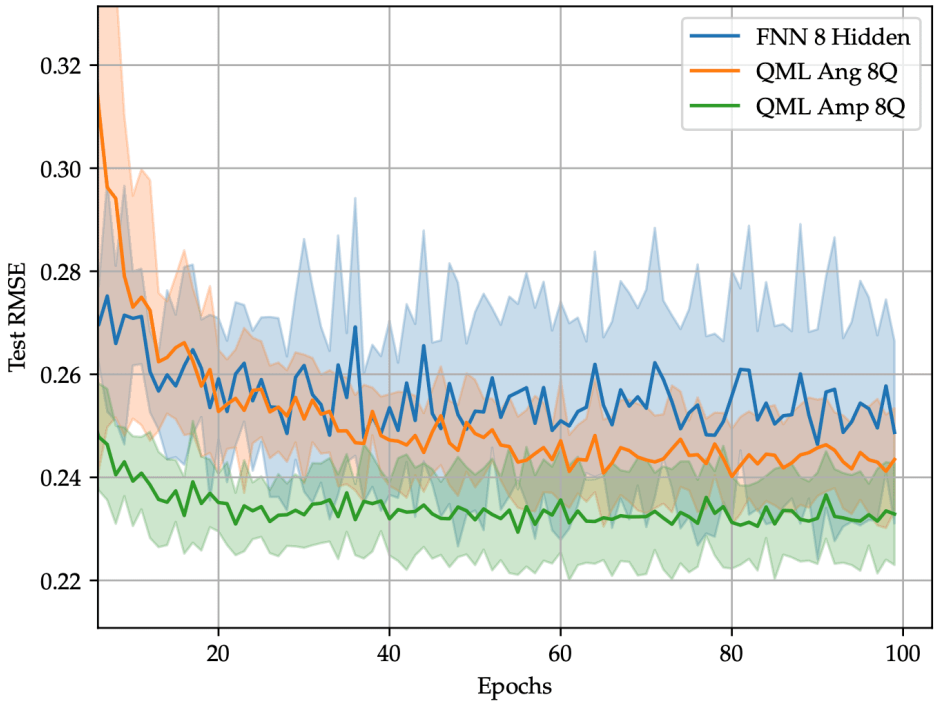



**Figure 8.** Average test RMSE and standard deviation (shaded area) calculated over four different cross-validation folds for the FNN, QML Ang, and QML Amp.

| Model | # Parameters | Best Average RMSE | At epoch # | Average STD | Average Time per Epoch (s) |
|---|---|---|---|---|---|
| FNN | 67,857 | 0.246 | 90 | 0.018 | 0.03 |
| QML Ang | 67,873 | 0.242 | 78 | 0.009 | 0.81 |
| **QML Amp** | **65,825** | **0.228** | **55** | **0.008** | **0.73** |

**Table 4.** Summary of the trainable parameters and key outputs for the selected FNN and QML models, including the best average RMSE, the epoch at which the lowest average RMSE is achieved, the average standard deviation of the RMSE, and the average execution time per epoch. All values are calculated over four cross-validation folds with one hundred epochs.

The results, summarized in Figure 8 and Tables 4, provide several key insights. First, the QML models


doi.org/10.32388/VDZ7JK

(using Amplitude Encoding and Angle Encoding) outperform the classical FNN model, showcasing superior generalizability. They achieve lower average RMSE on test data with fewer epochs (see columns two and three of Table 4). Moreover, their stability is evidenced by the smaller standard deviation observed across the four cross-validation folds (column four of Table 4). These findings underscore the effectiveness of replacing a classical layer with a parameterized quantum circuit (PQC) in reducing overfitting, particularly in tasks such as recovery rate prediction.

Second, the QML model with Amplitude Encoding (QML Amp) demonstrates the best overall performance. Despite its simplicity and minimal number of trainable parameters, it achieves the lowest average RMSE, requires the fewest epochs, and exhibits the smallest average standard deviation.

When comparing the QML models, the Amplitude Encoding model shows more remarkable advantages over Angle Encoding in both stability and accuracy, as detailed in Table 4 and Figure 8. While both models outperform the classical FNN with a comparable number of trainable parameters, the Amplitude Encoding model consistently achieves lower RMSE values and smaller standard deviations across multiple runs. This superior performance highlights the effectiveness of Amplitude Encoding in capturing and utilizing input information more efficiently. Its enhanced stability and accuracy can be attributed to its ability to represent input data compactly without requiring additional auxiliary layers to reduce input dimensions. By avoiding this added complexity, the Amplitude Encoding model minimizes the risk of overfitting, particularly with limited data. In contrast, the Angle Encoding model relies on an auxiliary classical layer to align the input size with the number of PQC qubits, which can introduce unnecessary complexity and negatively impact performance. Furthermore, the compactness of Amplitude Encoding enables it to encode a larger amount of information into the quantum state using fewer qubits, giving it a significant advantage over Angle Encoding, which scales less efficiently. These attributes make Amplitude Encoding a compelling choice for tasks demanding high accuracy and strong generalization, as reflected in our results.

Finally, we evaluate the computational efficiency of the proposed models. Given the experimental setup—using GPUs for the FNN and a CPU-based quantum simulator for the QML models—direct comparisons of execution times between classical and quantum models are not meaningful. Instead, we focus on the relative performance of the QML models. As shown in the seventh column of Table 4, the QML model with Amplitude Encoding exhibits slightly better time efficiency than the Angle Encoding model, primarily due to the absence of the auxiliary classical layer in the former.

## 5. Conclusion and Discussion

In this work, we proposed and evaluated a quantum machine learning (QML) model with Amplitude Encoding for recovery rate prediction, comparing its performance with a QML model using Angle Encoding and a classical feedforward neural network (FNN). Our experiments demonstrate that the QML model with Amplitude Encoding outperforms both the QML model with Angle Encoding and the classical FNN in terms of accuracy, stability, and generalization. The Amplitude Encoding model's lower RMSE and smaller standard deviation across multiple training runs highlight its superior ability to avoid overfitting, making it a promising approach for prediction tasks involving complex data.

Although the QML model with Angle Encoding shows improvements over the classical FNN, this is not as significant as the one achieved by the QML with Amplitude encoding. The performance of the QML Ang is hindered by the added complexity of the auxiliary classical layer, which increases the risk of overfitting. In contrast, the simplicity of the Amplitude Encoding model, with fewer layers and parameters, allows it to achieve better results with a more stable training process.

From a computational perspective, our results indicate that the QML model with Amplitude Encoding is slightly more time-efficient than the Angle Encoding model in simulation. However, the practical implementation of Amplitude Encoding on real quantum hardware faces challenges due to the need for deeper quantum circuits, which are more susceptible to noise and decoherence. These factors highlight the importance of ongoing advancements in quantum hardware to fully leverage the potential of Amplitude Encoding in practical applications. Despite these challenges, the superior simulation performance and scalability of Amplitude Encoding underscore its promise as a robust approach for quantum machine learning tasks.

Overall, our study provides valuable insights into the potential of quantum-enhanced machine learning and demonstrates the advantages of Amplitude Encoding for certain prediction tasks. In future work, we aim to explore two key directions. First, to improve the scalability and robustness of quantum models, adding more qubits in QML with Amplitude Encoding can enable the handling of exponentially higher-dimensional datasets. However, when the dataset dimension is not a perfect power of two, challenges such as data padding with appropriate schemes must be addressed. Second, regarding the practical deployment of quantum hardware, while it has the potential to achieve more time-efficient training compared to classical models, it is crucial to study the impact of noise and

decoherence. Understanding and potentially leveraging these phenomena could lead to more reliable and efficient training processes.

## Appendix A. The Classical FNNs and overfitting

In this section, we evaluate the performance of various FNN models, by increasing the number of hidden nodes. Figure A1 illustrates the average RMSE and standard deviation on test data across ten experiments for two extreme configurations: one with 8 hidden nodes and another with 8192 hidden nodes. Additionally, the table presents the lowest RMSE and corresponding standard deviation observed over ten experiments for each configuration.

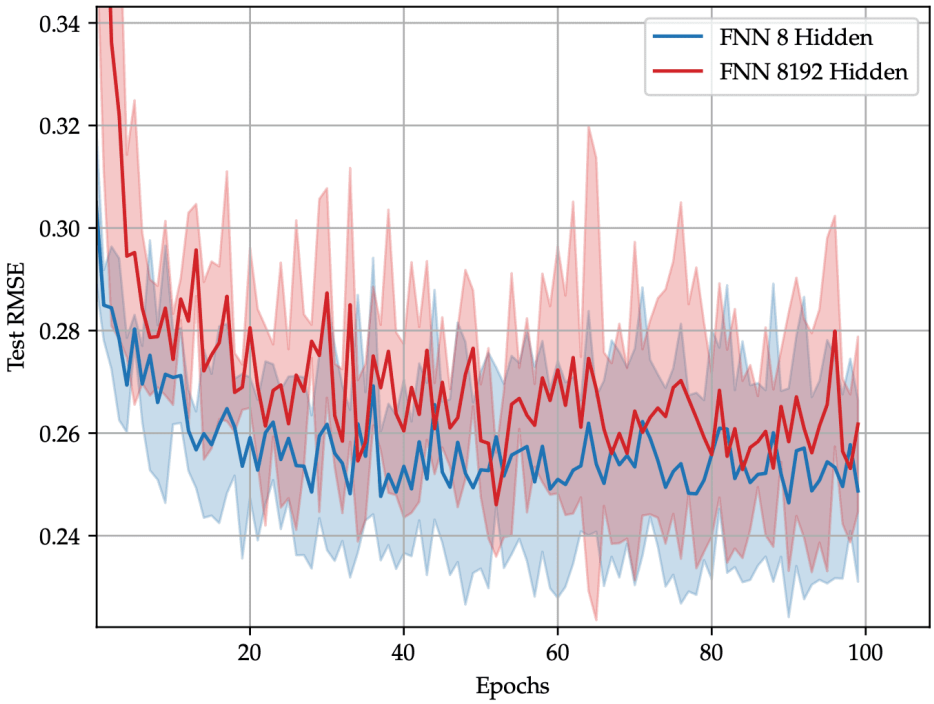



**Figure A1.** The average test RMSE and STD calculated over ten different experiments of the two extreme FNN configuration; specifically the one with 8 hidden nodes and the one with 8192.


doi.org/10.32388/VDZ7JK

| $n$ Hidden | # Parameters | Best Average RMSE # | Average STD |
|---|---|---|---|
| 8 | 67,857 | 0.246 | 0.018 |
| 16 | 69,921 | 0.244 | 0.017 |
| 128 | 98,817 | 0.244 | 0.023 |
| 512 | 197,889 | 0.246 | 0.021 |
| 2048 | 594,177 | 0.246 | 0.024 |
| 8192 | 2,179,329 | 0.250 | 0.031 |

**Table A1.** The total number of trainable parameters and some relevant outputs related to the different FNN configurations, specifically the best average RMSE, and the best STD calculated over the four cross-validation folds with one hundred epochs.

From Figure A1 and Table A1, it is evident that increasing the number of hidden nodes does not improve the FNN's prediction accuracy or stability, as indicated by both the RMSE and standard deviation. In fact, further increasing the number of hidden nodes worsens both prediction performance and stability. As discussed in Section 2, the recovery rate prediction problem is particularly susceptible to overfitting, and adding more complexity to the model is counterproductive.

## Appendix B. Performance increasing the number of qubits in the QML model with Angle Encoding

In the architecture proposed in Section 3, the number of qubits can be adjusted to identify the optimal configuration for the QML model using Angle Encoding. Figure B2 presents the average RMSE and standard deviation on the test data for two configurations: one with six qubits and another with fourteen qubits in the PQC. As in the previous section, Table B2 summarizes the best RMSE and corresponding standard deviation values observed across ten experiments for different qubit configurations.

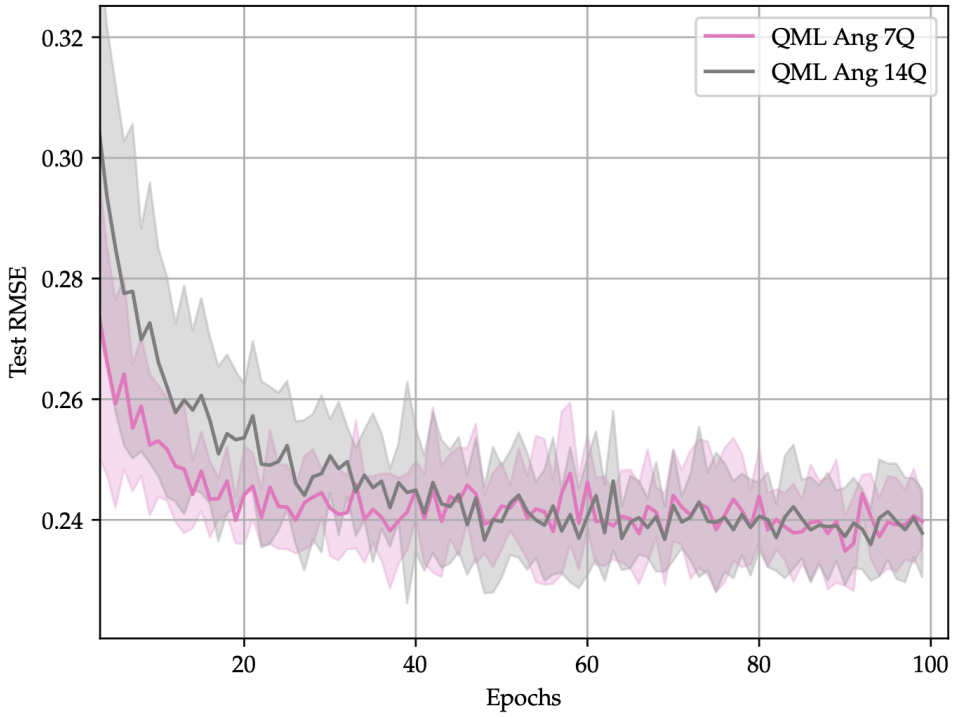



**Figure B2.** The average test RMSE and STD calculated over four corss-validation folds of the QML with Angle Encoding for seven and fourteen qubits in the PQC.

| $n$ **Qubits** | # **Parameters** | **Best Average RMSE** | **Average STD** |
|---|---|---|---|
| 6 | 67,353 | 0.241 | 0.012 |
| 7 | 67,613 | 0.241 | 0.009 |
| 8 | 67,873 | 0.242 | 0.009 |
| 10 | 68,393 | 0.242 | 0.011 |
| 12 | 68,913 | 0.242 | 0.010 |
| 14 | 69,433 | 0.242 | 0.013 |

**Table B2.** The total number of trainable parameters and some relevant outputs related to the QML models with angle encoding and different numbers of qubits in the PQC (specifically the best average RMS and the average STD calculated over four cross-validation folds with one hundred epochs).

From Table B2 and Figure B2, we can draw two key conclusions. First, the QML model with Angle Encoding does not demonstrate improved predictive performance with an increasing number of qubits. While there is a slight improvement with six and seven qubits, this effect saturates after eight qubits. Second, a comparison of Tables A1 and B2 reveals that the QML model with Angle Encoding offers a slight improvement over the classical FNN. In all reported cases, the QML Ang outperforms the classical FNN model in both stability (lower standard deviation) and effectiveness (lower average RMSE).

As a final remark, Figure B3 shows that the average execution time per epoch of the QML model with Angle Encoding increases exponentially with the number of qubits, consistent with the resource demands of the state-vector `default.qubit` simulator. For illustration purposes, in the same Figure B3, we also mark the execution time of the QML with Amplitude Encoding proposed in the current work, highlighting the slight advantage performance.

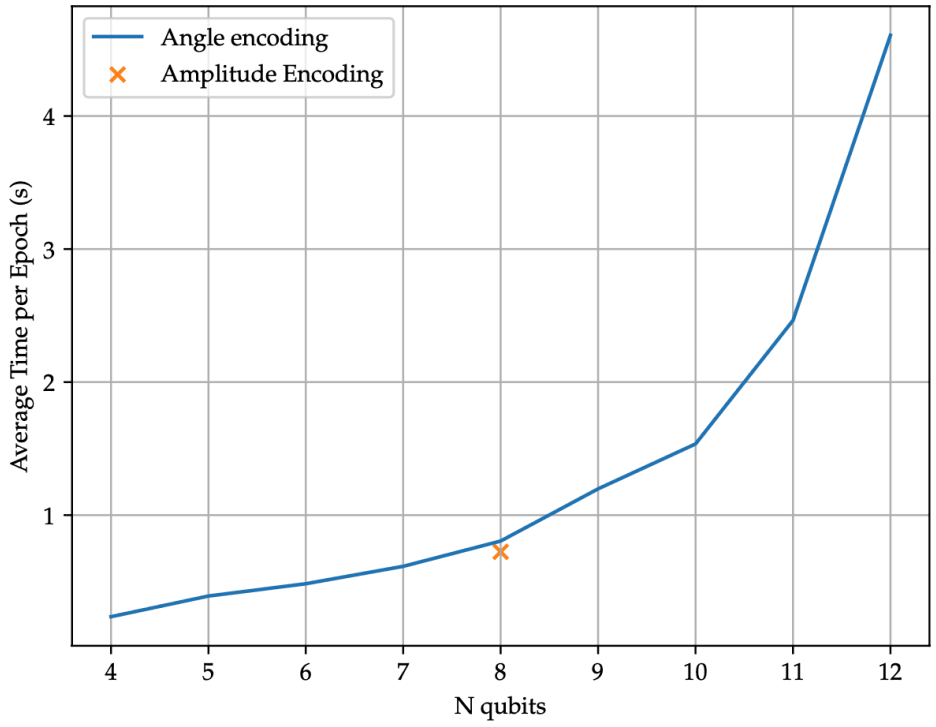



**Figure B3.** Average execution time per epoch in the QML with angle encoding (blue line) and amplitude encoding (orange marker).


doi.org/10.32388/VDZ7JK

## Statements and Declarations


### *Acknowledgements*

The authors gratefully acknowledge the support of Whitespace under grant number A-0003504-17-00 and the Quantum Engineering Program (QEP) under grant number A-8000339-01-00.


### *Data Availability*

The codes used in this study are available from the corresponding author upon reasonable request.

The data used to train the models presented in this work were obtained through the NRF Research Project UP5 at the National University of Singapore and are subject to a nondisclosure agreement.

## Footnotes

[1] [30] find an average recovery rate of 38.6% for 2002-2010, while The average ultimate recovery rate for US corporate bonds reported by [43] is 37% for defaults between 1987 and 2006. In general, market participants tend to assume constant recovery rates of around 40% within the pricing models [44].

## References


1. ^*Basel Committee on Banking Supervision. Basel Framework - Calculation of RWA for Credit Risk - IRB Approach: Risk Components. Basel, Switzerland: Bank for International Settlements; December 2023. Available from: https://www.bis.org/basel_framework/chapter/CRE/32.htm?inforce=20230101&published=20200327.*
2. ^*Pykthin M (2003). "Unexpected recovery risk". Risk. 16: 74–78.*
3. ^*Andersen L, Sidenius J (2004). "Extensions to the Gaussian Copula: Random Recovery and Random Factor Loadings". Journal of Credit Risk. 1: 29–70. doi:10.21314/JCR.2004.001.*
4. ^*Berd A (2005). "Recovery swaps". Journal of Credit Risk. 1: 61–70. doi:10.21314/JCR.2005.008.*
5. ^*Gambetti P, Gauthier G, Vrins F. Stochastic recovery rate: Impact of pricing measure's choice and financial consequences on single-name products. In: Mili M, Medina RS, Pietro FD, editors. New Methods in Fixed Income Analysis. Cham: Springer; 2018. p. 181–203. doi:10.1007/978-3-319-96032-3_9.*
6. ^*Saxe AM, McClelland JL, Ganguli S (2013). "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". arXiv preprint arXiv:1312.6120. Available from: https://arxiv.org/abs/1312.61*

20.

7. ^Le QV, Jaitly N, Hinton GE (2015). "A simple way to initialize recurrent networks of rectified linear unit s". arXiv preprint arXiv:1504.00941. Available from: https://arxiv.org/abs/1504.00941.
8. ^Henaff M, Szlam A, LeCun Y. "Recurrent orthogonal networks and long-memory tasks". In: Internatio nal Conference on Machine Learning. PMLR; 2016. p. 2034–2042.
9. ^Li S, Xu W, Liu S, Lin X, Zhang H (2019). "Orthogonal deep neural networks". IEEE Transactions on Pat tern Analysis and Machine Intelligence. 43 (4): 1352–1368.
10. ^Mashhadi PS, Nowaczyk S, Pashami S (2021). "Parallel orthogonal deep neural network". Neural Net works. 140: 167–183.
11. a, b, cSchuld M, Bocharov A, Svore KM, Wiebe N (2020). "Circuit-centric quantum classifiers". Physical R eview A. 101 (3): 032308.
12. ^Alcazar J, Leyton-Ortega V, Perdomo-Ortiz A (2020). "Classical versus quantum models in machine le arning: insights from a finance application". Machine Learning: Science and Technology. 1 (3): 035003.
13. ^Liu G, Ma W (2022). "A quantum artificial neural network for stock closing price prediction". Informati on Sciences. 598: 75–85.
14. ^Emmanoulopoulos D, Dimoska S (2022). "Quantum machine learning in finance: Time series forecasti ng". arXiv preprint arXiv:2202.00599. Available from: https://arxiv.org/abs/2202.00599.
15. ^Rivera-Ruiz MA, Mendez-Vazquez A, López-Romero JM. "Time series forecasting with quantum mac hine learning architectures". In: Mexican International Conference on Artificial Intelligence. Springer; 2 022. p. 66–82.
16. ^Brassard G, Hoyer P, Mosca M, Tapp A (2002). "Quantum amplitude amplification and estimation". Co ntemporary Mathematics. 305: 53–74.
17. ^Montanaro A (2015). "Quantum speedup of Monte Carlo methods". Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences. 471 (2181): 20150301.
18. ^Zoufal C, Lucchi A, Woerner S (2019). "Quantum generative adversarial networks for learning and load ing random distributions". npj Quantum Information. 5 (1): 103.
19. ^Plekhanov K, Rosenkranz M, Fiorentini M, Lubasch M (2022). "Variational quantum amplitude estima tion". Quantum. 6: 670.
20. ^Kyriienko O, Magnusson EB (2022). "Unsupervised quantum machine learning for fraud detection". ar Xiv preprint arXiv:2208.01203.

21. ^Tekkali CG, Natarajan K. "Smart payment fraud detection using qml--a major challenge". In: 2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS). IEEE; 2023. p. 523–526.
22. a, b, c, d, eSchetakis N, Aghamalyan D, Boguslavsky M, Rees A, Rakotomalala M, Griffin PR (2024). "Quantum machine learning for credit scoring". Mathematics. 12 (9): 1391.
23. a, b, c, dSchuld M. Supervised Learning with Quantum Computers. Springer; 2018.
24. a, b, cRanga D, Rana A, Prajapat S, Kumar P, Kumar K, Vasilakos AV. "Quantum machine learning: Exploring the role of data encoding techniques, challenges, and future directions". Mathematics. 12(21):3318 (2024).
25. a, b, cRath M, Date H (2024). "Quantum data encoding: a comparative analysis of classical-to-quantum mapping techniques and their impact on machine learning accuracy". EPJ Quantum Technology. 11 (1): 72.
26. a, bGong L-H, Pei J-J, Zhang T-F, Zhou N-R (2024). "Quantum convolutional neural network based on variational quantum circuits". Optics Communications. 550: 129993.
27. ^Benedetti M, Lloyd S, Sack S, Fiorentini M (2019). "Parameterized quantum circuits as machine learning models". Quantum Science and Technology. 4 (4): 043001.
28. a, bMottonen M, Vartiainen JJ, Bergholm V, Salomaa MM (2004). "Transformation of quantum states using uniformly controlled rotations". arXiv preprint quant-ph/0407010.
29. ^Moody's. Sovereign Default and Recovery Rates, 1983-2010. Special Comment, Moody's Investors Service; 2011.
30. a, bJankowitsch R, Nagler F, Subrahmanyam MG (2014). "The determinants of recovery rates in the US corporate bond market". Journal of Financial Economics. 114 (1): 155–177.
31. ^Altman EI, Kishore VM (1996). "Defaulted debt recovery rates of public utilities and industrial firms". Journal of Applied Corporate Finance. 9 (4): 88–98.
32. ^Altman EI, Brady B, Resti A, Sironi A (2005). "The link between default and recovery rates: Theory, empirical evidence, and implications". Journal of Business. 78 (6): 1807–1834.
33. ^Acharya VV, Bharath ST, Srinivasan A (2007). "Does industry-wide distress affect defaulted firms? Evidence from the credit market". Journal of Financial Economics. 85 (2): 452–479.
34. ^Altman EI, Kalotay EA (2014). "A mixture model for estimating recovery rates of defaulted debt". Journal of Fixed Income. 23 (2): 40–50.

35. ^Qi B, Zhao W (2011). "A support vector machine approach for predicting bond recovery rates". Quantitative Finance. 11 (3): 399–413.
36. ^Nazemi A, Fabozzi FJ (2018). "Macroeconomic factors in predicting corporate bond recovery rates: A LASSO approach". Journal of Fixed Income. 27 (4): 55–74.
37. ^Sim S, Johnson PD, Aspuru-Guzik A (2019). "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms". Advanced Quantum Technologies. 2 (12): 1900070.
38. ^Kingma DP (2014). "Adam: A method for stochastic optimization". arXiv preprint arXiv:1412.6980. Available from: https://arxiv.org/abs/1412.6980.
39. a, bPennylane. "Pennylane devices and ecosystem". Available from: https://pennylane.ai/plugins.
40. ^Pennylane. "Pennylane default.qubit". Available from: https://docs.pennylane.ai/en/stable/code/api/pennylane.devices.default_qubit.html.
41. ^Mitarai K, Negoro M, Kitagawa M, Fujii K (2018). "Quantum circuit learning". Physical Review A. 98 (3): 032309.
42. ^Jones T, Gacon J (2020). "Efficient calculation of gradients in classical simulations of variational quantum algorithms". arXiv preprint arXiv:2009.02823. Available from: https://arxiv.org/abs/2009.02823.
43. ^Cantor R, Emery K, Keisman D, Ou S (2007). Moody's ultimate recovery database: Special report. Moody's Investor Service.
44. ^Das SR, Hanouna P (2009). Implied recovery. J Econ Dyn Control. 33(11):1837-57.

## Declarations

**Funding:** The authors gratefully acknowledge the support of Whitespace under grant number A-0003504-17-00 and the Quantum Engineering Program (QEP) under grant number A-8000339-01-00.

**Potential competing interests:** No potential competing interests to declare.