scientific reports

OPEN

Contextual quantum neural networks for stock price prediction

Sharan Mourya[1,2]✉, Hannes Leipold[2] & Bibhas Adhikari[2]

In this paper, we apply quantum machine learning (QML) to predict the distribution of stock prices of multiple assets using a contextual quantum neural network. Our approach captures recent trends to predict future stock price distributions, moving beyond traditional models that focus on entire historical data. Utilizing the principles of quantum superposition, we introduce a new training technique called the quantum batch gradient update (QBGU), which accelerates the standard stochastic gradient descent (SGD) in quantum applications and improves convergence. Consequently, we propose a quantum multi-task learning (QMTL) architecture, specifically, the share-and-specify ansatz, that integrates task-specific operators controlled by quantum labels, enabling the simultaneous and efficient training of multiple assets on the same quantum circuit as well as enabling efficient portfolio representation with logarithmic overhead in the number of qubits. Through extensive experimentation on S&P 500 data for Apple, Google, Microsoft, and Amazon stocks, we demonstrate that our approach outperforms quantum single-task learning (QSTL) models by effectively capturing inter-asset correlations. Our findings highlight the transformative potential of QML in financial applications, paving the way for more advanced, resource-efficient quantum algorithms in stock price prediction and other complex financial modeling tasks.

Keywords Quantum machine learning, Quantum neural networks, Quantum finance, Quantum multi-task learning

Quantum computing is a computational paradigm that transcends the limitations of classical computing by harnessing the principles of quantum superposition and entanglement. These unique features enable quantum computers to tackle complex problems faster than classical systems can[1,2,4,5]. Owing to the potential exponential scaling of computational power with the number of qubits, quantum computing is expected to revolutionize diverse sectors, including medicine, engineering, energy, and finance[6,7]. Despite its immense potential, building a fully functional quantum computer is a monumental challenge that could take years, if not decades, to achieve a clear computational advantage over classical computers[8,9]. However, the near-term applications of quantum computing are particularly promising in fields like finance, where its ability to process vast amounts of complex data with fewer resources is transformative. Even with today's noisy intermediate scale quantum (NISQ) devices - limited by a small number of qubits and short coherence times[9] - quantum methods can provide approximate solutions to certain financial problems, making them highly relevant in the immediate future[10].

Machine learning has become essential for financial tasks like, asset management[10], risk analysis[11], crash detection[12], and portfolio optimization[13]. The ability of machine learning algorithms to analyze massive datasets, recognize patterns, and make fast predictions provides a significant competitive edge. Quantum machine learning (QML)[14] emerges at the intersection of these fields, combining quantum computing's ability to process and represent complex states efficiently with the powerful predictive tools of machine learning[15]. With the exponential growth of financial data, current machine learning systems are quickly reaching the boundaries of classical computational models. In this context, quantum algorithms present a promising alternative by offering faster or higher quality solutions for specific classes of problems. Additionally, breakthroughs in quantum learning theory suggest that, under certain conditions, there is a provable distinction between classical and quantum learnability[16]. This implies that problems deemed challenging for classical systems could see significant improvements through the adoption of QML approaches.

Quantum machine learning can be broadly categorized into two main components: parametric quantum circuit (PQC) optimization and classical-to-quantum information encoding[10]. These categories represent two core components of QML algorithms and workflows, each addressing a different aspect of how classical data interacts with quantum systems and how quantum models are trained. PQCs are quantum circuits that contain tunable parameters, interpretable as weights of a quantum neural network. These parameters are adjusted iteratively to minimize or maximize a cost function, similar to how classical machine learning algorithms

[1]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, US. [2]Fujitsu Research of America, Santa Clara, CA, US. ✉email: sharanmourya7@gmail.com

optimize parameters during training. PQCs can further be classified into two categories: variational quantum circuits (VQC) and quantum neural networks (QNN). VQCs involve a hybrid architecture where a classical computer works alongside a quantum computer in a loop, while QNNs consist of circuit architectures tailored to specific problems.

Recently, there have been a surge of research interest in these areas. For instance, hybrid architectures using parameterized quantum circuits in combination with classical optimization loops have been successfully deployed for classification tasks[17], while support vector machines (SVM) have been utilized for data classification[18]. In another study, quantum state space was used as the feature space to improve the learnability of QML and achieve quantum advantage in classification tasks[19]. Quantum versions of machine learning models, such as Boltzmann machines[20], recurrent neural networks[21], generative adversarial neural networks[22], reinforcement learning[23], and reservoir computing[24], have been extensively studied.

In finance, quantum machine learning has been applied to various tasks, including options pricing[25], time-series forecasting[27], and stock price prediction[29]. A notable example is a hybrid architecture developed for financial predictions[34], while quantum Wasserstein generative adversarial neural networks were employed for time-series predictions on the S&P 500[35]. Unsupervised quantum machine learning has also been explored for clustering and fraud detection[36]. On the other hand, significant progress has also been made in loading classical information onto quantum states. For example, quantum adversarial neural networks have been utilized to load random distributions onto quantum circuits using feature maps[37], and quantum Wasserstein GANs have achieved similar tasks with a gradient penalty, improving performance over previous approaches[35].

While many financial computational problems like portfolio management and risk analysis require training across multiple assets, only few studies have addressed this need. For instance, joint learning of two distributions was achieved in[38], though its direct application to financial problems like stock price prediction remains limited due to computational expense and the reliance on feature maps, which will be further explained in this paper. A related study[39] applied quantum reservoir computing and multi-task learning[40], which is hindered by the complexity of quantum reservoir systems, lacking generalization.

In this paper, we aim to address the challenge of training a parameterized quantum circuit over multiple assets on a single quantum device by utilizing minimal resources and optimizing various components of the training process. The contributions of this work are as follows:

1. We adopted fidelity loss over quantum representations of the entire data distribution and employed a training technique, quantum batch gradient update (QBGU), that loads the amplitude encoded context distribution and empirical context-continuation distribution to accelerate convergence and improve convergence quality compared to training by stochastic gradient descent (SGD) through expensive reconstruction of classical distributions.
2. We analyzed context-based stock price prediction for various companies using several training paradigms, including parameter shift and (simulation supported) backpropagation, across different loss functions.
3. We developed a new quantum multi-task learning (QMTL) architecture - *share-and-specify* ansatz - for predicting the stock price distribution over a portfolio of assets. By achieving logarithmic overhead in the number of assets in the portfolio and controllable scaling in the size of the context, we can load highly non-trivial non-static distributions on quantum devices, enabling the usage of potential quantum advantageous algorithms such as approaches utilizing quantum amplitude estimation[3] like quantum risk analysis[30] or other quantum algorithms for speed-up like quantum linear system solving[2] at inference time.

This paper is organized as follows: Section II gives a brief background of the required terminology used throughout this paper. Section III gives an overview of loading classical data onto quantum registers. Section IV and V introduces quantum single-task and multi-task learning respectively followed by numerical simulations in Section VI.

Background

In this section, we introduce all the background information needed to understand this paper including the time-series prediction model and contextual Quantum Neural Networks (QNN) for modeling asset futures.

Time-series prediction

A financial asset price prediction model is a time-series forecasting model designed to predict future asset prices by leveraging historical numeric price data and additional contextual information[26]. In this section, we outline the mathematical framework and notations that will be used subsequently to develop a quantum machine learning (QML) model for multi-asset price prediction. For any given financial asset, the associated *contextual string* represents a sequence of numeric values corresponding to the asset over a specific time frame. Typically, this sequence is formed by considering a series of consecutive past asset prices within the designated time window. In this work, we utilize asset price data derived from the S&P 500 index, which provides historical stock prices for corporations listed on the index. Stock prices are inherently volatile, influenced by market activity, news, and other external factors. These short-term fluctuations often introduce noise that can adversely affect model performance. Therefore, in this paper, we preprocess the stock prices by computing the finite difference between consecutive stock prices to capture the moving difference (returns). This is followed by performing a moving average, smoothing out these short-term fluctuations, revealing the underlying trends.

The value or movement of an asset at a time, denoted by *t*, is represented by a random variable X^t and we denote $X^t = x^t$ when it attains a value x^t. Thus, a context string of $T \geq 1$ numeric values is represented by a random vector $\boldsymbol{X}^{(T)} = (X^1, X^2, \ldots, X^T)$. Then the task of a prediction model *f* is to predict the value(s) of $\boldsymbol{X}^{(T+\tau)} \backslash \boldsymbol{X}^{(T)} := (X^{T+1}, X^{T+2}, \ldots, X^{T+\tau})$ as a probability distribution $f(\boldsymbol{X}^{(T+\tau)\backslash(T)}; \boldsymbol{X}^{(T)})$ for some values of the future time $\tau \geq 0$. To be specific, given $\boldsymbol{X}^{(T)} = \boldsymbol{x}^{(T)}$, the task of the prediction model is to assign probabilities to future states $\boldsymbol{x}^{(T+\tau)} \backslash \boldsymbol{x}^{(T)} := (x^{T+1}, x^{T+2}, \ldots, x^{T+\tau})$, the possible asset-price at the future times between $T+1$ and $T+\tau$, as $f(\boldsymbol{x}^{(T+\tau)\backslash T}; \boldsymbol{x}^{(T)})$ and the efficiency of *f* is determined by the closeness of a distance between $f(\boldsymbol{x}^{(T+\tau)\backslash T}; \boldsymbol{x}^{(T)})$ and the observed distribution over $\boldsymbol{x}^{(T+\tau)} \backslash \boldsymbol{x}^{(T)}$ to zero. Thus the design of a prediction model $f(\boldsymbol{X}^{(T+\tau)\backslash(T)}; \boldsymbol{X}^{(T)}, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$ is concerned with modeling *f* such that the loss function $\mathcal{L}(f(\boldsymbol{x}^{(T+\tau)\backslash(T)}; \boldsymbol{x}^{(T)}), \boldsymbol{x}^{(T+\tau)} \backslash \boldsymbol{x}^{(T)})$, which estimates a notion of closeness between the probability distribution $f(\boldsymbol{X}^{(T+\tau)\backslash(T)}; \boldsymbol{X}^{(T)})$ and observed frequencies $\boldsymbol{X}^{(T+\tau)} \backslash \boldsymbol{X}^{(T)}$ is minimized for all or a collection of assets in a financial market. In most occasions dealing with time-series data models, the loss function is considered as the mean squared error (MSE), binary cross entropy or any other custom function specifically designed for the task.

In our proposal of developing a QML model for multi-asset price prediction, it is customary to encode the context string data $\boldsymbol{x}^{(T)}$ into a quantum state which will be evolved under a unitary transformation. We define the T-qudit quantum state as

$$\left|\boldsymbol{x}^{(T)}\right\rangle = \left|x^1\right\rangle \otimes \left|x^2\right\rangle \otimes \cdots \otimes \left|x^T\right\rangle \tag{1}$$

to encode $\boldsymbol{x}^{(T)}$ after encoding the context data point x^t into its corresponding qudit $\left|x^t\right\rangle$ for $1 \leq t \leq T$. Here, $\otimes$ denotes the Kronecker product (also called tensor product) of two vectors. In this framework, the state of the qudit, $\left|x^t\right\rangle$, represents the quantized return of an asset at time *t*. The mapping of returns to the orthogonal states $\left|0\right\rangle, \left|1\right\rangle, \ldots, \left|d-1\right\rangle$ is determined by dividing the range of possible returns into *d* discrete intervals. Each interval corresponds to one of the orthogonal basis states of the qudit. For instance, let the minimum and maximum returns between $1 \leq t \leq T$ be $x_{\min}$ and $x_{\max}$, then the price range is divided into *d* equal intervals of length

$$\Delta x = \frac{x_{\max} - x_{\min}}{d-1}.$$

For any price x^t, its corresponding qudit state $\left|x^t\right\rangle$ is determined as:

$$\left|x^t\right\rangle = \left|i^t\right\rangle, \quad i^t = \left\lfloor \frac{x^t - x_{\min}}{\Delta x} \right\rfloor, \quad i^t \in \{0, 1, \ldots, d-1\}. \tag{2}$$

This mapping ensures that each basis state $\left|i^t\right\rangle$ represents a specific quantized interval of prices. Over time, as the asset price evolves, the state of the qudit $\left|x^t\right\rangle \in \mathbb{C}^d$ transitions between the basis states. These transitions can be modeled using a QNN $f(\boldsymbol{X}^{(T+\tau)/(T)}; \boldsymbol{X}^{(T)}, \boldsymbol{\theta})$ with trainable parameters $\boldsymbol{\theta}$ designed to capture the stochastic behavior of returns and generate an approximation to the quantum state $\sum_{\boldsymbol{x}^{(T+\tau)\backslash(T)} \in \{0,\ldots,d-1\}^\tau} \sqrt{f(\boldsymbol{x}^{(T+\tau)\backslash(T)}; \boldsymbol{x}^{(T)})} \left|\boldsymbol{x}^{(T+\tau)}\right\rangle$ such that direct measurement samples from the underlying distribution. Here, unlike the computational basis state $\left|\boldsymbol{x}^{(T)}\right\rangle$, the output vector $\left|\boldsymbol{y}^{(T+\tau)}\right\rangle$ is a superposition state of the form

$$\left|\boldsymbol{y}^{(T+\tau)}\right\rangle = \sum_{\phi \in \{0,1,\ldots,d-1\}^{T+\tau}} c(\phi) \left|\phi\right\rangle, \tag{3}$$

where $\left|\phi\right\rangle = \left|\phi^1\right\rangle \otimes \left|\phi^2\right\rangle \cdots \otimes \left|\phi^{T+\tau}\right\rangle$ is a $T+\tau$-qudit basis state with $\phi^t \in \{0, \ldots, d-1\}$ and $c(\phi) \in \mathbb{C}$ such that $\sum_\phi |c(\phi)|^2 = 1$. Now, the prediction is a density operator after taking the partial trace over the context:

$$\rho^{(T+\tau)\backslash(T)} = \mathrm{Tr}_{1\ldots T} \left|\boldsymbol{y}^{(T+\tau)}\right\rangle \left\langle \boldsymbol{y}^{(T+\tau)}\right|. \tag{4}$$

Note that the prediction of future returns constitutes only the last τ qudits. However, in our situation, the QNN may make transformations to the context qudits ($|\boldsymbol{x}^{(T)}\rangle$), making it not suitable for reuse for repeated predictions. To circumvent that, we require the whole input and output vectors ($|\boldsymbol{y}^{(T+\tau)}\rangle$, $|\boldsymbol{x}^{(T+\tau)}\rangle$) to be involved in the loss function to make sure that the context qudits are unchanged. With this, we can approximate the prediction as a wavefunction:

$$\left|\boldsymbol{y}^{(T+\tau)}\right\rangle \approx \sum_{\phi \in \{0,1,\ldots,d-1\}^\tau} c(\phi) \left|\boldsymbol{x}^{(T)}\right\rangle \left|\phi^{T+1}\right\rangle \ldots \left|\phi^{T+\tau}\right\rangle, \tag{5}$$

where we note that the context state $\left|\boldsymbol{x}^{(T+\tau)}\right\rangle$ leads to a prediction $\sum_{\phi\in\{0,1,\ldots,d-1\}^{\tau}} c(\phi)\left|\phi^{T+1}\right\rangle\ldots\left|\phi^{T+\tau}\right\rangle$, which is a superposition of all possible outcomes with different probabilities. Each probable state $\left|\phi^{T+t}\right\rangle$ at a time t $(T+1\leq t\leq T+\tau)$ can be mapped to the numerical stock movement of an asset by the transformation

$$\left|\phi^{T+t}\right\rangle \rightarrow x_{\min}+\phi^{T+t}\cdot\Delta x. \tag{6}$$

These mapped values can then be used to calculate the expected value or movement of the stock prices by combining with the probability ($|c(\phi)|^2$) of each possible outcome. Measuring the last τ qudits over the computation basis states yields a sample from the underlying probability distribution over futures by the model:

$$f(\phi^{T+1},\ldots,\phi^{T+\tau};\boldsymbol{x}^{(T)},\boldsymbol{\theta})=|c(\phi)|^2. \tag{7}$$

Given M measurement samples, we define the resulting distribution $f_M(\boldsymbol{X}^{(T+\tau)\backslash(T)};\boldsymbol{x}^{(T)})$ based on the frequency of observing specific continuation strings $\boldsymbol{x}^{(T+\tau)\backslash(T)}$; in the high sample limit $f_M\approx f$. We can also estimate the most probable future outcomes, which would be useful in repeated predictions. If we wish to predict a future state, $\tau R+T$, we can use R repeated application of the underlying QNN to generate a superposition over those outcomes. Fig. 1 shows how a contextual QNN can be utilized for such time-series based predictions.

We summarize the time-series prediction model as

$$\begin{aligned}\text{Context}:&\left|\boldsymbol{x}^{(T)}\right\rangle,\\ \text{Target}:&\left|\boldsymbol{x}^{(T+\tau)}\right\rangle,\\ \text{Model Prediction}:&\left|\boldsymbol{y}^{(T+\tau)}\right\rangle,\\ \text{Loss Function}:&\mathcal{L}\left(\left|\boldsymbol{y}^{(T+\tau)}\right\rangle,\left|\boldsymbol{x}^{(T+\tau)}\right\rangle\right),\end{aligned}$$

In this paper, we are particularly interested in $\tau=1$ scenario along with binary quantization ($d=2$), for which the qudits are reduced to qubits such that $\left|0\right\rangle$ and $\left|1\right\rangle$ correspond to the negative and positive stock price movement respectively. Hereafter, we stick to the binary quantization $d=2$ throughout the paper unless otherwise stated. This choice provides a simple and interpretable mapping between price movements and quantum measurement outcomes, where each qubit directly represents an upward or downward movement in returns. Using binary encoding allows us to focus on the model's ability to learn and reproduce the underlying conditional probability distributions, without additional complexity introduced by higher-dimensional encoding schemes. Moreover, binary quantization keeps the circuit depth and the number of qubits manageable, which is particularly important when studying scalability across multiple assets. By reducing representational overhead, we can isolate and analyze the learning dynamics of QMTL itself, establishing a clear baseline before extending the framework to multi-level quantization in later experiments. In addition, during training, we also consider the statistics of the contextual data including the contextual probability distribution $\mathcal{P}(\boldsymbol{X}^{(T)})$, the target probability distribution $\mathcal{P}(\boldsymbol{X}^{(T+1)})$, the conditional probability distribution $\mathcal{P}(X^{T+1}|\boldsymbol{X}^{(T)})$, and the total probability distribution $\mathcal{P}(X^{T+1},\boldsymbol{X}^{(T)})$. In particular, we train $\boldsymbol{\theta}$ such that $f(X^{T+1};\boldsymbol{X}^{(T)},\boldsymbol{\theta})\approx\mathcal{P}(X^{T+1}|\boldsymbol{X}^{(T)})$.

QNN framework

Now we recall that a QNN architecture on an n-qubit register represents a parametrized unitary matrix $\hat{U}(\boldsymbol{\theta})$ of dimension 2^n, which is defined by a sequence of parametrized quantum gates that produces an n-qubit output

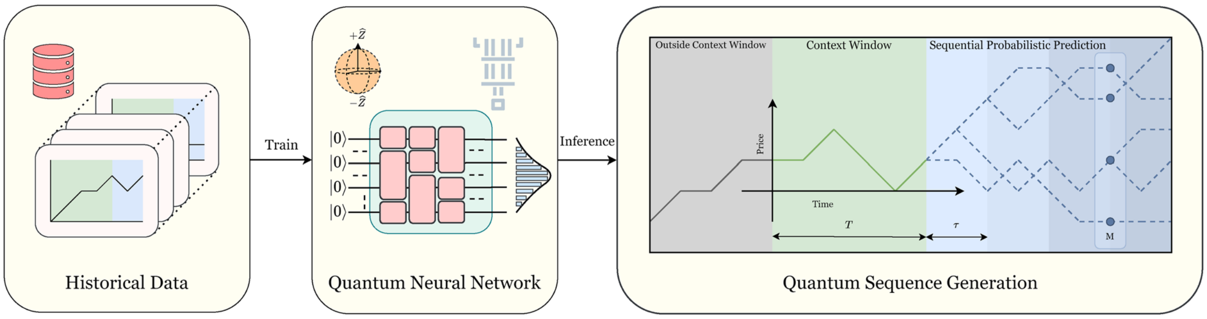


Fig. 1. Quantum Neural Networks for Contextual Sequence Generation. Given historical data of context and continuations, a Quantum Neural Network is trained to produce quantum distributions over future prices, enabling utilization of quantum advantageous algorithms, such as quantum risk analysis[30], downstream for tasks (labeled M in the rightmost diagram at a particular future) over all sequences in superposition.

state $\hat{U}(\boldsymbol{\theta})\Big|\Psi\Big\rangle$ for any input state $\Big|\Psi\Big\rangle \in \mathbb{C}^{2^n}$, where $\boldsymbol{\theta}$ is the set of (real) parameters in the QNN and $\Big|\Psi\Big\rangle$ encodes a classical input data for the problem. The parameters in $\boldsymbol{\theta}$ can be learned and trained to produce a desired output state which is approximated by performing several quantum measurements to all or a subset of the qubits. Deciding the parametrized quantum circuit (PQC), also known as *ansatz* which represents $\hat{U}(\boldsymbol{\theta})$ in a QNN model is one of the fundamental problems in QML applications. In this section, we will introduce the important components of a QNN.

Loading classical Data
The first step in a QNN involves encoding classical data into quantum states. This is typically done using quantum feature maps[19], where classical input data $\Psi \in \mathbb{C}^{\gamma}, 1 \leq \gamma \leq 2^n$ is encoded into a quantum state $\Big|\Psi\Big\rangle$, where 2^n is its dimension. This state can be prepared by

$$\Big|\Psi\Big\rangle = \hat{U}_F(\Psi)\Big|0\Big\rangle^{\otimes n}, \tag{8}$$

where $\hat{U}_F(\Psi)$ is a quantum feature map circuit that depends on the classical data Ψ, and $\Big|0\Big\rangle^{\otimes n}$ represents the initial quantum state. A feature map consists of a set of controlled rotation gates, parameterized by the contents in the classical register Ψ. Different feature maps can be employed depending on the application to project the classical data on the quantum state space. Some of the common feature maps used are the first and second-order Pauli-Z evolution circuits[28].

Parametric quantum circuits
Once the classical data is encoded into a quantum state, it is processed by a PQC. The goal of the training process is to optimize the parameters $\boldsymbol{\theta}$ such that the PQC outputs a quantum state that corresponds to accurate predictions for the learning task. In this paper, inspired from the traditional layered neural networks, we focus on PQCs with L repeated layers, composed on fixed and parameterized gates such that the unitary represented by the circuit at layer $l \in [1, L]$ is $\hat{U}^l(\boldsymbol{\theta}^l) = V^l \prod_j G^{lij}(\theta^{lij})$ as shown in Fig. 2, where V^l is a fixed unitary at layer l (which could be identity or a sequence of CNOT gates) and $G^{lij}(\theta^{lij})$ denote a single-qubit rotation gate acting at layer l, at position (i, j). Here, $i \in [1, n]$ and $j \in [1, c]$, with i and j corresponding to the row and column indices of the quantum circuit, respectively as shown in Fig. 2. In this notation, $\boldsymbol{\theta} = \bigoplus_{l=1}^{L} \boldsymbol{\theta}^l$, where $\boldsymbol{\theta}^l = \{\theta^{l11}, \theta^{l12}, \ldots, \theta^{lnc}\}$ such that $m(= Lnc)$ is the total number of parameters in the circuit, $n(= m/Lc)$ is the number of of qubits and $c(= m/Ln)$ is the number of sub-layers in each layer. This representation will be explored in detail in later sections. Each parameter $\theta^{lij} \in \boldsymbol{\theta}$ can correspond to the angles of single qubit rotation gates such as $R_X(\theta^{lij})$, $R_Y(\theta^{lij})$, $R_Z(\theta^{lij})$. In the proposed QML model in this paper, along with the context quantum state $\Big|\boldsymbol{x}^{(T)}\Big\rangle$, we need ancilla qubits to extract information from the output state of a QNN for the learning task. The ancilla qubit states control the application of certain quantum gates on the main quantum register to obtain a desired output state. Assuming that there are τ ancilla qubits, the QNN represents a unitary matrix $\hat{U}(\boldsymbol{\theta})$ with dimension $2^{T+\tau} \times 2^{T+\tau}$, resulting in the number of qubits $n = T + \tau$. Setting the initial state of the ancilla register as $\Big|0\Big\rangle^{\otimes\tau}$, the output state is given by

$$\Big|\boldsymbol{y}^{(T+\tau)}\Big\rangle = \hat{U}(\boldsymbol{\theta})\Big(\Big|\boldsymbol{x}^{(T)}\Big\rangle \otimes \Big|0\Big\rangle^{\otimes\tau}\Big), \tag{9}$$

where $\hat{U}(\boldsymbol{\theta}) = \prod_l \hat{U}^l(\boldsymbol{\theta}^l)$ is the unitary operator corresponding to the parametric quantum circuit. In addition to rotation gates, our circuit may include fixed gates, such as CNOT gates, that help spread entanglement in the system.

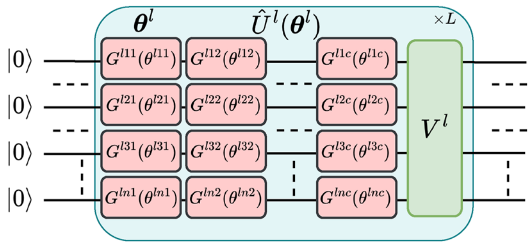


Fig. 2. Parameterized Quantum Circuit. Block diagram of the layered parametric quantum circuit showing various blocks in the l^{th} layer such as fixed unitary and parametric rotation gates.

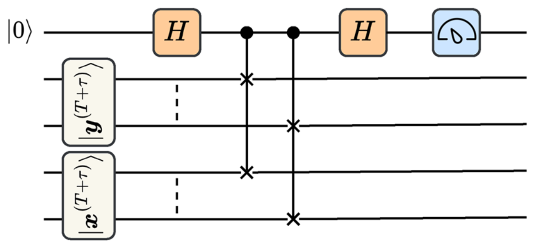

$|0\rangle$
H
H
$|\boldsymbol{y}^{(T+\tau)}\rangle$
$|\boldsymbol{x}^{(T+\tau)}\rangle$

Fig. 3. SWAP Test. A diagram of the SWAP test, which measures the fidelity loss between two wavefunction states $\left|\boldsymbol{x}^{(T+\tau)}\right\rangle$ and $\left|\boldsymbol{y}^{(T+\tau)}\right\rangle$.

Training
Classical neural networks are primarily trained by backpropagation, which is not directly possible in QNNs. Quantum states collapse upon measurement, meaning the quantum information is destroyed. Backpropagation relies on preserving intermediate computations (like activations) during the forward pass for use in the backward pass. In quantum systems, measurements required to extract information disrupt the state, preventing reuse. In addition, the no-cloning theorem prohibits copying quantum states for reuse, further complicating backpropagation. Consequently, QNNs have adopted to other techniques for gradient computation such as parameter shift[43] and simultaneous perturbation stochastic approximation (SPSA)[44]. Unlike backpropagation, which requires explicit differentiation through each layer, these methods compute gradients by evaluating the quantum circuit at slightly shifted parameter values. It works based on the fact that the output of quantum circuits often depends on the parameters through trigonometric functions (such as sine and cosine), enabling exact gradient computation.

If $\hat{B}$ be the observable that we would like to measure, then the expectation value of the output state of the PQC with respect to $\hat{B}$ is $\langle\hat{B}\rangle_{\boldsymbol{\theta}} = \left\langle \boldsymbol{x}^{(T+\tau)} \middle| \hat{U}^{\dagger}(\boldsymbol{\theta})\hat{B}\hat{U}(\boldsymbol{\theta}) | \boldsymbol{x}^{(T+\tau)} \right\rangle$. With this, the gradient update rules are given by[43,44]:

1. *Parameter-Shift*

$$\frac{\partial}{\partial\theta^{lij}}\langle\hat{B}\rangle_{\boldsymbol{\theta}} = \frac{1}{2}\left(\langle\hat{B}\rangle_{\theta^{lij}+\frac{\pi}{2}} - \langle\hat{B}\rangle_{\theta^{lij}-\frac{\pi}{2}}\right), \tag{10}$$

2. *SPSA*

$$\frac{\partial}{\partial\theta^{lij}}\langle\hat{B}\rangle_{\boldsymbol{\theta}} = \frac{1}{2\delta\alpha^{lij}}\left(\langle\hat{B}\rangle_{\boldsymbol{\theta}+\delta\boldsymbol{\alpha}} - \langle\hat{B}\rangle_{\boldsymbol{\theta}-\delta\boldsymbol{\alpha}}\right), \tag{11}$$

where δ is hyperparameter, typically between 0 and 1 and $\boldsymbol{\alpha}$ is a stochastic perturbation vector with dimensions same as $\boldsymbol{\theta}$. The vector $\boldsymbol{\alpha}$ is sampled from a zero mean distribution such that $\alpha^{lij} \in \{-1, 1\}$. The primary difference between the two approaches is in parameter updates: parameter-shift updates one parameter at a time, requiring $2m$ evaluations for m parameters, while SPSA updates all parameters simultaneously with just two evaluations per iteration.

Loss functions
Different loss functions are employed depending on the nature of the QNN and the gradient update rule. For instance, mean squared error (MSE) is commonly used in regression tasks due to its smooth gradients and simplicity in optimization. However, for classification tasks, cross-entropy loss might be preferred as it better captures the probabilistic nature of the output. Therefore, we introduce different loss functions that are relevant to our paper.

Mean squared error loss is the most common loss function used for optimization tasks and in our context, it is defined as:

$$\mathcal{L}^{m}(\boldsymbol{\theta}, \boldsymbol{x}^{(T+\tau)}) = |f_M(x^{(T+\tau)\setminus(T)}; x^{(T)}) - \delta(\boldsymbol{x}^{(T+\tau)})|^2. \tag{12}$$

Note that for the MSE loss, full state measurement is required i.e., all the qubits must be measured and the returns have to retrieved from the output state vector as per Eq. (6). To circumvent this, we use the fidelity loss or the SWAP test[41] to compare the output state $\left|\boldsymbol{y}^{(T+\tau)}\right\rangle$ with the target state $\left|\boldsymbol{x}^{(T+\tau)}\right\rangle$. An ancilla qubit is prepared in the state $\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + \left|1\right\rangle\right)$, whose logical state is then used to control SWAP the wavefunction (e.g. qubit by qubit). Thereafter, a Hadamard gate is applied on the ancilla, leading to a phase kick back, and then the ancilla is measured in the computational (Pauli-Z) basis.

The probability of obtaining $\left|0\right\rangle$ after measurement of the ancilla qubit is proportional to the fidelity $\left|\left\langle \boldsymbol{y}^{(T+\tau)} \middle| \boldsymbol{x}^{(T+\tau)} \right\rangle\right|^2$ and is given by:

$$\mathcal{P}(0) = \frac{1 + \left|\left\langle \boldsymbol{y}^{(T+\tau)} \middle| \boldsymbol{x}^{(T+\tau)} \right\rangle\right|^2}{2} \tag{13}$$

and the corresponding fidelity loss is:

$$\mathcal{L}^f(\boldsymbol{\theta}, \boldsymbol{x}^{(T+\tau)}) = 1 - \mathcal{P}(0) = \frac{1 - \left|\left\langle \boldsymbol{y}^{(T+\tau)} \middle| \boldsymbol{x}^{(T+\tau)} \right\rangle\right|^2}{2} \tag{14}$$

Loading classical data

Contextual data $\boldsymbol{x}^{(T)}$ can be encoded onto a quantum state using feature maps[28], as described in Eq. (8). However, feature maps are computationally intensive and slow to train due to the convoluted nature of $\hat{U}_f$. Moreover, training requires encoding a different contextual input during each iteration, further increasing the complexity. To address this challenge, we propose loading the entire contextual distribution onto the quantum state using a hardware-efficient ansatz[31]. By encoding the data only once, our approach achieves significantly faster training. In this section, we discuss the procedure for encoding the contextual probability distribution $\mathcal{P}(\boldsymbol{X}^{(T)})$, where $\boldsymbol{X}^{(T)}$ represents the contextual time-series data of an asset, onto a quantum state using the hardware efficient ansatz architecture.

Once the data has been preprocessed, we construct the Hilbert space as per Eq (2) to obtain a basis for the T-qubit representation of the returns, where the quantization levels for the returns are also chosen. Now, we need to represent the stochastic behavior of the asset in the space defined by the basis vectors such that the information can be used to train the QNN model and make reasonable predictions. To this end, we normalize the preprocessed data and compute its histogram to obtain the contextual probability distribution of the quantized returns as $\mathcal{P}(\boldsymbol{X}^{(T)} = (i^1, \dots, i^T))$, where i^t denotes the quantized return at time *t*. Thereafter, we load the contextual distribution onto the T-qubit state as follows

$$\left|\boldsymbol{\psi}^{(T)}\right\rangle = \sum_{i^1,\dots,i^T \in \{0,1\}} \sqrt{\mathcal{P}(i^1,\dots,i^T)} \left|i^1\right\rangle \otimes \left|i^2\right\rangle \dots \otimes \left|i^T\right\rangle \tag{15}$$

where $\mathcal{P}(i^1, \dots, i^T)$ is the value of the quantized distribution over the corresponding basis state $\left|i^1\right\rangle \otimes \left|i^2\right\rangle \dots \otimes \left|i^T\right\rangle$. Note that, using the definition of $\boldsymbol{x}^{(T)}$ and equations (1) and (2), we can rewrite Eq. (15) as

$$\left|\boldsymbol{\psi}^{(T)}\right\rangle = \sum_{\boldsymbol{x}^{(T)} \in \{0,1\}^T} \sqrt{\mathcal{P}(\boldsymbol{x}^{(T)})} \left|\boldsymbol{x}^{(T)}\right\rangle \tag{16}$$

Eq. (16) can be achieved using the Grover-Rudolph technique for state preparation[33]. However, due to its higher computational complexity, we adopt a simpler machine learning-based approach for state preparation. This method utilizes a hardware-efficient ansatz[31] combined with a mean squared error (MSE) loss function (Eq. (12)), where the parameters of the rotation gates are iteratively updated in a loop using the SPSA rule to minimize the loss function, as illustrated in Fig.4. This circuit is chosen due to its hardware-efficient architecture[32], which constitutes a repeated layers of R_Y, R_Z, and $CNOT$ gates to perform rotations and entanglement. Although several loss functions exist for comparing distributions, we choose MSE due to its smooth gradients and computational efficiency, enabling faster optimization. For each layer, the corresponding unitary transformation looks like:

$$\hat{O}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{j=0}^{T-1} CNOT_{j,(j+1)\%T} \bigotimes_{j=0}^{T-1} R_Z(\beta^j) \bigotimes_{j=0}^{T-1} R_Y(\alpha^j) \tag{17}$$

where $CNOT_{j,(j+1)\%T}$ acts on qubits *j* (control) and $(j+1)\%T$ (target) and $R_Y(\alpha^i)$, $R_Z(\beta^i)$ are 2×2 unitary matrices acting on the j^{th} qubit. To enhance the accuracy of loading the distribution, multiple such transformations are applied iteratively, as illustrated in the figure. This approach improves the learnability of the circuit, thereby increasing the fidelity of the loaded distribution. Note that the parameters in the circuit $\{\{\alpha^i\} \cup \{\beta^i\}\}$ are optimized through the training process to accurately load the contextual distribution.

Quantum single-task learning

Now that we have encoded the contextual distribution into the quantum states, we can move on to discussing predictions based on the given context. Before diving into quantum multi-task learning for predicting multiple asset prices, we'll first present a complete case of using quantum circuits for predicting the price of a single stock, which we will refer to as quantum single-task learning (QSTL).

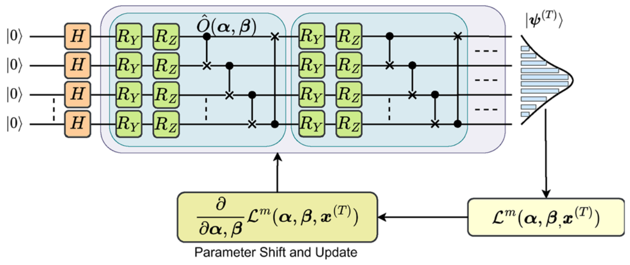
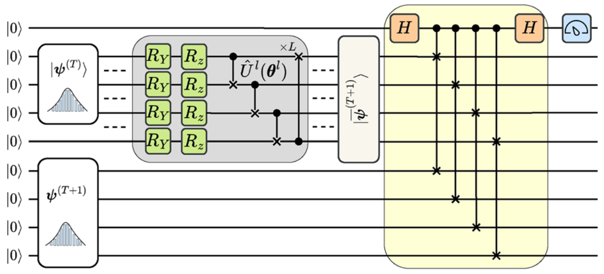

$\hat{O}(\boldsymbol{\alpha},\boldsymbol{\beta})$
$|\boldsymbol{\psi}^{(T)}\rangle$
$\frac{\partial}{\partial\boldsymbol{\alpha},\boldsymbol{\beta}}\mathcal{L}^m(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{x}^{(T)})$
$\mathcal{L}^m(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{x}^{(T)})$
Parameter Shift and Update

Fig. 4. Loading a distribution onto a hardware efficient ansatz. The circuit inside the blue box ($\hat{O}(\boldsymbol{\alpha},\boldsymbol{\beta})$) is applied sequentially for a sufficient number of iterations followed by an MSE loss with the SPSA update rule.


$|\boldsymbol{\psi}^{(T)}\rangle$
$\hat{U}^l(\boldsymbol{\theta}^l)$
$\times L$
$\boldsymbol{\psi}^{(T+1)}$
$\langle\bar{\boldsymbol{\psi}}^{(T+1)}|$

Figu 5. Quantum Batch Learning. A diagram showing the learning procedure of our proposed quantum batch gradient update for a context of T. The top most qubit is an ancilla qubit. For a batch, a distribution over inputs is loaded succeeding qubits (the input qubits) and the following qubit(s) is for the output. The joint distribution of the inputs and outputs for the batch is loaded on the subsequent qubits. The circuit inside the Grey box ($\hat{U}(\boldsymbol{\theta})$) is applied sequentially for a required number of iterations, is a contextual quantum neural network to prepare an approximate of the loaded joint distribution. A SWAP test is then used to take the fidelity loss between the two distributions.

Unlike previous approaches[35,37], which rely only on using the entire historical data to make predictions, we incorporate contextual information as well to forecast future outcomes, as shown in Fig. 5. This method offers the advantage of adapting to the constantly changing stock market, whereas past methods may become less relevant due to outdated data. In our approach, the contextual distribution is encoded onto the wavefunction $\left|\psi^{(T)}\right\rangle$, which, along with a prediction qubits, forms the input of the quantum circuit: $\left|\psi^{(T)}\right\rangle \otimes \left|0\right\rangle^{\otimes\tau}$. Hereafter, we will stick to $\tau = 1$ and binary quantization unless otherwise stated. We also load the target distribution $\left|\psi^{(T+1)}\right\rangle$ to the last $T+1$ qubits, which serves as an input to the SWAP test. The layered PQC forms the bulk of the circuit followed by the SWAP test between the predicted distribution and target distribution (as shown in Fig.5). Finally, measurement is done on the ancillary qubit, obtaining the fidelity loss, which guides the training process through SPSA gradient update rule.

A unitary transformation $\hat{U}(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the trainable parameters, enables the circuit to learn the conditional probability distribution represented by $\mathcal{P}(X^{T+1}|\boldsymbol{x}^{(T)},\boldsymbol{\theta})$. This probability distribution serves as the desired output state of the PQC at the end of the training process, allowing for the prediction of future returns via measurement. Learning conditional probability distributions is crucial for time-series prediction as they capture the dependency between past context and future outcomes. In contrast, marginal distributions are unsuitable for time-series prediction because they lack the ability to incorporate temporal dependencies and contextual information.

Quantum batch gradient update

Quantum data processing has been explored for Machine Learning due to the inherent memory compression associated. For example Ref[45]. utilizes access in superposition to models for quantum speedup in tasks like k-means clustering, while Ref[18]. achieves quantum speedup for Support Vector Machines by learning over a superposition of data points. In the context of QNNs, Ref[46]. recently considered a construction for parallel quantum batches in which a reduced density matrix is constructed over the ancilla, output, and label space. In comparison, our approach loads the entire contextual distribution at once according to Eq. (16), resulting in a straight forward loss function that leads to higher quality gradients. Then our QNN generates an *approximate* representation of the continuation for *each* context in superposition. This leverages the linearity of quantum circuits, allowing a superposition of all possible inputs to train the circuit without breaking the correspondence between the respective inputs and outputs. This correspondence reduces the multi-step stochastic gradient descent to a single step.

For example, consider Eq. (9), when the input of the PQC is a single context $\left|\boldsymbol{x}^{(T)}\right\rangle$ and the forward pass of this input through the PQC is $\left|\boldsymbol{y}^{(T+1)}\right\rangle = \hat{U}(\boldsymbol{\theta})\left(\left|\boldsymbol{x}^{(T)}\right\rangle \otimes \left|0\right\rangle\right)$ as shown in Fig. 5. Let the gradient update for $\boldsymbol{\theta}$, obtained from SPSA, be denoted as $g(\boldsymbol{\theta}, \boldsymbol{x}^{(T+1)})$, with entry *lij*:

$$\begin{aligned} g(\theta^{lij}, \boldsymbol{x}^{(T+1)}) &= \frac{\partial}{\partial \theta^{lij}} \mathcal{L}^f(\boldsymbol{\theta}, \boldsymbol{x}^{(T+\tau)}) \\ &= \frac{\partial}{\partial \theta^{lij}} \left(\frac{1 - \left|\left\langle \boldsymbol{y}^{(T+\tau)} \middle| \boldsymbol{x}^{(T+\tau)} \right\rangle\right|^2}{2} \right) \\ &= -\frac{1}{2} \frac{\partial}{\partial \theta^{lij}} \langle \boldsymbol{y}^{(T+\tau)} | \hat{B} \left| \boldsymbol{y}^{(T+\tau)} \right\rangle \\ &= -\frac{1}{4\delta\alpha^{lij}} \left(\langle \hat{B} \rangle_{\boldsymbol{\theta}+\delta\boldsymbol{\alpha}} - \langle \hat{B} \rangle_{\boldsymbol{\theta}-\delta\boldsymbol{\alpha}} \right), \end{aligned} \tag{18}$$

where the last step is obtained by using Eq. (11) with $\hat{B} = \left|\boldsymbol{x}^{(T+\tau)}\right\rangle\left\langle\boldsymbol{x}^{(T+\tau)}\right|$. Therefore, the gradient update of $\boldsymbol{\theta}$ becomes

$$\Rightarrow \boldsymbol{\theta} := \boldsymbol{\theta} - \beta\, g(\boldsymbol{\theta}, \boldsymbol{x}^{(T+1)}), \tag{19}$$

where β is the learning rate. Similarly, if the input to the PQC is a distribution of basis states (superposition of all possible contexts) such as $\left|\boldsymbol{\psi}^{(T)}\right\rangle$ from Eq. (16), then the forward pass through the PQC is given by:

$$\begin{aligned} \left|\overline{\boldsymbol{\psi}}^{(T+1)}\right\rangle &= \hat{U}(\boldsymbol{\theta}) \Big(\sum_{\boldsymbol{x}^{(T)} \in \{0,1\}^T} \sqrt{\mathcal{P}(\boldsymbol{x}^{(T)})} \left|\boldsymbol{x}^{(T)}\right\rangle \otimes \left|0\right\rangle \Big) \\ &= \sum_{\boldsymbol{x}^{(T)} \in \{0,1\}^T} \sqrt{\mathcal{P}(\boldsymbol{x}^{(T)})} \hat{U}(\boldsymbol{\theta}) \left(\left|\boldsymbol{x}^{(T)}\right\rangle \otimes \left|0\right\rangle \right) \\ &= \sum_{\boldsymbol{x}^{(T)} \in \{0,1\}^T} \sqrt{\mathcal{P}(\boldsymbol{x}^{(T)})} \left|\boldsymbol{y}^{(T+1)}\right\rangle, \end{aligned}$$

where $\left|\overline{\boldsymbol{\psi}}^{(T+1)}\right\rangle$ is the total distribution learned by the circuit $\hat{U}(\boldsymbol{\theta})$ over the contextual distribution $\left|\boldsymbol{\psi}^{(T)}\right\rangle$. Given that the gradient update corresponding to the output $\left|\boldsymbol{y}^{(T+1)}\right\rangle$ is $g(\boldsymbol{\theta}, \boldsymbol{x}^{(T+1)})$ (from Eq. 19), then the gradient update for the output $\left|\overline{\boldsymbol{\psi}}^{(T+1)}\right\rangle$ can be calculated from the derivative over summation rule as

$$\Rightarrow \boldsymbol{\theta} := \boldsymbol{\theta} - \beta \sum_{\boldsymbol{x}^{(T)} \in \{0,1\}^T} \mathcal{P}(\boldsymbol{x}^{(T)})\, g(\boldsymbol{\theta}, \boldsymbol{x}^{(T+1)}). \tag{20}$$

This expression (Eq. 20) effectively corresponds to applying stochastic gradient descent (SGD) across all input context samples from the dataset, achieving a single gradient update after processing the entire dataset. Notably, this result is obtained in one step due to the inherent linearity of quantum mechanics, facilitating the quantum circuit to train on large batches. Using the QBGU training process, the model $\hat{U}(\boldsymbol{\theta})$ in Fig. 5 is trained to learn the conditional probability distribution $\mathcal{P}(x^{T+1}|\boldsymbol{x}^{(T)}, \boldsymbol{\theta})$, which essentially maps each input $\left|\boldsymbol{x}^{(T)}\right\rangle$ to its corresponding output $\left|\boldsymbol{y}^{(T+1)}\right\rangle$ at the time of inference. As shown in Fig. 5, we preload both the contextual distribution $\left|\boldsymbol{\psi}^{(T)}\right\rangle$ and the target distribution $\left|\boldsymbol{\psi}^{(T+1)}\right\rangle$, and the SWAP test then measures the distance between the estimated and the original target distributions, guiding the training process.

Quantum multi-task learning

Multi-task learning (MTL)[40] is a machine learning approach where multiple tasks are learned at the same time, allowing the model to share information between them. It uses shared parameters to capture common patterns across all tasks, while task-specific parameters focus on unique aspects of each task. In financial time series prediction, MTL can be used to predict the prices or trends of multiple assets together. Shared parameters can represent factors that affect the entire market, such as economic indicators, while task-specific parameters account for unique characteristics of each asset, like individual volatility or trading patterns. This helps the model make better predictions by learning both shared and asset-specific information. In this section, we extend these ideas to QNNs by introducing the quantum multi-task learning (QMTL).

Circuit architecture

In order to incorporate MTL in a QNN framework, we introduce the *share-and-specify* ansatz (Fig. 6) which breaks each layer of the PQC into a block of universal gates (*shared ansatz*), followed by a block of asset specific label-controlled gates acting based on the state of the label registers (*specify ansatz*). The share ansatz for each layer can be embodied by the same gates used in the single-asset task, leading to identity operations applied to the $\log K$ label qubits,

$$\hat{U}_s^l(\boldsymbol{\theta}_s^l) = \left(V_s^l \otimes \mathbb{1}_K\right) \prod_{i,j} \left(G_s^{lij}(\theta_s^{lij}) \otimes \mathbb{1}_K\right), \tag{21}$$

where $\mathbb{1}_K$ is the identity operator with dimension $K \times K$. The label qubits, collectively represented by the qudit $\left|k\right\rangle = \left|k_1\right\rangle \otimes \left|k_2\right\rangle \cdots \otimes \left|k_{\log K}\right\rangle$ with $k_j \in \{0,1\}$, are used to distinguish the assets. Each asset is assigned a unique label $k \in [1, K]$, ensuring that the quantum operations are applied selectively to the corresponding asset based on its label. These label qubits are then used to form the task-specific unitary operator for asset *k* as

$$\hat{U}_k^l(\boldsymbol{\theta}_k^l) = \left(V_k^l \otimes \mathbb{1}_K\right) \prod_{i,j} \left(G_k^{lij}(\theta_k^{lij}) \otimes \mathbb{1}_K\right), \tag{22}$$

resulting in the specify ansatz (with control) for asset *k* as

$$\begin{aligned}\hat{U}_{ck}^l(\boldsymbol{\theta}_{ck}^l) = {} & (V_k^l \otimes \left|k\right\rangle\left\langle k\right| + \mathbb{1}_{2^{T+\tau}} \otimes \left|k^\perp\right\rangle\left\langle k^\perp\right|) \\ & \prod_{i,j} \left(G_k^{lij}(\theta_k^{lij}) \otimes \left|k\right\rangle\left\langle k\right| + \mathbb{1}_{2^{T+\tau}} \otimes \left|k^\perp\right\rangle\left\langle k^\perp\right|\right),\end{aligned} \tag{23}$$

where $\boldsymbol{\theta}_{ck}^l = \boldsymbol{\theta}_k^l$, but the subscript *c* is added for notational consistency and $\left|k^\perp\right\rangle\left\langle k^\perp\right| = \mathbb{1}_K - \left|k\right\rangle\left\langle k\right|$ is a projection onto the orthogonal space of $\left|k\right\rangle\left\langle k\right|$, such that $\bigotimes_i G_k^{lij}(\theta_k^{lij})$ is only applied when the label qudit is $\left|k\right\rangle$. In particular, if $G_k^{lij} \in \mathbb{1}_{2^{i-1}} \otimes \{\hat{R}_X(\theta_k^{lij}), \hat{R}_Y(\theta_k^{lij}), \hat{R}_Z(\theta_k^{lij})\} \otimes \mathbb{1}_{2^{n-i}}$, then $\left(G_k^{lij}(\theta_k^{lij}) \otimes \left|k\right\rangle\left\langle k\right| + \mathbb{1}_{2^{T+\tau}} \otimes \left|k^\perp\right\rangle\left\langle k^\perp\right|\right)$ is a control rotation based on label $\left|k\right\rangle$. We can then define the specify ansatz for one layer as $\hat{U}_c^l(\boldsymbol{\theta}_c^l) = \prod_k \hat{U}_{ck}^l(\boldsymbol{\theta}_{ck}^l)$ and the entire PQC as

$$\hat{U}(\boldsymbol{\theta}) = \prod_{l=1}^{L} \hat{U}_c^l(\boldsymbol{\theta}_c^l)\hat{U}_s^l(\boldsymbol{\theta}_s^l), \tag{24}$$

where the dimension of $\hat{U}(\boldsymbol{\theta})$ is $2^{T+\tau}K \times 2^{T+\tau}K$. Note that, $\boldsymbol{\theta} = \bigoplus_{l=1}^{L} \boldsymbol{\theta}_s^l \oplus \boldsymbol{\theta}_c^l$. For each asset *k*, the contextual price data of size *T*, represented as $\left|\boldsymbol{x}_k^{(T)}\right\rangle$, undergoes the transformation determined by the label qudit and the task-specific ansatz:

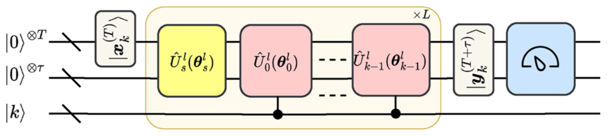

$|0\rangle^{\otimes T}$
$|0\rangle^{\otimes \tau}$
$|k\rangle$
$|\boldsymbol{x}_k^{(T)}\rangle$
$\times L$
$\hat{U}_s^l(\boldsymbol{\theta}_s^l)$
$\hat{U}_0^l(\boldsymbol{\theta}_0^l)$
$\hat{U}_{k-1}^l(\boldsymbol{\theta}_{k-1}^l)$
$|\boldsymbol{y}_k^{(T+\tau)}\rangle$

Fig. 6. Share-and-specify Ansatz. A diagram of our Quantum Multi-Task Learning Architecture showing various components for the QNN. A single asset state is loaded by setting the label $\left|k\right\rangle$ and the inference-time context $\boldsymbol{x}^{(T)}$ is loaded over qubits $\left|0\right\rangle^{\otimes T+\tau}$. The input can then be processed through the parameterized circuit, composed of *L* layers of the share-and-specify ansatz, to define a state $\left|\boldsymbol{y}^{(T)}\right\rangle$ that can be utilized for a downstream task. For example, as depicted in the figure, measurement can be used to sample possible continuations x^τ.

$$\begin{aligned}\left|\boldsymbol{y}_k^{(T+1)}\right\rangle &= \hat{U}(\boldsymbol{\theta})\left(\left|\boldsymbol{x}_k^{(T)}\right\rangle \otimes \left|0\right\rangle^{\otimes\tau} \otimes \left|k\right\rangle\right)\\ &= \prod_l \hat{U}_c^l(\boldsymbol{\theta}^l)\left(\hat{U}_s^l(\boldsymbol{\theta}_s^l)\left(\left|\boldsymbol{x}_k^{(T)}\right\rangle \otimes \left|0\right\rangle^{\otimes\tau}\right) \otimes \left|k\right\rangle\right)\\ &= \prod_l \hat{U}_k^l(\boldsymbol{\theta}_k^l)\hat{U}_s^l(\boldsymbol{\theta}_s^l)\left(\left|\boldsymbol{x}_k^{(T)}\right\rangle \otimes \left|0\right\rangle^{\otimes\tau}\right) \otimes \left|k\right\rangle.\end{aligned}$$

Intuitively, the share ansatz layer $\hat{U}_s^l$ helps facilitate learning across assets while the specify ansatz layer $\hat{U}_k^l$ allows focus on potential peculiarities of an individual asset. This entire architecture is demonstrated in Fig. 6.

Two-assets case

Extending the previous framework to handle two assets, the forward pass equations for a set of inputs $\left|\boldsymbol{x}_1^{(T)}\right\rangle$, $\left|\boldsymbol{x}_2^{(T)}\right\rangle$ for two different assets at layer l are given as

$$\begin{aligned}\left|\boldsymbol{y}_1^{(T+1)}\right\rangle &= \hat{U}_1^l(\boldsymbol{\theta}_1^l)\hat{U}_s^l(\boldsymbol{\theta}_s^l)\left(\left|\boldsymbol{x}_1^{(T)}\right\rangle \otimes \left|0\right\rangle \otimes \left|0\right\rangle\right)\\ \left|\boldsymbol{y}_2^{(T+1)}\right\rangle &= \hat{U}_2^l(\boldsymbol{\theta}_2^l)\hat{U}_s^l(\boldsymbol{\theta}_s^l)\left(\left|\boldsymbol{x}_2^{(T)}\right\rangle \otimes \left|0\right\rangle \otimes \left|1\right\rangle\right)\end{aligned}$$

The subscripts 1 and 2 denote first and second assets, respectively, with the last qubit serving as the control to switch between assets. As illustrated in Fig. 7, the unitary operator $\hat{U}_s(\boldsymbol{\theta}_s^l)$ is shared across all assets, while the operators $\hat{U}_1^l(\boldsymbol{\theta}_1^l)$ and $\hat{U}_2(\boldsymbol{\theta}_2^l)$ are task-specific trainable PQCs. The simplest way to switch between tasks is by applying an X gate, as illustrated in Fig. 7.

Four-assets case

For scenarios involving more than two assets or tasks, constructing the control qudit using only single-qubit gates and CNOT gates leads to a significant increase in the number of label qubits. To address this, our approach incorporates Toffoli gates, which enables us to minimize the number of label qubits to $\log K + 1$. For instance, with four assets, we designed a control circuit employing three qubits, Toffoli gates, and X gates, as depicted in Fig. 8. This configuration ensures that exactly one of the transformations $\hat{U}_k^l(\boldsymbol{\theta}_k^l)$ is active for each unique combination of $\left|k_1\right\rangle \otimes \left|k_2\right\rangle$, corresponding to a specific asset. The task-specific PQCs and their associated controls on the qubits $(\left|k\right\rangle \otimes \left|k_1\right\rangle \otimes \left|k_2\right\rangle)$ can be expressed as:

$$\hat{U}_{11}^l(\boldsymbol{\theta}_{11}^l) : \left(\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes \mathbb{1}_2\right)\mathrm{CCNOT}\left(\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes \mathbb{1}_2\right) \tag{25}$$

$$\hat{U}_{01}^l(\boldsymbol{\theta}_{01}^l) : \left(\mathbb{1}_2 \otimes X \otimes \mathbb{1}_2\right)\mathrm{CCNOT}\left(\mathbb{1}_2 \otimes X \otimes \mathbb{1}_2\right) \tag{26}$$

$$\hat{U}_{10}^l(\boldsymbol{\theta}_{10}^l) : \left(\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes X\right)\mathrm{CCNOT}\left(\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes X\right) \tag{27}$$

$$\hat{U}_{00}^l(\boldsymbol{\theta}_{00}^l) : \left(\mathbb{1}_2 \otimes X \otimes X\right)\mathrm{CCNOT}\left(\mathbb{1}_2 \otimes X \otimes X\right) \tag{28}$$

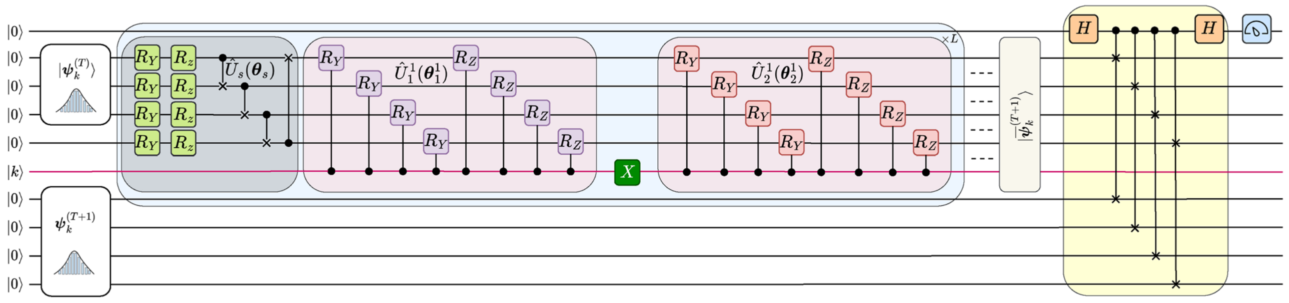

|0⟩
|k⟩
$|\psi_k^{(T)}\rangle$
$\psi_k^{(T+1)}$
R_Y
R_z
R_Z
$\hat{U}_s(\boldsymbol{\theta}_s)$
$\hat{U}_1^1(\boldsymbol{\theta}_1^1)$
$\hat{U}_2^1(\boldsymbol{\theta}_2^1)$
X
×L
$|\psi_k^{(T+1)}\rangle$
H

Fig. 7. Quantum Multi-Task Learning Architecture for $T = 3$ and $K = 2$. The circuits inside the Grey $(\hat{U}_s(\boldsymbol{\theta_s}))$ and Pink boxes $(\hat{U}_1^1(\boldsymbol{\theta}_1^1), \hat{U}_2^1(\boldsymbol{\theta}_2^1))$ are applied sequentially for a required number of iterations followed by the SWAP test.

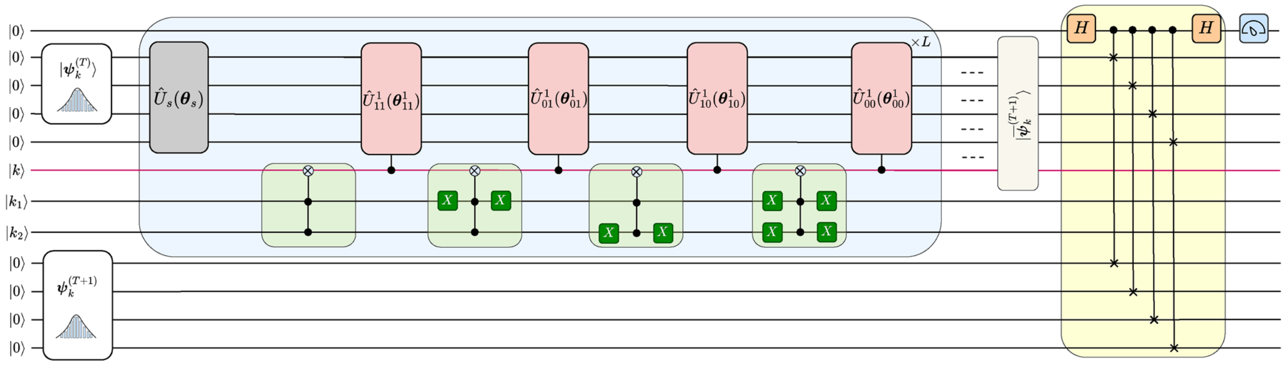


Fig. 8. Quantum Multi-Task Learning Architecture for $T = 3$ and $K = 4$. The circuits inside the Grey ($\hat{U}_s(\boldsymbol{\theta}_s)$) and Pink boxes ($\hat{U}^1_{11}(\boldsymbol{\theta}^1_{11})$, $\hat{U}^1_{01}(\boldsymbol{\theta}^1_{01})$, $\hat{U}^1_{10}(\boldsymbol{\theta}^1_{10})$, $\hat{U}^1_{00}(\boldsymbol{\theta}^1_{00})$) are applied sequentially for a required number of iterations followed by the SWAP test.

where, $\mathbb{1}_2$ represents the identity gate on one qubit, and CCNOT denotes the Toffoli gate, which operates on $\left|k\right\rangle$ with control inputs $\left|k_1\right\rangle$ and $\left|k_2\right\rangle$ such that $k_1, k_2 \in \{0, 1\}$. We denote the single-qubit gates as $G(k_1, k_2)$, such that the general expression for the control of the transformation $\hat{U}_{k_1k_2}(\boldsymbol{\theta}_{k_1k_2})$ can be given by (from Eq. 28):

$$\hat{U}^l_{k_1k_2}(\boldsymbol{\theta}^l_{k_1k_2}) : G^l(k_1, k_2) \text{ CCNOT } G^l(k_1, k_2) \tag{29}$$

where

$$\begin{aligned} G^l(k_1, k_2) = &\Big(\mathbb{1}_2 \otimes (k_1\mathbb{1}_2 + (1 - k_1)X) \\ &\otimes (k_2\mathbb{1}_2 + (1 - k_2)X)\Big). \end{aligned}$$

K-assets case
Furthermore, if we extend this logic to accommodate K tasks, then we require $\log(K) + 1$ qubits, which is more than $\log(K)$, indicating the need for more qubits to represent $\left|k\right\rangle$. The resulting gate composition can then be expressed as:

$$\begin{aligned} \hat{U}^l_{k_1k_2\ldots k_{logK}}(\boldsymbol{\theta}^l_{k_1k_2\ldots k_{logK}}) :&G^l(k_1, k_2 \ldots k_{logK}) \text{ CCNOT} \\ &G^l(k_1, k_2 \ldots k_{logK}) \end{aligned}$$

where

$$\begin{aligned} G^l(k_1, k_2 \ldots k_{logK}) = &\Big(\mathbb{1} \otimes (k_1\mathbb{1} + (1 - k_1)X) \\ &\otimes (k_2\mathbb{1} + (1 - k_2)X) \ldots \\ &\otimes (k_{logK}\mathbb{1} + (1 - k_{logK})X)\Big), \end{aligned}$$

where $k_1, \ldots, k_{logK} \in \{0, 1\}$. Note that this type of control provides exclusive task-specific layers for each task. In scenarios where a complete task-specific layer is not necessary for every task, the circuit can be optimized to reduce the number of gates by partially sharing the circuit among the tasks. Qubit overhead can also be reduced by using shared labels for similar assets.

Training
In classical MTL models, training typically involves optimizing weights using gradient-based methods applied to the entire network, with separate tasks handled through task-specific output layers or parameters. QMTL in this setting leverages shared parameters across tasks to capture common features while using task-specific parameters to model unique characteristics of each task. This approach often requires separate forward and backward passes for each task to compute gradients, making the process resource-intensive.

A parametric quantum circuit $\hat{U}(\boldsymbol{\theta})$ consists of fixed gates (such as CNOTs) and parameterized gates $\hat{G}^{lij}(\theta^{lij}) = e^{-i\frac{\theta^{lij}}{2}\hat{P}^{lij}}$, where $\hat{P}^{lij} \in \{\hat{X}_i, \hat{Y}_i, \hat{Z}_i\}^{T+\tau}_{i=1}$ is a single qubit Pauli generator. The parametric quantum circuit consists of m parameters, each corresponding to either a rotation gate $G^{lij}_s(\theta^{lij}_s) \otimes \mathbb{1}_K$ (from Eq. (21)) or a controlled rotation gate $\left(G^{lij}_k(\theta^{lij}_k) \otimes \left|k\right\rangle\left\langle k\right| + \mathbb{1}_{2^{T+\tau}} \otimes \left|k^\perp\right\rangle\left\langle k^\perp\right|\right)$ (from Eq. (23)). However, during training, we fix the value of k, and since the label qubits are also excluded from measurement (see Figs. 5, 7, 8), we effectively reduce all the controlled rotation gates to $G_k^{lij}(\theta_k^{lij}) \otimes \mathbb{1}_K$. This is because each task-specific layer is trained independently on its dataset, effectively making the control qubits function as a multiplexer. Mathematically, this leads to the equivalence of $\hat{U}_{ck}^l(\boldsymbol{\theta}_{ck}^l)$ and $\hat{U}_k^l(\boldsymbol{\theta}_k^l)$, resulting in

$$\hat{U}(\boldsymbol{\theta}) = \prod_{l=1}^{L} \hat{U}_k^l(\boldsymbol{\theta}_k^l)\hat{U}_s^l(\boldsymbol{\theta}_s^l), \tag{30}$$

where $\hat{U}(\boldsymbol{\theta})$ is now dimensionally reduced but retains the same structure as Eq. (24) from the perspective of gradient computation. From Eq.(30), $\hat{U}(\boldsymbol{\theta})$ consists of a sequence of non-controlled unitary gates of size $2^{T+\tau}$. Consequently, gradient update rules from equations (10) and (11) can be applied using the chain rule to train the QMTL model. This equivalence significantly reduces the complexity of the loss function and its dependence on hidden layers, resulting in training performance comparable to QSTL.

Sequential prediction

During inference, we load a sampled context from the overall contextual distribution but the circuit is capable of processing this single input to predict the future stock price for that specific context. For instance, we load a single context vector $\boldsymbol{x}_k^{(T)}$ to a state $\left|\boldsymbol{x}_k^{(T)}\right\rangle$ and find (from Eq. (5))

$$\left|\boldsymbol{y}_k^{(T+1)}\right\rangle \approx \sum_{x_k^{T+1}\in\{0,1\}} \sqrt{\mathcal{P}(x_k^{T+1}|\boldsymbol{x}_k^{(T)}, \boldsymbol{\theta})}\left|\boldsymbol{x}_k^{(T)}\right\rangle\left|x_k^{T+1}\right\rangle,$$

such that measuring the computational basis samples the most likely continuation based on the historical context given. However, we can repeatedly apply our QNN R times, as shown in Fig. 9, resulting in the final approximate state

$$\left|\boldsymbol{y}_k^{(T+R)}\right\rangle \approx \sum_{\boldsymbol{x}_k^{(T+R)/(T)}\in\{0,1\}^R} \Bigg(\sqrt{\mathcal{P}(\boldsymbol{x}_k^{(T+R)/(T)}|\boldsymbol{x}_k^{(T)}, \boldsymbol{\theta})} \left|\boldsymbol{x}_k^{(T)}\right\rangle\left|x_k^{T+1}\right\rangle\dots\left|x_k^{T+R}\right\rangle\Bigg).$$

This approach could, in future work, be extended to create a superposition state over all d^R possible paths with logarithmic qubit depth $\mathcal{O}(R)$, allowing efficient encoding of highly nontrivial context-specific distributions over futures. Such an extension would enable the use of quantum algorithms for statistical tasks, potentially achieving quantum advantage at inference time, for example through quadratic sampling speedups using amplitude estimation for risk analysis[30]. This is possible because QBGU preserves the mapping between input (context) and output (prediction) for individual samples within a batch, allowing the model to make predictions for each sample independently. This correspondence is further validated in the subsequent sections.

In the QMTL setting, we can load portfolio distributions (such as market capitalization weighted index of publicly traded corporate stocks) to do sequence generation over the portfolio with only logarithmic overhead for the asset labels. Let V_k correspond to the weight of asset k and $V = \sum_{k=1}^K V_k$. Then we can load the distribution $\sum_{k=1}^K \sqrt{V_k/V}\left|k\right\rangle$ over the $\log(K)$ label qubits and use label-control gates to initialize the corresponding context $\left|\boldsymbol{x}_k^{(T)}\right\rangle$ dependent on each label. Applying our QNN R times prepares:

$$\left|\boldsymbol{y}^{(T+R)}\right\rangle \approx \sum_{k=1}^{K}\sqrt{\frac{V_k}{V}}\left|\boldsymbol{y}_k^{(T+R)}\right\rangle.$$

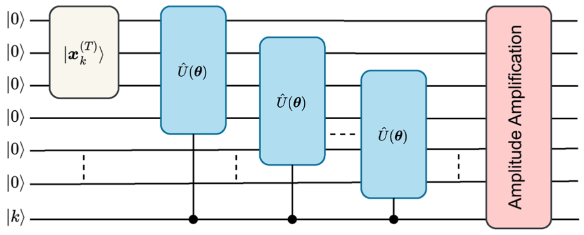

|0⟩
|0⟩
|0⟩
|0⟩
|0⟩
|0⟩
|k⟩
$|\boldsymbol{x}_k^{(T)}\rangle$
$\hat{U}(\boldsymbol{\theta})$
$\hat{U}(\boldsymbol{\theta})$
$\hat{U}(\boldsymbol{\theta})$
Amplitude Amplification

Fig. 9. Sequential Prediction for an Asset. A block diagram of our proposed sequential prediction approach, where a fully trained PQC ($\hat{U}(\boldsymbol{\theta})$) over K stocks using quantum multi-task learning is used to predict $\tau = t\,(\in \{1, \dots, R\})$ consecutive future values by loading the continuation on a new ancilla for each t.

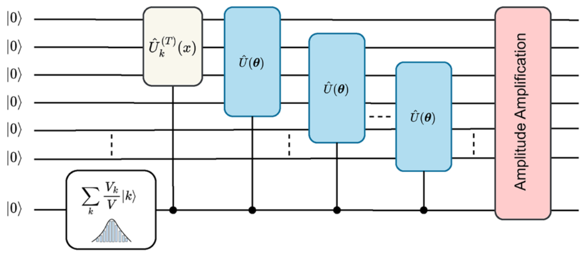

$|0\rangle$
$\hat{U}_k^{(T)}(x)$
$\hat{U}(\boldsymbol{\theta})$
$\sum_k \frac{V_k}{V}|k\rangle$
Amplitude Amplification

Fig. 10. Sequential Prediction for a Portfolio of Assets. A block diagram of our proposed sequential prediction approach, where a fully trained PQC ($\hat{U}(\boldsymbol{\theta})$) over K stocks using quantum multi-task learning is used to predict $\tau = t \, (\in \{1, \ldots, R\})$ consecutive future values for an entire portfolio $\sum_k \frac{V_k}{V} \left| k \right\rangle$.

Such a model can be utilized for quantum advantage on the downstream tasks, utilizing known quantum algorithms such as Ref[30]..

Numerical simulations

Loading contextual distribution

To encode a contextual distribution, $\mathcal{P}(\boldsymbol{x}^{(T)})$ onto the wavefunction, we use a quantum circuit, illustrated in Fig. 4. For this experiment, we use historical stock data from Apple, discretized to binary values where 0 indicates a price decrease, and 1 indicates an increase. We choose a context length of $T = 3$ and $\tau = 1$, allowing the model to incorporate three consecutive days of stock data to predict one day into the future. Consequently, the contextual distribution is encoded using three qubits, each representing a day's information. Our dataset comprises 10, 033 stock prices, which we average weekly with a stride length of 1 day, yielding 10, 029 samples.

This dataset is further divided into training (80%) and testing datasets (20%) by splitting. The quantum circuit includes a sequence of four layers ($L = 4$) of parameterized quantum sub-circuits (also shown in Fig. 4). We employ the SPSA rule in conjunction with the swap test to iteratively adjust the weights, an approach whose advantages are discussed in later sections. The model was trained for 3000 epochs with a learning rate of 0.1. The number of measurement shots was set to 10,000, and the SPSA perturbation parameter was fixed at 0.01. All experiments were initialized using a random seed of 42 to ensure reproducibility. These parameters are the same for all the subsequent results in the manuscript unless otherwise stated. This results in a contextual distribution loaded as depicted in Fig. 11. The corresponding training loss is illustrated in Fig. 12, demonstrating the model's convergence. As shown in Fig. 11, we observe that the contextual distribution is efficiently encoded onto the qubit states, accurately reflecting the intended probabilities.

Quantum single-task learning

With the contextual distribution now loaded, we proceed to predict the conditional probability $\mathcal{P}(x^{T+1}|\boldsymbol{x}^{(T)}, \boldsymbol{\theta})$ from the contextual distribution $\mathcal{P}(\boldsymbol{x}^{(T)})$ using a quantum circuit as shown in Fig. 5. To train this circuit, we use raw stock data from Apple and Google. The circuit employs eight qubits: the first three qubits store the contextual distribution loaded in the previous section, and a fourth qubit is designated for the prediction. The remaining four qubits encode the target conditional probability distribution $\mathcal{P}(x^{T+1}|\boldsymbol{x}^{(T)}, \boldsymbol{\theta})$, loaded similarly to the contextual. The circuit incorporates four layers ($L = 4$) of parameterized gates, which upon training, generate the wavefunction $\left| \boldsymbol{y}^{(T+1)} \right\rangle$, representing the predicted return for the context input $\left| \boldsymbol{x}^{(T)} \right\rangle$. To optimize this prediction, we apply the swap test, measuring the distance between the original and predicted conditional probabilities. This distance metric guides the training process by yielding gradients via the SPSA rule, allowing for precise adjustment of $\boldsymbol{\theta}$ to minimize prediction error and improve model accuracy. We trained the model for 3000 epochs with a learning rate of 0.1 to get the predicted conditional probability (or distributions) as displayed in Fig. 13.

The distance between the target conditional probability distribution and the predicted conditional probability distribution is quantified using the KL Divergence, providing a measure of similarity between the two distributions. These values are detailed in Table 1, offering insight into the model's accuracy and convergence during training. Lower KL Divergence values indicate closer alignment with the target distribution, demonstrating the effectiveness of our quantum circuit in capturing the underlying patterns of stock price movements.

Quantum multi-task learning

To train multiple stocks simultaneously, we employ a multi-task learning architecture, illustrated in Fig. 7, using both Apple and Google stock data. The circuit uses a control qubit, $\left| k \right\rangle$, to distinguish between the stocks, where $\left| k \right\rangle = \left| 0 \right\rangle$ represents Apple, and $\left| k \right\rangle = \left| 1 \right\rangle$ represents Google. We utilize a single set of parameterized gates, including $\hat{U}_s(\boldsymbol{\theta}_s)$, $\hat{U}_1^1(\boldsymbol{\theta}_1^1)$, and $\hat{U}_2^1(\boldsymbol{\theta}_2^1)$, running the circuit for 250 epochs per stock. Training begins with

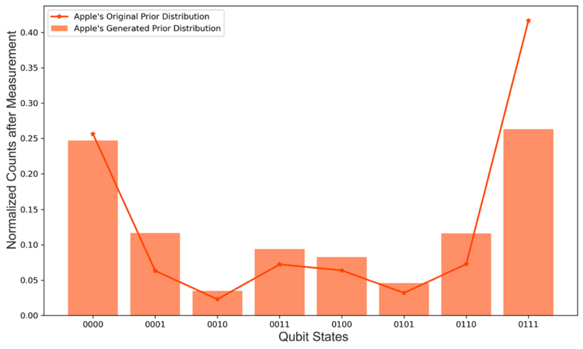


Fig. 11. Contextual Probability Distribution. A diagram showing the loaded contextual probability distribution ($\mathcal{P}(\boldsymbol{x}^{(T)})$) of Apple's stock data for a context size of $T = 3$.

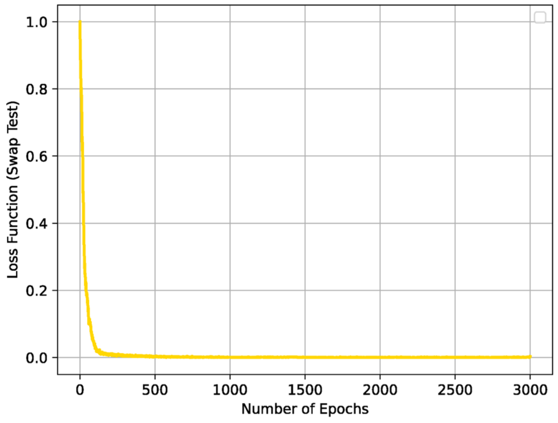


Fig. 12. Training Loss for loading context distribution. A diagram plotting the training loss for loading a contextual distribution ($\mathcal{P}(\boldsymbol{x}^{(T)})$) of Apple's stock data for a context size of $T = 3$.

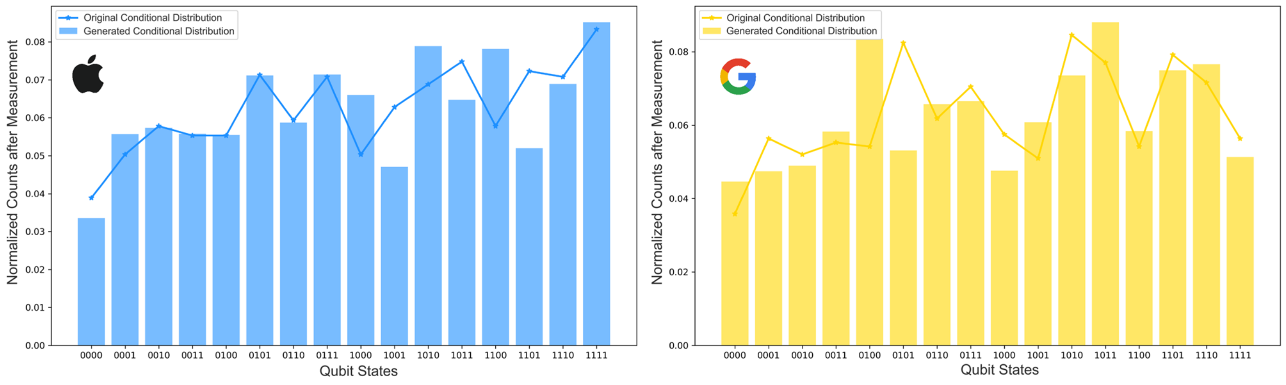


Fig. 13. Predictions of Quantum Single-Task Learning for $T = 3$. Generated conditional probability distributions of both Apple and Google datasets are shown comparing them to the original distributions. The model was trained for 3000 epochs with a learning rate of 0.1.

Apple ($\left|k\right\rangle = \left|0\right\rangle$) for 250 epochs, followed by Google ($\left|k\right\rangle = \left|1\right\rangle$) for another 250 epochs, both at a learning rate $\beta = 0.1$. The predicted conditional probability functions are shown in Fig. 14, where we observe a closer alignment to the target distributions compared to single-task learning. Table 1 further confirms this, showing that the KL Divergences in the multi-task learning case are lower. We attribute this improvement to correlations captured between the Apple and Google datasets, which induce a regularization effect on the circuit. This shared representation, attributed to the shared parameters $\hat{U}(\boldsymbol{\theta})$, allows the model to retain information from both stocks, creating an inductive bias that enhances performance.

Asset	Model	Parameters	KL Divergence
Apple	STL	32	0.1047
Google	STL	32	0.1239
Apple	MTL	16	0.0614
Google	MTL	16	0.0754

Table 1. Performance of QSTL vs QMTL for two selected stocks.

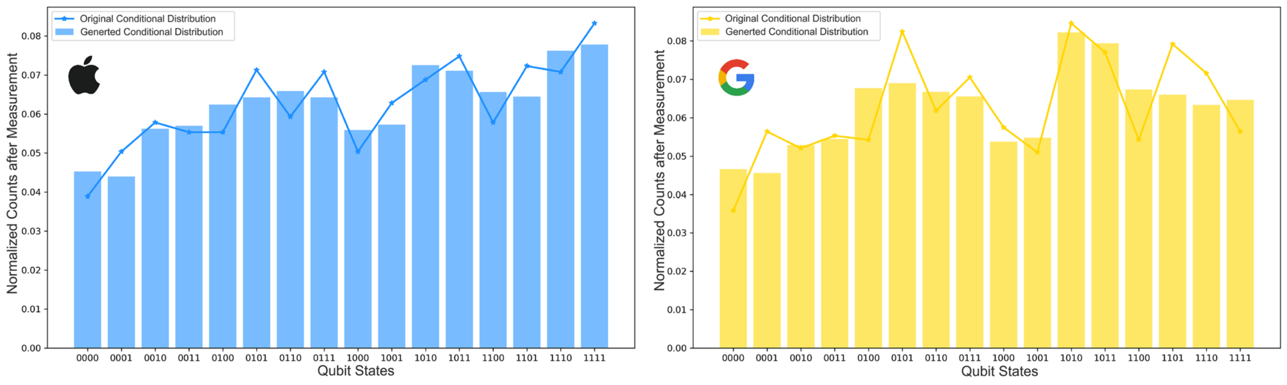


Fig. 14. Predictions of Quantum Multi-Task Learning for $T = 3$ and $K = 2$. Generated conditional probability distributions of both Apple and Google datasets are shown comparing them to the original distributions. The model was trained for 3000 epochs with a learning rate of 0.1.

It is important to note, however, that classical deep learning models can outperform quantum approaches in conventional large-scale time-series prediction tasks. This superiority arises from the scalability of classical architectures, which can accommodate millions of parameters, deep attention stacks, and long temporal contexts beyond the reach of near-term quantum systems. In such scenarios, Transformers and LSTMs achieve state-of-the-art performance in purely classical forecasting pipelines. Nevertheless, in this work, time-series prediction accuracy is not the sole key performance indicator. Our emphasis lies in efficient realization on quantum hardware, achieved through the QBGU mechanism, and enable sequential prediction as discussed in Section V. The proposed QMTL with QBGU framework can be repeatedly applied to predicted future stock states in superposition, allowing the exploration of all alternate future trajectories simultaneously. Such a process, where probabilistic futures are evolved and measured in parallel, remains infeasible for classical models, which must evaluate each scenario sequentially.

Convergence
The training losses for both models, QMTL and QSTL, are plotted in Fig. 16. We observe that the QMTL model converges significantly faster than the QSTL model, reflecting its improved learnability due to correlations across datasets. Notably, the QMTL loss curve for Google begins at a lower starting point. This is due to the sequential training, where Apple's dataset is processed first, followed by Google's. By the time Google's data is introduced, the model has already incorporated information from Apple, and the correlation between the stock prices of Apple and Google leads to a reduced initial loss for Google. This demonstrates how QMTL effectively leverages inter-dataset correlations to enhance learning efficiency and stability.

SWAP test versus mean-squared error loss
The motivation for using SPSA in optimizing a quantum neural network is well-established as it provides a gradient estimation technique compatible with quantum circuits, allowing efficient optimization by leveraging the circuit's differentiable structure without requiring classical backpropagation. Furthermore, we examine the differences between using the swap test and MSE loss in conjunction with SPSA. To investigate, we train two QSTL models on Apple and Google datasets, using MSE loss for 10, 000 epochs with a learning rate of 0.00001. The resulting predicted conditional probabilities are plotted in Fig. 17, where we observe that MSE loss paired with SPSA fails to capture the target distribution accurately. This limitation likely arises from the non-trigonometric nature of MSE, which does not align well with the periodicity inherent in quantum operations. Consequently, we exclusively utilized the swap test as the loss function throughout this paper to ensure optimal distribution learning. From the Fig. 17 and Fig. 13, we can conclude that our novel training method to exploit a conditionalized fidelity loss over quantum distributions shows significantly better performance compared to standard measurements for computing the MSE loss.

Scalability
To explore the scalability of QMTL, we expand the model to learn four stocks: Apple, Google, Microsoft, and Amazon. Using the circuit in Fig. 8, we get 3 control qubits, of which only two qubits ($\left|k_1\right\rangle$ and $\left|k_2\right\rangle$) are externally controlled. The circuit consists of one repetition of the trainable shared circuit $\hat{U}_s(\boldsymbol{\theta}_s)$, followed by the task-specific circuits $\hat{U}^1_{11}(\boldsymbol{\theta}^1_{11})$, $\hat{U}^1_{01}(\boldsymbol{\theta}^1_{01})$, $\hat{U}^1_{10}(\boldsymbol{\theta}^1_{10})$, and $\hat{U}^1_{00}(\boldsymbol{\theta}^1_{00})$. We train the model similarly to the two-stock case, with 250 epochs per stock and the same learning rate. The resulting predicted conditional probability distributions, plotted in Fig. 15, demonstrate that the model effectively learns the distributions for all four stocks while using the same number of trainable parameters as in the two-stock scenario. This scalability illustrates the robustness of the QMTL approach, which benefits from efficient parameter sharing and enhanced correlation capture, thereby outperforming single-task learning even as the number of assets grows.

Fig. 18 shows the loss function progression during the training of all stocks, illustrating the model's adaptability to each dataset. Each peak aligns with transitions between datasets, and we observe a gradual reduction in peak heights over time. This decline suggests that the model is capturing correlations between stocks, enhancing its compatibility across different datasets and resulting in consistently lower loss values. This trend underscores the QMTL model's ability to leverage shared patterns, improving overall training stability and efficiency across multiple assets.

To further evaluate the scalability and generalization capacity of the QMTL framework, we extend the architecture to simultaneously learn eight assets: Apple, Google, Microsoft, Amazon, Pepsi, Western Digital, Texas Instruments, and IBM. We average the asset prices over 3 days instead of 5 (weekly) for variability. This configuration expands the control space to four control qubits, of which three qubits ($\left|k_1\right\rangle$, $\left|k_2\right\rangle$, and $\left|k_3\right\rangle$) are externally modulated to select among eight task-specific circuits as shown in Fig. 20. The circuit structure remains analogous to the previous $K = 4$ setup.

We train the model for 100 epochs per stock and the same learning rate. The predicted conditional probability distributions, shown in Fig. 21, confirm that the model accurately learns all eight stocks while keeping the number of trainable parameters the same. Each distribution closely matches its target, showing that QMTL maintains good prediction quality even as the number of tasks increases. The shared circuit continues to capture common patterns across different stocks, while each task-specific part learns the unique behavior of an individual stock. Overall, the eight-stock results highlight the robustness and efficiency of the QMTL model in handling larger and more complex multi-asset learning problems.

To further explore how QMTL behaves under richer data representations, we conducted an additional experiment where stock prices were quantized into four discrete levels instead of two. This setting provides

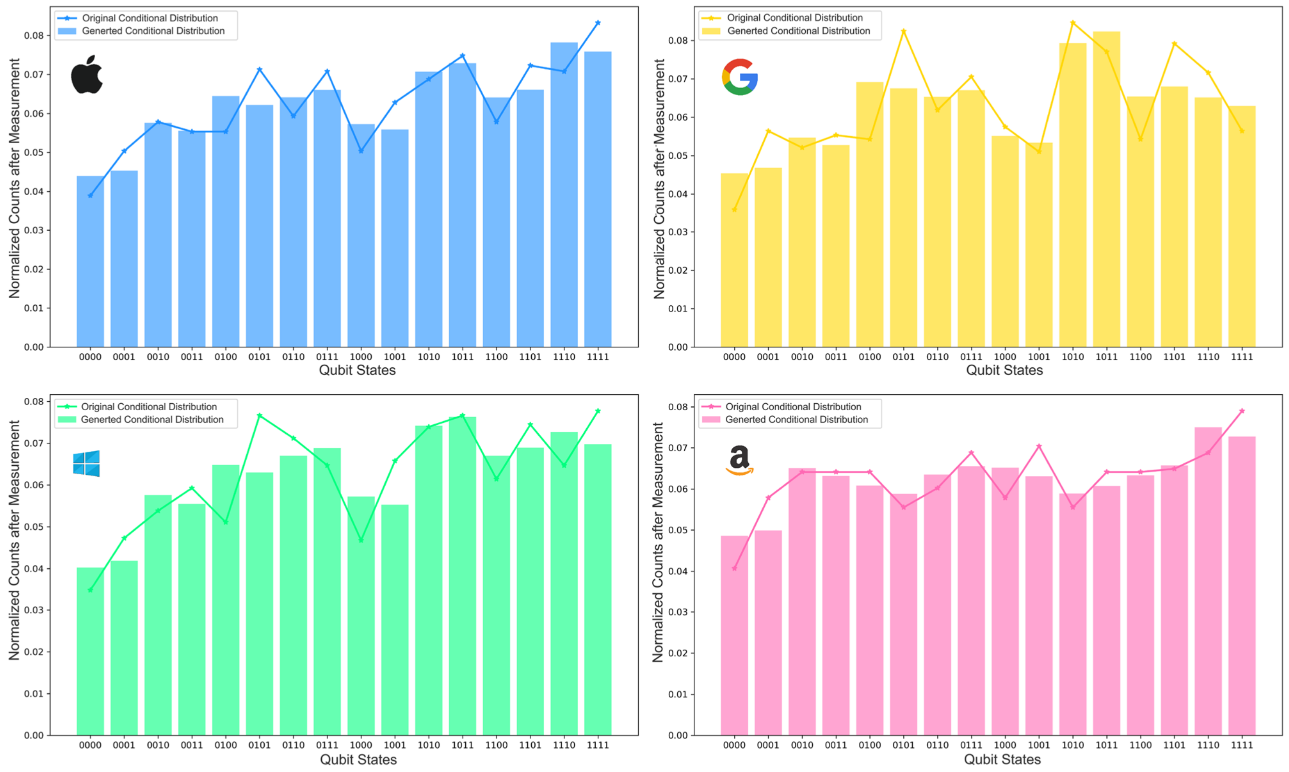


Fig. 15. Predictions of Quantum Multi-Task Learning for $T = 3$ and $K = 4$. Generated conditional probability distributions of Apple, Google, Microsoft, and Amazon datasets are shown comparing them to the original distributions. The model was trained for 3000 epochs with a learning rate of 0.1.

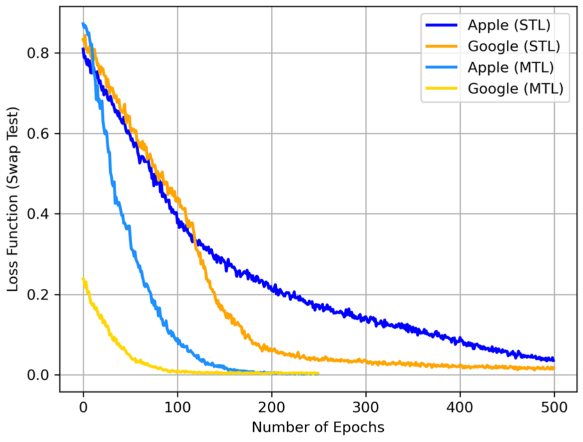


Fig. 16. Loss over training epochs for QSTL and QMTL. A diagram plotting loss function versus epochs for QMTL and QSTL.

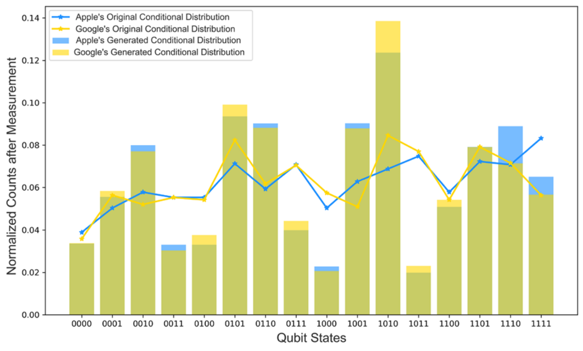


Fig. 17. MSE Loss. Representing the underlying conditional probability distributions of Apple and Google stocks after training with MSE Loss.

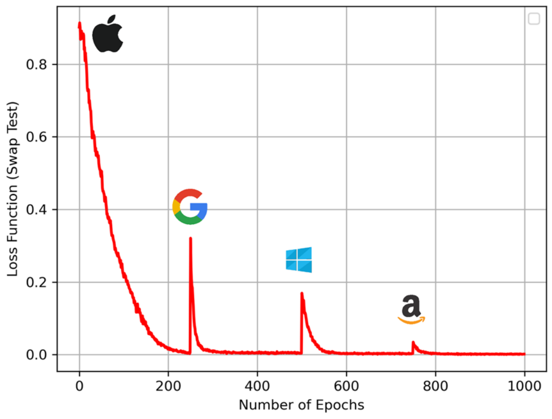


Fig. 18. QMTL Loss over training epochs of 4 assets. A diagram plotting loss function vs. epochs for QMTL ($K = 4$) with a learning rate of 0.1.

finer granularity in capturing subtle variations in stock movements while maintaining a manageable circuit size. Rather than dividing the range of returns into evenly spaced bins, we employed a non-uniform, density-based quantization scheme in which the boundaries between quantization levels are determined by the empirical distribution of the data. In this approach, the full range of price changes is partitioned according to their cumulative probability density so that each quantization level contains approximately the same proportion of data points. This means that smaller intervals are assigned to regions where the data is dense, while larger

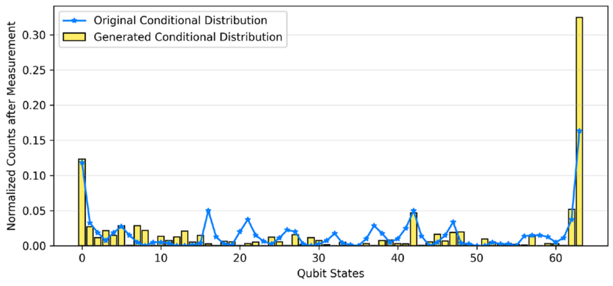


Fig. 19. Predictions of QMTL for $T = 2$ and $d = 4$. Generated conditional probability distribution of Apple dataset is shown comparing it to the original distribution. The model was trained for 3000 epochs with a learning rate of 0.1.

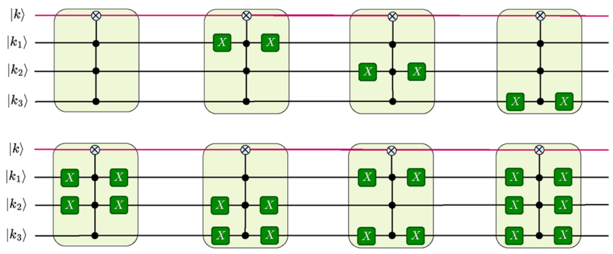


Fig. 20. QMTL Control Circuit for K=8 assets. Share-and-specify ansatz is not shown for simplicity.

intervals cover the sparse, tail regions where extreme price changes are rare. Such a representation ensures that all quantization levels are statistically meaningful and that the model allocates more expressive capacity to the most frequently occurring variations.

For this experiment, we focused on a context size of two ($T = 2$), rather than three, to keep the results intuitive and the number of possible qubit states within a reasonable range. The circuit is similar to Fig. 7 but context and prediction bits now correspond to two qubits each. As shown in Fig. 19, the predicted and original conditional probability distributions align closely, indicating that the model effectively captures the underlying statistical structure of the quantized data.

Noise

To study how hardware imperfections affect the model, we used Qiskit's AerSimulator to test QMTL on a single stock under different levels of depolarizing gate noise and readout error. Figure 22 shows the resulting average KL divergence between the noisy and ideal output distributions as the noise probability increases from 0 to 50%.

Both noise models were implemented using Qiskit's built-in noise framework. For depolarizing noise, the noise probability represents how likely it is that a qubit's state is randomly disturbed during a gate operation, replacing it with a completely mixed state[48]. For readout error, the noise probability indicates how often a measured bit is flipped during measurement[49]. Note that there are other types of noise prevalent in quantum circuits which are extensively studied in literature[50].

As the figure shows, both noise sources cause the KL divergence to increase as the noise becomes stronger, meaning that the output distribution moves further away from the noiseless reference. Depolarizing noise (red curve) shows a faster and larger rise in divergence, reaching around 0.1 at high noise levels, while readout error (purple curve) increases more gradually and almost linearly. This difference reflects the fact that gate noise affects the computation throughout the circuit, whereas readout noise influences only the final measurement.

Conclusion

In this paper, we presented a quantum multi-task learning (QMTL) architecture tailored for predicting stock prices distribution across multiple assets. By encoding contextual distributions into quantum states and leveraging a compact, parameter-efficient circuit, our approach capitalizes on shared patterns across stocks, resulting in enhanced accuracy and faster convergence compared to quantum single-task learning (QSTL). We optimized our training using quantum batch gradient update (QBGU) that enabled training over a distribution of inputs. The scalability of the QMTL model was validated by extending predictions to multiple stocks with the same circuit, showing its potential for broader financial applications. Our results underscore the viability of quantum machine learning for loading complex financial distributions, paving the way for future studies in quantum finance that harness the unique capabilities of multi-task quantum neural networks for more efficient and adaptive forecasting models and the utilization of such networks for downstream quantum prediction that enable quantum advantage.

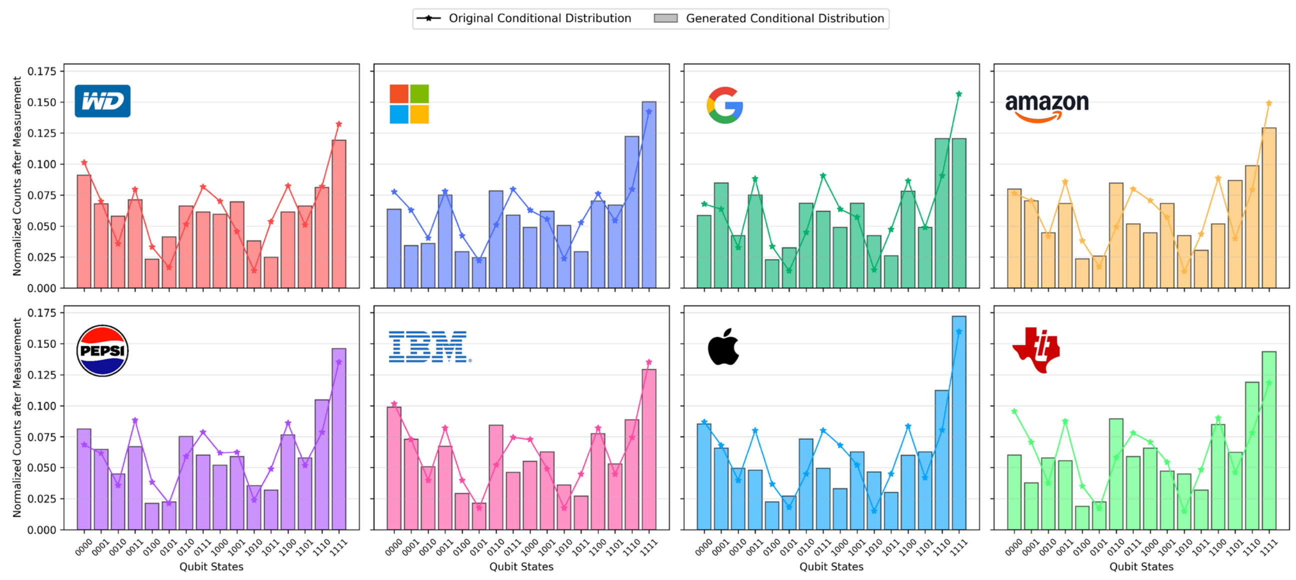


Fig. 21. Predictions of Quantum Multi-Task Learning for $T = 3$ and $K = 8$. Generated conditional probability distributions of various datasets are shown comparing them to the original distributions. The model was trained for 800 epochs with a learning rate of 0.1.

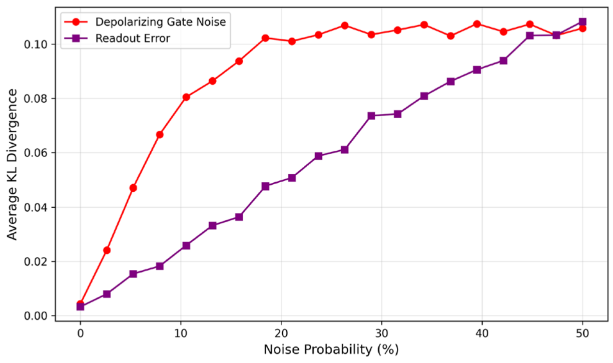


Fig. 22. Effect of Depolarizing Noise and Readout Error on QMTL. Measured KL divergences between noisy and noiseless output distributions from the QTML model for Apple stock prices with varying noise probabilities.

Data availability
Data generated in this study are available from the corresponding author upon reasonable request.

Received: 8 March 2025; Accepted: 29 December 2025
Published online: 07 January 2026

References
1. Shor, P. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review* **41**(2), 303–332 (1999).
2. Harrow, A., Hassidim, A. & Lloyd, S. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.* **103**, 150502 (2009).
3. Brassard, G., Hoyer, P., Mosca, M. & Tapp, A. Quantum amplitude amplification and estimation. *Contemporary Mathematics* **305**, 53–74 (2002).
4. Aharonov, D., Jones, V., & Landau, Z. A polynomial quantum algorithm for approximating the Jones polynomial. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*. 427–436 (2006).
5. Rebentrost, P., Mohseni, M. & Lloyd, S. Quantum support vector machine for big data classification. *Physical review letters* **113**(13), 130503 (2014).
6. Rietsche, R. et al. Quantum computing. *Electronic Markets* **32**(4), 2525–2536 (2022).
7. Arute, F. et al. Quantum supremacy using a programmable superconducting processor. *Nature* **574**(7779), 505–510 (2019).
8. Preskill John. Quantum computing: pro and con-*Proc. R. Soc. Lond.* A.454469–486 (1998)
9. Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum* **2**, 79. https://doi.org/10.22331/q-2018-08-06-79 (2018).
10. Egger, D. J. et al. Quantum Computing for Finance: State-of-the-Art and Future Prospects. *IEEE Transactions on Quantum Engineering* **1**, 1–24. https://doi.org/10.1109/TQE.2020.3030314 (2020).

11. Egger, D. J., Gutierrez, R. G., Mestre, J. & Woerner, S. Credit Risk Analysis Using Quantum Computers. *IEEE Transactions on Computers* **70**(12), 2136–2145. https://doi.org/10.1109/TC.2020.3038063 (2021).
12. Orús, R., Mugel, S. & Lizaso, E. Forecasting financial crashes with quantum computing. *Phys. Rev. A* **99**, 060301. https://doi.org/10.1103/PhysRevA.99.060301 (2019).
13. Rebentrost, P. & Lloyd, S. Quantum Computational Finance: Quantum Algorithm for Portfolio Optimization. *Künstl Intell* (2024). https://doi.org/10.1007/s13218-024-00870-9
14. Biamonte, J. et al. Quantum machine learning. *Nature* **549**(7671), 195–202. https://doi.org/10.1038/nature23474 (2017).
15. Ciliberto, C., et al. Quantum machine learning: a classical perspective. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, **474**(2209), (2018). https://doi.org/10.1098/rspa.2017.0551
16. Abbas, A., et al. The power of quantum neural networks. *Nat Comput Sci* 1, 403–409 (2021). https://doi.org/10.1038/s43588-021-00084-1
17. Sim, S., Johnson, P. D. & Aspuru-Guzik, A. Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms. *Adv. Quantum Technol.* **2**(12), 1900070. https://doi.org/10.1002/qute.201900070 (2019).
18. Rebentrost, P., Mohseni, M. & Lloyd, S. Quantum Support Vector Machine for Big Data Classification. *Phys. Rev. Lett.* **113**, 130503. https://doi.org/10.1103/PhysRevLett.113.130503 (2014).
19. Havlíček, V. et al. Supervised learning with quantum-enhanced feature spaces. *Nature* **567**(7747), 209–212. https://doi.org/10.1038/s41586-019-0980-2 (2019).
20. Amin, M. H., Andriyash, E., Rolfe, J., Kulchytskyy, B. & Melko, R. *Quantum Boltzmann Machine. Phys. Rev. X* **8**, 021050. https://doi.org/10.1103/PhysRevX.8.021050 (2018).
21. Bausch, J. Recurrent Quantum Neural Networks. In Vol. 33 (eds. H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin), *Advances in Neural Information Processing Systems*, 1368–1379 (2020).
22. Lloyd, S. & Weedbrook, C. Quantum Generative Adversarial Learning. *Phys. Rev. Lett.* **121**, 040502. https://doi.org/10.1103/PhysRevLett.121.040502 (2018).
23. Dong, D., Chen, C., Li, H., & Tarn, T.-J. Quantum Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **38**(5), 1207–1220. (2008). https://doi.org/10.1109/TSMCB.2008.925743
24. Fujii, K. & Nakajima, K. Harnessing Disordered-Ensemble Quantum Dynamics for Machine Learning. *Phys. Rev. Appl.* **8**, 024030. https://doi.org/10.1103/PhysRevApplied.8.024030 (2017).
25. Stamatopoulos, N. et al. Option Pricing using Quantum Computers. *Quantum* **4**, 291. https://doi.org/10.22331/q-2020-07-06-291 (2020).
26. Idrees, S., Alam, M. & Agarwal, P. A Prediction Approach for Stock Market Volatility Based on Time Series Data. *IEEE Access* **7**, 17287–17298 (2019).
27. Emmanoulopoulos, D., & Dimoska, S. Quantum Machine Learning in Finance: Time Series Forecasting. (2022). arXiv [Quant-Ph]. Retrieved from http://arxiv.org/abs/2202.00599
28. Li, Gushu, Anbang, Wu., Shi, Yunong, Javadi-Abhari, Ali & Ding, Yufei. *& Yuan Xie* (Paulihedral, 2021).
29. Liu, G. & Ma, W. A quantum artificial neural network for stock closing price prediction. *Information Sciences* **598**, 75–85. https://doi.org/10.1016/j.ins.2022.03.064 (2022).
30. Woerner, S. & Egger, D. Quantum risk analysis. *npj Quantum Information* **5**(1), 15 (2019).
31. Kandala, A., et al. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* 549, 242–246 (2017). https://doi.org/10.1038/nature23879
32. Leone, L., Oliviero, S., Cincio, L. & Cerezo, M. On the practical usefulness of the Hardware Efficient Ansatz. *Quantum* **8**, 1395 (2024).
33. Lov Grover, & Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. (2002).
34. Paquet, E. & Soleymani, F. QuantumLeap: Hybrid quantum neural network for financial predictions. *Expert Systems with Applications* **195**, 116583. https://doi.org/10.1016/j.eswa.2022.116583 (2022).
35. Orlandi, F., Barbierato, E. & Gatti, A. Enhancing Financial Time Series Prediction with Quantum-Enhanced Synthetic Data Generation: A Case Study on the S&P 500 Using a Quantum Wasserstein Generative Adversarial Network Approach with a Gradient Penalty. *Electronics* **13**, 2158 (2024).
36. Pistoia, M., et al. Quantum Machine Learning for Finance ICCAD Special Session Paper. *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)* (2021).
37. Zoufal, C., Lucchi, A. & Woerner, S. "Quantum Generative Adversarial Networks for learning and loading random distributions". *npj Quantum Inf* 5, 103 (2019).
38. Zhu, E. Y. et al. Generative quantum learning of joint probability distribution functions. *Phys. Rev. Res.* **4**, 043092. https://doi.org/10.1103/PhysRevResearch.4.043092 (2022).
39. Xia, W. et al. Configured quantum reservoir computing for multi-task machine learning. *Science Bulletin* **68**(20), 2321–2329. https://doi.org/10.1016/j.scib.2023.08.040 (2023).
40. Caruana, R. Multitask Learning. *Machine Learning* **28**(1), 41–75. https://doi.org/10.1023/A:1007379606734 (1997).
41. Buhrman, H., Cleve, R., Watrous, J. & de Wolf, R. Quantum fingerprinting. *Physical Review Letters* **87**(16), 167902. https://doi.org/10.1103/PhysRevLett.87.167902 (2001).
42. Takaki, Y., Mitarai, K., Negoro, M., Fujii, K. & Kitagawa, M. Learning temporal data with a variational quantum recurrent neural network. *Phys. Rev. A* **103**, 052414. https://doi.org/10.1103/PhysRevA.103.052414 (2021).
43. Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. Quantum circuit learning. *Phys. Rev. A* **98**, 032309. https://doi.org/10.1103/PhysRevA.98.032309 (2018).
44. Spall, J. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control* **37**(3), 332–341 (1992).
45. Harrow, A. W., "Small quantum computers and large classical data sets," arXiv preprint arXiv:2004.00026, (2020).
46. Wu, S., Zhang, Y. & Li, J. Quantum data parallelism in quantum neural networks. *Physical Review Research* **7**(1), 013177 (2025).
47. Srivastava, N., Belekar, G., Shahakar, N., & A. Babu H., The Potential of Quantum Techniques for Stock Price Prediction, *2023 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE), Kerala, India,* 1-7 (2023) https://doi.org/10.1109/RASSE60029.2023.10363533.
48. https://qiskit.github.io/qiskit-aer/stubs/qiskit_aer.noise.depolarizing_error.html#qiskit_aer.noise.depolarizing_error
49. https://qiskit.github.io/qiskit-aer/stubs/qiskit_aer.noise.ReadoutError.html#qiskit_aer.noise.ReadoutError
50. Rivas, ´angel, & Huelga, S. F. Open Quantum Systems: An Introduction. (2012). https://doi.org/10.1007/978-3-642-23354-8

Acknowledgements
The authors thank Sarvagya Upadhyay, Yasuhiro Endo, and Hirotaka Oshima for their thoughtful comments. The authors thank anonymous reviewers for their insightful and constructive feedback.

Author contributions
All authors wrote the main manuscript text and prepared figures 1-10. All authors contributed to the mathematical framework and designed the associated methods. S.M. prepared figures 11-19 and ran the associated

numerical simulations.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to S.M.

Reprints and permissions information is available at www.nature.com/reprints.