

Quantum reservoir computing for realized volatility forecasting

Qingyu Li^{1,2,*}, Chiranjib Mukhopadhyay^{1,2}, Abolfazl Bayat^{1,2,3} and Ali Habibnia^{4,5}

¹*Institute of Fundamental and Frontier Sciences, University of Electronic Sciences and Technology of China, Chengdu 611731, China*

²*Key Laboratory of Quantum Physics and Photonic Quantum Information, Ministry of Education, University of Electronic Science and Technology of China, Chengdu 611731, China*

³*Shimmer Center, Tianfu Jiangxi Laboratory, Chengdu 641419, China*

⁴*Department of Economics, Virginia Tech, Blacksburg, Virginia 24061, USA*

⁵*Dataism Laboratory for Quantitative Finance, Virginia Tech, Blacksburg, Virginia 24061, USA*



(Received 2 October 2025; accepted 12 March 2026; published 8 April 2026)

Recent advances in quantum computing have demonstrated its potential to significantly enhance the analysis and forecasting of complex classical data. Among these, quantum reservoir computing (QRC) has emerged as a particularly powerful approach, combining quantum computation with machine learning for modeling nonlinear temporal dependencies in high-dimensional time series. As with many data-driven disciplines, quantitative finance and econometrics can hugely benefit from emerging quantum technologies. In this work, we investigate the application of QRC for realized volatility forecasting. Our model employs a fully connected transverse-field Ising Hamiltonian as the reservoir with distinct input and memory qubits to capture temporal dependencies. The QRC approach is benchmarked against several econometric models and standard machine learning algorithms. The models are evaluated using multiple error metrics and the model confidence set procedures. To enhance interpretability and mitigate current quantum hardware limitations, we utilize wrapper-based forward selection for feature selection, identifying optimal subsets and quantifying feature importance via Shapley values. Our results indicate that the proposed quantum reservoir approach consistently outperforms benchmark models across various metrics, highlighting its potential for financial forecasting despite existing quantum hardware constraints. This work serves as a proof of concept for the applicability of quantum computing in econometrics and financial analysis, paving the way for further research into quantum-enhanced predictive modeling as quantum hardware capabilities continue to advance.

DOI: [10.1103/rbj7-4wnq](https://doi.org/10.1103/rbj7-4wnq)

I. INTRODUCTION

Quantum mechanics promises to enhance computing capacity in a fundamental way [1–3]. In recent years, quantum computers, albeit with severe limitations, are rapidly emerging in various physical platforms including superconducting qubits [4–7], ion traps [8–11], Rydberg atoms [12,13], photonic setups [14,15], nitrogen vacancy centers in diamond [16], and topological qubits [17]. While near-term quantum computers have demonstrated quantum supremacy over their classical counterparts, they suffer from various imperfections such as finite coherence time and a limited number of qubits [18]. Therefore, many algorithms with proven quantum advantages cannot be implemented on noisy near-term quantum computers. A natural emergent question is—*can near-term quantum computers be used for machine learning?* The advent of variational quantum algorithms [19] and related quantum approximate optimization algorithms [20]

are attempts to answer this question where they have been adapted to solving problems in a truly diverse array of disciplines, including molecular simulations [21,22], biology [23], or particle physics [24]. In parallel, methods of classical time series analysis like Crutchfield computational mechanics framework [25,26] have also been shown to be improved by quantum encodings that reduce model complexity [27]. Quantum computing methods are also starting to become important for modern quantitative finance problems like asset pricing [28], portfolio optimization [29,30], credit sales classification [31], risk management [32], and loan eligibility prediction [33], among others. See Refs. [34–36] for recent surveys. The key approach behind these algorithms is to optimize a parametrized quantum circuit to optimize the loss function in an iterative way like a classical neural network [19]. However, there is a different paradigm of classical machine learning, namely, *reservoir computing* [37], which does not seek to optimize the parameters of a neural network, and training takes place only at the final output layer. While this approach to machine learning has transformed time series modeling, it still operates in the realm of classical computing [38,39]. Quantum versions of this approach are particularly promising for near-term quantum computers and have been proposed with various underlying platforms like disordered spin chains [40], quantum optical energy levels [41], spin-boson platforms [42], or superconducting devices [43] for some mathematical and

*Contact author: lqy859405@gmail.com

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

physical problems. They are exceptionally useful for time-series forecasting, as demonstrated with canonical models like Mackey-Glass [44] and autoregressive moving average [45]. Thus, it is natural to ask whether quantum reservoir computing (QRC) can be useful for accurate forecasting of real-world financial time series data.

Volatility modeling is a fundamental aspect of financial econometrics, essential to understand and manage the uncertainty or risks associated with financial markets. Accurate modeling and forecasting of volatility are crucial for various applications, including risk management, portfolio optimization, and derivative pricing [46,47]. Given its critical role, developing robust volatility forecasting models has been a major focus of financial research. While traditional models such as generalized autoregressive conditional heteroskedasticity (GARCH) [48] and its standard extensions are widely used, their reliance on low-dimensional parametric recursions for conditional variance, typically linear in squared returns, imposes strong functional restrictions. These constraints may limit their ability to flexibly represent complex nonlinear and multiscale volatility dynamics governing realized volatility (RV) at medium and long horizons. Recognizing these limitations, subsequent research has emphasized the importance of understanding the distribution of realized stock return volatility for more effective financial analysis, as highlighted by Andersen *et al.* [49]. Furthermore, econometric analysis of realized volatility, particularly its application in estimating stochastic volatility models, has provided crucial insights into the behavior of financial time series, as demonstrated by Barndorff-Nielsen and Shephard [50]. To address some of the shortcomings of traditional models, the heterogeneous autoregressive (HAR) model [51] was developed, offering a more nuanced approach by incorporating realized volatility over different time horizons, thus providing a more accurate and comprehensive measure of market risk. Its variations, including HAR with jumps and continuous HAR [52], semivariance-HAR [53], and HAR with realized quarticity [54], further enhanced its ability to capture different aspects of market volatility. The nonlinear and often non-Gaussian nature of financial data has driven researchers to explore more sophisticated approaches, such as machine learning techniques to capture complex relationships that traditional econometric models may miss [55–63]. In the realm of realized volatility forecasting, machine learning models have shown promising results. While some studies found machine learning models to perform similarly to or slightly worse than traditional HAR-family models [64–67], others reported significant improvements using machine learning approaches [58,61,68]. For instance, Christensen *et al.* [68] applied machine learning to volatility forecasting, demonstrating that these models can significantly improve prediction accuracy, especially when combined with macroeconomic variables. Similarly, Zhang *et al.* [69] explored the integration of machine learning with intraday commonality, further enhancing the precision of volatility forecasts. Reference [58] also demonstrated the effectiveness of long short-term memory (LSTM) neural networks in forecasting monthly S&P 500 realized volatility, and have found strong links between volatility and macroeconomic factors, paving the way for more advanced machine learning applications in this field. The importance of capturing

the multifaceted nature of volatility is underscored by the work of Ghysels *et al.* [70], who emphasized the value of high-frequency data for more accurate volatility estimation. By leveraging data sampled at different frequencies, their approach offers a richer understanding of market dynamics, which is crucial for effective risk management and portfolio allocation. Recent reviews, such as the one by Gunnarsson *et al.* [71], have highlighted the growing role of machine learning in volatility modeling, particularly in the prediction of realized and implied volatility indices. These advancements underscore the trend toward more data-driven approaches in finance, which offer superior adaptability to the complexities of modern financial markets. Among machine learning algorithms, recurrent neural networks and their variant, LSTM networks, have shown particular success in modeling time series data due to their inherent ability to capture temporal dependencies. Recurrent neural networks are designed to recognize patterns in sequences of data by maintaining a hidden state that is influenced by previous inputs, making them ideal for time-dependent data such as financial time series [58,72,73].

This work introduces an approach using QRC for realized volatility forecasting of the S&P 500 index. QRC, an extension of classical reservoir computing, has been chosen for this study due to its ability to efficiently process temporal data on near-term quantum computers [40], while maintaining a low training cost and providing a state space that grows exponentially with the number of qubits to capture features. QRC models harness quantum features to enhance the prediction power of time series data. They utilize the dynamics of fixed quantum systems known as the “reservoir.” Unlike conventional neural networks, in reservoir computing, the parameters of the reservoir are not trained, and the learning procedure takes place at the final output layer after performing measurements [74]. This makes reservoir computing particularly advantageous for tasks requiring the capture of temporal dynamics without the heavy computational burden associated with training traditional neural networks. QRC extends this concept into the quantum domain, leveraging quantum states to enhance computational power and efficiency. This approach can be promising for volatility forecasting, where capturing complex, time-dependent relationships is critical [40,75–80]. Such quantum reservoir platforms have already been conceptualized in various atomic and spin lattices [40,77,81] as well as photonic platforms [76,82,83]. It is of particular interest to note that very recent results on reservoir computing already hint that quantum properties may lead to fast and reliable forecasts with smaller resources [84]. We aim to showcase the proof of concept and capability of quantum machine learning in real-world time series analysis, particularly in financial forecasting. We benchmark our quantum model against several classical models, including the HAR and HARX models, as well as traditional neural network algorithms. By leveraging a comprehensive set of features, including market microstructure and macroeconomic variables, we explore whether quantum nonlinear models can more effectively capture the intricate relationships governing market volatility. Prior research, such as that by Alaminos *et al.* [85] and Thakkar *et al.* [79], has demonstrated the potential of quantum machine learning in

financial forecasting, suggesting that quantum models may eventually outperform classical machine learning algorithms, particularly when dealing with large datasets and complex patterns.

The paper is organized as follows. Section II outlines the classical models, including traditional regression methods and machine learning models, for realized volatility forecasting. Section III demonstrates our method of volatility forecasting with QRC. If you are not familiar with quantum computing, Appendix A gives a brief discussion of the principle of quantum computing, bridging concepts from both financial econometrics and quantum computing to facilitate interdisciplinary understanding. Section IV compares our results with other volatility forecasting strategies, before concluding discussions in Sec. V.

II. TECHNICAL BACKGROUND

This paper lies at the intersection of three different subjects, namely, econometrics, machine learning, and quantum computation. In this section, we provide a brief overview of the former two subjects for physicists, introducing the concepts and methodologies that we will use later. For non-physicists, we include an appendix on quantum computation preliminaries as well as a notation table. Depending on their expertise, the readers can skip the following subsections.

A. Classical models for realized volatility forecasting

Stock market volatility is an important economic factor that reflects the risk of investment at a given time. The concept of realized volatility, formally introduced by Andersen and Bollerslev [86], marked a significant advancement in volatility measurement by providing an accurate, model-free estimate utilizing high-frequency financial data. Realized volatility, RV_t , at time t , is defined as the square root of the sum of squared returns within a given time interval:

$$RV_t = \sqrt{\sum_{i=1}^{N_t} r_{i,t}^2}, \quad \text{and we model } \ln(RV_t), \quad (1)$$

where $r_{i,t}$ represents the return on the day i within period t and N_t indicates the number of observations (e.g., trading days) in that period. Autoregressive (AR) models and their various extensions have since become essential tools for modeling and forecasting realized volatility due to their computational simplicity and effectiveness as baseline methodologies in time series analysis. Although traditional AR models effectively capture short-term volatility dependencies, they often fail to reflect the long-memory characteristics typically present in volatility dynamics. To overcome this limitation, the HAR model proposed by Corsi [51] incorporates realized volatility at multiple time horizons, thus effectively capturing both persistence and scaling features. The general form of the HAR model is expressed as the following:

$$RV_t = \beta_0 + \beta_d RV_{t-1}^{(d)} + \beta_w RV_{t-1}^{(w)} + \beta_m RV_{t-1}^{(m)} + \varepsilon_t, \quad (2)$$

where $RV_{t-1}^{(d)}$, $RV_{t-1}^{(w)}$, and $RV_{t-1}^{(m)}$ represent realized volatility averages computed over daily, weekly, and monthly horizons. The coefficients β_d , β_w , and β_m measure the contributions of

these short-term, intermediate-term, and long-term volatility components to the current volatility level. The parameter ε_t is the residual term assumed to follow a conditionally heteroskedastic process, with $\mathbb{E}[\varepsilon_t | \mathcal{F}_{t-1}] = 0$ and $\text{Var}(\varepsilon_t | \mathcal{F}_{t-1}) = \sigma_t^2$, where $\mathcal{F}_{t-1}$ denotes the information set available at time $t-1$. This specification allows the HAR model to capture volatility at multiple frequencies, accommodating heterogeneity in market participants' investment horizons. The model assumes weak stationarity of the log-volatility series and serially uncorrelated errors. Estimation is typically performed via ordinary least squares on the log-transformed realized volatility series to stabilize variance and reduce the impact of heteroskedasticity. For statistical inference, heteroskedasticity and autocorrelation-consistent standard errors are employed following Newey and West [87], ensuring robustness to serial correlation and time-varying error variance.

To incorporate exogenous macroeconomic and financial variables, we adopt a monthly version of the HAR model, resulting in a HARK specification that captures realized volatility dynamics at multiple timescales and permits additional predictors. Specifically, we define the monthly HARK model as

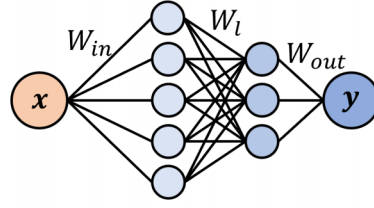
$$RV_t = \beta_0 + \beta_1 RV_{t-1} + \beta_2 \left(\frac{1}{3} \sum_{i=1}^3 RV_{t-i} \right) + \beta_3 \left(\frac{1}{12} \sum_{i=1}^{12} RV_{t-i} \right) + \sum_{k=1}^K \gamma_k^T X_{t-k} + \varepsilon_t, \quad (3)$$

where RV_{t-1} represents the monthly realized volatility lagged by one month, capturing short-term dynamics; $\frac{1}{3} \sum_{i=1}^3 RV_{t-i}$ and $\frac{1}{12} \sum_{i=1}^{12} RV_{t-i}$ represent the quarterly and annual realized volatility averages, respectively; and X_{t-k} denotes the vector of macroeconomic and financial features available at lag k , with corresponding coefficient vector γ_k . Residual diagnostics and model stability tests are performed to ensure robustness. The HARK model serves as a high-performing linear benchmark, capturing persistence at different time horizons while allowing flexible enhancement through X_t .

However, linear models such as the HAR may still fall short in capturing the nonlinear dynamics inherent in financial time series. This limitation has motivated the development of more advanced models, such as the realized GARCH model by Hansen *et al.* [88], which integrates realized measures into the GARCH framework, and extensions of the standard HAR framework designed to accommodate nonlinearities and regime-switching behaviors [64,89]. More recently, hybrid approaches combining HAR structures with advanced methodologies, such as neural networks and regularization-based procedures (e.g., least absolute shrinkage and selection operator or elastic net), have further enhanced forecast performance and adaptability in high-dimensional settings [66].

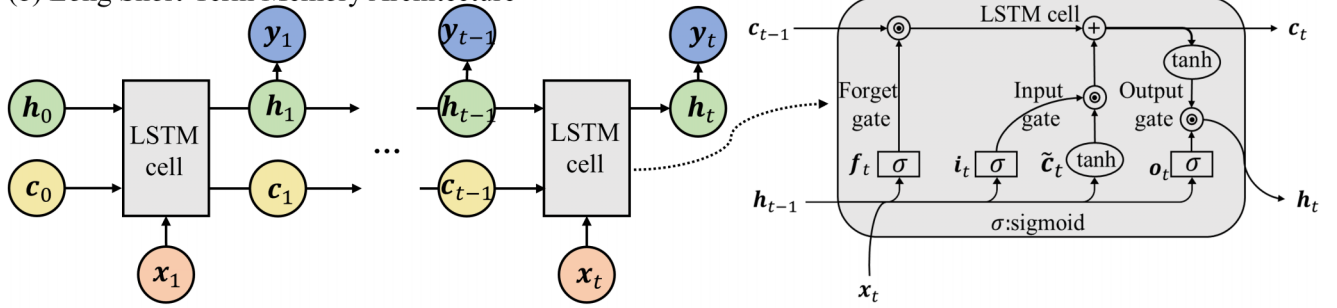
Machine learning techniques, briefly discussed later in this section, offer promising alternatives due to their ability to effectively capture complex nonlinear relationships without requiring explicit parametric functional forms. Models like LSTM networks and reservoir computing (RC) have demonstrated effectiveness in financial forecasting tasks

(a) Traditional Feedforward NN

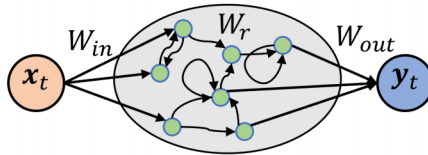


$$\mathbf{h}_l = \mathcal{F}(W_l \mathbf{h}^{l-1} + \mathbf{b}_l)$$

(b) Long Short-Term Memory Architecture



(c) Reservoir Computing



$$\mathbf{h}_t = (1 - \alpha)\mathbf{h}_{t-1} + \mathcal{F}(W_r \mathbf{h}_{t-1} + W_{in} \mathbf{x}_t + \mathbf{b})$$

$$\mathbf{y}_t = W_{out} \mathbf{h}_t$$

FIG. 1. Schematic of various machine-learning paradigms. (a) Traditional feedforward neural networks, where W_{in} , $\{W_l\}$, and W_{out} are the trainable weight matrices for the input layer, hidden layer(s), and output layer, respectively. Noticeably, the information flows unidirectionally layer by layer from the input to the output. (b) LSTM networks are a special form of the recurrent neural network architecture. LSTM introduces two key states as cell state c_i for long-term memory and hidden state h_i for short-term memory. In addition, LSTM layers utilize additional gates to control which information in the hidden state is output and passed to the next hidden state. These additional gates overcome the common issues recurrent neural networks face in learning long-term dependencies. (c) Reservoir computing is a computational framework that inputs x_i at each step are mapped to a high-dimensional space by a fixed random W_{in} . The hidden states h_i evolve dynamically through a fixed random reservoir W_r , which models the system's temporal dependencies. Notably, both W_{in} and W_r are randomly initialized and remain fixed throughout the process, while only the output weights matrix W_{out} is trainable. This feature make reservoir computing efficient for handling time series and dynamic system tasks, requiring minimal training effort.

[90,91]. Incorporating exogenous variables leads to extensions such as long short-term memory with exogenous features (LSTMX) and reservoir Computing with Exogenous features (RCX), enhancing forecast performance by utilizing additional information.

B. Machine learning preliminaries

Machine learning involves the development of algorithms and statistical models that allow systems to perform specific tasks effectively by analyzing data, identifying patterns, and making predictions. The artificial neural network [92], as one of the most powerful algorithms in machine learning, is widely used in many domains, including image recognition, natural language processing, recommendation systems, predictive analytics, and time series processes.

1. Feedforward neural networks

Feedforward neural networks are the simplest type of artificial neural network, consisting of an input layer, an output layer, and one or more hidden layers that connect the input layer to output layer [see Fig. 1(a)]. The term “feedforward” indicates that the architecture of these networks relies on transforming inputs into outputs through a series of operations, where each operation involves multiplying the input by a weight matrix and applying an activation function. Specifically, consider the output of the l th layer, denoted as $\mathbf{h}_l$, such that the input of the network is represented as $\mathbf{h}_0 = \mathbf{x}$, and the final output is $\mathbf{h}_L = \hat{\mathbf{y}}$. For the l th layer in the feedforward neural networks, the output takes the form

$$\mathbf{h}_l = \mathcal{F}(W_l \mathbf{h}_{l-1} + \mathbf{b}_l), \quad (4)$$

where W_l is the weight matrix, $\mathbf{b}_l$ is the bias vector, and $\mathcal{F}(\cdot)$ is a certain activation function. The overall mathematical model of feedforward neural networks is a nested composition of these computations, where the output $\mathbf{h}_l$ of each layer serves as the input of the next layer. Because there are no cycles or loops in the networks, the information flows strictly in a single direction: from the input node, through the hidden nodes, and to the output nodes [see Fig. 1(a)]. Training for a neural network is based on a set of labeled data $\{(\mathbf{x}_i, \mathbf{y}_i)\}$, where $\mathbf{x}_i$ and $\mathbf{y}_i$ represent input and output, respectively. We train the weight matrices $\{W_l\}_l$ as well as the bias vectors $\{\mathbf{b}_l\}_l$ so that for any input data $\mathbf{x}_i$, the corresponding output of the neural network $\hat{\mathbf{y}}_i$ closely approximates the real label $\mathbf{y}_i$. This is accomplished by minimizing a loss function such as

$$L = \min_{\{W_l, \mathbf{b}_l\}} \frac{1}{n} \sum_i^n (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2, \quad (5)$$

where n is the size of the training set. Feedforward neural networks are well suited for tasks where each input is independent of the others, such as image classification or regression. However, they are not ideal for tasks that require capturing temporal or sequential dependencies.

2. Long short-term memory neural networks

For addressing tasks with spatiotemporal dependencies, it is often necessary to utilize information from past data to make accurate predictions about future results. For example, when reading an article, the meaning of each word is interpreted based on the understanding of the preceding words. LSTM networks [72] address this problem by introducing a specialized architecture designed to capture both short-term and long-term dependencies in sequential data. Unlike standard recurrent neural networks, which rely solely on a hidden state $\mathbf{h}_t$ to carry forward information, LSTM introduces an additional cell state $\mathbf{c}_t$, which serves as long-term memory. The cell state enables LSTM to selectively retain or discard information over time. At each time step, as Fig. 1(b) shows, the LSTM cell receives the following inputs:

- (1) Hidden state $\mathbf{h}_{t-1}$: The short-term memory from the previous time step.
- (2) Cell state $\mathbf{c}_{t-1}$: The long-term memory from the previous time step.
- (3) Current input $\mathbf{x}_t$: The input in the current time step.

The operation of the LSTM cell can be described in terms of three different gates: (1) forget gate; (2) input gate; and (3) output gate [see Fig. 1(b)]. The forget gate determines which parts of the previous cell state $\mathbf{c}_{t-1}$ should be “forgotten” or retained. It uses a sigmoid activation function to produce values between 0 (completely forget) and 1 (completely keep) for each element, defined as

$$\mathbf{f}_t = \text{sigmoid}(W_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f), \quad (6)$$

where W_f is the weight matrix for the forget gate, $\mathbf{b}_f$ is the bias term, and $\mathbf{f}_t$ is a vector whose elements take values between 0 and 1. The input gate determines which parts of the current input $\mathbf{x}_t$ should be added to the cell state $\mathbf{c}_t$. It has two components, a sigmoid layer to decide which values to update and a tanh layer to create candidate values to potentially add

to the cell state $\mathbf{c}_t$, for which formulas are defined as

$$\begin{aligned} \mathbf{i}_t &= \text{sigmoid}(W_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i), \\ \tilde{\mathbf{c}}_t &= \tanh(W_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c), \end{aligned} \quad (7)$$

where $\mathbf{i}_t$ is the input gate output, a vector of values between 0 and 1, and $\tilde{\mathbf{c}}_t$ is the candidate cell state. The information from the forget gate and the input gate is used to update the cell state $\mathbf{c}_t$, as

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \quad (8)$$

where $\odot$ is the element-wise multiplication operator. As a result, $\mathbf{c}_t$ contains a nonlinear combination of the input state $\mathbf{x}_t$ and the previous input data. The output gate determines the vector $\mathbf{o}_t$ that is a nonlinear function of the hidden state $\mathbf{h}_{t-1}$ as well as the input $\mathbf{x}_t$ as

$$\mathbf{o}_t = \text{sigmoid}(W_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o). \quad (9)$$

The next hidden state $\mathbf{h}_t$ is then determined through combination of $\mathbf{o}_t$ and the cell state $\mathbf{c}_t$ as

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \quad (10)$$

In general, the output $\hat{\mathbf{y}}_t$ is a function of $\mathbf{h}_t$ as $\hat{\mathbf{y}}_t = g(\mathbf{h}_t)$, where $g(\cdot)$ depends on the specific problem. Training is carried out by optimizing a loss function, of the form of Eq. (5), which aims to minimize the difference between $\hat{\mathbf{y}}_t$ and the real output $\mathbf{y}_t$. During the training procedure, the weight matrices $\{W_f, W_i, W_c, W_o\}$ and bias vectors $\{\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_c, \mathbf{b}_o\}$ are all trained and updated iteratively [see Fig. 1(b)]. The LSTM architecture uses gating mechanisms [see Fig. 1(b)] to regulate information flow and gradient propagation [72], effectively mitigating vanishing and exploding gradients common in standard recurrent neural networks and enabling the retention of long-term information. These characteristics make LSTM widely used in fields such as natural language processing and time-series prediction.

3. Classical reservoir computing

In the early 2000s, echo state networks [37] and liquid state machine [93] were independently proposed as the seminal approach of the time series model, which are grouped into the framework of classical reservoir computing. Reservoir computing models share the common principle of using a “reservoir,” comprising W_r (the recurrent weight matrix) and W_{in} (the input-to-reservoir weight matrix), to project inputs into a high-dimensional feature space. This transformation enables the effective capture of complex patterns, relationships, and temporal dynamics in the data. Unlike conventional neural networks, reservoir computing does not train weight matrices W_r and W_{in} and bias vectors $\mathbf{b}$. Instead, these parameters are randomly initialized and remain fixed during training. A simple and trainable readout mechanism, i.e., linear regression, is then used to generate information for this high-dimensional representation. In the whole reservoir computing process, only the weights W_{out} in the linear regression layer are trained, significantly reducing training cost. The mathematical representations of these models can be expressed as

$$\begin{aligned} \mathbf{h}_t &= (1 - \alpha)\mathbf{h}_{t-1} + \mathcal{F}(W_r\mathbf{h}_{t-1} + W_{\text{in}}\mathbf{x}_t + \mathbf{b}), \\ \mathbf{y}_t &= W_{\text{out}}\mathbf{h}_t, \end{aligned} \quad (11)$$

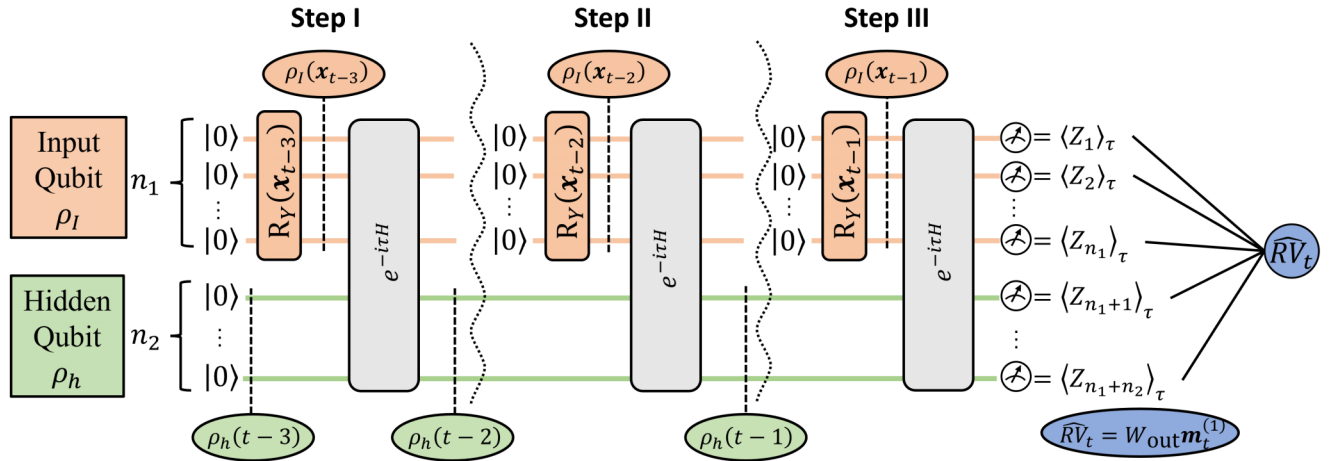


FIG. 2. Quantum reservoir computing schematic. The task is to forecast $\widehat{RV}_t$ based on the past information of features $\mathbf{x}$ at discrete time from $t-3$ to $t-1$. The quantum reservoir consists of two subsystems, the input qubits ρ_I for inputting features and the hidden qubits ρ_h for hidden state, composed of n_1 and n_2 qubits, respectively. In the first, the feature variables $\mathbf{x}_{t-3}$ are encoded into the input qubits via single-qubit gates $R_Y(\mathbf{x}_{t-3})$ applied on $|0\rangle^{\otimes n_1}$. Therefore, the input density matrix is given as $\rho_I(\mathbf{x}_{t-3})$. The hidden state at first is initialized as $\rho_h(t-3) = |0\rangle\langle 0|^{\otimes n_2}$. Thus, the quantum reservoir state is given by $\rho_I(\mathbf{x}_{t-3}) \otimes \rho_h(t-3)$. The state evolves under the action of the Hamiltonian H for time τ , described by $e^{-iH\tau}$. To update the hidden state for the next step, the input n_1 qubits are discarded such that the hidden state is given by a partial trace as $\rho_h(t-2)$. This process continues until all features from $t-3$ to $t-1$ have been encoded within the quantum reservoir. After processing all features, measurements on Pauli-Z basis of all qubits are performed to read out useful information from the quantum reservoir, yielding the expectation value $\langle Z_j \rangle_\tau$ for the j th qubit. These expectation values collectively form the readout vector $\mathbf{m}_t^{(1)} = [\langle Z_1 \rangle_\tau, \dots, \langle Z_{n_1+n_2} \rangle_\tau]$. A linear regression model is used to predict $\widehat{RV}_t$ based on this readout vector as $\widehat{RV}_t = W_{\text{out}} \mathbf{m}_t^{(1)}$, where W_{out} is a weight matrix trained using the ridge regression method.

where $\mathbf{h}_t$ represents the hidden state at time t and α is the leak rate, a hyperparameter controlling the update speed of hidden state [see Fig. 1(c)].

Since only W_{out} needs to be trained, reservoir computing is highly suitable for deployment in artificial or natural physical systems characterized by high dimensionality and nonlinear transformations. To date, reservoir computing has been widely implemented in various systems, such as cellular automata [94,95], coupled oscillators [96], analog circuits [97,98], optical node arrays [99–101], and biological organization [102].

III. QRC FOR REALIZED VOLATILITY FORECASTING

By harnessing the unique properties of quantum mechanics, such as superposition and entanglement, quantum computing demonstrates a quantum advantage in solving specific problems, including integer factorization [103], random circuit sampling [4], and quantum simulation [104]. These achievements identify quantum computing as a powerful method for information processing and thus drive researchers to explore more fields that might benefit from quantum computing. A natural extension of the quantum computing application is the investigation of reservoir computing with quantum systems. Fujii and Nakajima took the first theoretical step by proposing disordered quantum spin ensembles as reservoirs [40], which has since been extended to photonic [76,82,105], nonlinear oscillator [41], and neutral atomic Rydberg array [106] platforms. In terms of physical implementation, nuclear-spin-based reservoirs [107] as well as superconducting qubit platforms such as IBM [108] have already been successful in demonstrating various aspects of

reservoir computing. See Ref. [109] for a more detailed recent overview.

Here, we introduce a QRC approach (see Fig. 2) based on a qubit system driven by a fully connected transverse-field Ising model as a reservoir. The quantum reservoir consists of two subsystems: (1) input qubits, whose quantum state is shown as ρ_I , for encoding variables; and (2) hidden qubits, whose quantum state is shown as ρ_h , to store past information to predict future outcomes. The whole reservoir is controlled by fully connected Hamiltonian H that is given by

$$H = \sum_{ij} J_{ij} X_i X_j + v \sum_i Z_i, \quad (12)$$

where X_i and Z_i are the Pauli operators acting on i th qubits (see Table IV for the definition of Pauli matrices), v is the strength of the magnetic field that is used as the unit of energy and thus set to be $v = 1$, and finally J_{ij} are exchange couplings between qubits i and j that are randomly sampled from $J_{ij}/v \in [0, 1]$. After being randomly initialized, the Hamiltonian H is fixed throughout the process, similar to a classical reservoir.

As shown in the leftmost part of Fig. 2, the state of the ensemble quantum system is initialized as $|0\rangle$ at the beginning of the QRC process:

$$\rho_I \otimes \rho_h = |0\rangle\langle 0|^{\otimes n_1} \otimes |0\rangle\langle 0|^{\otimes n_2}, \quad (13)$$

where n_1 and n_2 are the respective sizes of the input and hidden subsystems, which can be adjusted for different tasks. In this paper, we consider $n_1 + n_2 = 10$, which is a more accessible size on current quantum computers.

TABLE I. Economic variables used in this study. Unless otherwise stated, equity data refer to S&P 500 index.

Variable	Symbol	Description
Realized volatility	RV	The natural log of the square root of the sum of squared daily returns for a month RV _q and RV _a , respectively, represent the quarterly and annual realized volatility averages.
Dividend yield ratio	DP	Dividends over the past year relative to current market prices
Earning-price ratio	EP	Earning over the past year relative to current market prices index
Market excess return	MKT	Fama-French's market factor: return of U.S. stock index minus the one-month T-bill rate
Value factor	HML	Fama-French's HML factor: average return on value stocks minus growth stocks
Size premium factor	SMB	Fama-French's SMB factor: average return on small-cap stocks minus large-cap stocks
Short-Term Reversal	STR	Fama-French's STR: average return on stocks with low prior minus high prior returns
Factor		
T-bill rate	TB	Three-month T-bill rate
Monthly inflation	INF	US monthly inflation rate
Default spread	DEF	Estimate the credit risk
Monthly Industrial	IP	Monthly growth rate of U.S. industrial production
Production Growth Rate		

The goal in this paper is to predict RV_t by using past k steps features, i.e., the time series $\mathbf{x}_{t-k}, \dots, \mathbf{x}_{t-1}$. For convenience of notation, we denote $\mathbf{x}_{t-k} = [x_{t-k,1}, x_{t-k,2}, \dots, x_{t-k,n_1}]^T$, where each element $x_{t-k,i}$ represents the i th economic feature at time step $t-k$. For example, the input vector $\mathbf{x}_{t-k}$ may contain three input data at time $t-k$ such as realized volatility RV_{t-k} , Dividend Yield (DP_{t-k}) ratio, and Earning-Price (EP_{t-k}) ratio (note that all external features used in this paper are defined in Table I). In this situation, $\mathbf{x}_{t-k} = [RV_{t-k}, DP_{t-k}, EP_{t-k}]^T$ contains three input features. In this work, we limit $k = 3$; that is, we consider the memory depth for learning to be only three steps. Although generalizing to other memory depths is straightforward and it may appear that considering large memories may yield better results, one must also consider the fact that larger memory depths require deeper quantum circuits. Given the finite coherence times available in near-term quantum devices, this trade-off quickly becomes considerable. As we shall see, our choice of $k = 3$ already yields a satisfactory predictive accuracy. Note that all features are scaled between $[-\pi, +\pi]$, fulfilling the requirement of the R_Y gate. The input vectors $\{\mathbf{x}_{t-3}, \mathbf{x}_{t-2}, \mathbf{x}_{t-1}\}$ are fed into the reservoir model sequentially. The reservoir performs an iterative loop consisting of three steps:

Step I: Encoding input data $\mathbf{x}_{t-3}$. The classical variables $\mathbf{x}_{t-3}$, assumed to have n_1 features, are encoded to the input qubits of the quantum reservoir through phase encoding, i.e., single-qubit rotation gates around the Y direction, such that the input state is given by

$$|\psi_I(\mathbf{x}_{t-3})\rangle = R_Y(\mathbf{x}_{t-3})|0\rangle^{\otimes n_1},$$

$$\text{where } R_Y(\mathbf{x}_{t-3}) = \bigotimes_{j=1}^{n_1} e^{-i\frac{x_{t-3,j}}{2}Y_j}. \quad (14)$$

Rotation encoding is a common choice for near-term quantum hardware because it is easy to implement and embeds classical features into quantum states via parametrized single-qubit rotations, thereby effectively leveraging the continuous degrees of freedom of a qubit state. Therefore, the input density matrix is given by $\rho_I(\mathbf{x}_{t-3}) = |\psi_I(\mathbf{x}_{t-3})\rangle\langle\psi_I(\mathbf{x}_{t-3})|$. The hidden n_2

qubits (the total reservoir thus consists of $n_1 + n_2$ qubits) at this stage are initialized as

$$\rho_h(t-3) = [|0\rangle\langle 0|]^{\otimes n_2}. \quad (15)$$

Thus, the collective input state of the quantum reservoir state is given by $\rho_I(\mathbf{x}_{t-3}) \otimes \rho_h(t-3)$. Then, the whole reservoir evolves under the action of the Hamiltonian H for a specific time τ , described by $U = e^{-iH\tau}$. The evolution scrambles the input data $\mathbf{x}_{t-3}$ across the entire reservoir. Note that since the reservoir is a fully connected graph we set $\tau = 1/v$, which is enough to guarantee that the information is scrambled across the whole system. Following this, while the original n_1 qubits are discarded, the quantum state of the hidden n_2 qubits carry this information forward to the next step. The quantum state of the hidden qubits after discarding the input qubits now becomes

$$\rho_h(t-2) = \text{Tr}_I[U(\tau)[\rho_I(\mathbf{x}_{t-3}) \otimes \rho_h(t-3)]U^\dagger(\tau)], \quad (16)$$

where $\text{Tr}_I[\cdot]$ represents partial trace over all the n_1 input qubits.

Step II: Encoding input data $\mathbf{x}_{t-2}$. After discarding the input n_1 qubits at the end of the last round, we replace them with fresh n_1 qubits with the encoding

$$|\psi_I(\mathbf{x}_{t-2})\rangle = R_Y(\mathbf{x}_{t-2})|0\rangle^{\otimes n_1},$$

$$\text{where } R_Y(\mathbf{x}_{t-2}) = \bigotimes_{j=1}^{n_1} e^{-i\frac{x_{t-2,j}}{2}Y_j}. \quad (17)$$

Therefore, the state of the whole reservoir becomes $\rho_I(\mathbf{x}_{t-2}) \otimes \rho_h(t-2)$. Similar to the previous step, the whole system undergoes another evolution for time τ that scrambles the input data $\mathbf{x}_{t-2}$ across the reservoir. Once again, the n_1 input qubits are discarded and the quantum state of the hidden qubits is naturally updated to

$$\rho_h(t-1) = \text{Tr}_I[U(\tau)[\rho_I(\mathbf{x}_{t-2}) \otimes \rho_h(t-2)]U^\dagger(\tau)]. \quad (18)$$

Step III: Encoding input data $\mathbf{x}_{t-1}$ and final measurement. Similar to the previous step, the n_1 input qubits are replaced

by fresh qubits to encode the input data $\mathbf{x}_{t-1}$ as

$$|\psi_I(\mathbf{x}_{t-1})\rangle = R_Y(\mathbf{x}_{t-1})|0\rangle^{\otimes n_1},$$

where $R_Y(\mathbf{x}_{t-1}) = \bigotimes_{j=1}^{n_1} e^{-i\frac{x_{t-1,j}}{2}Y_j}$. (19)

The state of the whole reservoir thus becomes $\rho_I(\mathbf{x}_{t-1}) \otimes \rho_h(t-1)$. Again, the whole system evolves freely for time τ that scrambles the input data $\mathbf{x}_{t-1}$ across the whole reservoir. At this stage, no qubit is discarded and all of them are measured in the Pauli-Z basis. The expectation of measurement outcomes of qubit j is given by

$$\langle Z_j \rangle_\tau = \text{Tr}[U(\tau)[\rho_I(\mathbf{x}_{t-1}) \otimes \rho_h(t-1)]U^\dagger(\tau)Z_j]. \quad (20)$$

At any given step t , the measurement outcomes on all the qubits form a vector $\mathbf{m}_t^{(1)} = [\langle Z_1 \rangle_\tau, \dots, \langle Z_{n_1+n_2} \rangle_\tau]^T$. The training is performed on these measured data, just as the classical reservoir learning. This is accomplished by a linear regression that is used to approximate $\widehat{RV}_t$ by

$$\widehat{RV}_t = W_{\text{out}} \mathbf{m}_t^{(1)}, \quad (21)$$

where W_{out} is a weight matrix that has to be trained. If we consider the loss function as mean squared error (MSE), defined as

$$\text{MSE} = \frac{1}{T} \sum_{t=1}^T (RV_t - \widehat{RV}_t)^2, \quad (22)$$

then the weight matrix takes an analytical form given by the ridge regression as

$$W_{\text{out}} = RV^\top M_1^\top (M_1 M_1^\top + \delta I)^{-1}, \quad (23)$$

where $M_1 = [\dots, \mathbf{m}_t^{(1)}, \dots, \mathbf{m}_T^{(1)}]$, I is the identity matrix, and $\delta = 10^{-8}$ is a small number that is used to guarantee that the matrix is nonsingular. We call this model as quantum reservoir 1 (QR1), since here we only use one quantum reservoir. See Appendix B for more information.

Furthermore, to obtain richer information from the quantum reservoir, we adopt the ensemble reservoir approach [40]. In this approach, at any given step t apart from the measurement outcomes $\mathbf{m}_t^{(1)}$, one can use another reservoir that is almost identical to the previous setup except that the evolution at step III is run for a time duration of $\tau/2$, instead of τ (see Fig. 3).

The ensemble of these two reservoirs are combined to make a larger vector $\mathbf{m}_t^{(2)} = [\langle Z_1 \rangle_\tau, \dots, \langle Z_{n_1+n_2} \rangle_\tau, \langle Z_1 \rangle_{\tau/2}, \dots, \langle Z_{n_1+n_2} \rangle_{\tau/2}]^T$, where $\langle Z_j \rangle_{\tau/2}$ represents the measurement of the Pauli operator Z_j in the second reservoir setup in which the last evolution is run for time $\tau/2$, as schematically shown in Fig. 3. We call this model quantum reservoir 2 (QR2), which is expected to be more precise than the QR1 as it uses twice as much resources. It is worth noting that, in the simulation stage, we directly compute the measurement outcomes of the quantum system. In an actual experiment, however, one must perform many repeated measurements (shots) and use their average as the output. Recent studies [110] have shown that, as the system dimension increases, QRC may suffer from an exponential

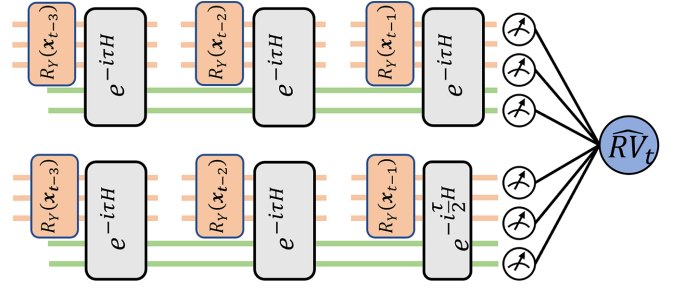


FIG. 3. Ensemble reservoir. The ensemble reservoir approach repeats the procedure twice, with the final step running the quantum dynamical evolution for times τ and $\tau/2$ in two separate ensembles, respectively. This is referred to as QR2 in this work, contrasted with QR1 not employing ensemble learning.

concentration of measurement results toward a value that is independent of the input. Concretely, the variance of measurement outcomes across different input variables can decay exponentially to zero. As a result, an exponential number of shots would be required to reliably distinguish the outputs corresponding to different inputs, which undermines the potential advantage of QRC. In our approach, by contrast, the QRC system is kept at a fixed size (ten qubits), so this issue does not arise (see Appendix C for more information). The measurement outputs corresponding to different inputs still exhibit sufficiently large variance, indicating that our scheme does not require an excessive number of measurements to obtain accurate output estimates.

IV. EMPIRICAL ANALYSIS

In this section, we present the empirical findings of our study, with a focus on the performance of quantum algorithms in forecasting the realized volatility of the S&P 500 index. We begin by detailing the dataset and the variables used, which include monthly realized volatility, as well as a set of macroeconomic and financial features. Following this, we outline the fitting procedure for competing models, both classical and quantum, and describe the training and evaluation methodologies applied.

A key technical contribution of this study lies in our approach to feature selection within the quantum framework. To optimize the set of features for the QRC models, we employ a forward selection method, a wrapper-based approach that incrementally identifies the most impactful features. This process enables the model to systematically build an optimal feature set by evaluating the performance impact of each addition of variables. Furthermore, to assess the relative importance of each selected feature, we utilize the Shapley value, a method grounded in game theory that fairly distributes the contribution of each feature across different model configurations. The use of Shapley values provides valuable information on the explanatory power of individual features, enhancing the interpretability of the forecast results of the QRC model.

Finally, we analyze the predictive performance of the models, comparing them based on accuracy metrics and statistical tests. This comprehensive evaluation demonstrates the effectiveness of the QRC approach in capturing the complex and

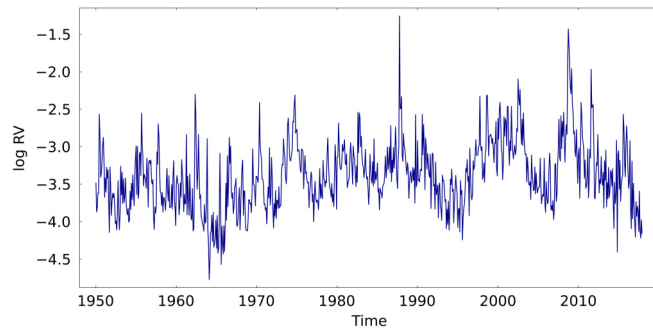


FIG. 4. Time series of the logarithm of monthly realized volatility data for the S&P 500 index. Data are taken from Yahoo Finance's API [111] between January 1950 and December 2017.

nonlinear dynamics of financial market volatility, underscoring its potential advantages over classical models in volatility forecasting.

A. Data

For this study, we utilize a dataset comprising monthly observations of realized volatility for the S&P 500 index, covering the period from February 1950 to December 2017, resulting in a total of 815 data points. This approach aligns with that of Bucci [58], who employed a similar time frame to assess the performance of neural network models in forecasting realized volatility. Our primary variable of interest, realized volatility (RV_t), provides a foundational measure of market uncertainty. As illustrated in Fig. 4, realized volatility exhibits significant variability over time, reflecting alternating phases of market turbulence and stability. Capturing these dynamics accurately is critical for financial decision-making and risk management.

To facilitate robust comparison, we align our study with the data and sample period used by Bucci [58]. This alignment enables a direct and fair comparison between our QRC approach and established neural network models while maintaining consistency in data characteristics and market environments.

Beyond realized volatility, we include a set of macroeconomic and financial variables known to capture important economic forces and market behavior. Research has shown that adding these features can significantly enhance the performance of volatility forecasting models. For example, Schwert [112] and Engle *et al.* [113] demonstrate that indicators such as inflation rates (INF) and industrial production growth (IP) provide an essential context about the state of the broader economy, improving a model's reliability in forecasting financial volatility. Similarly, valuation measures such as the dividend-price ratio (DP) and the earnings-price ratio (EP) reflect market expectations and investor sentiment, providing valuable predictive power by signaling changes in expected returns and market risk [114]. Additionally, market-based factors derived from the Fama-French model [115,116], including market excess returns (MKT), value factor (HML), size premium factor (SMB), and short-term reversal factor (STR), capture critical aspects of equity market dynamics, offering insights into investor behavior and systematic market risk. The inclusion of financial indicators such as the three-

month Treasury Bill Rate (TB) and Default Spread (DEF) further enhances the predictive framework by incorporating measures of interest rate and credit risk conditions.

Including these diverse factors aligns with the findings of Bucci [58], Christensen *et al.* [68], and Zhang *et al.* [69], who showed that combining macroeconomic and financial variables with machine learning models significantly enhances both explanatory power and forecasting accuracy. By incorporating this set of features into our QRC framework, we aim to capture the multifaceted drivers of market volatility. This approach is expected to not only improve forecast precision but also yield deeper insights into the complex interplay between financial markets and economic fundamentals.

Table I provides a detailed summary of the features utilized in our study, consistent with the variables commonly employed in prior research.

B. Training and benchmarking

We employ our quantum reservoir models QR1 and QR2 to predict realized volatility. To assess their performance, we conduct a comprehensive benchmarking study against established classical models. In particular, we compare QRC models QR1 and QR2 with the classical linear models AR1 (which is AR with memory of one previous step), AR3 (which is AR with memory of three previous steps), ARMAX, HAR, and HARX as well as the nonlinear machine learning based methods LSTM, LSTMX, RC, and RCX; see the relevant parts of Sec. II for a short introduction to these models. For all models, including classical models and QRC, we employ a rolling-window approach to estimate and train the models, optimize their parameters, and perform one-step-ahead and five-steps-ahead out-of-sample predictions. Rolling reestimation at each monthly step is consistent with standard out-of-sample evaluation practices in the volatility forecasting literature, ensuring alignment with recent information and accommodating structural change [58,117–119]. This design promotes comparability across models by isolating learning effects from differences in estimation windows and is particularly important for monthly realized volatility, which reflects aggregated market activity and is influenced by slowly evolving macroeconomic regimes. Without regular updating, parameter estimates may become obsolete in the presence of structural breaks. Our approach, therefore, maintains regime adaptiveness while avoiding excessive sensitivity to high-frequency noise. While our study uses monthly updates to maximize predictive accuracy for benchmarking, a lower reestimation frequency could be adopted in industrial settings to enhance stability and ease of debugging.

The initial training window spans February 1950 to June 1997 (approximately 571 months). After training and optimizing these models on the initial window, we generate a forecast for the next out-of-sample month (August 1997). Subsequently, the rolling window advances by one month, using data from March 1950 to August 1997 to predict September 1997. This process continues iteratively, rolling through all 245 out-of-sample observations, spanning from August 1997 to December 2017. During this process, all models are reestimated at each step to ensure optimal performance for the rolling window.

Among the classical linear models that we use, AR1, AR3, ARMAX, HAR, and HARX are estimated using ordinary least squares or maximum likelihood methods, depending on the structure of the model. In addition, the machine learning based LSTMX model is implemented with two layers of LSTM cells. The hidden state size is set to 60 for the LSTM model and 50 for the LSTMX variant. After that, a linear regression is used to transform the output of LSTMX into the prediction of $\widehat{RV}_t$. The model parameters are optimized using the ADAM optimizer, with a learning rate of 0.001, the batch size of 64, and 100 epochs. For the classical reservoir computing models (RC and RCX), the reservoir consists of 50 hidden neurons for RC and 20 hidden neurons for RCX. The leak rate, which controls the speed at which the state of the reservoir evolves, is set at 0.6. The spectral radius, which influences the stability and nonlinearity of the reservoir, is fixed at 0.9. The input scaling, which is a coefficient applied on W_{in} , is set at 0.1. The readout matrix W_{out} , responsible for mapping the reservoir states to the output, is estimated using ridge regression to mitigate overfitting and improve generalization. All machine learning hyperparameters, including architecture choices and training settings, are selected using time-series-aware cross-validation procedures within the training window to ensure generalization and robustness (see Appendix D for more information).

For QRC, taking into account the current limitations of quantum devices, we used a quantum reservoir with ten qubits (comprising both input and hidden qubits such that $n_1 + n_2 = 10$) and three layers, which means that features that lagged as $t - 1$, $t - 2$, and $t - 3$ will be used to predict RV_t . As mentioned before, we implement two QRC models, QR1 and QR2. QR1 does not utilize the ensemble reservoir approach, while QR2 incorporates an additional reservoir to enhance computational capacity (see Fig. 3). The quantum reservoir outputs, corresponding to evolved quantum states driven by lagged input encodings, are used as nonlinear regressors in a regularized linear model estimated via ridge regression. In both quantum reservoir models, the weight matrix W_{out} is estimated using ridge regression, as given in Eq. (23). Implementation-wise, we consider 100 different quantum reservoir instances and train and evaluate each of them separately. We then select the best-performing quantum reservoir and report the results. Similarly, for LSTM, we test multiple hyperparameter configurations and report the best-performing results for presentation. Notice that we are reporting the best-performing results for both LSTM and QRC; hence, QRC does not specifically gain any unfair advantage.

C. Closed-loop prediction of multistep future outcomes

So far, we have focused on the prediction of outcomes that are only one step ahead of the input. In other words, all models are trained under a one-step open-loop setting: The model inputs are ground-truth values, and performance is evaluated against the ground truth. For example,

$$\widehat{RV}_t = f(RV_{t-3}, X_{S_{t-3}}, RV_{t-2}, X_{S_{t-2}}, RV_{t-1}, X_{S_{t-1}}),$$

where Xs denotes other (exogenous) variables (see Table I). If a model name does not include the X suffix, then Xs is empty (i.e., no exogenous variables are used). This is called open-loop strategy in which historic data are used for predicting

one step ahead. However, one might be interested to predict the outcomes well ahead of the current data. For instance, by accessing the historic data one might be interested to know the outcome of S steps ahead. This can be done through a closed-loop procedure that is explained below. To accomplish this, we rely on a conventional trained open-loop model that predicts $S = 1$ step ahead. For predicting $S > 1$ steps in the future, we repeatedly call the model and feed its previous predictions back to the model as part of the next input. Specifically,

(1) For $S = 1$,

$$\widehat{RV}_t = f(RV_{t-3}, X_{S_{t-3}}, RV_{t-2}, X_{S_{t-2}}, RV_{t-1}, X_{S_{t-1}}).$$

(2) For $S = 2$,

$$\widehat{RV}_{t+1} = f(RV_{t-2}, X_{S_{t-2}}, RV_{t-1}, X_{S_{t-1}}, \widehat{RV}_t, X_{S_t}).$$

At this step, $\widehat{RV}_t$ is taken from the previous step's output. Since our models predict only RV (and not the other variables), X_{S_t} is still provided as ground truth. However, if such data are not available then a predictor has to be trained targeting X_{S_t} and use its predicted value (i.e., $\widehat{X}_{S_t}$) as the input of the closed-loop model. Here, for simplicity we assume that the ground truth of X_{S_t} is available.

Noted that when $S = 1$, the closed-loop setting degenerates to the open-loop setting. Using the closed-loop strategy allows us to assess a model's ability to forecast into the future, which is crucial in many forecasting scenarios. We consider $S = 1$ and $S = 5$ to evaluate both short-term and longer-term predictive performance. It is worth emphasizing that by increasing S the accuracy is expected to go down as the inputs come from prediction rather than the actual ground truth data. In other words, at every step, a small error can propagate to the next steps, affecting the accuracy of the model for predicting distant future outcomes.

D. Feature selection for QRC

For the classical models that we selected, the features listed in Table I can be easily applied to the models. However, considering the performance of currently available quantum computers, we have limited the system size of the quantum reservoir model to ten qubits. Among these qubits, we use n_1 qubits for encoding the features and reserve the rest (i.e., $n_2 = 10 - n_1$) for hidden qubits. Consequently, we cannot apply all the features listed in Table I to the quantum model at the same time. Therefore, we need to further select the most significant features for our QRC models QR1 and QR2.

To refine the key features that best fit our model, we employ one of the wrapper methods, which are model-related feature selection algorithms. The wrapper method treats the machine learning model as a black box, optimizing it to evaluate different subsets of features. This approach simplifies feature selection by framing it as a search problem to identify the best subset of features. The wrapper method requires a strategy to search through possible feature subsets. Common search strategies include exhaustive search, forward selection, backward elimination, and various heuristic approaches, such as genetic algorithms or simulated annealing.

In this study, we use forward selection as the search strategy, which is an iterative method (see Fig. 5). In forward

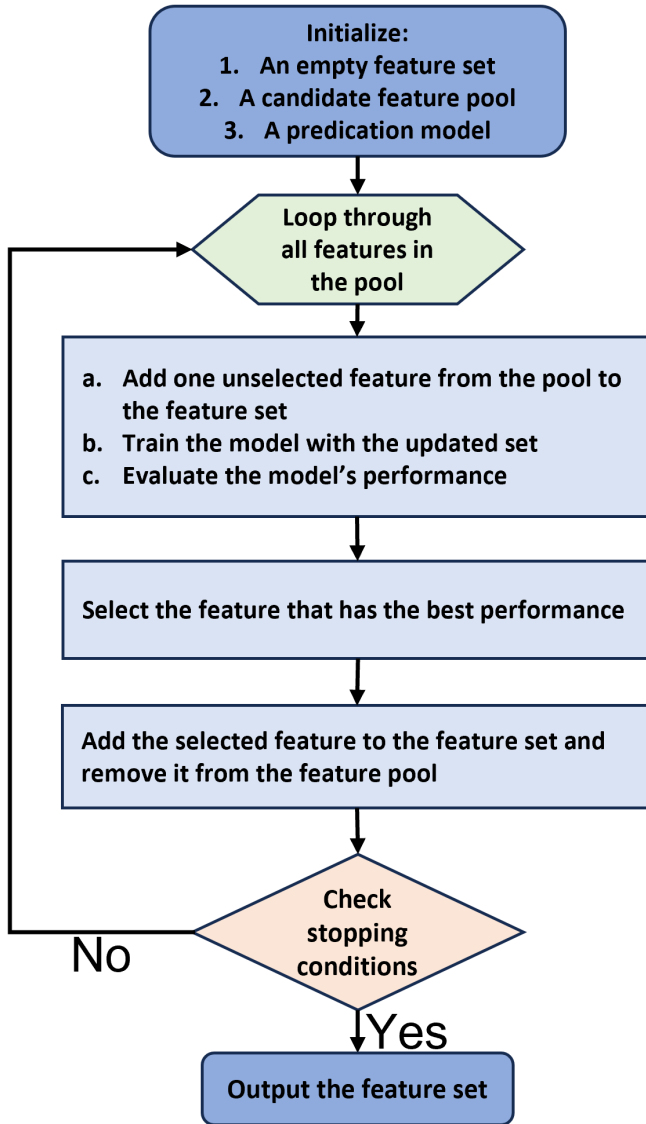


FIG. 5. Feature selection by wrapper method. The process begins with an initial pool of features, which contains all candidate features without prior filtering. At each iteration, one feature from the pool is considered as a candidate for inclusion in the optimal feature set (initially empty). Each candidate feature is temporarily added to the current optimal feature set, and the resulting feature set is then used to train a quantum reservoir model. The performance of the trained model is evaluated using the MSE loss function. Among the candidates, the feature that leads to the lowest MSE is selected and permanently added to the optimal feature set, while the remaining candidates return to the pool. This selection process is repeated iteratively until a predefined stopping criterion is met, such as reaching a target performance threshold or exhausting all candidate features.

selection, the process begins with three key components: (1) a prediction model; (2) an initially empty feature set; and (3) a feature pool containing all candidate features available for selection. At each iteration, the method evaluates the performance of the prediction model by temporarily adding one feature at a time from the candidate pool to the current feature set. The feature that results in the best performance in the current iteration is permanently added to the optimal

feature set and removed from the feature pool. The process continues until no feature from the candidate pool can significantly improve the model's performance or until a predefined stopping criterion is met. The algorithm is schematically shown in Fig. 5. Through this incremental feature selection process, the dimension of the feature space can be effectively reduced while retaining the most predictive features. This not only improves the explainability and stability of the model, but also avoids overfitting, thus enhancing the accuracy of predicting volatility.

We use the forward selection method with the stop condition as the maximum features in the optimal features is 10 (see Fig. 5) for both the QR1 and the QR2 models to identify the best subsets of features. For the QR1, the optimal subset is $F_{QR1}^* = \{RV, MKT, DP, IP, RVq, STR, DEF\}$, and for the QR2, it is $F_{QR2}^* = \{RV, MKT, STR, RVq, EP, INF, DEF\}$. Thus, we have selected $n_1 = 7$ qubits to input features for each model, allowing $n_2 = 3$ hidden qubits to carry information from the past data to be used for future prediction. We measure the forecast mean squared error of QR1 and QR2 with different subsets of features and present the results in Figs. 6(a) and 6(b). These figures clearly show that as the number of optimal features increases, the performance of the model initially improves but then deteriorates. One reason for this is that as the number of features increases, more qubits are required to input the values. Since the total size of the quantum system is fixed, the number of qubits for hidden states decreases, preventing the quantum model from capturing the long-memory property of past information. Based on the results in Fig. 6, the optimal number of features for the QR1 and QR2 models is $n_1 = 7$. In addition, they share several features, $\{RV, MKT, STR, RVq, DEF\}$.

To see how increasing the features affects the prediction quality, in Fig. 7(a), we plot the realized volatility as well as the QR1 forecast for the first one, four, seven, and nine features, which are shown in Fig. 6. As the figure clearly shows by increasing the features, the prediction improves until the number of features reach $n_1 = 7$. By further increasing the number of features, the prediction quality cannot be further improved because the number of hidden qubits decreases; thus, the reservoir lacks sufficient memory to incorporate past information for future prediction. The corresponding results for the QR2 model are also shown in Fig. 7(b). The same conclusion about the impact of features on prediction quality can be deduced from the QR2 model. In addition, by comparing the corresponding panels in Figs. 7(a) and 7(b) one can clearly see that the QR2 outperforms the QR1 prediction. This is expected as the QR2 uses twice more measurement data for predicting the future outcomes.

E. Model interpretability

Any machine learning based algorithm should not only minimize its loss function, but preferably also explain which features contribute the most toward this goal. In the context of this work, we want to understand which external features contribute the most toward realized volatility. In the previous section devoted to the forward feature selection method, we adopted a constructive approach of adding one feature at a time from scratch. However, this leaves an equally important

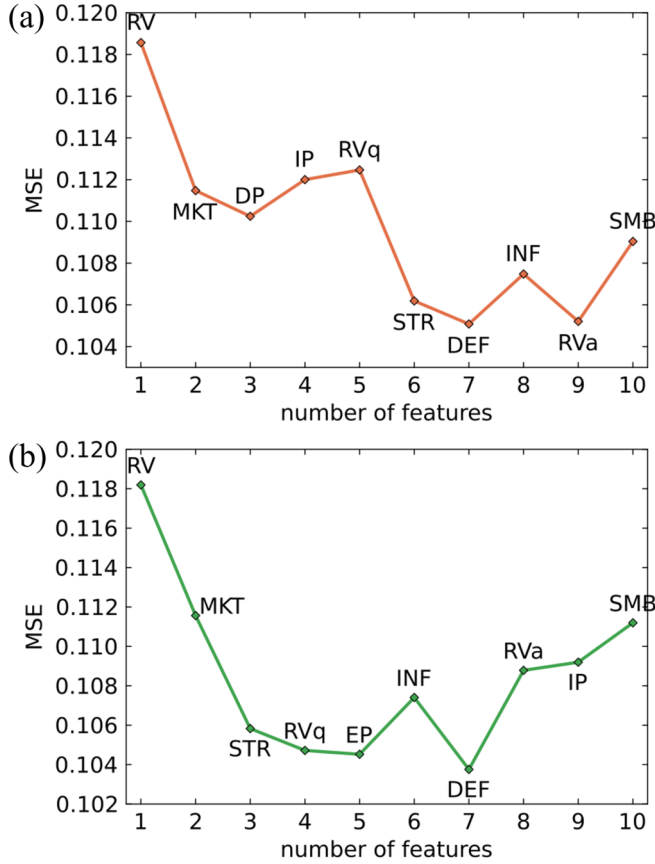


FIG. 6. Performance (MSE) of QR1 and QR2 models with their optimal feature sets. (a) QR1 model (orange), the best performance is achieved with the optimal features {RV, MKT, DP, IP, RVq, STR, DEF}. (b) QR2 model (green), the best performance is achieved with the optimal features {RV, MKT, STR, RVq, EP, INF, DEF}. Order of features in the optimal feature set is left to right along horizontal axis. Note that the features included in these two sets are not the same and have different orderings.

question untouched—given a set of features already being encoded into our simulation, how do we separate out individual contributions coming from each feature toward the predictive success of our QRC based method?

To answer this, we employ the Shapley-value method, a model-agnostic technique grounded in cooperative game theory. Originally introduced in Ref. [120], the Shapley value offers a principled framework for fairly distributing the total payoff among participants in a coalition based on their individual contributions. This approach considers all possible combinations and permutations of features, helping to quantify the contribution of each feature to the model prediction. Lundberg and Lee [121] adapted this concept to machine learning, proposing Shapley values as a unified and interpretable measure of feature importance that satisfies key axioms such as consistency and local accuracy. In this spirit, our analysis applies Shapley values to the QRC framework to assess how macrofinancial variables contribute to realized volatility forecasts, bridging interpretability with quantum-enhanced prediction. In theory, exact computation of Shapley values is exponential in the number of features; therefore,

we typically rely on approximation methods in practice. In this work, we estimate Shapley values for the QR1 and QR2 models using the Monte Carlo sampling method [122], as implemented in the Julia package ShapML. The choice of the features is decided according to one of following three strategies:

(1) *Individual feature contribution.* In this strategy, we evaluate the contribution of each feature at each lag separately, e.g., $\{RV_{t-3}, MKT_{t-3}, \dots, RV_{t-2}, MKT_{t-2}, \dots, RV_{t-1}, MKT_{t-1}, \dots\}$. The results are shown in Figs. 8(a) and 8(b), for QR1 and QR2 models, respectively. As expected, in both models, RV_{t-1} has the most contribution in predicting the future outcome RV_t .

(2) *Feature-family contribution.* In this strategy, we evaluate the contribution of each feature family irrespective of the time lag. Therefore, during the evaluation of the Shapley value when we evaluate the contribution of one feature type, e.g., for realized volatility RV, we consider contributions from all $\{RV_{t-3}, RV_{t-2}, RV_{t-1}\}$ time-lagged steps. The results are shown in Figs. 8(c) and 8(d) for the QR1 and QR2 models, respectively. Interestingly, the Shapley-value-based method furnishes a somewhat different ordering of feature importance compared to the forward selection method results depicted in Fig. 6. Note that this is not inconsistent and stems from the fact that the Shapley value is computed when all the features are embedded in the model while the forward selection method works by considering each individual feature in isolation.

(3) *Time-lagged feature contribution.* In this strategy, we evaluate the contribution of all the features at a given time lag, e.g. $F_{t-k} = \{RV_{t-k}, MKT_{t-k}, \dots\}$. We expect that the more recent data contribute the most toward future prediction compared to historical data. The results are shown in Figs. 8(e) and 8(f) for QR1 and QR2, respectively. As expected, in both models our expectation is borne out and the more recent data are indeed more useful.

F. Performance metrics and forecast evaluation

The selection of appropriate performance metrics is pivotal in the evaluation of volatility forecasting models [123]. In all the models, either classical or quantum, we have used MSE, defined in Eq. (22), as a loss function that is minimized. Although MSE is a popular figure of merit for evaluating the performance of a time series predictor, we stress that it is blind to the direction of error. In other words, overestimation as well as underestimation, both get equally penalized. However, in the concrete case of realized volatility prediction, underestimation is far more dangerous as it can lead to serious undermining of emerging risks in the market. Thus, we specifically need a figure of merit that quantifies the degree of underestimation. In order to accomplish this task, we employ the widely recognized Quasilikelihood (QLIKE) as our figure of merit that captures both symmetric and asymmetric aspects of forecast errors, thereby providing a comprehensive assessment of model performance. The QLIKE is defined as

$$\text{QLIKE} = \frac{1}{T} \sum_{t=1}^T \left(\log \widehat{RV}_t^2 + \left(\frac{RV_t}{\widehat{RV}_t} \right)^2 \right).$$

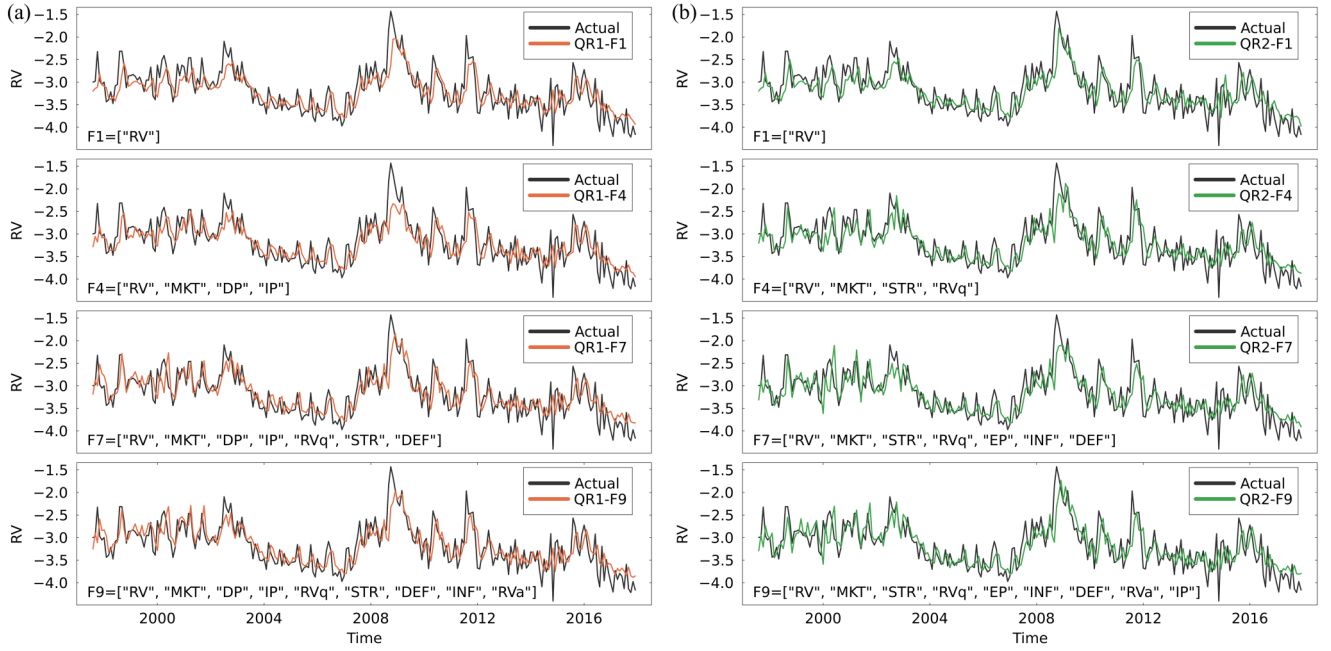


FIG. 7. Forecasting performance of QR1 [panel (a)] and QR2 [panel (b)] models on realized volatility with different feature subsets. Each subfigure presents the actual realized volatility (black line) of the S&P 500 index alongside predictions from the QRC models (QR1 in orange and QR2 in green) using varying feature configurations. QR2 demonstrates higher accuracy, especially during volatility peaks, suggesting it has better responsiveness to market fluctuations.

As it is evident from the definition, the QLIKE function penalizes underpredictions more heavily, reflecting the higher costs of underestimating volatility in risk management and option pricing. Furthermore, Patton [124] demonstrates that QLIKE

is robust to measurement errors in volatility proxies. For statistical evaluation of predictive accuracy, we employ the Model Confidence Set (MCS) procedure proposed by Hansen *et al.* [125] and the Diebold-Mariano (DM) test [126]. We

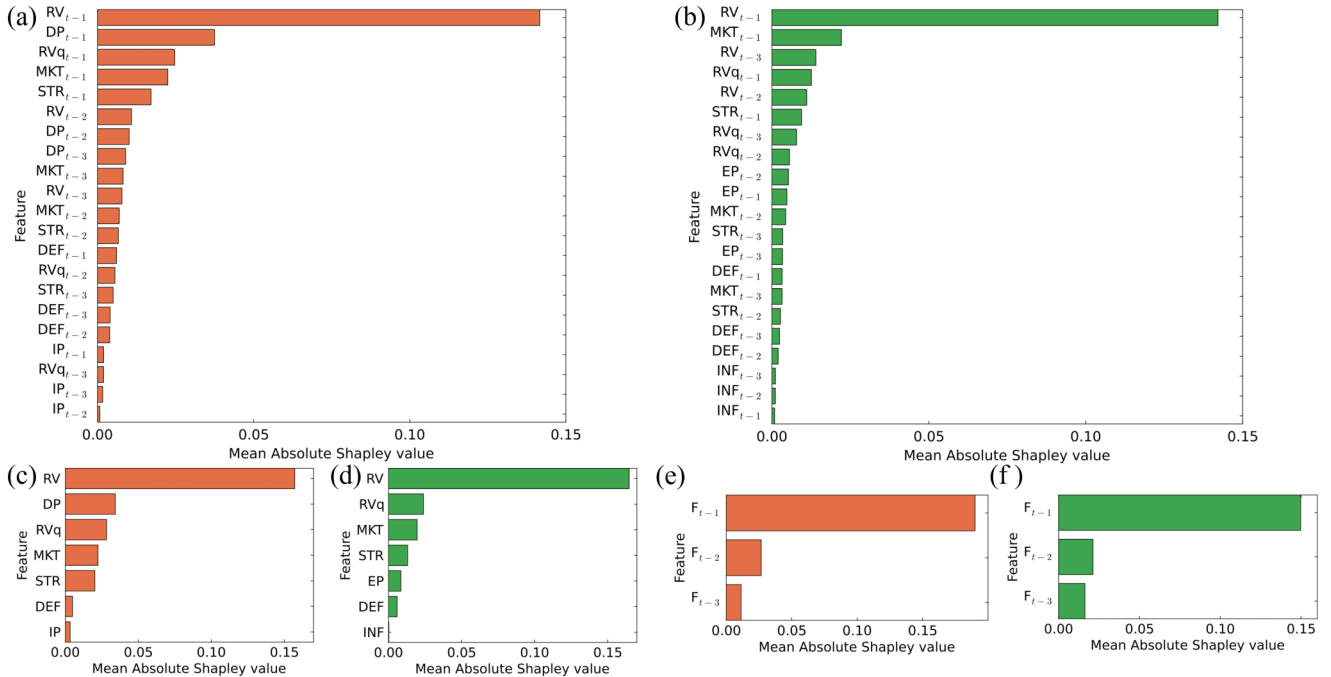


FIG. 8. Shapley-value-based feature importance test. The Shapley values for the optimal feature set are presented for QR1 and QR2. Panels (a), (c), and (e) correspond to QR1, while panels (b), (d), and (f) correspond to QR2. Panels (a) and (b) show the Shapley values for each feature at different time lags. Panels (c) and (d) show the Shapley values for each feature, aggregated across all time lags. Panels (e) and (f) show the Shapley values for each lagged input.

utilize MSE as our primary loss function due to its simplicity, while also reporting QLIKE values to assess asymmetry and penalize underestimation of volatility more strongly.

Model confidence set. The MCS procedure is designed to identify a subset of forecasting models whose predictive performance is statistically indistinguishable from that of the best model at a specified confidence level. The procedure begins with an initial set of candidate models, denoted by $\mathcal{M}_0$, and iteratively eliminates inferior models based on tests of equal predictive ability. Formally, let $L_{i,t}$ denote the loss (e.g., squared error) incurred by model i at time t and define the loss differential between models i and j as

$$d_{ij,t} = L_{i,t} - L_{j,t}, \quad i, j \in \mathcal{M}_0$$

The null hypothesis of equal predictive ability is

$$H_{0,\mathcal{M}} : \mu_{i,j} = \mathbb{E}[d_{ij,t}] = 0, \quad \forall i, j \in \mathcal{M},$$

where $\mathcal{M} \subseteq \mathcal{M}_0$ denotes the set of models under consideration at a given iteration. Rejection of $H_{0,\mathcal{M}}$ implies that at least one model in $\mathcal{M}$ exhibits significantly inferior forecasting performance.

To test $H_{0,\mathcal{M}}$, a range-type test statistic is employed:

$$T_{R,\mathcal{M}} = \max_{i \in \mathcal{M}} t_i,$$

where t_i denotes a studentized statistic comparing the performance of model i to others (e.g., via average loss differences). If the null is rejected, the model with the most evidence against it is eliminated according to the elimination rule:

$$e_{\mathcal{M}} = \arg \max_{i \in \mathcal{M}} \left(\sup_{j \in \mathcal{M}} \hat{d}_{i,j} \right),$$

where $\hat{d}_{i,j}$ is the sample mean of the loss differential. This elimination process is repeated until the null hypothesis is no longer rejected.

The final surviving set of models is denoted by $\widehat{\mathcal{M}}_{1-\alpha}^*$, the $(1 - \alpha)$ MCS, which is asymptotically guaranteed to contain the model(s) with the lowest expected loss with probability at least $1 - \alpha$. In this study, we set $\alpha = 0.05$, ensuring that the resulting confidence set contains the best-performing model(s) with at least 95% probability in large samples. We complement the MCS analysis with pairwise Diebold-Mariano tests to assess the statistical significance of forecast performance differentials between models. This dual approach allows us to validate robustness in model rankings and mitigate the limitations associated with relying on a single evaluation metric.

Diebold-Mariano test. The DM test is a particular adaptation of the above procedure for time-series data and evaluates the null hypothesis of equal predictive accuracy between two competing models, formally stated as

$$H_0 : \mathbb{E}[d_t] = 0,$$

where $d_t = L_{1,t} - L_{2,t}$ is the loss differential at time t between models 1 and 2. Here, $L_{i,t}$ represents the loss at time t for model i , and the choice of loss function $L_{i,t}$ (e.g., MSE in this paper) reflects the predictive objective. The DM test statistic

is computed as

$$\text{DM statistic} = \frac{\bar{d}}{\sqrt{\hat{\sigma}^2/T}},$$

where $\bar{d} = \frac{1}{T} \sum_{t=1}^T d_t$ is the sample mean of the loss differential, $\hat{\sigma}^2$ is the Newey-West adjusted variance of d_t , and T is the sample size. The test accounts for potential autocorrelation and heteroskedasticity in d_t , making it suitable for time-series data with serially correlated forecast errors.

MCS results. The results of MCS are presented in Table II. The table reports the performance metrics (MSE and QLIKE) along with the MCS p values for classical time series models and QRC models. In particular, when $S = 1$, namely, one-step-ahead prediction, quantum models, especially QR2, demonstrate superior performance across all measures. From the upper table in Table II, we further observe that the QR2 model achieves the lowest MSE and QLIKE values, indicating its superior predictive accuracy. The MCS p values further reinforce this finding, as QR2 attains a p value of 1.0, signifying its inclusion in the superior set of models at the 95% significance level. In contrast, most classical models exhibit significantly higher loss values and lower MCS p values, suggesting inferior performance. Examining the MCS p values, models with asterisks are included in the Model Confidence Set $\mathcal{M}_{\text{MCS}}$, indicating that their predictive accuracy is statistically indistinguishable from the best-performing model. Among the classical models, only ARMAX, LSTMX, and RCX are included in the MCS based on their p values, although their loss metrics are still higher than those of the quantum models. Additionally, QR1 is also included in the MCS, further highlighting the strong performance of quantum models. Examining the results, we also observe that including additional features sometimes enhances the model performance. For example, ARMAX and LSTMX exhibit lower loss values and higher MCS p values compared to their counterparts AR3 and LSTM, suggesting that the inclusion of characteristics improves the accuracy of forecasting in these models. However, this improvement is not consistent across all models. Notably, HARX performs slightly worse than HAR in both MSE and QLIKE metrics, indicating that additional features do not always contribute to better performance. This suggests that the effectiveness of including features depends on the model structure and the relevance of the features. When considering $S = 5$, namely, the long-term prediction, both quantum and classical reservoir demonstrated outstanding performance compared with other models. The performances of QR1 and QR2 are very similar to that of the classic reservoir computing. Considering that we only used a simple reservoir model, these indirectly demonstrate the capabilities of quantum reservoir.

Diebold-Mariano test results. Table III presents the Diebold-Mariano test statistic(s) and corresponding p value(s) for pairwise model comparisons, providing further evidence of the quantum models' outperformance. Analyzing the DM test results, we find that the quantum models (QR1 and QR2) significantly outperform most classical models. For instance, the p values for comparisons between QR2 and classical models like HAR and AR1 are effectively zero, indicating strong evidence against the null hypothesis of equal predictive

TABLE II. Model Confidence Set. Performance of classical and quantum models in predicting realized volatility with predicting S -ahead step. The MSE and QLIKE are presented alongside the MCS p values (P_{MCS}). Asterisks (*) indicate inclusion in the model confidence set. The QRC models outperform all classical models, with QR2 exhibiting the best performance across all measures.

$S = 1$ (entire sample: August 1997 to December 2017) (January 1950 to December 2017)				
Model	MSE		QLike	
	Loss of MSE	P_{MCS} of MSE	Loss of QLike	P_{MCS} of QLike
Classical Time Series Models				
HAR	0.1476	0.0004	2.0431	0.0008
HARX	0.1508	0.0004	2.2436	0.0008
AR1	0.1304	0.0065	1.7279	0.0050
AR3	0.1178	0.0936	1.5893	0.0861
ARMAX	0.1145	0.4406*	1.6196	0.4355*
LSTM	0.1295	0.0221	1.7909	0.0188
LSTMX	0.1185	0.4406*	1.7571	0.4355*
RC	0.1441	0.0084	2.1011	0.0061
RCX	0.1089	0.6086*	1.6480	0.6106*
QRC				
QR1	0.105	0.7603*	1.4427	0.7510*
QR2	0.103	1.0000*	1.4004	1.0000*
$S = 5$ (entire sample: January 1998 to December 2017) (January 1950 to December 2017)				
Model	MSE		QLike	
	Loss of MSE	P_{MCS} of MSE	Loss of QLike	P_{MCS} of QLike
Classical Time Series Models				
HAR	0.2143	0.1291*	2.9041	0.1275*
HARX	0.2934	0.0044	4.5800	0.0040
AR1	0.2642	0.0938*	3.4136	0.0937*
AR3	0.2134	0.1302*	2.8369	0.1297*
ARMAX	0.2134	0.1302*	3.0703	0.1297*
LSTM	0.1831	0.1291*	2.4600	0.1275*
LSTMX	0.2200	0.0925*	3.4512	0.0851*
RC	0.1528	1.0000*	2.0551	1.0000*
RCX	0.1667	0.6333*	2.4605	0.6251*
QRC				
QR1	0.1556	0.7642*	2.1518	0.7703*
QR2	0.1663	0.6333*	2.2332	0.6251*

TABLE III. Diebold-Mariano test. DM statistic values (lower triangular matrix) and p values (upper triangular matrix) for model comparisons. The statistic values closer to zero indicate that the pair of models have similar predictive accuracy. Lower p values suggest rejection of the null hypothesis of equal predictive ability.

	HAR	HARX	AR1	AR3	ARMAX	LSTM	LSTMX	RC	RCX	QR1	QR2
HAR		0.575	0.203	0.004	0.009	0.006	0.01	0.801	0.001	0	0
HARX	-0.562		0.116	0.001	0.003	0.004	0.002	0.597	0	0	0
AR1	1.275	1.577		0.002	0.077	0.925	0.301	0.154	0.009	0.001	0
AR3	2.929	3.313	3.12		0.652	0.065	0.937	0.003	0.17	0.036	0.014
ARMAX	2.636	2.999	1.775	0.451		0.134	0.657	0.006	0.393	0.12	0.111
LSTM	2.764	2.938	0.094	-1.856	-1.502		0.268	0.168	0.017	0.006	0.002
LSTMX	2.583	3.168	1.037	-0.079	-0.445	1.109		0.028	0.229	0.114	0.09
RC	0.252	0.53	-1.431	-3.032	-2.758	-1.383	-2.206		0	0	0
RCX	3.395	4.001	2.629	1.376	0.856	2.403	1.205	3.639		0.501	0.296
QR1	3.695	4.149	3.322	2.104	1.561	2.788	1.584	3.82	0.674		0.771
QR2	3.806	4.289	3.577	2.465	1.6	3.111	1.701	4.584	1.048	0.291	

accuracy. Additionally, the test statistics are positive and relatively large, further supporting the superiority of the quantum models. Interestingly, the comparison between QR1 and QR2 yields a high p value (0.771), suggesting no significant difference in predictive accuracy between the two quantum models. This implies that both quantum models perform comparably well, although QR2 has a slight edge based on the loss metrics in Table II. The results also reveal that among classical models, AR3 and ARMAX exhibit relatively better performance, with higher p values when compared to other classical models. However, they still fall short when compared to the quantum models. The results further illustrate the mixed impact of including additional features. For some model pairs, such as LSTM and LSTMX, the inclusion of features leads to significant improvements, as evidenced by lower test statistics and higher p values. In contrast, comparisons between AR3 and ARMAX yield p values that indicate no significant difference, implying that the additional features in ARMAX do not provide substantial benefits over AR3. Overall, the results suggest that while incorporating additional features can enhance model performance in certain cases, it does not universally guarantee better forecasts. The effectiveness of additional features appears to be model specific, highlighting the importance of selecting relevant variables that contribute meaningfully to volatility forecasting.

Overall, the combined evidence from the loss functions, MCS procedure, and DM test underscores the superior predictive performance of the QRC models over classical time series and machine learning models in forecasting realized volatility.

V. CONCLUSIONS AND OUTLOOK

QRC presents a promising paradigm for the modeling and forecasting of time series data. In this study, we have demonstrated how a specific instantiation of QRC that utilizes disordered spin systems as reservoirs can be effectively employed for forecasting realized volatility in financial markets. Our empirical evaluation reveals that even with a modest quantum architecture comprising only ten qubits and a moderate number of macroeconomic input features per instance, the proposed framework outperforms classical models in predictive accuracy. While we do not advance any claim to quantum supremacy in the rigorous sense of the term since this is a phenomenological study, the results obtained herein do indicate a potential future application of QRC toward applying them to other similar financial problems. Our protocol is especially suitable to use in real-life quantum computers based on trapped ion platforms, in particular, Refs. [8,127,128]. These setups offer excellent all-to-all connectivity between spins, as required in our specific choice of the quantum reservoir. We anticipate that alternative quantum reservoir architectures with different connectivity constraints may yield comparable predictive performance, although empirical verification would be necessary to substantiate this speculation.

Quantum computing for machine learning still faces significant challenges, both in hardware and algorithmic development. On the hardware side, current quantum devices are still in their infancy, constrained by limited qubit counts,

restricted connectivity, finite coherence times, and imperfect readout. Overcoming these limitations will require advances in error correction—a milestone likely still a decade away. Nevertheless, near-term quantum devices provide valuable opportunities to develop and test small-scale algorithms, paving the way for large-scale applications on future error-corrected quantum computers. From an algorithmic perspective, it remains unproven whether quantum computers can outperform classical machine learning methods on classical data. In this work, we demonstrate that quantum computers can indeed solve prediction tasks in stochastic time series. While our quantum reservoir learning approach shows advantages over several classical methods, we do not claim a definitive quantum advantage, as such a conclusion would require rigorous theoretical proof beyond the scope of this paper. Moreover, intrinsic limitations on QRC like the exponential concentration problem [110], whereby the predictions may become input-independent due to concentration of expectation values for many-qubit reservoirs, exist and have to be addressed in future work.

ACKNOWLEDGMENT

A.B. acknowledges support from the National Natural Science Foundation of China (Grants No. W2541020, No. 12274059, No. 12574528, and No. 1251101297).

DATA AVAILABILITY

The data that support the findings of this article are openly available [111].

APPENDIX A: QUANTUM COMPUTATION PRELIMINARIES

Let us briefly provide an overview of the quantum theory necessary for understanding the subsequent sections of this paper. Quantum mechanics offers a mathematical framework originally developed to explain several experimental phenomena that challenged classical Newtonian physics in the late 19th and early 20th centuries. Readers interested in a more comprehensive exploration are encouraged to consult standard references such as Dirac [129] and Nielsen and Chuang [130]. Throughout this paper, we employ the Dirac notation commonly used in quantum physics. To assist readers from financial and econometric backgrounds, we present a mini-dictionary in Table IV, where the departures of the Dirac notation from the standard econometric notation as laid out in Ref. [131] are codified. In modern axiomatic form, quantum mechanics is completely described through the following four postulates.

Postulate 1: Quantum states. The complete information about a quantum system is encoded in its state vector $|\psi\rangle$, which is a ray of the system Hilbert space $\mathcal{H}$. The quantum bit (or qubit for short) is the smallest nontrivial unit in quantum computing, similar to the bit in classical computing. It is physically implemented by a two-level quantum system, such as the spin of an electron. Its mathematical form can be represented by a state vector in a Hilbert space $\mathcal{H}$, such that

TABLE IV. A minidictionary of Dirac notation for econometricians.

Object	Econometric notation	Dirac notation	Comments
Scalar variable	a	a	
Complex conjugate of Scalar	$\bar{a}$ or a^*	a^*	
Vector (column)	$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$	$ a\rangle$	
Dual vector (row)	$\mathbf{a}^\top = [a_1^* \ a_2^* \ \cdots \ a_n^*]$	$\langle a $	$\langle a = (a\rangle)^\dagger$
Complex Conjugate of Vector	$\bar{\mathbf{a}}$	$ a\rangle^*$	Element-wise complex conjugate
Adjoint (Conjugate Transpose) of Vector	$\mathbf{a}^\dagger = (\mathbf{a}^*)^\top$	$\langle a $	
Norm of a Vector	$\ \mathbf{a}\ = \sqrt{\mathbf{a}^\dagger \mathbf{a}}$	$\ \mathbf{a}\ = \sqrt{\langle a a\rangle}$	
Inner Product	$\mathbf{a}'\mathbf{b}$	$\langle a b\rangle$	Results in a scalar
Outer Product	$\mathbf{a}\mathbf{b}'$	$ a\rangle\langle b $	Results in a matrix
Composite vector	$\mathbf{a} \otimes \mathbf{b}$	$ a\rangle \otimes b\rangle \equiv ab\rangle$	Produces a higher-dimensional vector
Composite vector*	$\mathbf{a}^{\otimes n} = \mathbf{a} \otimes \mathbf{a} \otimes \cdots \otimes \mathbf{a}$	$ a\rangle^{\otimes n} \equiv \underbrace{ a\rangle \otimes a\rangle \otimes \cdots \otimes a\rangle}_n$	Tensor product with n $ a\rangle$
Transpose of matrix $\mathbf{A}$	$\mathbf{A}^\top$	$\mathbf{A}^\top$	Reflect over main diagonal
Complex conjugate of Matrix $\mathbf{A}$	$\bar{\mathbf{A}}$	$\mathbf{A}^*$	Element-wise complex conjugate
Conjugate Transpose of Matrix $\mathbf{A}$	$\mathbf{A}^\dagger = (\mathbf{A}^*)^\top$	$\mathbf{A}^\dagger$	
Adjugate of Matrix $\mathbf{A}$	$\mathbf{A}^\#$	$\text{adj}(\mathbf{A})$	
Determinant of Matrix $\mathbf{A}$	$\det(\mathbf{A})$	$\det(\mathbf{A})$	Scalar value
Trace of Matrix $\mathbf{A}$	$\text{tr}(\mathbf{A})$	$\text{Tr}(\mathbf{A})$	Sum of diagonal elements
Identity matrix	$\mathbf{I}_n$	$\mathbb{I}$	
Zero Vector/matrix	$\mathbf{0}$	$\mathbf{0}$	
Expectation Value	$\mathbb{E}[X]$	$\langle X \rangle = \langle \psi X \psi \rangle$	$ \psi\rangle$ is the state vector
Variance	$\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$	$\langle (X - \langle X \rangle)^2 \rangle$	Measure of spread
Covariance	$\text{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$	$\langle XY \rangle - \langle X \rangle \langle Y \rangle$	
Hermitian operator	$\mathbf{A} = \mathbf{A}^\dagger$	$\mathbf{A} = \mathbf{A}^\dagger$	Self-adjoint operator
Unitary operator	$\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}_n$	$\mathbf{U}\mathbf{U}^\dagger = \mathbb{I}$	Preserves norms
Projection operator	$\mathbf{P} = \mathbf{a}\mathbf{a}'$	$\mathbf{P} = a\rangle\langle a $	$\mathbf{P}^2 = \mathbf{P}$
Commutator of operators $\mathbf{A}$ and $\mathbf{B}$	$[\mathbf{A}, \mathbf{B}] = \mathbf{AB} - \mathbf{BA}$	$[\mathbf{A}, \mathbf{B}] = \mathbf{AB} - \mathbf{BA}$	Measures noncommutativity
Anticommutator of operators $\mathbf{A}$ and $\mathbf{B}$	$\{\mathbf{A}, \mathbf{B}\} = \mathbf{AB} + \mathbf{BA}$	$\{\mathbf{A}, \mathbf{B}\} = \mathbf{AB} + \mathbf{BA}$	
Kronecker delta	δ_{ij}	δ_{ij}	$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$
Dirac delta function	$\delta(x - x')$	$\delta(x - x')$	Generalized function
Exponential of operator	$\exp(\mathbf{A})$	$e^{\mathbf{A}}$	Defined via power series
Fourier Transform of Function $f(x)$	$\mathcal{F}\{f(x)\} = \int_{-\infty}^{\infty} e^{-i\omega x} f(x) dx$	Same as econometric notation	
Tensor Product of Matrices $\mathbf{A}$ and $\mathbf{B}$	$\mathbf{A} \otimes \mathbf{B}$	$\mathbf{A} \otimes \mathbf{B}$	
Spectral Decomposition	$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\dagger$	$\mathbf{A} = \sum_i \lambda_i i\rangle\langle i $	λ_i are eigenvalues
Eigenvalue Equation	$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$	$\mathbf{A} v\rangle = \lambda v\rangle$	
Density Matrix	N/A	$\rho = \sum_i p_i i\rangle\langle i $	Represents mixed states
Born Rule	$P(x) = \psi(x) ^2$	$P(a) = \langle a \psi \rangle ^2$	Probability of outcome a
Heisenberg uncertainty principle	N/A	$\Delta X \Delta P \geq \frac{\hbar}{2}$	Fundamental limit
Pauli matrices	N/A	$\begin{cases} X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \\ Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \\ Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{cases}$	SU(2) rotation elements
Spin operators	N/A	$S_\alpha = \frac{\hbar}{2}\alpha$	$\alpha = \{X, Y, Z\}$
Rotation operator	N/A	$U(\theta, \hat{n}) = e^{-i\theta \hat{n} \cdot \mathbf{S} / \hbar}$	Rotates spin state

the single-qubit state is defined as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad (\text{A1})$$

where $|\alpha|^2 + |\beta|^2 = 1$ and $\alpha, \beta \in \mathbb{C}$. Note that $|0\rangle$ and $|1\rangle$ are the basis of a two-dimension Hilbert space. The vector representation of these bases is

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad (\text{A2})$$

and

$$\langle 0| = [1, 0], \quad \langle 1| = [0, 1]. \quad (\text{A3})$$

It is easy to see how qubits are fundamentally different from classical bits from this definition itself. By choosing α and β arbitrarily, the state vector $|\psi\rangle$ can be expressed in an arbitrary but unique linear combination of $|0\rangle$ and $|1\rangle$ simultaneously, while classical bits can only be in one of these states. This characteristic of quantum systems is called *superposition*.

Postulate 2: Composition of quantum systems. When we consider a joint quantum system consisting of multiple qubits, the state of this system is described by the tensor-product Hilbert space of each qubit $\mathcal{H}_i$. For example, if two qubits are described by their Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively, the joint system comprising both these systems as subsystems is described by the tensor-product Hilbert space $\mathcal{H}_1 \otimes \mathcal{H}_2$. Thus, the state of two qubits is

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle, \quad (\text{A4})$$

where we ignore the $\otimes$ between qubits for notational simplicity (e.g., $|0\rangle \otimes |0\rangle = |00\rangle$). In the same way, for n -qubit state, its formulation is

$$|\psi\rangle = \sum_{x_1, x_2, \dots, x_n}^{0,1} \alpha_{x_1, x_2, \dots, x_n} |x_1 x_2 \dots x_n\rangle. \quad (\text{A5})$$

The dimension of the n -qubit system is 2^n , and hence the tensor-product space is a 2^n -dimensional complex vector space $\mathbb{C}^{2^n}$. The dimension of the n -qubit system increases exponentially in the number n of the qubits. Quantum entanglement is another important property of quantum states that is different from the classical states. It occurs when a joint quantum system cannot be represented as a tensor product of the subsystems that make up it. Mathematically, if two quantum systems A and B are entangled, the joint system AB might be represented as

$$|\psi\rangle_{AB} = \alpha|0\rangle_A|0\rangle_B + \beta|1\rangle_A|1\rangle_B, \quad (\text{A6})$$

where $|\alpha|^2 + |\beta|^2 = 1$. One cannot then find any representation of $|\psi\rangle_A$ and $|\psi\rangle_B$ to satisfy $|\psi\rangle_{AB} = |\psi\rangle_A \otimes |\psi\rangle_B$. This interconnectedness leads to correlations between the qubits that are stronger than any classical correlation. These characteristics of quantum states—viz., *superposition*, *entanglement*, and *exponentially growing state space*, give quantum computing the potential to solve problems efficiently. Another way to describe the quantum state is the density operator of the density matrix, which is mathematically equivalent to the state vector approach, but it provides a much more convenient

language for thinking about some commonly encountered scenarios in quantum mechanics. The density operator of a pure n -qubit state is defined as

$$\begin{aligned} \rho &= |\psi\rangle\langle\psi| \\ &= \sum_{x_1, \dots, x_n, x'_1, \dots, x'_n}^{0,1} \alpha_{x_1, \dots, x_n} \alpha_{x'_1, \dots, x'_n}^* |x_1 x_2 \dots x_n\rangle \langle x'_1 x'_2 \dots x'_n|. \end{aligned} \quad (\text{A7})$$

Moreover, suppose a quantum system is in one of several states $|\psi_i\rangle$, where i is an index, with respective probabilities p_i . The pair $\{p_i, |\psi_i\rangle\}$ is named an ensemble of pure states. The density operator for the system is defined by the equation

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|, \quad (\text{A8})$$

where the normalization condition for probabilities implies $\sum_i p_i = 1$. Mathematically, we can recognize the pure states of Eq. (A7) from the mixed states in Eq. (A8) by the trace on the second order as $\text{tr}(\rho^2) = 1$ for pure states and $\text{tr}(\rho^2) < 1$ for mixed states.

Postulate 3: Quantum dynamics. The time evolution of a closed quantum system describes how the state of the system evolves over time, which is the most important way to deal with the information. The *closed* means the evolution of systems that do not interact with the rest of the world. The time evolution is governed by the Schrödinger equation, which is a fundamental partial differential equation in quantum mechanics.

The time-dependent Schrödinger equation is given by

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H |\psi(t)\rangle, \quad (\text{A9})$$

where i is the imaginary unit, $\hbar$ is the reduced Planck's constant, $|\psi(t)\rangle$ is the state vector at time t , and H is the Hamiltonian operator, which represents the total energy of the system. The solution of the Schrödinger equation for a time-independent Hamiltonian H is given by

$$|\psi(t)\rangle = e^{-\frac{i}{\hbar} H t} |\psi(0)\rangle, \quad (\text{A10})$$

where $|\psi(0)\rangle$ is the initial state of the system at time 0 and $e^{-\frac{i}{\hbar} H t}$ is the time evolution operator. In practice, it is common to absorb the factor $\hbar$ into H , effectively setting $\hbar = 1$. If the Hamiltonian H depends on time, the situation is more complex, but we do not use it in this paper. It is not difficult to see that the time evolution operator $e^{-\frac{i}{\hbar} H t}$ is a unitary matrix, so this time evolution is a unitary transformation, which preserves the norm of the quantum states, meaning the probability conservation. For simplicity, we usually use $U(t)$ to represent e^{-iHt} as

$$|\psi(t)\rangle = U(t) |\psi(0)\rangle. \quad (\text{A11})$$

In the context of digital quantum computing, time evolutions or quantum operations are also named quantum logical gates, similar to the logical gates in classical digital computing. Similarly, the time evolution implemented on the density operator can be easily defined as

$$\rho(t) = U(t) \rho(0) U^\dagger(t), \quad (\text{A12})$$

where $U^\dagger(t)$ is the transpose conjugate of $U(t)$ as $U^\dagger(t) = e^{iHt}$.

Postulate 4: Quantum measurement. Above, we described the evolution of a *closed* quantum system. However, to gain any information about the system, one has to necessarily consider an experimentalist as an external physical system observing it. Unlike classical mechanics, where in principle, such observations can be made without changing the observed system itself, quantum mechanics distinguishes itself by postulating that the quantum system undergoes irreversible change due to such a measurement. The general quantum measurement postulate is described by a collection measurement operator $\{M_m\}$. The index m refers to the measurement outcomes that may occur in the experiment with the probability p_m given by the Born rule $p_m = \langle \psi | M_m^\dagger M_m | \psi \rangle$, where $|\psi\rangle$ is the state being measured. To ensure the probabilities sum up to unity, the measurement operators satisfy the completeness equation

$$\sum_m M_m^\dagger M_m = I. \quad (\text{A13})$$

In most cases of quantum computing, we care about the quantum measurement of an observable. An observable, i.e., a dynamical variable like position or momentum, is described by a self-adjoint operator A (namely, $A = A^\dagger$), acting on the system Hilbert space $\mathcal{H}$. Via the spectral theorem, any observable has a spectral decomposition

$$A = \sum_m a_m |a_m\rangle \langle a_m|, \quad (\text{A14})$$

where a_m 's are the eigenvalue and $|a_m\rangle$'s are the eigenvectors of A . Note that $\langle a_m|$ is the dual of the vector $|a_m\rangle$, and is an element of the so-called dual space of $\mathcal{H}$ connected to $\mathcal{H}$ via the Riesz representation theorem. For a general quantum state $|\psi\rangle$, given in Eq. (A1), the dual state is given by $\langle \psi| = [\alpha^*, \beta^*]$ (see Table IV). According to quantum physics, the outcome of measuring the operator A on a quantum system described by state $|\psi\rangle$ probabilistically results in one of the eigenvalues a_m . The probability of each outcome is determined by the projector $M_m = |a_m\rangle \langle a_m|$ as $p_m = \langle \psi | M_m^\dagger M_m | \psi \rangle = |\langle a_m | \psi \rangle|^2$. Hence, the expectation value of the measurement is given by

$$\langle A \rangle = \sum_m p_m a_m = \langle \psi | A | \psi \rangle. \quad (\text{A15})$$

Note here that the Hilbert space is a linear vector space; thus, for any two states $|\psi_1\rangle$ and $|\psi_2\rangle$, the linear combination $\alpha|\psi_1\rangle + \beta|\psi_2\rangle$ is also a quantum state. Thus, in contrast to classical digital computation, where each bit (either 0 or 1) encodes one unit of classical information, theoretically an infinite number of superpositions are accessible to a single “qubit” (a two-level quantum system). From a computational standpoint, the superposition property of quantum mechanics opens up the possibility of encoding multiple gates in a single step. Indeed, famous quantum algorithms like Shor’s algorithm [103] for factoring, or Grover’s algorithm [132] for search, use this inherent parallelism of quantum mechanics to solve classical computational problems far more efficiently than the best possible classical algorithms known. However, in practice, these quantum properties are extremely fragile to noise, and reliable implementation of error-correction schemes is still in its infancy [133].

APPENDIX B: NUMERICAL SIMULATION METHOD

To simulate QRC on a classical computer, we explicitly model the system-state evolution. Since our QRC contains only ten qubits, we represent the quantum state using a density matrix ρ . The Hamiltonian dynamics are implemented via exact diagonalization, and the time-evolution operator is applied using a product formula $e^{-itH} \rho e^{itH}$, which is a standard method for numerically simulating quantum dynamics with small quantum systems.

In the partial-trace step, we obtain the reduced density matrix of the hidden state ρ_h by tracing out subsystem I from the joint state ρ_{Ih} , i.e.,

$$\rho_h = \text{Tr}_I(\rho_{Ih}).$$

Finally, the measurement outcome on each qubit is computed as the expectation value of the Pauli-Z operator,

$$\langle Z_i \rangle = \text{Tr}(\rho_{Ih} Z_i).$$

APPENDIX C: EXPONENTIAL CONCENTRATION OF QUANTUM MEASUREMENT

In this article [110], the authors discuss exponential concentration: As the circuit depth increases, the system size increases, entanglement increases, or the noise strength increases, the measurement expectation values converge

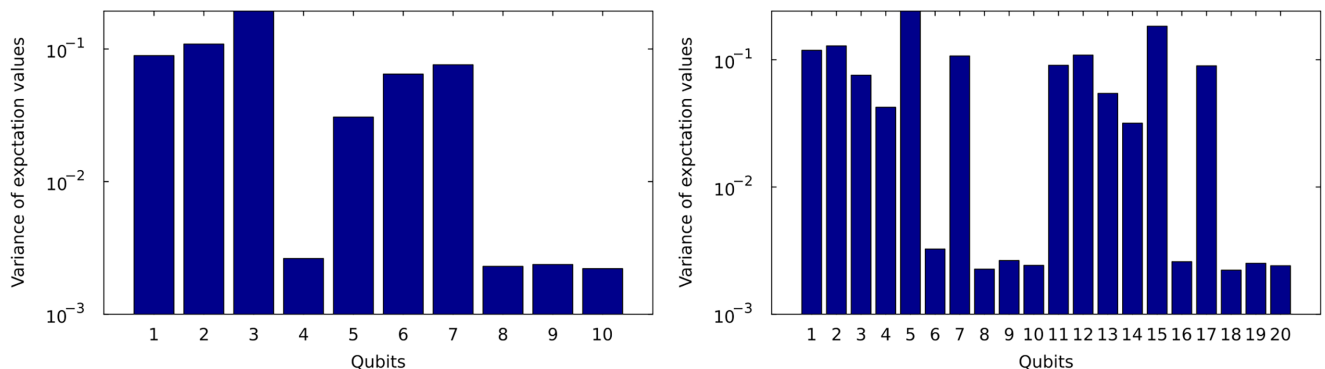


FIG. 9. Variance of the measured expectation values across different qubits (left: QR1 model; right: QR2 model).

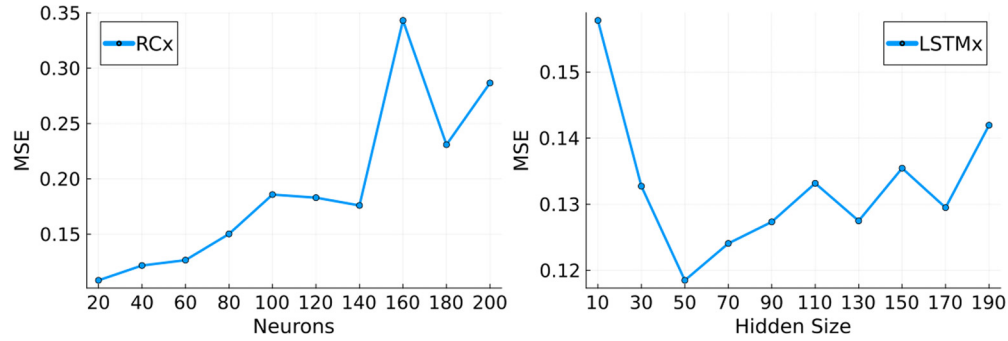


FIG. 10. The MSE of RCX (left) and LSTMX (right) as the hidden state size increases.

exponentially to a variable-independent value, such that the number of measurement shots grows exponentially to identify different expectation values. In our paper, however, we consider a specific example—namely, the application of QRC to predicting RV. We use a fixed system size (the number of qubits is 10) and circuit depth (three layers). By calculating the variance of the expectation value on each qubit, we find that it does not exhibit exponential concentration (see Fig. 9), thereby demonstrating the feasibility of QRC for this problem.

APPENDIX D: THE HYPERPARAMETERS OF CLASSICAL RESERVOIR COMPUTING AND LSTM

For the LSTMX, we used stack LSTM model with two layers. The optimizer is ADAM, with 0.001 learning rate. The number of iterations is 100 epochs, and the batch size is 64. After these hyperparameters are chosen, we train these LSTMs with different hidden sizes from 10 to 200 at intervals of 10. LSTM

model gets the best result (the lowest MSE) with 60 hidden size, and LSTMX model get the best performance (see Fig. 10, right) with 50 hidden size.

For classical reservoir computing, RC(X), apart from the number of neurons in reservoir, the other key hyperparameters are the leak rate (l_r), controlling the time constant of the reservoir, the spectral radius of W (sr), the maximum absolute eigenvalue of the reservoir, and the input scaling (is), the coefficient applied on W_{in} . We set them as $l_r = 0.6$, $sr = 0.9$, and $is = 0.1$ by trying a lot of configurations. This document [134] provides a detailed description of these hyperparameters. Based on these hyperparameters, we train RC(X) models with the number of neurons varying from 10 to 200 at intervals of 10. We get for the RC model, the number of neurons is 50 achieving best result, and RCX model, 20 achieving the best performance (see Fig. 10, left). As shown in Fig. 10, increasing the number of hidden nodes (neurons) does not always reduce errors. So, we selected the best model by comparing the performance of different hidden nodes (neurons).

-
- [1] P. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, NW Washington, DC, 1994), pp. 124–134.
 - [2] A. Steane, Quantum computing, *Rep. Prog. Phys.* **61**, 117 (1998).
 - [3] A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi, *Classical and Quantum Computation* (American Mathematical Society, Providence, RI, 2002).
 - [4] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, *et al.*, Quantum supremacy using a programmable superconducting processor, *Nature (London)* **574**, 505 (2019).
 - [5] Y. Wu, W.-S. Bao, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan, *et al.*, Strong quantum computational advantage using a superconducting quantum processor, *Phys. Rev. Lett.* **127**, 180501 (2021).
 - [6] Z. Han, C. Lyu, Y. Zhou, J. Yuan, J. Chu, W. Nuerbolati, H. Jia, L. Nie, W. Wei, Z. Yang, L. Zhang, Z. Zhang, C.-K. Hu, L. Hu, J. Li, D. Tan, A. Bayat, S. Liu, F. Yan, and D. Yu, Multilevel variational spectroscopy using a programmable quantum simulator, *Phys. Rev. Res.* **6**, 013015 (2024).
 - [7] W. Ren, W. Li, S. Xu, K. Wang, W. Jiang, F. Jin, X. Zhu, J. Chen, Z. Song, P. Zhang, *et al.*, Experimental quantum adversarial learning with programmable superconducting qubits, *Nat. Comput. Sci.* **2**, 711 (2022).
 - [8] D. Kielpinski, C. Monroe, and D. J. Wineland, Architecture for a large-scale ion-trap quantum computer, *Nature (London)* **417**, 709 (2002).
 - [9] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, P. Becker, H. Kaplan, A. V. Gorshkov, Z.-X. Gong, and C. Monroe, Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator, *Nature (London)* **551**, 601 (2017).
 - [10] M. Ringbauer, M. Meth, L. Postler, R. Stricker, R. Blatt, P. Schindler, and T. Monz, A universal qudit quantum processor with trapped ions, *Nat. Phys.* **18**, 1053 (2022).
 - [11] C. Monroe, W. C. Campbell, L.-M. Duan, Z.-X. Gong, A. V. Gorshkov, P. W. Hess, R. Islam, K. Kim, N. M. Linke, G. Pagano, P. Richerme, C. Senko, and N. Y. Yao, Programmable quantum simulations of spin systems with trapped ions, *Rev. Mod. Phys.* **93**, 025001 (2021).
 - [12] H. Bernien, S. Schwartz, A. Keesling, H. Levine, A. Omran, H. Pichler, S. Choi, A. S. Zibrov, M. Endres, M. Greiner, *et al.*, Probing many-body dynamics on a 51-atom quantum simulator, *Nature (London)* **551**, 579 (2017).
 - [13] S. Ebadi, T. T. Wang, H. Levine, A. Keesling, G. Semeghini, A. Omran, D. Bluvstein, R. Samajdar, H. Pichler, W. W. Ho,

- et al.*, Quantum phases of matter on a 256-atom programmable quantum simulator, *Nature (London)* **595**, 227 (2021).
- [14] H.-S. Zhong, Y.-H. Deng, J. Qin, H. Wang, M.-C. Chen, L.-C. Peng, Y.-H. Luo, D. Wu, S.-Q. Gong, H. Su, *et al.*, Phase-programmable Gaussian Boson sampling using stimulated squeezed light, *Phys. Rev. Lett.* **127**, 180502 (2021).
 - [15] L. Xiao, T. Deng, K. Wang, G. Zhu, Z. Wang, W. Yi, and P. Xue, Non-hermitian bulk–boundary correspondence in quantum dynamics, *Nat. Phys.* **16**, 761 (2020).
 - [16] K. Nemoto, M. Trupke, S. J. Devitt, A. M. Stephens, B. Scharfenberger, K. Buczak, T. Nöbauer, M. S. Everitt, J. Schmiedmayer, and W. J. Munro, Photonic architecture for scalable quantum information processing in diamond, *Phys. Rev. X* **4**, 031022 (2014).
 - [17] M. A. Quantum, M. Aghaee, A. Alcaraz Ramirez, Z. Alam, R. Ali, M. Andrzejczuk, A. Antipov, M. Astafev, A. Barzegar, B. Bauer, *et al.*, Interferometric single-shot parity measurement in InAs–Al hybrid devices, *Nature (London)* **638**, 651 (2025).
 - [18] J. Preskill, Quantum computing in the NISQ era and beyond, *Quantum* **2**, 79 (2018).
 - [19] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, Variational quantum algorithms, *Nat. Rev. Phys.* **3**, 625 (2021).
 - [20] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, [arXiv:1411.4028](https://arxiv.org/abs/1411.4028).
 - [21] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik, Quantum chemistry in the age of quantum computing, *Chem. Rev.* **119**, 10856 (2019).
 - [22] Q. Li, C. Mukhopadhyay, and A. Bayat, Fermionic simulators for enhanced scalability of variational quantum simulation, *Phys. Rev. Res.* **5**, 043175 (2023).
 - [23] A. Baiardi, M. Christandl, and M. Reiher, Quantum computing for molecular biology, *ChemBioChem* **24**, e202300120 (2023).
 - [24] D. Paulson, L. Dellantonio, J. F. Haase, A. Celi, A. Kan, A. Jena, C. Kokail, R. van Bijnen, K. Jansen, P. Zoller, and C. A. Muschik, Simulating 2D effects in lattice gauge theories on a quantum computer, *PRX Quantum* **2**, 030334 (2021).
 - [25] J. P. Crutchfield and K. Young, Inferring statistical complexity, *Phys. Rev. Lett.* **63**, 105 (1989).
 - [26] C. R. Shalizi and J. P. Crutchfield, Computational mechanics: Pattern and prediction, structure and simplicity, *J. Stat. Phys.* **104**, 817 (2001).
 - [27] M. Gu, K. Wiesner, E. Rieper, and V. Vedral, Quantum mechanics can reduce the complexity of classical models, *Nat. Commun.* **3**, 762 (2012).
 - [28] W. da Silva Coelho, L. Henriët, and L.-P. Henry, Quantum pricing-based column-generation framework for hard combinatorial problems, *Phys. Rev. A* **107**, 032426 (2023).
 - [29] S. Mugel, M. Abad, M. Bermejo, J. Sánchez, E. Lizaso, and R. Orús, Hybrid quantum investment optimization with minimal holding period, *Sci. Rep.* **11**, 19587 (2021).
 - [30] S. Mugel, C. Kuchkovsky, E. Sánchez, S. Fernández-Lorenzo, J. Luis-Hita, E. Lizaso, and R. Orús, Dynamic portfolio optimization with real datasets using quantum processors and quantum-inspired tensor networks, *Phys. Rev. Res.* **4**, 013006 (2022).
 - [31] J. Wiśniewska and M. Sawerwain, Variational quantum eigensolver for classification in credit sales risk, [arXiv:2303.02797](https://arxiv.org/abs/2303.02797).
 - [32] L. Leclerc, L. Ortiz-Gutiérrez, S. Grijalva, B. Albrecht, J. R. Cline, V. E. Elfving, A. Signoles, L. Henriët, G. Del Bimbo, U. A. Sheikh, *et al.*, Financial risk management on a neutral atom quantum processor, *Phys. Rev. Res.* **5**, 043117 (2023).
 - [33] N. Innan, A. Marchisio, M. Bennai, and M. Shafique, Lep-qnn: Loan eligibility prediction using quantum neural networks, in *Proceedings of the 2025 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, New York, 2025), pp. 1864–1872.
 - [34] R. Orús, S. Mugel, and E. Lizaso, Quantum computing for finance: Overview and prospects, *Rev. Phys.* **4**, 100028 (2019).
 - [35] D. Herman, C. Googin, X. Liu, A. Galda, I. Safro, Y. Sun, M. Pistoia, and Y. Alexeev, A survey of quantum computing for finance, [arXiv:2201.02773](https://arxiv.org/abs/2201.02773).
 - [36] A. S. Naik, E. Yeniaras, G. Hellstern, G. Prasad, and S. K. L. P. Vishwakarma, From portfolio optimization to quantum blockchain and security: A systematic review of quantum computing in finance, *Financ. Innov.* **11**, 88 (2025).
 - [37] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, *Science* **304**, 78 (2004).
 - [38] W. Li and K. L. E. Law, Deep learning models for time series forecasting: A review, *IEEE Access* **12**, 92306 (2024).
 - [39] J. Kim, H. Kim, H. Kim, D. Lee, and S. Yoon, A comprehensive survey of deep learning for time series forecasting: Architectural diversity and open challenges, *Artif. Intell. Rev.* **58**, 216 (2025).
 - [40] K. Fujii and K. Nakajima, Harnessing disordered-ensemble quantum dynamics for machine learning, *Phys. Rev. Appl.* **8**, 024030 (2017).
 - [41] L. C. G. Govia, G. J. Ribeill, G. E. Rowlands, H. K. Krovi, and T. A. Ohki, Quantum reservoir computing with a single nonlinear oscillator, *Phys. Rev. Res.* **3**, 013077 (2021).
 - [42] S. Das, G. L. Giorgi, and R. Zambrini, Quantum reservoir computing using Jaynes-Cummings model, [arXiv:2510.00171](https://arxiv.org/abs/2510.00171).
 - [43] T. Yasuda, Y. Suzuki, T. Kubota, K. Nakajima, Q. Gao, W. Zhang, S. Shimono, H. I. Nurdin, and N. Yamamoto, Quantum reservoir computing with repeated measurements on superconducting devices, [arXiv:2310.06706](https://arxiv.org/abs/2310.06706).
 - [44] M. C. Mackey and L. Glass, Oscillation and chaos in physiological control systems, *Science* **197**, 287 (1977).
 - [45] P. A. Moran and P. Whittle, Hypothesis testing in time series analysis, *J. R. Stat. Soc.* **114**, 579 (1951).
 - [46] F. Black and M. Scholes, The pricing of options and corporate liabilities, *J. Polit. Econ.* **81**, 637 (1973).
 - [47] S.-H. Poon and C. W. J. Granger, Forecasting volatility in financial markets: A review, *J. Econ. Lit.* **41**, 478 (2003).
 - [48] T. Bollerslev, Generalized autoregressive conditional heteroskedasticity, *J. Econom.* **31**, 307 (1986).
 - [49] T. G. Andersen, T. Bollerslev, F. X. Diebold, and H. Ebens, The distribution of realized stock return volatility, *J. Financ. Econ.* **61**, 43 (2001).
 - [50] O. E. Barndorff-Nielsen and N. Shephard, Econometric analysis of realized volatility and its use in estimating stochastic volatility models, *J. R. Stat. Soc.* **64**, 253 (2002).
 - [51] F. Corsi, A Simple approximate long-memory model of realized volatility, *J. Financ. Econ.* **7**, 174 (2009).

- [52] T. G. Andersen, T. Bollerslev, and F. X. Diebold, Roughing it up: Including jump components in the measurement, modeling, and forecasting of return volatility, *Rev. Econ. Stat.* **89**, 701 (2007).
- [53] A. J. Patton and K. Sheppard, Good volatility, bad volatility: Signed jumps and the persistence of volatility, *Rev. Econ. Stat.* **97**, 683 (2015).
- [54] T. Bollerslev, A. J. Patton, and R. Quaadvlieg, Exploiting the errors: A simple approach for improved volatility forecasting, *J. Econom.* **192**, 1 (2016).
- [55] C.-M. Kuan and H. White, Artificial neural networks: An econometric perspective, *Econom. Rev.* **13**, 1 (1994).
- [56] A. Habibnia, Essays in high-dimensional nonlinear time series analysis, Ph.D. thesis, London School of Economics and Political Science, 2016.
- [57] S. Gu, B. Kelly, and D. Xiu, Empirical asset pricing via machine learning, *Rev. Financ. Stud.* **33**, 2223 (2020).
- [58] A. Bucci, Realized volatility forecasting with neural networks, *J. Financ. Econ.* **18**, 502 (2020).
- [59] S. Gu, B. Kelly, and D. Xiu, Autoencoder asset pricing models, *J. Econom.* **222**, 429 (2021).
- [60] A. Habibnia and E. Maasoumi, Forecasting in big data environments: An adaptable and automated shrinkage estimation of neural networks (AAShNet), *J. Quantum Econ.* **19**, 363 (2021).
- [61] H. Zhu, L. Bai, L. He, and Z. Liu, Forecasting realized volatility with machine learning: Panel data perspective, *J. Empir. Finan.* **73**, 251 (2023).
- [62] J. Jiang, B. Kelly, and D. Xiu, (Re-)Imag(in)ing price trends, *J. Finance* **78**, 3193 (2023).
- [63] L. Chen, M. Pelger, and J. Zhu, Deep learning in asset pricing, *Manage. Sci.* **70**, 714 (2024).
- [64] E. Hillebrand and M. C. Medeiros, The benefits of bagging for forecast models of realized volatility, *Econom. Rev.* **29**, 571 (2010).
- [65] M. Fernandes, M. C. Medeiros, and M. Scharth, Modeling and predicting the CBOE market volatility index, *J. Bank. Finan.* **40**, 1 (2014).
- [66] F. Audrino and S. D. Knaus, Lassoing the HAR model: A model selection perspective on realized volatility dynamics, *Econom. Rev.* **35**, 1485 (2016).
- [67] R. R. Branco, A. Rubesam, and M. Zevallos, Forecasting realized volatility: Does anything beat linear models? *J. Empir. Finan.* **78**, 101524 (2024).
- [68] K. Christensen, M. Siggaard, and B. Veliyev, A machine learning approach to volatility forecasting, *J. Financ. Econ.* **21**, 1680 (2023).
- [69] C. Zhang, Y. Zhang, M. Cucuringu, and Z. Qian, Volatility forecasting with machine learning and intraday commonality, *J. Financ. Econ.* **22**, 492 (2024).
- [70] E. Ghysels, P. Santa-Clara, and R. Valkanov, Predicting volatility: Getting the most out of return data sampled at different frequencies, *J. Econom.* **131**, 59 (2006).
- [71] E. S. Gunnarsson, H. R. Isern, A. Kaloudis, M. Rissstad, B. Vigdel, and S. Westgaard, Prediction of realized volatility and implied volatility indices using AI and machine learning: A review, *Int. Rev. Financ. Anal.* **93**, 103221 (2024).
- [72] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.* **9**, 1735 (1997).
- [73] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, Massachusetts, 2016).
- [74] M. Lukoševičius and H. Jaeger, Reservoir computing approaches to recurrent neural network training, *Comput. Sci. Rev.* **3**, 127 (2009).
- [75] P. Mujal, R. Martínez-Peña, G. L. Giorgi, M. C. Soriano, and R. Zambrini, Time-series quantum reservoir computing with weak and projective measurements, *npj Quantum Inf.* **9**, 16 (2023).
- [76] J. García-Beni, G. L. Giorgi, M. C. Soriano, and R. Zambrini, Scalable photonic platform for real-time quantum reservoir computing, *Phys. Rev. Appl.* **20**, 014051 (2023).
- [77] G. Llodrà, P. Mujal, R. Zambrini, and G. L. Giorgi, Quantum reservoir computing in atomic lattices, *Chaos Solit. Fractals* **195**, 116289 (2025).
- [78] J. Garcia-Beni, G. L. Giorgi, M. C. Soriano, and R. Zambrini, Quantum reservoir computing for time series processing, in *Quantum Communications and Quantum Imaging XXII*, edited by K. S. Deacon and R. E. Meyers (SPIE, San Diego, California, 2024), Vol. PC13148, p. PC131480E.
- [79] S. Thakkar, S. Kazdaghi, N. Mathur, I. Kerenidis, A. J. Ferreira-Martins, and S. Brito, Improved financial forecasting via quantum machine learning, *Quantum Mach. Intell.* **6**, 27 (2024).
- [80] M. A. Rivera-Ruiz, A. Mendez-Vazquez, and J. M. López-Romero, Time series forecasting with quantum machine learning architectures, in *Advances in Computational Intelligence*, edited by O. Pichardo Lagunas, J. Martínez-Miranda, and B. Martínez Seis (Springer, Cham, 2022), pp. 66–82.
- [81] J. Settino, L. Salatino, L. Mariani, M. Channab, L. Bozzolo, S. Vallisa, P. Barillà, A. Policicchio, N. L. Gullo, A. Giordano, *et al.*, Memory-augmented hybrid quantum reservoir computing, [arXiv:2409.09886](https://arxiv.org/abs/2409.09886).
- [82] S. Nerenberg, O. D. Neill, G. Marcucci, and D. Faccio, Photon number-resolving quantum reservoir computing, *Opt. Quantum* **3**, 201 (2025).
- [83] A. Suprano, D. Zia, L. Innocenti, S. Lorenzo, V. Cimini, T. Giordani, I. Palmisano, E. Polino, N. Spagnolo, F. Sciarrino, G. M. Palma, A. Ferraro, and M. Paternostro, Experimental property reconstruction in a photonic quantum extreme learning machine, *Phys. Rev. Lett.* **132**, 160802 (2024).
- [84] A. H. Abbas and I. S. Maksymov, Reservoir computing using measurement-controlled quantum dynamics, *Electronics* **13**, 1164 (2024).
- [85] D. Alaminos, M. Belen Salas, and M. A. Fernandez-Gamez, Forecasting stock market crashes via real-time recession probabilities: A quantum computing approach, *Fractals* **30**, 2240162 (2022).
- [86] T. G. Andersen and T. Bollerslev, Answering the skeptics: Yes, standard volatility models do provide accurate forecasts, *Int. Rev. Econ.* **39**, 885 (1998).
- [87] W. K. Newey and K. D. West, A simple, positive semi-definite, heteroskedasticity and autocorrelationconsistent covariance matrix, *Econometrica* **55**, 703 (1987).
- [88] P. R. Hansen, Z. Huang, and H. H. Shek, Realized GARCH: A joint model for returns and realized measures of volatility, *J. Appl. Econom.* **27**, 877 (2012).
- [89] M. McAleer and M. C. Medeiros, Realized volatility: A review, *Econom. Rev.* **27**, 10 (2008).

- [90] T. Fischer and C. Krauss, Deep learning with long short-term memory networks for financial market predictions, *Eur. J. Oper.* **270**, 654 (2018).
- [91] J. B. Butcher, D. Verstraeten, B. Schrauwen, C. R. Day, and P. W. Haycock, Reservoir computing and extreme learning machines for non-linear time-series data analysis, *Neural Netw.* **38**, 76 (2013).
- [92] J. Zou, Y. Han, and S.-S. So, Overview of artificial neural networks, in *Artificial Neural Networks: Methods and Applications*, edited by D. J. Livingstone (Humana Press, Totowa, NJ, 2009), pp. 14–22.
- [93] W. Maass, T. Natschl ger, and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, *Neural Comput.* **14**, 2531 (2002).
- [94] O. Yilmaz, Symbolic computation using cellular automata-based hyperdimensional computing, *Neural Comput.* **27**, 2661 (2015).
- [95] N. McDonald, Reservoir computing & extreme learning machines using pairs of cellular automata rules, in *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)* (IEEE, Anchorage, AK, 2017), pp. 2429–2436.
- [96] T. Yamane, Y. Katayama, R. Nakane, G. Tanaka, and D. Nakano, Wave-based reservoir computing by synchronization of coupled oscillators, in *Neural Information Processing*, edited by S. Arik, T. Huang, W. K. Lai, and Q. Liu (Springer International, Cham, 2015), pp. 198–205.
- [97] S. Roy, A. Banerjee, and A. Basu, Liquid state machine with dendritically enhanced readout for low-power, neuromorphic VLSI implementations, *IEEE Trans. Biomed. Circuits Syst.* **8**, 681 (2014).
- [98] A. Katumba, J. Heyvaert, B. Schneider, S. Uvin, J. Dambre, and P. Bienstman, Low-loss photonic reservoir computing with multimode photonic integrated circuits, *Sci. Rep.* **8**, 2653 (2018).
- [99] F. Duport, B. Schneider, A. Smerieri, M. Haelterman, and S. Massar, All-optical reservoir computing, *Opt. Express* **20**, 22783 (2012).
- [100] C. Mesaritakis, V. Papataxiarhis, and D. Syvridis, Micro ring resonators as building blocks for an all-optical high-speed reservoir-computing bit-pattern-recognition system, *J. Opt. Soc. Am. B* **30**, 3048 (2013).
- [101] A. Dejonckheere, F. Duport, A. Smerieri, L. Fang, J.-L. Oudar, M. Haelterman, and S. Massar, All-optical reservoir computer based on saturation of absorption, *Opt. Express* **22**, 10868 (2014).
- [102] A. Goudarzi, M. R. Lakin, and D. Stefanovic, DNA reservoir computing: a novel molecular computing approach, in *DNA Computing and Molecular Programming*, edited by D. Soloveichik and B. Yurke (Springer International, Cham, 2013), pp. 76–89.
- [103] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* **26**, 1484 (1997).
- [104] S. Lloyd, Universal quantum simulators, *Science* **273**, 1073 (1996).
- [105] S. Ghosh, A. Opala, M. Matuszewski, T. Paterek, and T. C. Liew, Quantum reservoir processing, *npj Quantum Inf.* **5**, 35 (2019).
- [106] R. A. Bravo, K. Najafi, X. Gao, and S. F. Yelin, Quantum reservoir computing using arrays of Rydberg atoms, *PRX Quantum* **3**, 030325 (2022).
- [107] M. Negoro, K. Mitarai, K. Fujii, K. Nakajima, and M. Kitagawa, Machine learning with controllable quantum dynamics of a nuclear spin ensemble in a solid, [arXiv:1806.10910](https://arxiv.org/abs/1806.10910).
- [108] T. Kubota, Y. Suzuki, S. Kobayashi, Q. H. Tran, N. Yamamoto, and K. Nakajima, Temporal information processing induced by quantum noise, *Phys. Rev. Res.* **5**, 023057 (2023).
- [109] P. Mujal, R. Mart nez-Pe a, J. Nokkala, J. Garc a-Beni, G. L. Giorgi, M. C. Soriano, and R. Zambrini, Opportunities in quantum reservoir computing and extreme learning machines, *Adv. Quantum Technol.* **4**, 2100027 (2021).
- [110] W. Xiong, G. Facelli, M. Sahebi, O. Agnel, T. Chotibut, S. Thanasilp, and Z. Holmes, On fundamental aspects of quantum extreme learning machines, *Quantum Mach. Intell.* **7**, 20 (2025).
- [111] Ranaroussi/yfinance: Download market data from yahoo! Finance’s API, GitHub (2025), <https://github.com/ranaroussi/yfinance>.
- [112] G. W. Schwert, Why does stock market volatility change over time? *J. Finance* **44**, 1115 (1989).
- [113] R. F. Engle, E. Ghysels, and B. Sohn, On the economic sources of stock market volatility, in *AFA 2008 New Orleans Meetings Paper* (Social Science Research Network, Rochester, NY, 2008).
- [114] I. Welch and A. Goyal, A comprehensive look at the empirical performance of equity premium prediction, *Rev. Financ. Stud.* **21**, 1455 (2008).
- [115] E. F. Fama and K. R. French, Common risk factors in the returns on stocks and bonds, *J. Financ. Econ.* **33**, 3 (1993).
- [116] E. F. Fama and K. R. French, Multifactor explanations of asset pricing anomalies, *J. Finance* **51**, 55 (1996).
- [117] Y. Feng, Y. Zhang, and Y. Wang, Out-of-sample volatility prediction: Rolling window, expanding window, or both? *J. Forecasting* **43**, 567 (2024).
- [118] A. J. Patton and K. Sheppard, Good volatility, bad volatility: Signed jumps and the persistence of volatility, *J. Bus. Econ. Stat.* **33**, 22 (2015).
- [119] M. H. Pesaran and A. Timmermann, Selection of estimation window in the presence of breaks, *J. Econom.* **137**, 134 (2007).
- [120] L. S. Shapley, A value for n -person games, in *Contribution to the Theory of Games* (Princeton University Press, Princeton, New Jersey, 1953), Vol. 2, pp. 307–318.
- [121] S. M. Lundberg and S.-I. Lee, A unified approach to interpreting model predictions, in *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17* (Curran Associates, Red Hook, NY, 2017), pp. 4768–4777.
- [122] E.  trumbelj and I. Kononenko, Explaining prediction models and individual predictions with feature contributions, *Knowl. Inf. Syst.* **41**, 647 (2014).
- [123] P. R. Hansen and A. Lunde, A forecast comparison of volatility models: Does anything beat a GARCH(1, 1)? *J. Appl. Econom.* **20**, 873 (2005).

- [124] A. J. Patton, Volatility forecast comparison using imperfect volatility proxies, *J. Econom.* **160**, 246 (2011).
- [125] P. R. Hansen, A. Lunde, and J. M. Nason, The model confidence set, *Econometrica* **79**, 453 (2011).
- [126] F. X. Diebold and R. S. Mariano, Comparing predictive accuracy, *J. Bus. Econ. Stat.* **13**, 253 (1995).
- [127] A. Erhard, H. Poulsen Nautrup, M. Meth, L. Postler, R. Stricker, M. Stadler, V. Negnevitsky, M. Ringbauer, P. Schindler, H. J. Briegel, *et al.*, Entangling logical qubits with lattice surgery, *Nature (London)* **589**, 220 (2021).
- [128] M. Akhtar, F. Bonus, F. Lebrun-Gallagher, N. Johnson, M. Siegele-Brown, S. Hong, S. Hile, S. Kulmiya, and W. Hensinger, A high-fidelity quantum matter-link between ion-trap microchip modules, *Nat. Commun.* **14**, 531 (2023).
- [129] P. A. M. Dirac, *The Principles of Quantum Mechanics*, 4th ed., International Series of Monographs on Physics, Vol. 27 (Oxford University Press, Oxford, 2010).
- [130] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, Cambridge, 2012).
- [131] K. Abadir and J. Magnus, Notation in econometrics: a proposal for a standard, *Econom. J.* **5**, 76 (2002).
- [132] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96* (ACM Press, New York, NY, 1996), pp. 212–219.
- [133] R. Acharya, D. A. Abanin, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, J. Atalaya, R. Babbush, D. Bacon, B. Ballard, J. C. Bardin, J. Bausch, A. Bengtsson, and other, Quantum error correction below the surface code threshold, *Nature (London)* **638**, 920 (2025).
- [134] Understand and optimize ESN hyperparameters, https://reservoirpy.readthedocs.io/en/latest/user_guide/hyper.html.