

Contextual Quantum Neural Networks for Stock Price Prediction

Sharan Mourya*

*Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign and
Fujitsu Research of America*

Hannes Leipold and Bibhas Adhikari

Fujitsu Research of America

(Dated: March 5, 2025)

In this paper, we apply quantum machine learning (QML) to predict the 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, enhancing adaptability and precision. 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 (QM TL) 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. This architecture represents the first of its kind in quantum finance, offering superior predictive power and computational efficiency for multi-asset stock price forecasting. Through extensive experimentation on S&P 500 data for Apple, Google, Microsoft, and Amazon stocks, we demonstrate that our approach not only outperforms quantum single-task learning (QSTL) models but also effectively captures inter-asset correlations, leading to enhanced prediction accuracy. 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

I. INTRODUCTION

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[1–4]. 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 [5, 6]. 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 [7, 8]. 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 [8] - quantum methods can provide approximate solutions to certain financial problems, making them highly relevant in the immediate future [9].

Machine learning has become essential for financial tasks like, asset management [9], risk analysis [10], crash detection [11], and portfolio optimization [12]. 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) [13] 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 [14]. 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 [15]. This implies that problems deemed challenging for classical systems could see significant improvements through the

* Correspondence email address: sharanmourya7@gmail.com

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 [9]. 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 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 [16], while support vector machines (SVM) have been utilized for data classification [17]. 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 [18]. Quantum versions of machine learning models, such as Boltzmann machines [19], recurrent neural networks [20], generative adversarial neural networks [21], reinforcement learning [22], and reservoir computing [23], have been extensively studied.

In finance, quantum machine learning has been applied to various tasks, including options pricing [24], time-series forecasting [26], and stock price prediction [28]. A notable example is a hybrid architecture developed for financial predictions [33], while quantum Wasserstein generative adversarial neural networks were employed for time-series predictions on the S&P 500 [34]. Unsupervised quantum machine learning has also been explored for clustering and fraud detection [35]. 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 [36], and quantum Wasserstein GANs have achieved similar tasks with a gradient penalty, improving performance over previous approaches [34].

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 [37], 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 [38] applied quantum reservoir computing and multi-task learning [39], 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 developed a training technique, quantum batch gradient update (QBGU), that exploits the linearity of quantum mechanics 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 prices of multiple assets, which, to the best of our knowledge, is the first of its kind. Our model allows for high quality multi-asset prediction, enabling quantum advantage in downstream tasks over *portfolios* of assets with logarithmic overhead in the number of qubits.

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.

II. 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.

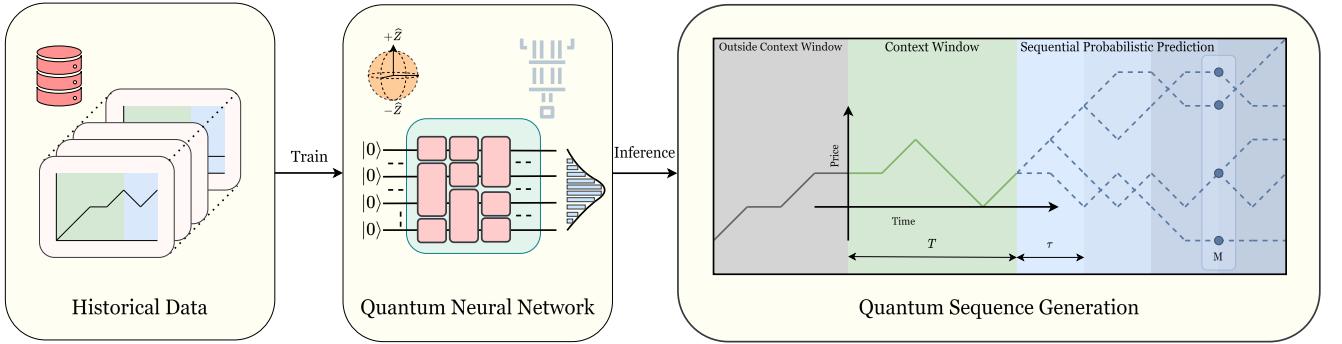


Figure 1. Quantum Neural Networks for Contextual Sequence Generation. Given historical data of context and continuations, a Quantum Neural Network is trained to produce quantum distributions over future prices, enabling downstream quantum advantage for tasks (labeled M in the rightmost diagram at a particular future) over all sequences in superposition.

A. 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 [25]. 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 preprocessed value 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 $\mathbf{X}^{(T)} = (X^1, X^2, \dots, X^T)$. Then the task of a prediction model f is to predict the value(s) of $\mathbf{X}^{(T+\tau)} \setminus \mathbf{X}^{(T)} := (X^{T+1}, X^{T+2}, \dots, X^{T+\tau})$ as a probability distribution $f(\mathbf{X}^{(T+\tau)} \setminus \mathbf{X}^{(T)}; \mathbf{X}^{(T)})$ for some values of the future time $\tau \geq 0$. To be specific, given $\mathbf{X}^{(T)} = \mathbf{x}^{(T)}$, the task of the predic-

tion model is to assign probabilities to future states $\mathbf{x}^{(T+\tau)} \setminus \mathbf{x}^{(T)} := (x^{T+1}, x^{T+2}, \dots, x^{T+\tau})$, the possible asset-price at the future times between $T + 1$ and $T + \tau$, as $f(\mathbf{x}^{(T+\tau)} \setminus \mathbf{x}^{(T)}; \mathbf{x}^{(T)})$ and the efficiency of f is determined by the closeness of a distance between $f(\mathbf{x}^{(T+\tau)} \setminus \mathbf{x}^{(T)}; \mathbf{x}^{(T)})$ and the observed distribution over $\mathbf{x}^{(T+\tau)} \setminus \mathbf{x}^{(T)}$ to zero. Thus the design of a prediction model $f(\mathbf{X}^{(T+\tau)} \setminus \mathbf{X}^{(T)}; \mathbf{X}^{(T)}, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$ is concerned with modeling f such that the loss function $\mathcal{L}(f(\mathbf{x}^{(T+\tau)} \setminus \mathbf{x}^{(T)}; \mathbf{x}^{(T)}), \mathbf{x}^{(T+\tau)} \setminus \mathbf{x}^{(T)})$, which estimates a notion of closeness between the probability distribution $f(\mathbf{X}^{(T+\tau)} \setminus \mathbf{X}^{(T)}; \mathbf{X}^{(T)})$ and observed frequencies $\mathbf{X}^{(T+\tau)} \setminus \mathbf{X}^{(T)}$ 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 $\mathbf{x}^{(T)}$ into a quantum state which will be evolved under a unitary transformation. We define the T -qudit quantum state as

$$|\mathbf{x}^{(T)}\rangle = |x^1\rangle \otimes |x^2\rangle \otimes \cdots \otimes |x^T\rangle \quad (1)$$

to encode $\mathbf{x}^{(T)}$ after encoding the context data point x^t into its corresponding qudit $|x^t\rangle$ 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, $|x^t\rangle$, represents the quantized return of an asset at time t . The mapping of returns to the orthogonal states $|0\rangle, |1\rangle, \dots, |d-1\rangle$ 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 inter-

vals of length

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

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

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

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

$$|\mathbf{y}^{(T+\tau)}\rangle = \sum_{\phi \in \{0, 1, \dots, d-1\}^{T+\tau}} c(\phi) |\phi\rangle, \quad (3)$$

where $|\phi\rangle = |\phi^1\rangle \otimes |\phi^2\rangle \cdots \otimes |\phi^{T+\tau}\rangle$ is a $T + \tau$ -qudit basis state with $\phi^t \in \{0, \dots, d-1\}$ 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:

$$\rho^{(T+\tau)/(T)} = \text{Tr}_{1\dots T} |\mathbf{y}^{(T+\tau)}\rangle\langle \mathbf{y}^{(T+\tau)}|. \quad (4)$$

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 ($|\mathbf{x}^{(T)}\rangle$), making it not suitable for reuse for repeated predictions. To circumvent that, we require the whole input and output vectors ($|\mathbf{y}^{(T+\tau)}\rangle, |\mathbf{x}^{(T+\tau)}\rangle$) 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:

$$|\mathbf{y}^{(T+\tau)}\rangle \approx \sum_{\phi \in \{0, 1, \dots, d-1\}^\tau} c(\phi) |\mathbf{x}^{(T)}\rangle |\phi^{T+1}\rangle \dots |\phi^{T+\tau}\rangle, \quad (5)$$

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

$$|\phi^{T+t}\rangle \rightarrow x_{\min} + \phi^{T+t} \cdot \Delta x. \quad (6)$$

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:

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

Given M measurement samples, we define the resulting distribution $f_M(\mathbf{X}^{(T+\tau)/(T)}; \mathbf{x}^{(T)})$ based on the frequency of observing specific continuation strings $\mathbf{x}^{(T+\tau)/(T)}$; 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

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

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 $|0\rangle$ and $|1\rangle$ 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. In addition, during training, we also consider the statistics of the contextual data including the contextual probability distribution $\mathcal{P}(\mathbf{X}^{(T)})$, the target probability distribution $\mathcal{P}(\mathbf{X}^{(T+1)})$, the conditional probability distribution $\mathcal{P}(X^{T+1}|\mathbf{X}^{(T)})$, and the total probability distribution $\mathcal{P}(X^{T+1}, \mathbf{X}^{(T)})$. In particular, we train $\boldsymbol{\theta}$ such that $f(X^{T+1}; \mathbf{X}^{(T)}, \boldsymbol{\theta}) \approx \mathcal{P}(X^{T+1}|\mathbf{X}^{(T)})$.

B. 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 state $\hat{U}(\boldsymbol{\theta})|\Psi\rangle$ for any input state $|\Psi\rangle \in \mathbb{C}^{2^n}$, where $\boldsymbol{\theta}$ is the set of (real) parameters in the QNN and $|\Psi\rangle$ 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.

1. Loading Classical Data

The first step in a QNN involves encoding classical data into quantum states. This is typically done using quantum feature maps [18], where classical input data $\Psi \in \mathbb{C}^\gamma, 1 \leq \gamma \leq 2^n$ is encoded into a quantum state $|\Psi\rangle$, where 2^n is its dimension. This state can be prepared by

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

where $\hat{U}_F(\Psi)$ is a quantum feature map circuit that depends on the classical data Ψ , and $|0\rangle^{\otimes n}$ 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 [27].

2. 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 $\boldsymbol{\theta}^l = \{\theta^{l11}, \theta^{l12}, \dots, \theta^{lnc}\}$ such that $m (= Lnc)$ is the total number of parameters in the circuit, $n (= m/Ln)$ is the number of qubits and $c (= m/Ln)$ is the number of sub-layers in each layer. This representation will be explored in detail

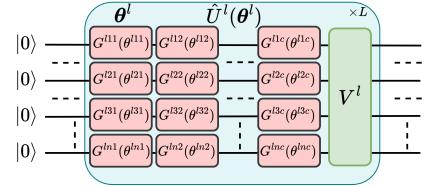


Figure 2. Parameterized Quantum Circuit. Block diagram of the layered parametric quantum circuit showing various blocks in the l^{th} layer such as fixed unitary and parametric rotation gates.

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 $|\mathbf{x}^{(T)}\rangle$, 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 $|0\rangle^{\otimes \tau}$, the output state is given by

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

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.

3. 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 [42] and simultaneous perturbation stochastic approximation (SPSA) [43]. 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 $\langle \hat{B} \rangle_{\theta} = \langle \mathbf{x}^{(T+\tau)} | \hat{U}^{\dagger}(\theta) \hat{B} \hat{U}(\theta) | \mathbf{x}^{(T+\tau)} \rangle$. With this, the gradient update rules are given by[42][43]:

1. Parameter-Shift

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

2. SPSA

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

where δ is hyperparameter, typically between 0 and 1 and α is a stochastic perturbation vector with dimensions same as θ . The vector α 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.

4. 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:

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

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 [40] to compare the output state $|\mathbf{y}^{(T+\tau)}\rangle$ with the target state $|\mathbf{x}^{(T+\tau)}\rangle$. An ancilla qubit is prepared in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, whose

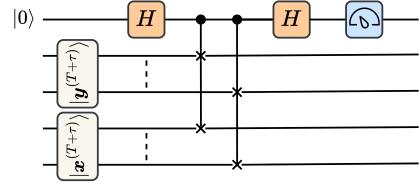


Figure 3. **SWAP Test.** A diagram of the SWAP test, which measures the fidelity loss between two wavefunction states $|\mathbf{x}^{(T+\tau)}\rangle$ and $|\mathbf{y}^{(T+\tau)}\rangle$.

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 $|0\rangle$ after measurement of the ancilla qubit is proportional to the fidelity $|\langle \mathbf{y}^{(T+\tau)} | \mathbf{x}^{(T+\tau)} \rangle|^2$ and is given by:

$$\mathcal{P}(0) = \frac{1 + |\langle \mathbf{y}^{(T+\tau)} | \mathbf{x}^{(T+\tau)} \rangle|^2}{2} \quad (13)$$

and the corresponding fidelity loss is:

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

III. LOADING CLASSICAL DATA

Contextual data $\mathbf{x}^{(T)}$ can be encoded onto a quantum state using feature maps [27], 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 [30]. 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 $\mathcal{P}(\mathbf{X}^{(T)})$, where $\mathbf{X}^{(T)}$ 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

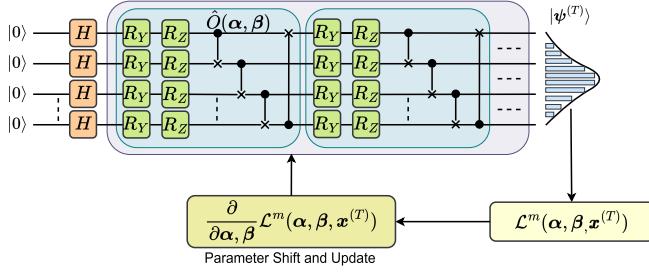


Figure 4. **Loading a distribution onto a hardware efficient ansatz.** The circuit inside the blue box ($\hat{O}(\alpha, \beta)$) is applied sequentially for a sufficient number of iterations followed by an MSE loss with the SPSA update rule.

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 $\mathcal{P}(\mathbf{X}^{(T)} = (i^1, \dots, i^T))$, where i^t denotes the quantized return at time t . Thereafter, we load the contextual distribution onto the T -qubit state as follows

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

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

$$|\psi^{(T)}\rangle = \sum_{\mathbf{x}^{(T)} \in \{0,1\}^T} \sqrt{\mathcal{P}(\mathbf{x}^{(T)})} |\mathbf{x}^{(T)}\rangle \quad (16)$$

Eq. (16) can be achieved using the Grover-Rudolph technique for state preparation [32]. 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 [30] 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 [31], 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:

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

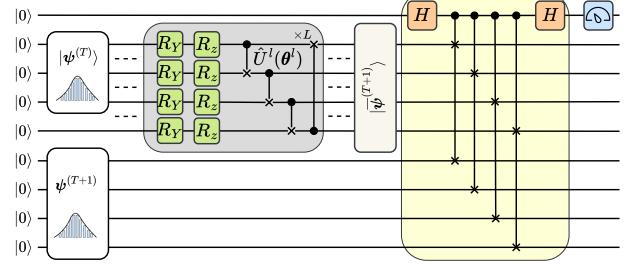


Figure 5. **Quantum Batch Learning.** A diagram showing the learning procedure of our proposed quantum batch gradient update for a context of T . The top most qubit is an ancilla qubit. For a batch, a distribution over inputs is loaded succeeding qubits (the input qubits) and the following qubit(s) is 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 ($\hat{U}(\theta)$) 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.

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.

IV. 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).

Unlike previous approaches [34, 36], 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 $|\psi^{(T)}\rangle$, which, along with a predic-

tion qubits, forms the input of the quantum circuit: $|\psi^{(T)}\rangle \otimes |0\rangle^{\otimes \tau}$. Hereafter, we will stick to $\tau = 1$ and binary quantization unless otherwise stated. We also load the target distribution $|\psi^{(T+1)}\rangle$ 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 $\mathcal{P}(X^{T+1}|\mathbf{x}^{(T)}, \boldsymbol{\theta})$. 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.

A. Quantum Batch Gradient Update

Unlike previous models [34], we do not rely on a feature map to load individual contextual samples during training. Instead, we load the entire contextual distribution at once according to Eq. (16), resulting in a less convoluted loss function which leads to higher quality gradients. Additionally, this approach 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), where the input of the PQC is $|\mathbf{x}^{(T)}\rangle$ and the forward pass of this input through the PQC is $|\mathbf{y}^{(T+1)}\rangle = \hat{U}(\boldsymbol{\theta})(|\mathbf{x}^{(T)}\rangle \otimes |0\rangle)$ as shown in Fig. 5. Let the gradient update for $\boldsymbol{\theta}$, obtained from SPSA, be denoted as $g(\boldsymbol{\theta}, \mathbf{x}^{(T+1)})$, then

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

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

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

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 $|\psi^{(T)}\rangle$ from Eq. (16), then the forward pass through the PQC is given by:

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

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

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

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. With this new quantum batch gradient update (QBGU) rule, we efficiently leverage the quantum circuit's ability to process the entire dataset in a single step, dramatically reducing the computational overhead of traditional gradient-based optimization methods. Consequently, we enable faster convergence and scalability in quantum machine learning applications, making it especially suitable for large datasets and complex probabilistic models. Note that these advantages depend on the accuracy of the state preparation as described in Eq. (16). Using the QBGU training process, the model $\hat{U}(\boldsymbol{\theta})$ in Fig. 5 is trained to learn the conditional probability distribution $\mathcal{P}(x^{T+1}|\mathbf{x}^{(T)}, \boldsymbol{\theta})$, which essentially maps each input $|\mathbf{x}^{(T)}\rangle$ to its corresponding output $|\mathbf{y}^{(T+1)}\rangle$ at the time of inference. As shown

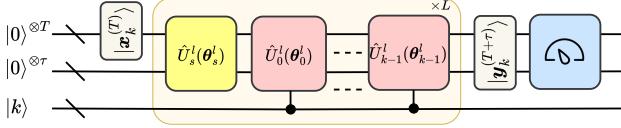


Figure 6. **Share-and-specify Ansatz.** A diagram of our Quantum Multi-Task Learning Architecture showing various components for the QNN. A single asset state is loaded by setting the label $|k\rangle$ and the inference-time context $\mathbf{x}^{(T)}$ is loaded over qubits $|0\rangle^{\otimes T+\tau}$. The input can then be processed through the parameterized circuit, composed of L layers of the share-and-specify ansatz, to define a state $|\mathbf{y}^{(T)}\rangle$ 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 .

in Fig. 5, we preload both the contextual distribution $|\psi^{(T)}\rangle$ and the target distribution $|\psi^{(T+1)}\rangle$, and the SWAP test then measures the distance between the estimated and the original target distributions, guiding the training process.

V. QUANTUM MULTI-TASK LEARNING

Multi-task learning (MTL) [39] 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).

A. 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,

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

where $\mathbb{1}_K$ is the identity operator with dimension $K \times K$. The label qubits, collectively represented by the qudit $|k\rangle = |k_1\rangle \otimes |k_2\rangle \cdots \otimes |k_{\log K}\rangle$ 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

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

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

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

where $\boldsymbol{\theta}_{ck}^l = \boldsymbol{\theta}_k^l$, but the subscript c is added for notational consistency and $|k^\perp\rangle\langle k^\perp| = \mathbb{1}_K - |k\rangle\langle k|$ is a projection onto the orthogonal space of $|k\rangle\langle k|$, such that $\otimes_i G_k^{lij}(\theta_k^{lij})$ is only applied when the label qudit is $|k\rangle$. 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 $(G_k^{lij}(\theta_k^{lij}) \otimes |k\rangle\langle k| + \mathbb{1}_{2^{T+\tau}} \otimes |k^\perp\rangle\langle k^\perp|)$ is a control rotation based on label $|k\rangle$. 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

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

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 $|\mathbf{x}_k^{(T)}\rangle$, undergoes the transformation determined by the label qudit and the task-specific ansatz:

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

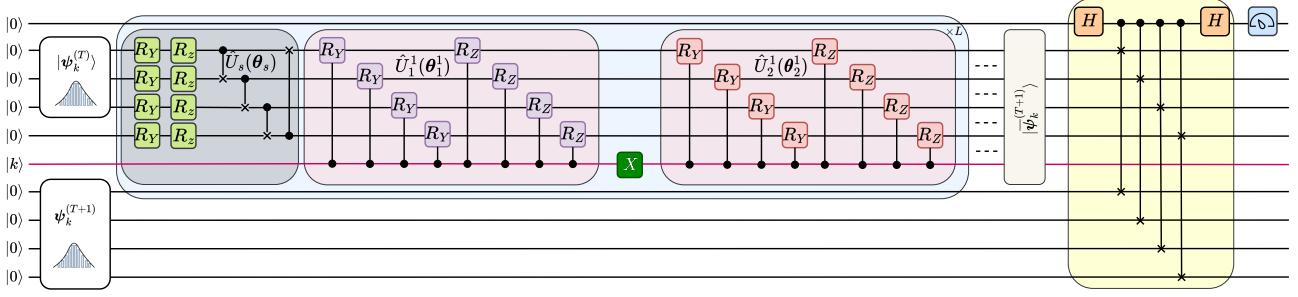


Figure 7. **Quantum Multi-Task Learning Architecture for $T = 3$ and $K = 2$.** The circuits inside the Grey ($\hat{U}_s(\theta_s)$) and Pink boxes ($\hat{U}_1^l(\theta_1^l)$, $\hat{U}_2^l(\theta_2^l)$) are applied sequentially for a required number of iterations followed by the SWAP test.

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.

1. Two-Assets Case

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

$$|\mathbf{y}_1^{(T+1)}\rangle = \hat{U}_1^l(\theta_1^l)\hat{U}_s^l(\theta_s^l)(|\mathbf{x}_1^{(T)}\rangle \otimes |0\rangle \otimes |0\rangle)$$

$$|\mathbf{y}_2^{(T+1)}\rangle = \hat{U}_2^l(\theta_2^l)\hat{U}_s^l(\theta_s^l)(|\mathbf{x}_2^{(T)}\rangle \otimes |0\rangle \otimes |1\rangle)$$

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 $\hat{U}_s(\theta_s^l)$ is shared across all assets, while the operators $\hat{U}_1^l(\theta_1^l)$ and $\hat{U}_2^l(\theta_2^l)$ are task-specific trainable PQCs. The simplest way to switch between tasks is by applying an X gate, as illustrated in Fig. 7.

2. 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 $\hat{U}_k^l(\theta_k^l)$ is active for

each unique combination of $|k_1\rangle \otimes |k_2\rangle$, corresponding to a specific asset. The task-specific PQCs and their associated controls on the qubits ($|k\rangle \otimes |k_1\rangle \otimes |k_2\rangle$) can be expressed as:

$$\hat{U}_{11}^l(\theta_{11}^l) : (\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes \mathbb{1}_2) \text{CCNOT}(\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes \mathbb{1}_2) \quad (25)$$

$$\hat{U}_{01}^l(\theta_{01}^l) : (\mathbb{1}_2 \otimes X \otimes \mathbb{1}_2) \text{CCNOT}(\mathbb{1}_2 \otimes X \otimes \mathbb{1}_2) \quad (26)$$

$$\hat{U}_{10}^l(\theta_{10}^l) : (\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes X) \text{CCNOT}(\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes X) \quad (27)$$

$$\hat{U}_{00}^l(\theta_{00}^l) : (\mathbb{1}_2 \otimes X \otimes X) \text{CCNOT}(\mathbb{1}_2 \otimes X \otimes X) \quad (28)$$

where, $\mathbb{1}_2$ represents the identity gate on one qubit, and CCNOT denotes the Toffoli gate, which operates on $|k\rangle$ with control inputs $|k_1\rangle$ and $|k_2\rangle$ 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 $\hat{U}_{k_1 k_2}(\theta_{k_1 k_2})$ can be given by (from Eq. 28):

$$\hat{U}_{k_1 k_2}^l(\theta_{k_1 k_2}^l) : G^l(k_1, k_2) \text{CCNOT} G^l(k_1, k_2) \quad (29)$$

where

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

3. 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 $|k\rangle$. The resulting gate composition can then be expressed as:

$$\hat{U}_{k_1 k_2 \dots k_{\log K}}^l(\theta_{k_1 k_2 \dots k_{\log K}}^l) : G^l(k_1, k_2 \dots k_{\log K}) \text{CCNOT} G^l(k_1, k_2 \dots k_{\log K})$$

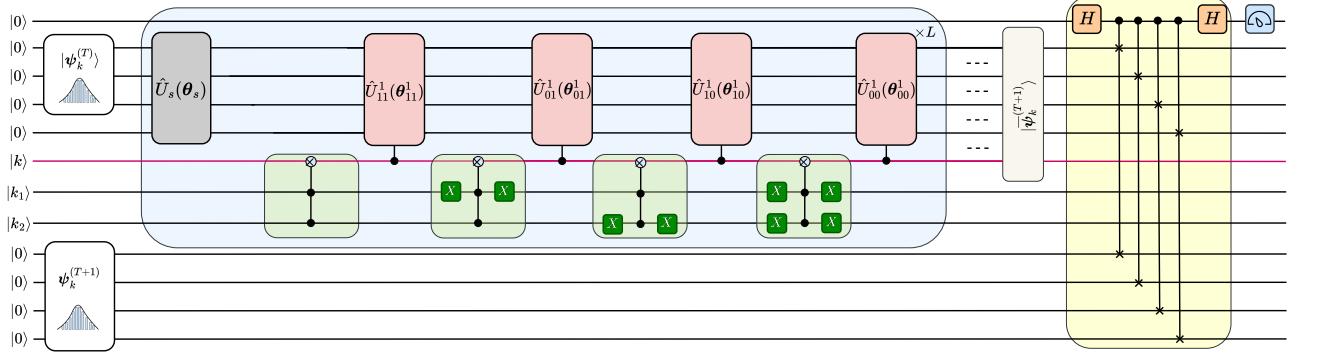


Figure 8. **Quantum Multi-Task Learning Architecture for $T = 3$ and $K = 4$.** The circuits inside the Grey ($\hat{U}_s(\theta_s)$) and Pink boxes ($\hat{U}_{11}^1(\theta_{11}^1)$, $\hat{U}_{01}^1(\theta_{01}^1)$, $\hat{U}_{10}^1(\theta_{10}^1)$, $\hat{U}_{00}^1(\theta_{00}^1)$) are applied sequentially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

where

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

where $k_1, \dots, k_{logK} \in \{0, 1\}$. 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.

B. 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 $\hat{U}(\theta)$ 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 $\hat{P}^{lij} \in \{\hat{X}_i, \hat{Y}_i, \hat{Z}_i\}_{i=1}^{T+\tau}$ is a single qubit Pauli generator. The parametric quantum circuit consists of m parameters, each corresponding to either a rotation gate $G_s^{lij}(\theta_s^{lij}) \otimes \mathbb{1}_K$ (from Eq. (21)) or a controlled rotation gate $(G_k^{lij}(\theta_k^{lij}) \otimes |k\rangle\langle k| + \mathbb{1}_{2^{T+\tau}} \otimes |k^\perp\rangle\langle k^\perp|)$ (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 $\hat{U}_{ck}^l(\theta_{ck}^l)$ and $\hat{U}_k^l(\theta_k^l)$, resulting in

$$\hat{U}(\theta) = \prod_{l=1}^L \hat{U}_k^l(\theta_k^l) \hat{U}_s^l(\theta_s^l), \quad (30)$$

where $\hat{U}(\theta)$ is now dimensionally reduced but retains the same structure as Eq. (24) from the perspective of gradient computation. From Eq.(30), $\hat{U}(\theta)$ 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.

C. 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 $\mathbf{x}_k^{(T)}$ to a state $|\mathbf{x}_k^{(T)}\rangle$ and find (from Eq. (5))

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

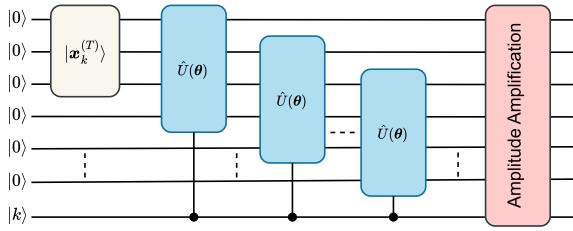


Figure 9. Sequential Prediction for an Asset. A block diagram of our proposed sequential prediction approach, where a fully trained PQC ($\hat{U}(\theta)$) over K stocks using quantum multi-task learning is used to predict $\tau = t$ ($\in \{1, \dots, R\}$) consecutive future values by loading the continuation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

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

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

This gives us a superposition state over possible paths with logarithmic qubit depth, enabling us to efficiently encode highly nontrivial distributions over futures and utilize quantum algorithms for statistical tasks to achieve quantum advantage at inference time, such as quadratic sampling advantage through amplitude estimation on risk analysis [29]. This is possible because QBCU 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 $\sum_{k=1}^K \sqrt{V_k/V} |k\rangle$ over the $\log(K)$ label qubits and use label-control gates to initialize the corresponding context $|x_k^{(T)}\rangle$ dependent on each label. Applying our QNN R times prepares:

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

Such a model can be utilized for quantum advantage

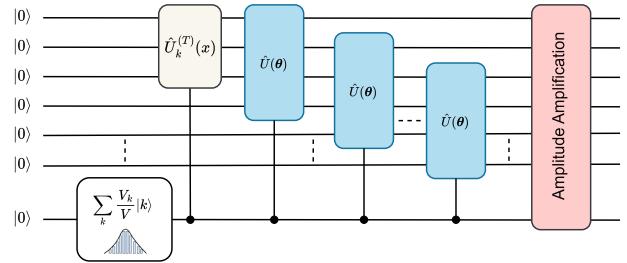


Figure 10. Sequential Prediction for a Portfolio of Assets. A block diagram of our proposed sequential prediction approach, where a fully trained PQC ($\hat{U}(\theta)$) over K stocks using quantum multi-task learning is used to predict $\tau = t$ ($\in \{1, \dots, R\}$) consecutive future values for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

on the downstream tasks, utilizing known quantum algorithms such as Ref. [29].

VI. NUMERICAL SIMULATIONS

A. Loading Contextual Distribution

To encode a contextual distribution, $\mathcal{P}(x^{(T)})$ 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 is trained over 3000 epochs with a learning rate of 0.1, resulting 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.

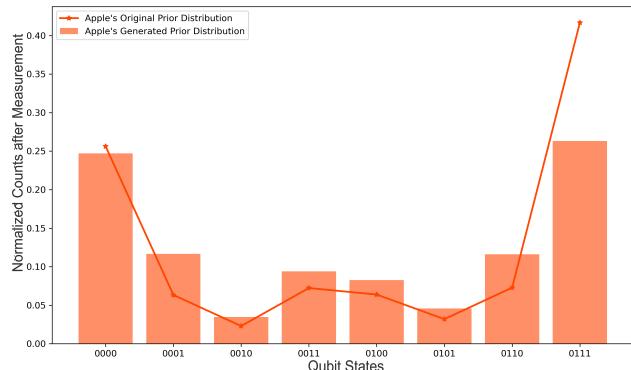


Figure 11. **Contextual Probability Distribution.** A diagram showing the loaded contextual probability distribution ($\mathcal{P}(\mathbf{x}^{(T)})$) of Apple’s stock data for a context size of $T = 3$.

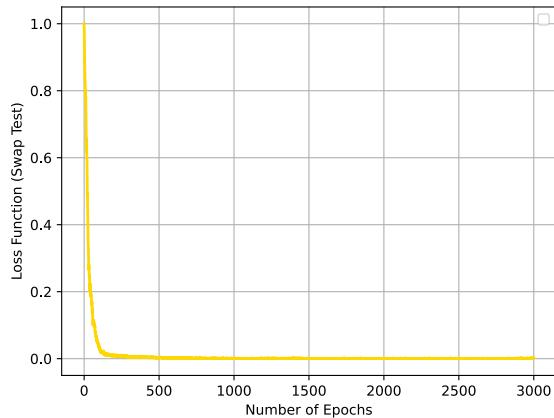


Figure 12. **Training Loss for loading context distribution.** A diagram plotting the training loss for loading a contextual distribution ($\mathcal{P}(\mathbf{x}^{(T)})$) of Apple’s stock data for a context size of $T = 3$.

B. Quantum Single-Task Learning

With the contextual distribution now loaded, we proceed to predict the conditional probability $\mathcal{P}(x^{T+1}|\mathbf{x}^{(T)}, \boldsymbol{\theta})$ from the contextual distribution $\mathcal{P}(\mathbf{x}^{(T)})$ 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 $\mathcal{P}(x^{T+1}|\mathbf{x}^{(T)}, \boldsymbol{\theta})$, loaded similarly to the contextual. The circuit incorporates four layers ($L = 4$) of parameterized gates, which upon training, generate the

wavefunction $|\mathbf{y}^{(T+1)}\rangle$, representing the predicted return for the context input $|\mathbf{x}^{(T)}\rangle$. 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.

C. 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, $|k\rangle$, to distinguish between the stocks, where $|k\rangle = |0\rangle$ represents Apple, and $|k\rangle = |1\rangle$ 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 Apple ($|k\rangle = |0\rangle$) for 250 epochs, followed by Google ($|k\rangle = |1\rangle$) for another 250 epochs, both at a learning rate $\beta = 0.1$. The predicted conditional probability functions are shown in Fig. 19, 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 at least an order of magnitude 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 prediction accuracy.

1. Resource Utilization

As illustrated in Fig. 19, the QMTL model shows markedly better performance compared to the QSTL

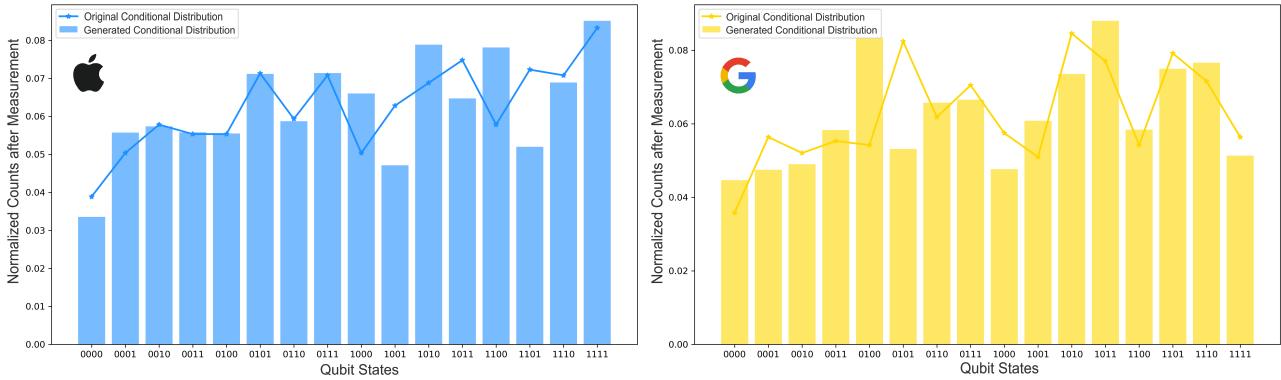


Figure 13. Predictions of Quantum Single-Task Learning for $T = 3$. Generated conditional probability distributions of both Apple and Google datasets are shown comparing them to the original distributions. The model was trained for 3000 epochs with a learning rate of 0.1.

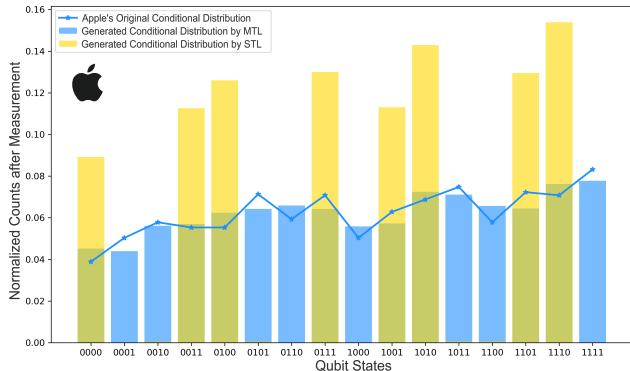


Figure 14. Predictions of MTL vs. STL. A diagram displaying the difference between original and generated conditional probability distributions for QMTL and QSTL for an equal number of trainable parameters (16).

model. To further verify this, we conducted an additional experiment using the same number of resources for QSTL—specifically, 16 trainable parameters—and plotted the resulting conditional probability distribution alongside QMTL in Fig. 14. Despite keeping the training parameters, learning rate, and datasets identical, QMTL successfully captured the target distribution, while QSTL struggled to do so. This demonstrates that the multi-task learning approach not only outperforms single-task learning but also utilizes resources more effectively, underscoring its advantages for complex financial data modeling.

2. Convergence

The training losses for both models, QMTL and QSTL, are plotted in Fig. 15. 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.

3. Time-series Prediction

With the trained model, we predict future stock prices by measuring the ancillary qubit from the swap test in the Z -basis. Depending on the context, this measurement yields probabilistic predictions that follow the conditional probability distribution shown in Fig. 19. We tested the model on an unseen testing dataset and the accuracy of predictions is tabulated in Table I. Notably, the multi-task learning approach achieves higher prediction accuracy for both stocks, outperforming the single-task learning approach even with half the number of trainable parameters.

This efficiency highlights the model’s capability to leverage shared patterns between stocks, underscoring the advantages of multi-task learning for robust stock price prediction. The authors in [44] achieved a maximum prediction accuracy of 58.94% using support vector machines and quantum annealing, whereas our model demonstrates significantly improved performance. Similarly, the best reported accuracy in [28] is under 60% when using quantum Elman neural networks. Furthermore, numerous studies on the encoding

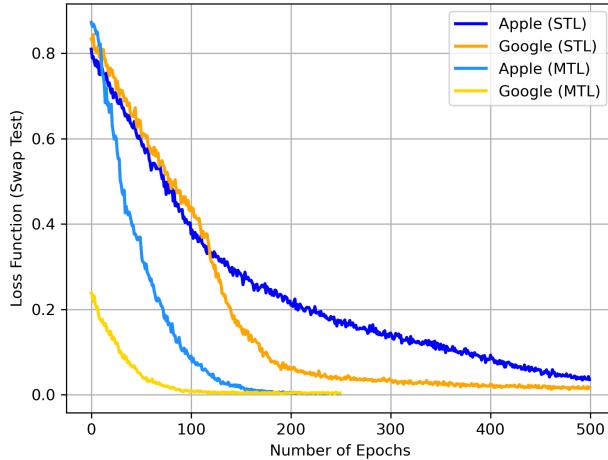


Figure 15. **Loss over training epochs for QSTL and QMTL.** A diagram plotting loss function versus epochs for QMTL and QSTL.

Table I. Performance of QSTL vs. QMTL for various stocks.

Asset	Model	Parameters	KL Divergence	Accuracy
Apple	STL	32	7.4668e-05	62.21%
Google	STL	32	8.2665e-05	56.46%
Apple	MTL	16	3.5392e-07	71.83%
Google	MTL	16	1.5697e-06	68.30%

of financial data into quantum circuits, such as [36], [37], and [26], do not perform time series prediction experiments or report prediction accuracy metrics. Our model thus not only surpasses prior accuracy benchmarks but also addresses the gap in the existing literature by validating the quantum neural network’s performance on actual time-series predictions.

4. 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 ($|k_1\rangle$ and $|k_2\rangle$) are externally controlled. The circuit consists of one repetition of the trainable shared circuit $\hat{U}_s(\theta_s)$, followed by the task-specific circuits $\hat{U}_{11}^1(\theta_{11}^1)$, $\hat{U}_{01}^1(\theta_{01}^1)$, $\hat{U}_{10}^1(\theta_{10}^1)$, and $\hat{U}_{00}^1(\theta_{00}^1)$. 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. 18, 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

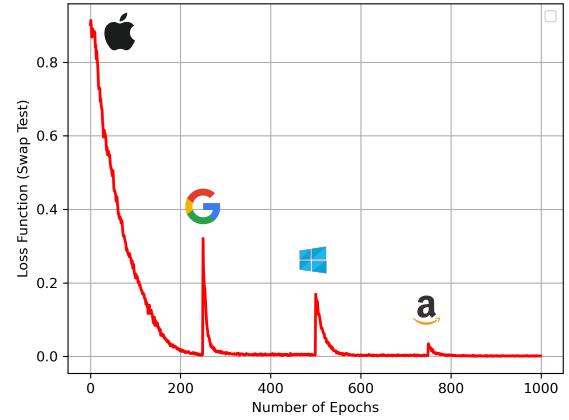


Figure 16. **QMTL Loss over training epochs of 4 assets.** A diagram plotting loss function vs. epochs for QMTL ($K = 4$) with a learning rate of 0.1.

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. 16 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.

D. 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

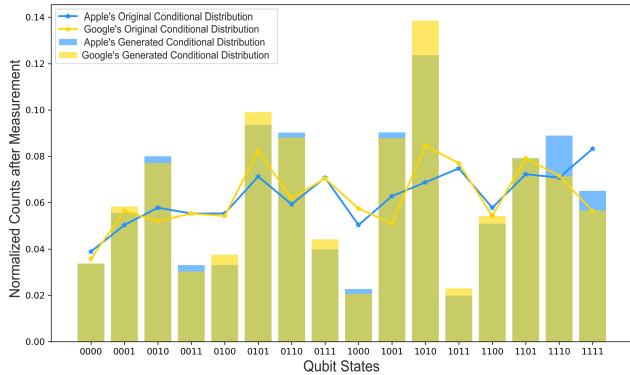


Figure 17. SWAP Test vs. MSE Loss. Comparison of representing the underlying conditional probability distributions of Apple and Google stocks after training with SWAP Test versus MSE Loss.

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, 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.

VII. CONCLUSION

In this paper, we presented a quantum multi-task learning (QMTL) architecture tailored for predicting stock prices 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). Our experiments demonstrated that the QMTL model effectively captures correlations between different assets, enabling improved prediction accuracy while requiring fewer trainable parameters.

We also develop the quantum batch gradient update (QBGU) and compared this method with the mean squared error (MSE) loss, concluding that this method for training over quantum distributions significantly improves convergence and solution quality. 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.

-
- [1] Shor, P. (1999). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, 41(2), 303-332.
 - [2] Harrow, A., Hassidim, A., & Lloyd, S. (2009). Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.*, 103, 150502.
 - [3] Aharonov, D., Jones, V., & Landau, Z. (2006). A polynomial quantum algorithm for approximating the Jones polynomial. In Proceedings of the thirty-eighth annual ACM symposium on Theory of computing (pp. 427–436).
 - [4] Rebentrost, P., Mohseni, M., & Lloyd, S. (2014). Quantum support vector machine for big data classification. *Physical review letters*, 113(13), 130503.
 - [5] Rietsche, R., Dremel, C., Bosch, S., Steinacker, L., Meckel, M., & Leimeister, J. M. (2022). Quantum computing. *Electronic Markets*, 32(4), 2525–2536.
 - [6] Arute, F., Arya, K., Babbush, R., et al. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779), 505–510.
 - [7] Preskill John 1998Quantum computing: pro and con-
Proc. R. Soc. Lond. A.454469–486
 - [8] Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. *Quantum*, 2, 79. doi:10.22331/q-2018-08-06-79
 - [9] Egger, D. J., Gambella, C., Marecek, J., McFaddin, S., Mevissen, M., Raymond, R., Yndurain, E. (2020). Quantum Computing for Finance: State-of-the-Art and Future Prospects. *IEEE Transactions on Quantum Engineering*, 1, 1–24. doi:10.1109/TQE.2020.3030314
 - [10] Egger, D. J., Gutierrez, R. G., Mestre, J., & Woerner, S. (2021). Credit Risk Analysis Using Quantum Computers. *IEEE Transactions on Computers*, 70(12), 2136–2145. doi:10.1109/TC.2020.3038063
 - [11] Orús, R., Mugel, S., & Lizaso, E. (2019). Forecasting financial crashes with quantum computing. *Phys. Rev. A*, 99, 060301. doi:10.1103/PhysRevA.99.060301
 - [12] Rebentrost, P., Lloyd, S. Quantum Computational Finance: Quantum Algorithm for Portfolio Optimization. Künstl Intell (2024). <https://doi.org/10.1007/s13218-024-00870-9>
 - [13] Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature*, 549(7671), 195–202. doi:10.1038/nature23474
 - [14] Ciliberto, C., Herbster, M., Ialongo, A. D., Pontil, M., Rocchetto, A., Severini, S., & Wossnig, L. (2018). Quantum machine learning: a classical per-

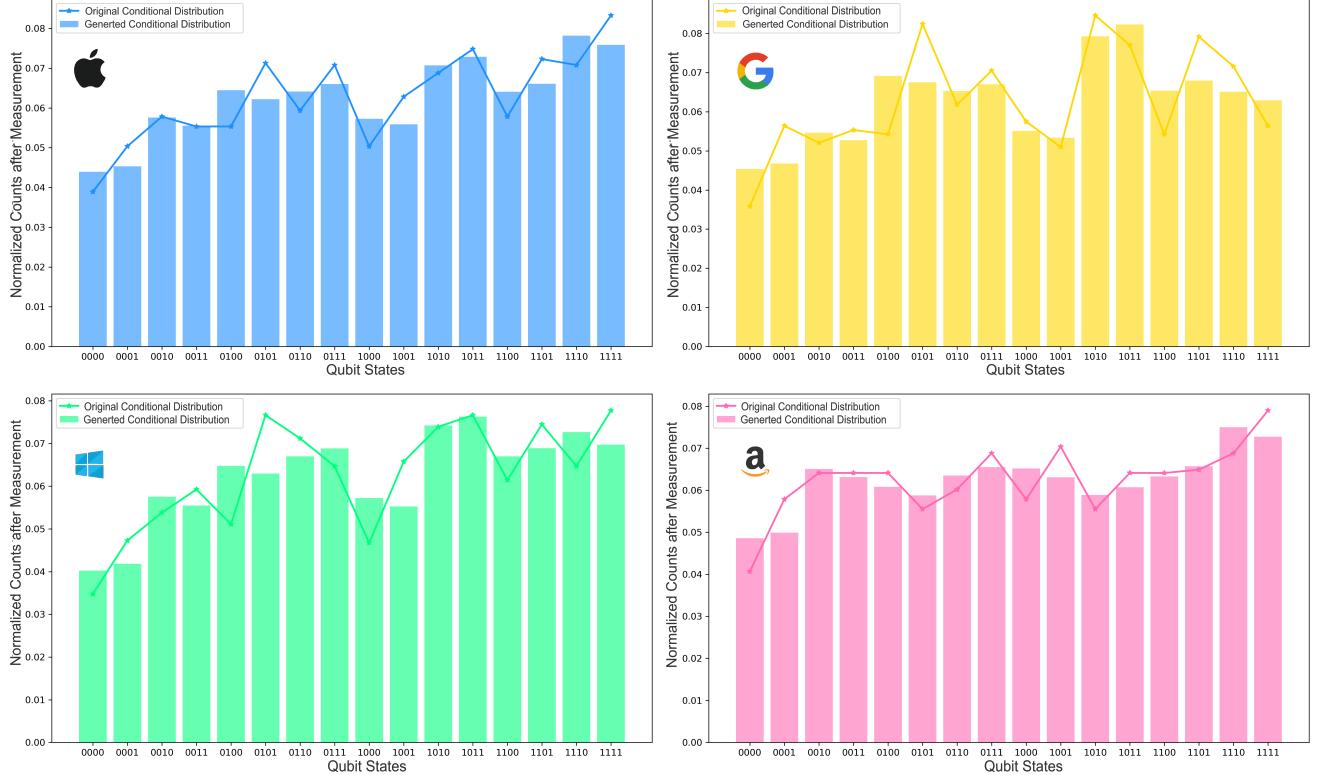


Figure 18. Predictions of Quantum Multi-Task Learning for $T = 3$ and $K = 4$. Generated conditional probability distributions of Apple, Google, Microsoft, and Amazon datasets are shown comparing them to the original distributions. The model was trained for 3000 epochs with a learning rate of 0.1.

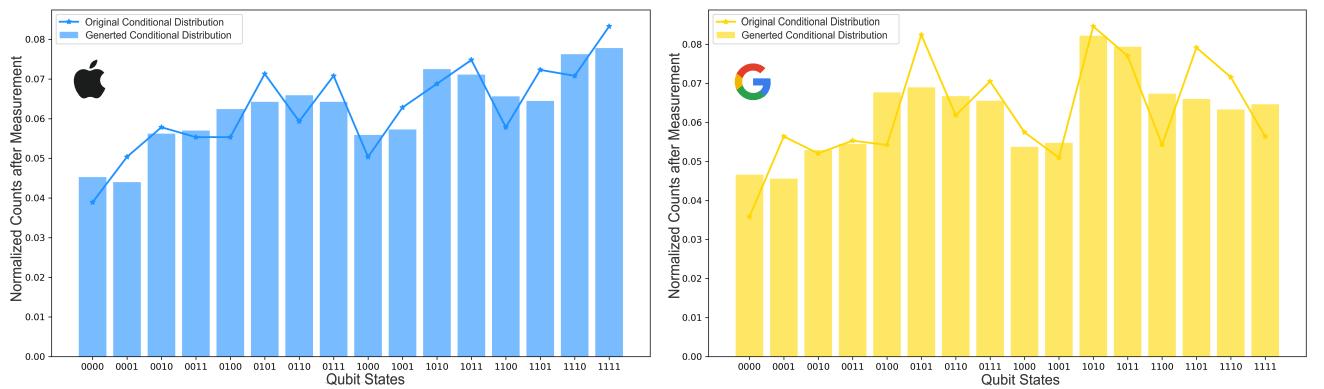


Figure 19. Predictions of Quantum Multi-Task Learning for $T = 3$ and $K = 2$. Generated conditional probability distributions of both Apple and Google datasets are shown comparing them to the original distributions. The model was trained for 3000 epochs with a learning rate of 0.1.

- spective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209). <https://doi.org/10.1098/rspa.2017.0551>
- [15] Abbas, A., Sutter, D., Zoufal, C. et al. The power of quantum neural networks. *Nat Comput Sci* 1, 403–409 (2021). <https://doi.org/10.1038/s43588-021-00084-1>
- [16] Sim, S., Johnson, P. D., & Aspuru-Guzik, A. (2019). Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms. *Adv. Quantum Technol.*, 2(12), 1900070. doi:10.1002/qute.201900070
- [17] Rebentrost, P., Mohseni, M., & Lloyd, S. (2014). Quantum Support Vector Machine for Big Data Classification. *Phys. Rev. Lett.*, 113, 130503. doi:10.1103/PhysRevLett.113.130503
- [18] Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747), 209–212. doi:10.1038/s41586-019-0980-2
- [19] Amin, M. H., Andriyash, E., Rolfe, J., Kulchytskyy, B., & Melko, R. (2018). Quantum Boltzmann Machine. *Phys. Rev. X*, 8, 021050. doi:10.1103/PhysRevX.8.021050
- [20] Bausch, J. (2020). Recurrent Quantum Neural Networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (Vol. 33, pp. 1368–1379).
- [21] Lloyd, S., & Weedbrook, C. (2018). Quantum Generative Adversarial Learning. *Phys. Rev. Lett.*, 121, 040502. doi:10.1103/PhysRevLett.121.040502
- [22] Dong, D., Chen, C., Li, H., & Tarn, T.-J. (2008). Quantum Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5), 1207–1220. doi:10.1109/TSMCB.2008.925743
- [23] Fujii, K., & Nakajima, K. (2017). Harnessing Disordered-Ensemble Quantum Dynamics for Machine Learning. *Phys. Rev. Appl.*, 8, 024030. doi:10.1103/PhysRevApplied.8.024030
- [24] Stamatopoulos, N., Egger, D. J., Sun, Y., Zoufal, C., Iten, R., Shen, N., & Woerner, S. (2020). Option Pricing using Quantum Computers. *Quantum*, 4, 291. doi:10.22331/q-2020-07-06-291
- [25] Idrees, S., Alam, M., & Agarwal, P. (2019). A Prediction Approach for Stock Market Volatility Based on Time Series Data. *IEEE Access*, 7, 17287–17298.
- [26] Emmanoulopoulos, D., & Dimoska, S. (2022). Quantum Machine Learning in Finance: Time Series Forecasting. arXiv [Quant-Ph]. Retrieved from <http://arxiv.org/abs/2202.00599>
- [27] Gushu Li, Anbang Wu, Yunong Shi, Ali Javadi-Abhari, Yufei Ding, & Yuan Xie. (2021). Paulihedral: A Generalized Block-Wise Compiler Optimization Framework For Quantum Simulation Kernels.
- [28] Liu, G., & Ma, W. (2022). A quantum artificial neural network for stock closing price prediction. *Information Sciences*, 598, 75–85. doi:10.1016/j.ins.2022.03.064
- [29] Woerner, S., & Egger, D. (2019). Quantum risk analysis. *npj Quantum Information*, 5(1), 15.
- [30] Kandala, A., Mezzacapo, A., Temme, K. et al. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* 549, 242–246 (2017). <https://doi.org/10.1038/nature23879>
- [31] Leone, L., Oliviero, S., Cincio, L., & Cerezo, M. (2024). On the practical usefulness of the Hardware Efficient Ansatz. *Quantum*, 8, 1395.
- [32] Lov Grover, & Terry Rudolph. (2002). Creating superpositions that correspond to efficiently integrable probability distributions.
- [33] Paquet, E., & Soleymani, F. (2022). QuantumLeap: Hybrid quantum neural network for financial predictions. *Expert Systems with Applications*, 195, 116583. doi:10.1016/j.eswa.2022.116583
- [34] 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* 2024, 13, 2158.
- [35] Pistoia, M., Ahmad, S. F., Ajagekar, A., Buts, A., Chakrabarti, S., Herman, D., ... Yalovetzky, R. (2021). Quantum Machine Learning for Finance ICCAD Special Session Paper. 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)
- [36] Zoufal, C., Lucchi, A. & Woerner, S. “Quantum Generative Adversarial Networks for learning and loading random distributions”. *npj Quantum Inf* 5, 103 (2019).
- [37] Zhu, E. Y., Johri, S., Bacon, D., Esencan, M., Kim, J., Muir, M., Wright, K. (2022). Generative quantum learning of joint probability distribution functions. *Phys. Rev. Res.*, 4, 043092. doi:10.1103/PhysRevResearch.4.043092
- [38] Xia, W., Zou, J., Qiu, X., Chen, F., Zhu, B., Li, C., ... Li, X. (2023). Configured quantum reservoir computing for multi-task machine learning. *Science Bulletin*, 68(20), 2321–2329. doi:10.1016/j.scib.2023.08.040
- [39] Caruana, R. (1997). Multitask Learning. *Machine Learning*, 28(1), 41–75. doi:10.1023/A:1007379606734
- [40] Buhrman, H., Cleve, R., Watrous, J., de Wolf, R. (2001). Quantum fingerprinting. *Physical Review Letters*, 87(16), 167902. doi:10.1103/PhysRevLett.87.167902
- [41] Takaki, Y., Mitarai, K., Negoro, M., Fujii, K., & Kitagawa, M. (2021). Learning temporal data with a variational quantum recurrent neural network. *Phys. Rev. A*, 103, 052414. doi:10.1103/PhysRevA.103.052414
- [42] Mitarai, K., Negoro, M., Kitagawa, M., & Fujii, K. (2018). Quantum circuit learning. *Phys. Rev. A*, 98, 032309. doi:10.1103/PhysRevA.98.032309
- [43] Spall, J. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3), 332–341.
- [44] N. Srivastava, G. Belekar, N. Shahakar and 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, 2023, pp. 1-7, doi: 10.1109/RASSE60029.2023.10363533.