An Artificial Neural Networks-based Quantum Finance System

Taige Hu 2130026048 Zhengqi Lin 2230026097 Yanhua Wan 2230026147 Zhuowei Ruan 2230026133 Yiyu Mao 2130026106 Jinyu Gan 2130024051

***Abstract*—In this paper, we propose a novel approach to financial market prediction by integrating principles of quantum mechanics with advanced deep learning models. Traditional forecasting methods, such as those based on Geometric Brownian Motion (GBM), often fail to accurately capture the complex, non-linear, and highly uncertain nature of financial markets. To address these limitations, we introduce quantum price levels, inspired by quantum theory's discrete energy levels, wave functions, and superposition and calculated by the modeling of market price fluctuations using discrete energy structures in quantum theory. . These levels offer a probabilistic and multidimensional representation of market behavior, providing a richer feature space that reflects the inherently uncertain dynamics of financial price movements.**

We employ these quantum-derived price levels as input features for three deep learning architectures: Convolutional Neural Networks (CNNs) for capturing local spatial patterns; Long Short-Term Memory (LSTM) networks for learning sequential and temporal dependencies; and Attention mechanisms, which selectively focus on critical patterns across time steps to improve model interpretability and accuracy. Each model leverages the unique structure of quantum-based features to enhance its predictive capabilities.

The results of the back test suggest that quantum-inspired representations, when combined with deep learning, can significantly improve forecasting performance. This study bridges quantum theory and financial artificial intelligence, offering innovative tools for more robust and adaptive market prediction.

***Index Terms*—Quantum financial, Schrödinger equation, Quantum price level**

I. Introduction

The financial market is known for its complex and dynamic nature, making accurate predictions a huge challenge. Traditional forecasting techniques, such as those based on Geometric Brownian Motion, have achieved some success in modeling market behavior. However, these models often struggle to fully capture the complexity and uncertainty of the market. For example, GBM considered changes in asset prices are random and there will be no sudden changes or jumps. In recent years, short frequent trading is becoming a mainstreaming trading method in financial market, making the market and price flow more chaotic and unpredictable.

Under such a situation, more and more scientists and researchers start turning their sight on the quantum finance, a emerging research orientation. By combine two subject based on the similar property. We may can have a better forecast on the coming market trend.

And the core of this paper is the application of the quantum price level (QPL), which Lee [1] put forward as a indicator for the market trend based on quantum mechanics' theory.

In this paper, we explores the application of quantum mechanics theory to the calculation of quantum price levels, which are then used as input features for various deep learning models. Quantum mechanics, with its foundations in wave-particle duality and superposition, offers a new perspective on price prediction by introducing quantum states that represent potential future prices. These quantum price levels are calculated by the price fluctuation modeling of financial market based on the simulation of the discrete energy level structure in quantum mechanics, capturing a broader range of market possibilities.

To have a further forecast, we apply three different deep learning models: Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) networks, and Attention mechanisms. CNNs capture spatial patterns in data. They find hierarchies in grid-like inputs. LSTMs handle time-based patterns. They work well for sequences like text data. Attention models pick the most important information over time. They focus on key parts of the input.

Each model has a different purpose. CNNs work with space. LSTMs work with time. Attention improves focus on relevant details.

II. Literature review

Financial market prediction has been revolutionized by developing more sophisticated methods using quantum mechanics and Deep learning. Quantum price levels modeled from quantum mechanics is a novel approach to represent market price based on simulating discrete energy level structure of quantum mechanics for price fluctuation modeling in financial markets offering a panoramic view of price movements. CNNs have proven successful for financial time series forecasting as they provide the possibility to learn hierarchies of spatial features in the data set, i.e., a visual representation of the data that is meaningful for the problem at hand. Similarly, LSTMs are efficient in learning temporal dependency and long-term trends in sequential financial data. Furthermore, Attention model is brought more attention into recent years because it can concentrate on the vital region of input sequence,which enhance the accurate and efficiency of predicting. Using these kind of deep learning models in combination with our quantum price levels, can explore the deep connected characteristic from the internal feature of stock.

A. Quantum price level

The integration of quantum mechanics into financial modeling, referred to as quantum finance, has emerged as a robust

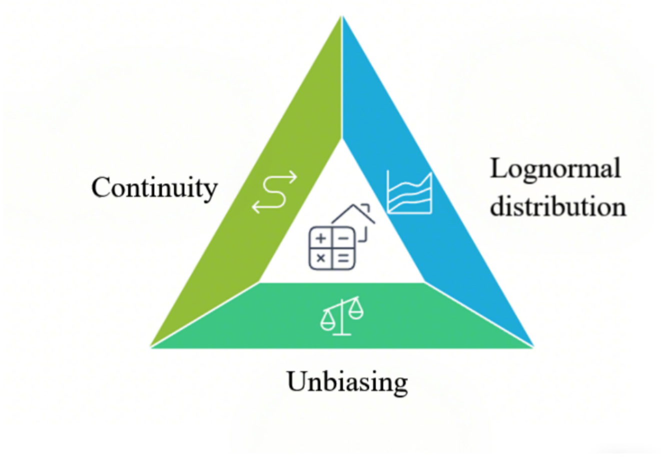


Fig. 1. GBM

framework for addressing the limitations of classical financial models. Specifically, it excels in capturing complex market behaviors such as price jumps, volatility clustering, and non-Gaussian distributions. Traditional stochastic models, exemplified by those based on Geometric Brownian Motion (GBM), typically assume continuous price changes. However, financial markets exhibit discrete, probabilistic behaviors that are more effectively modeled using quantum mechanical principles. Quantum finance is grounded in the analogy that financial markets can be conceptualized similarly to quantum systems, where asset prices exist in discrete states akin to the energy levels of quantum particles. Focardi et al. [2] substantiate this analogy in detail by arguing that prices can be meaningfully construed as quantum observables in order to model the market as a process of state-transition. In fact, exogenous events can drive instantaneous "jumps" between levels, in an analogous manner to quantum transitions, thus endowing a richer account of markets' dynamics, especially when they are under stress or when there is uncertainty. The quantum paradigm allows modelling price fluctuation and market uncertainties in more profound manners than classical models do.

Quantum finance is a recent subject which considers the utilization of quantum mechanics to study the modelling of financial markets, and uses as an analogy the relationship between quantum energy levels and states in a quantum system. Within this framework, Lee [1] has proposed the formalism of Quantum Price Levels (QPL) which suggests that the prices of the markets will appear in intrinsic, discrete levels, analogous to energy levels appearing in quantum systems. The above mentioned levels, computed as statistical distribution of historical price returns, behave similarly to eigenstates in quantum mechanics and evolve on the external fields (say, economic news, shock) through a stepwise, rather than continuous, jumping response. The foundational theory is formalized through a Quantum Finance Schrödinger Equation (QFSE), which is solved numerically using methods such as the Finite Difference Method (FDM) and Cardano's method for depressed cubic equations. These solutions yield key parameters, including Quantum Financial Energy Levels (QFEL), Quantum Price Returns (QPR), and Normalized Quantum Price Returns (NQPR). In this study, we implement Lee's framework in Python and extend it as a feature extraction module, generating structured QPL features (from qpl-5 to qpl5) for integration into deep learning models. This hybrid approach bridges physically interpretable quantum structures with the predictive capabilities of neural networks, providing a novel perspective on market dynamics and feature representation.

B. CNN

In recent years, CNNs (Convolutional Neural Networks) have gained widespread attention and use in the prediction of financial time series. This is mainly attributed to its ability to automatically extract hierarchical data from raw data. Traditional models usually assume linear relationships and stability in financial data, but these relationships do not hold in most cases in real forecasting, and CNNs are able to capture the complex nonlinear dependencies in financial time series. Several studies have shown that CNNs have great potential for financial forecasting, especially in futures forecasting and arbitrage strategies.

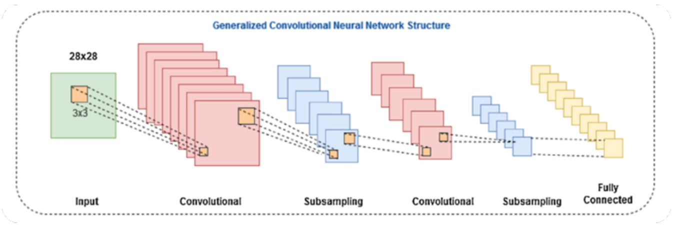


Fig. 2. CNN Structure

1) *Advantages of CNN in financial forecasting:*

- **Automatic feature extraction:** CNNs use convolutional layers to extract local patterns in the time series, which in turn eliminates the need to manually extract relevant features. It is very efficient when dealing with high-frequency data or large-scale historical data, for example, Sezer and Ozbayoglu (2018) showed that CNNs can effectively predict the price differences of stock index futures, thus enhancing the effectiveness of arbitrage strategies [3].
- **Modeling nonlinear relationships:** Real financial markets have very complex nonlinear dynamics. CNNs can use their hierarchical learning structure to effectively model these nonlinear dynamic dependencies, capturing complex patterns that traditional linear models cannot directly capture. [3].
- **Parallel processing and efficiency:** CNNs allow data to be processed in parallel so that large-scale financial datasets can be processed efficiently. Parallel processing is especially important when trading at high frequencies or when dealing with large historical trading datasets. [4].

2) *Challenges and mixed methods:* Despite its many advantages, CNNs still face some challenges in financial time series forecasting:

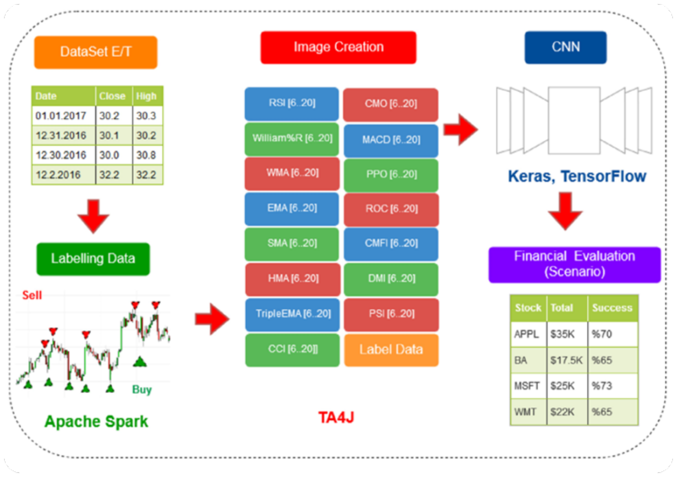


Fig. 3. Proposed method for CNN-TA

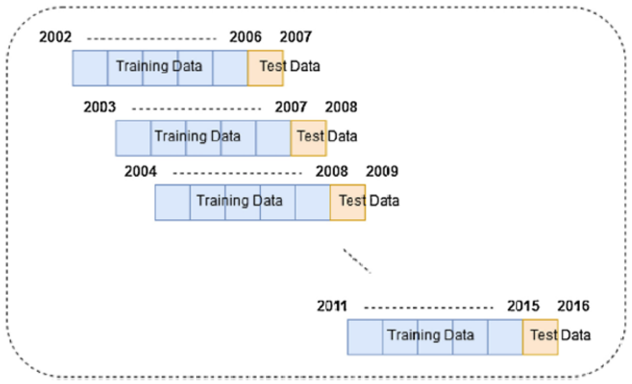


Fig. 4. Training and testing approach

- **Capturing long-term dependence:** Although CNNs have something to offer in capturing the dynamic dependencies of short-term nonlinearities, their local sensibilities may not be able to effectively model long-term relationships in financial time series. To address this issue, Yue and Li (2021) explored the use of hybrid models using CNN models together with other architectural models (LSTM).[3].
- **Sensitivity to noise:** Financial datasets contain noise and fluctuations from many unpredictable factors (e.g., market sentiment, breaking news, new policies, etc.). CNNs are more sensitive to noise and can easily lead to overfitting. This problem can be mitigated by applying regularization techniques (e.g., Dropout) to increase the generalization ability of CNN models. Sezer and Ozbayoglu show that using moving average preprocessing is effective in reducing noise and improving prediction accuracy in financial applications. [4].
- **Explanatory questions:** Deep learning models (including CNNs) are often criticized as “black boxes”, making their decision-making process difficult to understand. In finance, understanding the prediction results of models is crucial for decision making, so improving the interpretability of CNNs is still an active research direction.

3) Future direction: Future research may focus on developing advanced CNN architectures that can better capture long-term dependencies and handle noisy financial data. In addition, improving the interpretability of CNN models is crucial for their application in the financial industry. Hybrid models, combining CNNs with other deep learning architectures such as LSTM or Transformer networks, are expected to improve prediction accuracy and robustness. In addition, as shown in the study by Yue and Li, incorporating CNNs into arbitrage strategies may provide a significant boost to the predictive accuracy and profitability of financial models [3].

C. Long Short-Term Memory (LSTM)

In recent years, deep learning has rapidly transformed how we process sequential data in areas ranging from natural language processing and speech recognition to financial forecasting and bioinformatics. However, standard recurrent neural networks often hit a snag when dealing with lengthy sequences—they can struggle with gradients that either fade away or explode, making it hard to capture patterns that span long intervals [6]. To overcome this limitation, Hochreiter and Schmidhuber introduced the Long Short-Term Memory network in 1997. By incorporating three distinct gates—input, forget, and output—this architecture adeptly manages the flow of information, greatly enhancing its capacity to track long-term dependencies.

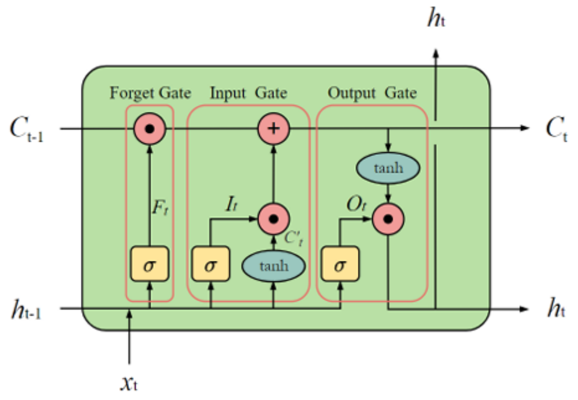


Fig. 5. A Basic LSTM with a forget gate

1) Theoretical Foundations and Improvements: LSTM’s breakthrough innovation is its clever gating system. One gate decides how much new information to let in, another weeds out unnecessary or outdated details, and a third produces the output at each step [6]. This setup keeps gradient flow steady over long sequences, effectively sidestepping the challenges that standard RNNs face. Over time, to tackle even more demanding applications, researchers have introduced a variety of improved versions. For instance, peephole LSTM utilizes the cell state directly in gating decisions for more refined state control [5]. Bidirectional LSTM processes sequences in both forward and backward directions, further enhancing the capture of contextual information; it is particularly effective in tasks such as machine translation and sequence labeling [8]. In addition, stacked LSTM employs a multi-layer network structure to progressively extract higher-level features, while the gated recurrent unit (GRU), with its simpler structure, serves as a lightweight alternative in certain scenarios [8]. More recently, scholars have experimented with integrating attention mechanisms with LSTM to build hybrid models that can dynamically focus on key portions of the input during generation [5] [7].

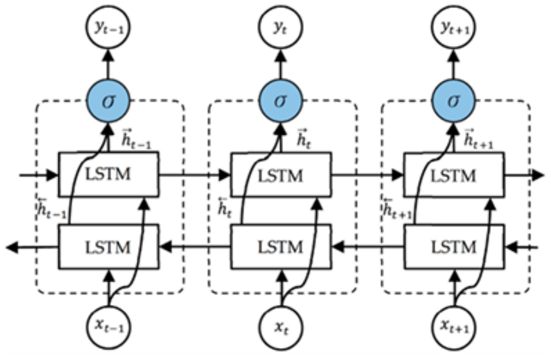


Fig. 6. Architecture of BiLSTM network

2) Applications and Future Prospects: Long Short-Term Memory networks have been widely applied. In natural language processing, LSTM-based models effectively capture the continuous contextual information of texts and are commonly used in machine translation, sentiment analysis, and dialogue systems [9]. In speech recognition and financial time-series forecasting, stacked LSTM models extract deep-level features from complex data to improve prediction accuracy [7]. However, LSTM also has some limitations: its complex gating structure results in a large number of parameters and longer training times, and in handling extremely long sequences, its efficiency and performance still lag behind self-attention-based models such as Transformers [9]. Looking ahead, researchers

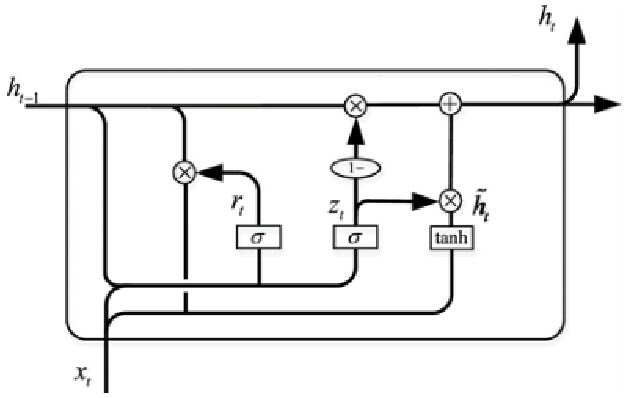


Fig. 7. Architecture of GRU network

are committed to enhancing LSTM's training efficiency and lightweight design. On one hand, integrating LSTM with attention mechanisms and Transformer architectures can enable parallel computation to accelerate training while retaining robust long-term dependency modeling [5] [7]. On the other hand, exploring novel gating techniques and cross-model integration (e.g., fusion with graph neural networks) is expected to further broaden LSTM's application in various complex scenarios [8]. Overall, despite challenges in computational complexity and model tuning, the unique gating advantages of LSTM and ongoing improvements promise flexible and efficient solutions for a wide range of practical tasks [5] [6] [8] [9].

D. Attention

In recent years, Attention mechanisms have emerged as a critical innovation in deep learning, particularly for sequence-based tasks such as natural language processing (NLP), speech recognition, and financial forecasting.The primary strength of attention models lies in their ability to dynamically focus on the most relevant parts of the input sequence, effectively addressing the limitations of traditional models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks.The introduction of Attention mechanisms has significantly improved the performance of models by allowing them to prioritize important features while disregarding irrelevant information.

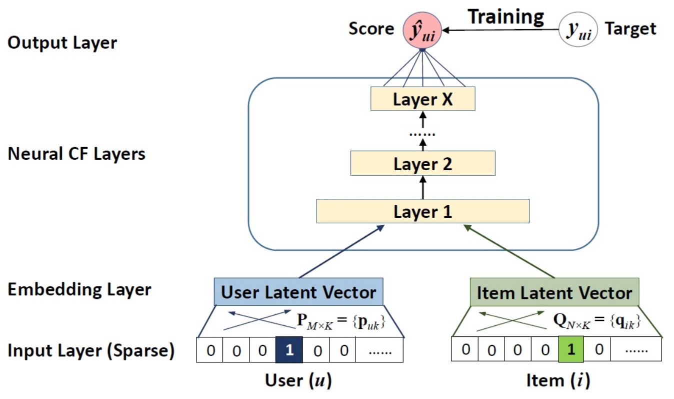


Fig. 8. Neural Cooperative Filtering (NCF) model architecture combined with attention mechanism

1) Theoretical Foundations and Mechanism: The Attention mechanism was first introduced by Bahdanau [10] in the context of machine translation.In traditional sequence-to-sequence models, such as RNNs and LSTMs, information from earlier parts of the sequence is compressed into a fixed-length vector, this may lead to loss of important details in long sequences.Attention models overcome this issue by allowing the model to focus on different parts of the input sequence at each decoding step, and assigning different weights (or attention scores) to different input elements.This dynamic pattern is very useful for preserving and using sequences of distant periods and important information elements, which is especially important for tasks like financial time-series forecasting ,where long-range dependencies and subtle patterns are prevalent.One of the key development to Attention-based models is the Transformer architecture, introduced by Vaswani [12], which is entirely based on self-attention mechanisms.Unlike RNNs or LSTMs, the Transformer leverages self-attention to process all tokens in the sequence simultaneously.So, this can allow for more efficient parallelization and faster training times.This approach has made Transformer become an important model for many NLP tasks.Also, this is now being explored in other domains, such as financial forecasting.

2) Advantages of Attention in Financial Forecasting: Attention mechanisms can focus on relevant features of financial time series, such as key events, price spikes, or market sentiment shifts, which may not be captured by traditional models.By assigning higher weights to critical points in the sequence, Attention can improve the accuracy of predictions, particularly when forecasting volatile or unpredictable market movements.

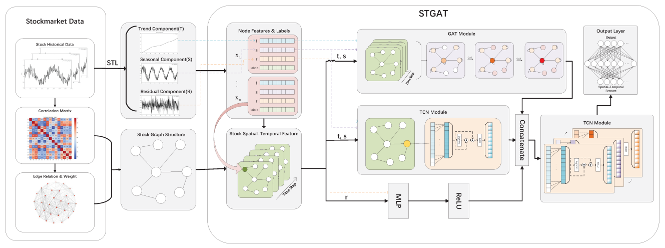


Fig. 9. The architecture of the STGAT (spatial-temporal Graph Attention Neural Network) model, which uses the graph attention mechanism and the time convolution module to effectively capture the Spatial dependence between stocks and the time pattern of price changes in the stock market

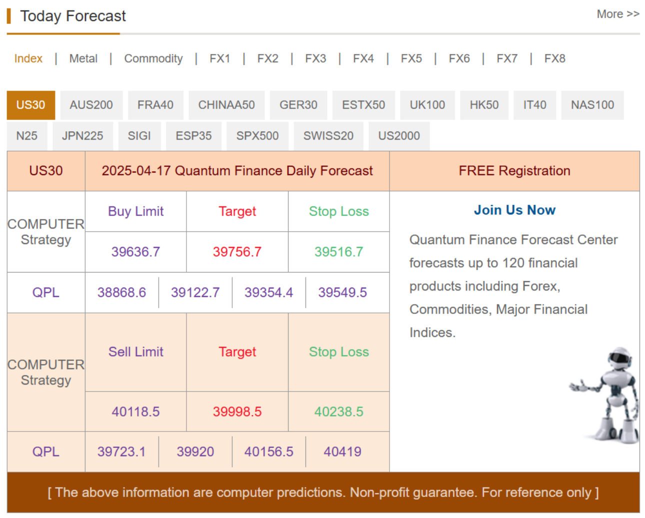
3) Challenges and Limitations: Despite the promising advantages, Attention mechanisms also face challenges, particularly in terms of scalability and computational complexity.As the sequence length increases, the computational cost of calculating attention scores grows quadratically, which can be inefficient for very long sequences.Additionally, while Attention excels at focusing on important elements, it may struggle to interpret or explain the rationale behind its focus, making it difficult to gain insights into model decisions, an issue that is especially important in financial applications where model interpretability is crucial for decision-making.

4) Future Directions: Future research in Attention mechanisms may focus on improving their scalability and reducing computational complexity.One approach is the development of efficient variants such as the Linformer [13], which reduces the time complexity of the self-attention mechanism, making it more practical for long sequences.Another promising direction is the integration of Attention mechanisms with other deep learning models, such as LSTMs and CNNs, to capture both temporal and spatial dependencies in financial time series.Hybrid models that combine Attention with domain-specific features, such as quantum price levels (QPL), may further enhance the accuracy and robustness of financial forecasts by leveraging multiple sources of information.

E. Trading Strategy

1) Short-term reversal strategy: Lehmann [15] proposes a trading strategy which is based on weekly or monthly return reversal. In a nutshell, this strategy goes long the stock with the most negative cumulative return in the past week, and short the stock with the most positive cumulative return in the past week. Jegadesh [16] also identifies this pattern and demonstrate evidence that this pattern exists over the short term, such as one month. For the machine learning aspect, Thomas and Christopher [17] observe that LSTM is able to effectively extract the short-term reversal patterns in stock prices. Thomas and Christopher [17] implement LSTM model to predict whether each stock's next day return exceeds the cross-section median of the market-wide stock returns for that day (Class 1). Class 1 (predicted outperformance of the median) represents the model's judgment that the stock has relative strength, while the opposite is true for Class 0. Top k stocks (high probability of beating the market) and stocks in the bottom k (high probability of losing the market) are predicted using LSTM. Subsequently, they go long the top k stocks and short the bottom 10. The study reveals that high volatility stocks are overweighted through feature attribution analysis, and approximately 52% of the strategy's return can be explained by short-term reversal strategy.

2) Quantum Trading Strategy: According to Lee [1], quantum trading strategy can be simply implemented by setting a buy limit at the forecast low and a sell limit at the forecast high combined with daily calculated QPLs (Quantum Price Levels) as stop loss and target profit to perform daily trading. Furthermore, Lee [1] states that the averaging strategy can also be implemented with QPLs for averaging levels. For example, multiple buy limits can be set at progressively lower QPLs and one-third of the daily investment is allocated to each level. The quantum forecast oscillation trading with averaging strategy is especially effective when a financial market experiences an extremely high likelihood of oscillation. In addition, Lee [1] demonstrates that the lower three QPLs can be used for breakout strategy, and the market should be considered a breakout when the market rises or falls beyond the oscillating range.


Today Forecast More >>

Index | Metal | Commodity | FX1 | FX2 | FX3 | FX4 | FX5 | FX6 | FX7 | FX8

US30 AUS200 FRA40 CHINAA50 GER30 ESTX50 UK100 HK50 IT40 NAS100
N25 JPN225 SIGI ESP35 SPX500 SWISS20 US2000

US30	2025-04-17 Quantum Finance Daily Forecast				FREE Registration
COMPUTER Strategy	Buy Limit	Target	Stop Loss		Join Us Now
	39636.7	39756.7	39516.7		Quantum Finance Forecast Center forecasts up to 120 financial products including Forex, Commodities, Major Financial Indices.
QPL	38868.6	39122.7	39354.4	39549.5	
COMPUTER Strategy	Sell Limit	Target	Stop Loss		
	40118.5	39998.5	40238.5		
QPL	39723.1	39920	40156.5	40419	

[The above information are computer predictions. Non-profit guarantee. For reference only]

Fig. 10. Quantum Financial Forecast Center (QFFC), with the integration of quantum price level calculation algorithm invented by Dr. Raymond LEE [18].

In this review, we have explored five cutting-edge methodologies for financial forecasting and trading strategies. The integration of quantum-inspired models and deep learning architectures—namely Quantum Price Levels (QPL), Convolutional Neural Networks (CNNs), Long Short-Term Memory networks (LSTMs), and Attention mechanisms—along with traditional trading strategies, offers a multi-dimensional framework for understanding and predicting market dynamics.

QPL introduces a novel perspective by applying principles of quantum mechanics to model asset prices as discrete quantum states, thereby capturing complex market phenomena that conventional models struggle with. CNNs provide efficient feature extraction from raw data, highlighting non-linear relationships and offering robust performance in high-frequency markets. LSTMs address the challenges of long-term dependencies within sequential data through their unique gating mechanisms, while Attention mechanisms enhance the ability to focus on the most relevant information across time steps, improving overall predictive accuracy.

Furthermore, trading strategies that leverage short-term reversal patterns and quantum trading principles exemplify the practical applications of these advanced techniques, enabling the development of dynamic and adaptive portfolios. Together, these methodologies illustrate a promising pathway towards hybrid models that combine theoretical depth with data-driven precision, paving the way for more robust, explainable, and efficient financial forecasting systems.

Future research should focus on further integrating these approaches to capitalize on their complementary strengths, addressing computational challenges, and ensuring greater interpretability in dynamic market environments.

TABLE I
STOCK COMPONENTS

Stock Symbol	Company Name
AXP	American Express
AMGN	Amgen
AAPL	Apple
BA	Boeing
CAT	Caterpillar
CSCO	Cisco Systems
CVX	Chevron
GS	Goldman Sachs
HD	Home Depot
HON	Honeywell
IBM	IBM
INTC	Intel
JNJ	Johnson & Johnson
KO	Coca-Cola
JPM	JPMorgan Chase
MCD	McDonald's
MMM	3M
MRK	Merck
MSFT	Microsoft
NKE	Nike
PG	Procter & Gamble
TRV	Travelers Companies
UNH	UnitedHealth Group
CRM	Salesforce
VZ	Verizon
V	Visa
WBA	Walgreens Boots Alliance
WMT	Walmart
DIS	Disney
DOW	Dow Inc.

III. METHODOLOGY

In this section, we will provide a detailed explanation of the dataset used in this study, the construction of the training model, and the methods employed. In order to further investigate the significance of quantum price in financial market time series prediction, this project constructed a simple self-organizing composite neural network, trained on a specially processed dataset, and finally conducted simple backtesting tests based on historical data.

A. Data set

1) Data sources: The most important thing in training artificial intelligence models is a large amount of high-quality data. This project uses a popular open-source library developed by Ran Aroussi, the yfinance library, to obtain financial data available on Yahoo Finance. The project adopts thirty constituent stocks belonging to the Dow Jones Industrial Average (Table I). The index composed of them is also one of the most representative indices in the US stock market.

We believe that using stocks from companies in various industries as training data can better simulate the behavior and patterns of the entire market in a limited dataset and train artificial intelligence models.

2) Data processing: Data preprocessing is a crucial step in training high-quality machine learning models.

a) Fill in missing data: This project adopts forward padding method to fill missing data, which is a commonly used method in time series data. Its principle is to replace missing values with the previous valid data. This method assumes that the data changes little in a short period of time and is suitable for many continuous trading data.

b) Feature engineering: Just as a mature technical analyst would use various technical indicators as auxiliary tools when analyzing stocks, training models also requires data with different features for training. This project has added multiple technical indicators in addition to traditional indicators. These indicators can to some extent reflect whether investors in the market will choose to retain capital. At the same time, when facing different levels of market volatility, it will also affect the trading strategy of each individual. However, not all technical indicators can be used as features, as we also need to pay attention to the correlation between feature indicators. Feature correlation refers to the degree of correlation between two or more variables, usually measured by correlation coefficients. When two features have a high correlation, the information they provide may overlap, which is to some extent redundant. Simultaneously selecting features with low correlation can reduce the risk of multicollinearity when constructing statistical models, and improve the predictive performance and interpretability of the model. Features with low correlation can also enhance the model's generalization ability, allowing the model to more clearly capture independent information behind different features, which is beneficial for reducing overfitting and enhancing the model's adaptability to new data.

Correlation coefficient matrix:

$$\mathbf{R} = \begin{bmatrix} \rho_{1,1} & \rho_{1,2} & \cdots & \rho_{1,p} \\ \rho_{2,1} & \rho_{2,2} & \cdots & \rho_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{p,1} & \rho_{p,2} & \cdots & \rho_{p,p} \end{bmatrix},$$

$$\text{where} \quad \rho_{ij} = \frac{\operatorname{cov}(X_i, X_j)}{\sigma_{X_i}\sigma_{X_j}}.$$

This project uses a correlation coefficient matrix to determine the level of correlation between two indicators (Figure 11). The correlation coefficient matrix is used to statistically measure the degree of linear correlation between two random variables. The Pearson correlation coefficient between two variables is defined as the product of their covariance divided by their standard deviations.

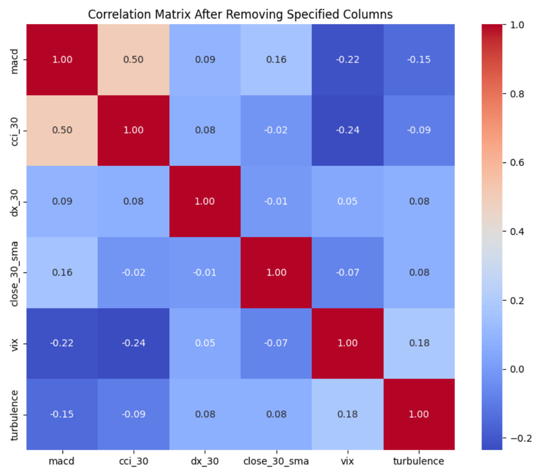


Fig. 11. Thermogram of the correlation coefficient matrix.

The technical indicators added to this project are detailed in Table II.

TABLE II
INDICATORS AND THEIR DESCRIPTIONS

Indicator Name	Description
MACD	Moving Average Convergence Divergence. It captures changes in price trends by comparing short-term and long-term moving averages.
CCI_30	30-day Commodity Channel Index. It identifies potential price reversal opportunities by comparing the deviation of price from its moving average.
DX_30	30-day Directional Index. It is used to measure the strength of the price trend and is often employed for further trend analysis.
Close_30_sma	30-day Simple Moving Average of closing prices. It reflects the average price over the past 30 days, smoothing out price fluctuations and capturing the overall trend.
VIX	Volatility Index. It reflects the market's expectation of volatility in the next 30 days and is often regarded as the 'fear index.'
Turbulence	Market Turbulence Index. It examines abnormal price fluctuations to help identify periods of market risk and instability.

c) QPL: This paper proposes a quantum-inspired stock return model that processes historical closing price data to extract a series of Quantum Price Level (QPL) indicators, thereby supporting the development of quantitative trading strategies. The method is divided into the following nine steps, which are interconnected to form a complete indicator calculation process.

1. Dynamic Return Calculation: This function takes as input an array of closing prices for a stock and computes the dynamic return (denoted as `dt_rt`), defined as the ratio of the current day's closing price to the next day's closing price, as follows:

$$\text{dt\_rt}[d] = \frac{\text{close\_prices}[d]}{\text{close\_prices}[d+1]}, \quad \text{if close\_prices}[d+1] \neq 0, \\ \text{otherwise, set to } 1.$$

When the next day's closing price is 0, to avoid a division-by-zero error, the ratio is set to 1. The resulting dynamic return sequence reflects the basic "step" in price changes, providing the raw signal for constructing the frequency distribution in subsequent steps.

2. Constructing the Frequency Distribution: Given the dynamic return sequence and a discretization step dr (for example, $3\sigma/50$, where σ is the standard deviation of `dt_rt`), this function divides the continuous dynamic returns into 100 bins (by default) and counts the frequency of the samples in each bin to generate a frequency distribution Q. Simultaneously, Q is normalized to obtain the probability distribution NQ, and the corresponding values r for each bin are generated, starting from $1 - 50 \times dr$ and incrementing by dr for each bin. This discretization step helps capture the statistical structure of the return data and provides the basis for calculating local characteristics in subsequent steps.

3. Calculation of the L Coefficient: This function identifies the bin with the highest frequency (i.e., the peak) in the normalized frequency distribution NQ and uses the r values of that bin and its two adjacent bins (adjusted to $r_0 \pm dr$) to calculate the L coefficient:

$$L = \left| \frac{r_{n-1}^2 NQ[i-1] - r_1^2 NQ[i+1]}{r_{n-1}^4 NQ[i-1] - r_1^4 NQ[i+1]} \right|$$

If the peak is at the boundary or the denominator is zero, L is set to 0. The L coefficient reflects the local structure of the frequency distribution, providing key input for solving the quantum parameters later.

4. Calculation of Quantum Level K Parameters: For each quantum level eL (ranging from 0 to max_levels-1), this function calculates the K parameter using the following formula:

$$K[eL] = \left(\frac{1.1924 + 33.2383 \times eL + 56.2169 \times eL^2}{1 + 43.6106 \times eL} \right)^{\frac{1}{3}},$$

and stores each K parameter in an array. The K parameter is a factor that adjusts the characteristics of each quantum level in the model and directly influences the solution of the subsequent cubic equation, thus determining the quantum energy level.

5. Solving the Cubic Equation:

For each quantum level, this function sets up the parameters for a cubic equation as follows:

$$p = -(2eL+1)^2, \quad q = -L \cdot (2eL+1)^3 \cdot K_{\text{val}}^3.$$

It then constructs the discriminant:

$$\Delta = \frac{q^2}{4} + \frac{p^3}{27},$$

and applies Cardano's formula by computing:

`cube_root(x)` computes the cube root of a number; if the result is complex with a negligible imaginary part (which can be regarded as a numerical error), only the real part is returned.

$$u = \text{cube\_root}\left(-\frac{q}{2} + \sqrt{\Delta}\right), \quad v = \text{cube\_root}\left(-\frac{q}{2} - \sqrt{\Delta}\right),$$

Returning $u+v$ as the raw quantum parameter for that level (denoted as QFEL), which forms the basis for the subsequent calculation of the quantum price return.

6. Calculation of Quantum Energy Level and Price Return:

1) Calculation of the Quantum Energy Level (QFEL): QFEL for each quantum level eL is obtained by iteratively calling the solution of the cube equation (Function above, Part 5).
2) Calculation of the Quantum Price Return (QPR) and Normalized Quantum Price Return (NQPR): These are defined as:

$$QPR[eL] = \frac{QFEL[eL]}{QFEL[0]},$$

and, using the market volatility σ,

$$NQPR[eL] = 1 + 0.21 \times \sigma \times QPR[eL].$$

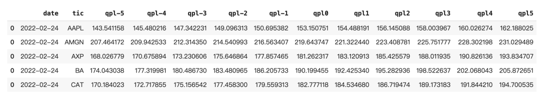
The resulting QFEL, QPR, and NQPR serve as the raw parameters, relative returns, and normalized adjustment factors for each quantum level, with NQPR being directly used for calculating the QPL indicators in the next step.

7. Calculation of QPL Series Indicators:

Using the opening prices from the processed stock data, this function calculates a series of QPL indicators based on the normalized quantum price return NQPR for different quantum levels:

- For lower-level indicators (such as `qpl-5` to `qpl-1`), the opening price is divided by the corresponding NQPR.
- For `qpl0` and higher-level indicators (such as `qpl1` to `qpl5`), the opening price is multiplied by the corresponding NQPR.

The following illustration shows the effect of this function (Figure 12).

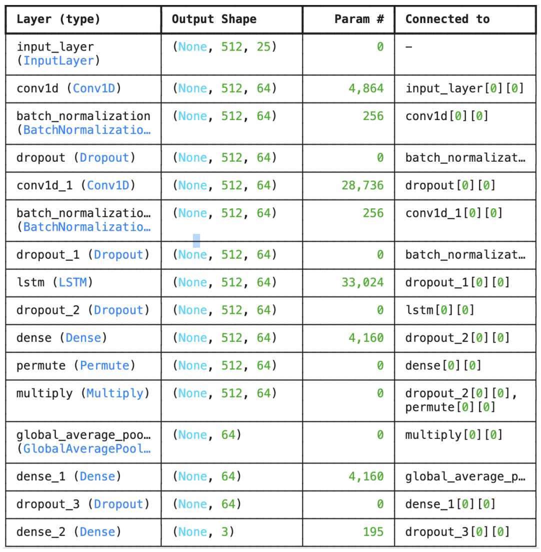


	date	tic	qpl-5	qpl-4	qpl-3	qpl-2	qpl-1	qpl0	qpl1	qpl2	qpl3	qpl4	qpl5
0	2022-02-24	AAPL	143.541158	145.480216	147.342231	149.096313	150.695382	153.150751	154.488191	156.145088	158.003967	160.026274	162.188025
0	2022-02-24	AMGN	207.464172	209.942533	212.314350	214.540993	216.563407	219.643747	221.322440	223.408781	225.751777	228.302198	231.029489
0	2022-02-24	AXP	168.026779	170.675894	173.230606	175.646864	177.857465	181.262317	183.120913	185.425579	188.011935	190.826136	193.834707
0	2022-02-24	BA	174.043038	177.319981	180.486730	183.480965	186.205733	190.199455	192.425340	195.282936	198.522637	202.068043	205.872651
0	2022-02-24	CAT	170.184023	172.717855	175.156542	177.458300	179.559313	182.777118	184.534680	186.719474	189.173183	191.844210	194.700535

Fig. 12. Quantum price illustration of stock time series.

B. Model Construction

The following narrative explains each main component of our training approach—from data splitting to defining an attention-based deep learning architecture and a custom data generator for sliding-window training.



Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 512, 25)	0	-
conv1d (Conv1D)	(None, 512, 64)	4,864	input_layer[0][0]
batch_normalization (BatchNormalizatio…	(None, 512, 64)	256	conv1d[0][0]
dropout (Dropout)	(None, 512, 64)	0	batch_normalizat…
conv1d_1 (Conv1D)	(None, 512, 64)	28,736	dropout[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 512, 64)	256	conv1d_1[0][0]
dropout_1 (Dropout)	(None, 512, 64)	0	batch_normalizat…
lstm (LSTM)	(None, 512, 64)	33,024	dropout_1[0][0]
dropout_2 (Dropout)	(None, 512, 64)	0	lstm[0][0]
dense (Dense)	(None, 512, 64)	4,160	dropout_2[0][0]
permute (Permute)	(None, 512, 64)	0	dense[0][0]
multiply (Multiply)	(None, 512, 64)	0	dropout_2[0][0], permute[0][0]
global_average_poo… (GlobalAveragePool…	(None, 64)	0	multiply[0][0]
dense_1 (Dense)	(None, 64)	4,160	global_average_p…
dropout_3 (Dropout)	(None, 64)	0	dense_1[0][0]
dense_2 (Dense)	(None, 3)	195	dropout_3[0][0]

Fig. 13. Model Structure Diagram.

Our model leverages a combination of convolutional neural networks (CNNs), Long Short-Term Memory (LSTM) units, and an attention mechanism to forecast stock price indicators (Figure 13). The overall system is designed in several sequential steps:

1) Data Splitting and Preprocessing: The complete dataset (referred to as `processed_full`) is split into distinct subsets based on predefined date ranges for training, trading, prediction, and comparison purposes. This is accomplished via a custom helper function called `data_split`. For each resulting subset, we extract the date column and print the corresponding sample counts (e.g., printing "Train samples" and "Compare samples"). In addition, stock identifiers (tickers) are mapped to unique numerical codes (stored in `tic_encoded`) so that the model can handle categorical stock information efficiently.

2) Defining an Attention Layer: To enhance the model's capability of focusing on the most relevant time steps in the sequential data, we define an attention mechanism in the form of an `attention_3d_block` function. The input tensor (shape (batch, time_steps, features)) is passed through a Dense layer with a softmax activation to generate attention weights. Optionally, the weights can be averaged along the time-axis (controlled by the `SINGLE_ATTENTION_VECTOR` flag), repeated to match the dimensions, and then used to reweight the LSTM output via element-wise multiplication. This attention mechanism enables the model to learn which time steps are most influential for the final prediction.

3) Network Architecture: CNN + LSTM + Attention: The core of the model is implemented in the `attention_model_optimized` function. The architecture follows these main steps:

- **Local Feature Extraction:**
  The input sequence (with dimensions defined by the number of time steps and features) is first processed by two successive 1D convolutional layers with filter sizes of 64. The first convolution uses a kernel size of 3, while the second uses a kernel size of 7. Batch normalization and dropout (with a dropout rate of 0.3) are applied after each convolution to stabilize training and reduce overfitting.
- **Temporal Modeling:**
  The output from the convolutional layers is fed into a unidirectional LSTM layer with 64 units. With `return_sequences` set to True, the LSTM layer captures the sequential dynamics across all time steps, and dropout is applied to its outputs to further regularize the network.
- **Attention Mechanism:**
  The LSTM outputs are then passed through the previously defined attention block. The attention-augmented features are globally aggregated using a `GlobalAveragePooling1D` layer, which produces a fixed-dimensional feature vector summarizing the entire sequence.
- **Prediction Head:**
  The aggregate features are passed through a dense layer with 64 neurons and ReLU activation followed by additional dropout. Finally, a fully connected output layer with three neurons (and linear activation) predicts the target values, which correspond to stock indicators such as the closing, high, and low prices.

4) Data Normalization: To ensure robust training across different stocks, the training subset is scaled using a `MinMaxScaler` (with values rescaled between 0 and 1). The model uses a predefined list of technical indicators and other features (including the encoded ticker identifier) so that all features are normalized uniformly. The scaler is first fitted on the entire training data to avoid inter-stock discrepancies and is then applied to the training, prediction, and comparison datasets.

5) Custom Data Generator: A custom data generator is implemented by extending Keras' `Sequence` class. This generator is designed to produce sliding windows (with a configurable look-back of 512 time steps) for each stock separately. The data generator:

- Groups the dataset by stock ticker and sorts each group by date.
- Applies the previously fitted `MinMaxScaler` to obtain normalized data.
- Iterates through each group to generate sliding window samples along with their corresponding target values (selected from specific indices corresponding to close, high, and low prices).
- Batches the samples dynamically during training to ensure efficient memory utilization.

6) Training and Validation Setup: After constructing the complete list of sliding window samples, the dataset is partitioned into training and validation sets based on a predefined validation split (e.g., 20% for validation). Two distinct instances of the data generator are then created—one for training and one for validation. These generators feed mini-batches to the model during training, enabling the use of real-time data augmentation without the need to pre-compute the full sliding window matrix.

C. Backtesting Process

To simulate the trading process for each stock, our study designs a backtesting function, `backtest_stock_hourly`, which simulates the trades based on the following logic:

1) Trading Signal Generation: For each trading day, the prediction data includes the predicted high and low prices for that day. The backtesting function first converts the `date` field in the prediction data to a date format and extracts the date (denoted as `date_only`) from the minute-level data in the `datetime` field to match the prediction data.
For each hour of the day:

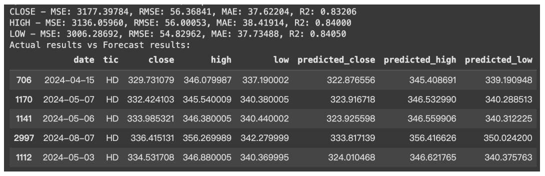
- If there is no active position and the hourly low price is lower than the predicted low price, a buy signal is generated. At this moment, the entire available capital (after accounting for transaction costs) is used to purchase shares at the hourly low price.
- If a position is held and the hourly high price exceeds the predicted high price, a sell signal is triggered. In this case, the position is liquidated at the hourly high price, and detailed transaction information (including buy time, sell time, transaction price, quantity, and profit) is recorded.
- To avoid multiple trades within the same day, only one buy and one sell are executed per day. If a position remains at the end of the day, it is closed at the closing price of the last hour.

2) Recording Trades and Equity Calculation: For every buy and sell operation, detailed transaction records are maintained and the account balance is updated in real time. In addition, the current account equity is recorded at each hour. If a position is held, the equity is valued based on the current hourly closing price; if not, the equity equals the current cash balance. This process constructs a complete equity curve.

3) Control group: For comparison, we also construct an equity curve for a buy-and-hold strategy. The buy-and-hold strategy is built by using the closing price of the first K-line at the beginning of the backtesting period as the buy price, calculating the number of shares purchased accordingly, and then computing the equity based on the hourly closing prices. The equity from all stocks is then aggregated to form the overall portfolio equity curve.

IV. Result Analysis

In this section, we conducted an in-depth analysis of the performance of our model in predicting the high, low, and closing prices of a specific financial product's future T+1. At the same time, we also constructed a simple and naive buying and selling trading strategy for backtesting, and compared the returns of our strategy with those of a benchmark trading strategy called long-term holding.



```
CLOSE - MSE: 3177.39784, RMSE: 56.36841, MAE: 37.62204, R2: 0.83206
HIGH - MSE: 3136.05960, RMSE: 56.00053, MAE: 38.41914, R2: 0.84000
LOW - MSE: 3006.28692, RMSE: 54.82962, MAE: 37.73488, R2: 0.84050
Actual results vs Forecast results:
```

	date	tic	close	high	low	predicted_close	predicted_high	predicted_low
706	2024-04-15	HD	329.731079	346.079987	337.190002	322.876556	345.408691	339.190948
1170	2024-05-07	HD	332.424103	345.540009	340.380005	323.916718	346.532990	340.288513
1141	2024-05-06	HD	333.985321	346.380005	340.440002	323.925598	346.559906	340.312225
2997	2024-08-07	HD	336.415131	356.269989	342.279999	333.817139	356.416626	350.024200
1112	2024-05-03	HD	334.531708	346.880005	340.369995	324.010468	346.621765	340.375763

Fig. 14. Model Result Diagram.

1) *Model performance:* In this study, we used mean square error (MSE), mean absolute error (MAE), and coefficient of determination (R²) as evaluation metrics to systematically measure the performance of the model in predicting future T+1 high, low, and closing prices (Figure 14). Taking predicting low prices as an example, the results show that the MSE of the model in low price prediction is 3006, indicating a high prediction error; If the RMSE is 54, it indicates that the model's predictions usually deviate from the true values by about 50 units (for example, if predicting stock prices, these 50 units correspond to the range of price fluctuations); The MAE is 37, and the average difference between each prediction and the true value is about 37 units, especially in low price and closing price predictions; The coefficient of determination (R²) is 0.84, 0.84 indicates that the model can explain about 84% of the data variation, which usually indicates that the model has strong explanatory power in prediction. The remaining 16% is the unexplained part of the model, which may be related to noise in the data or limitations of the model itself (Figure 15). Although the performance of the corresponding

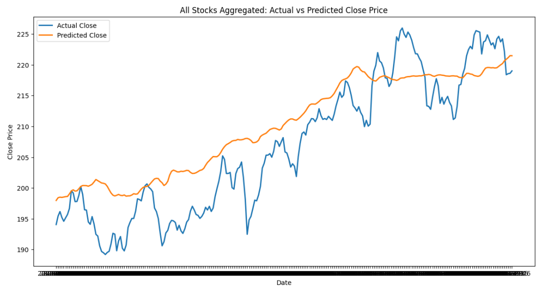
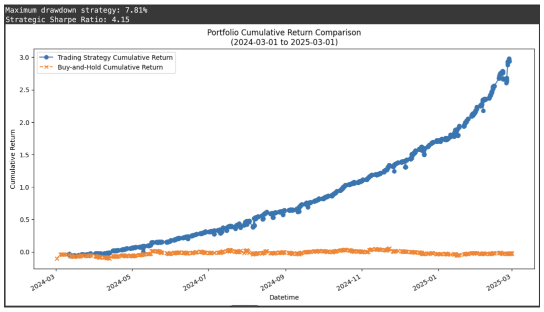


Fig. 15. All Stocks Aggregated: Actual vs Predicted Close Price.

indicators fluctuates slightly, the overall prediction accuracy is relatively ideal.

2) *Backtesting effect:* Based on historical data backtesting, the model driven dynamic trading strategy exhibits significant excess returns. Throughout the entire simulation period, the strategy achieved a cumulative return of 300%, significantly higher than the benchmark "buy hold" strategy of 15% during the same period (Figure 16). In addition, from the perspective of risk indicators, the maximum drawdown control of the strategy is around 7.85%, and the Sharpe ratio has reached 4.15, indicating that it has a relatively stable risk management ability while pursuing higher returns.



```
Maximum drawdown strategy: 7.81%
Strategic Sharpe Ratio: 4.15
```

Fig. 16. Portfolio Cumulative Return Comparison.

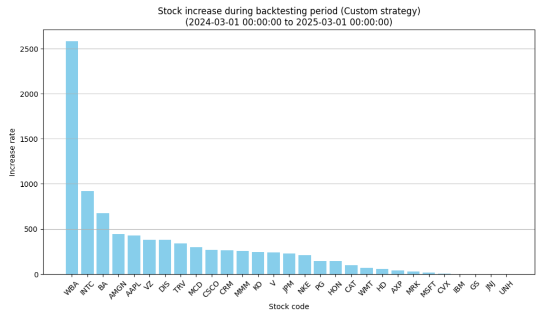


Fig. 17. Stock increase during backtesting period (Custom strategy).

From the perspective of equity curve, the model strategy can respond to market fluctuations in a timely manner and achieve rapid appreciation of funds; In most market conditions, its smooth trend also reflects good resistance to volatility. It is worth noting that in some extreme market conditions, the strategy may experience a decline in returns after forced liquidation, which suggests the need to introduce more comprehensive risk control measures in the future, such as dynamic stop loss strategies, to reduce the adverse effects of sudden market trends (Figure 17). Overall, these backtesting results fully validate the practical utility of real-time trading decisions based on model predicted signals, and provide clear directions for further optimizing trading strategies and improving risk control levels.

V. Conclusion

This article proposes a novel financial market prediction method that integrates quantum mechanics principles and deep learning techniques. It constructs a quantum price level indicator based on historical closing price data and combines it with convolutional neural networks, long short-term memory networks, and attention mechanisms. By discretizing traditional continuous asset price fluctuations into a structure similar to quantum energy states, this method not only successfully captures complex phenomena such as inherent nonlinearity, jumps, and volatility clustering in the market, but also provides a probabilistic and multi-scale new perspective for modeling multidimensional market features.

Firstly, in the process of data preprocessing and feature engineering, the study achieved deep mining and quality assurance of raw data by filling in missing data, constructing technical indicators, and analyzing the correlation between features, laying a solid foundation for the subsequent calculation of quantum indicators. By utilizing dynamic return calculation, frequency distribution discretization, and local structure extraction (through L coefficients), and finally combining with the Cardano formula to solve for various quantum energy level parameters, a series of QPL indicators are generated through normalization and standardization processing, providing multi-dimensional and market detail rich information input for deep learning models. In terms of model construction, the self-organizing composite neural network architecture designed in this article effectively captures the complex spatial and temporal dependencies in financial time-series data through data splitting, feature remapping, and sliding window training; Among them, CNN focuses on local pattern extraction, LSTM has unique advantages for long-period dependencies, and the Attention mechanism exhibits extremely high flexibility in dynamically focusing on key information. The complementary advantages of each sub model not only improve the accuracy and robustness of predictions, but also provide theoretical support and empirical evidence for short-term reversal strategies and practical trading systems based on quantum trading ideas. Overall, this research work breaks through the limitations of traditional continuity models such as geometric Brownian motion, successfully introduces quantum mechanics ideas into financial market prediction, and demonstrates its unique advantages in explaining and predicting discontinuous behaviors such as market price jumps and volatility clustering through the integration with deep learning techniques. Although current methods still face challenges in terms of data dimension, model complexity, and computational efficiency, this hybrid model provides clear development directions and rich insights for further integrating new Transformer architectures, improving model interpretability, and optimizing data processing in high-frequency trading environments in the future.

Future work can validate the adaptability of models on larger and more diverse datasets, while combining more external information such as market sentiment and news events to further refine the synergistic effects of various deep architectures, in order to achieve more efficient and stable financial forecasting and risk management systems, and provide investors with more guiding decision support tools.

References

[1] R. S. T. Lee, “Quantum price levels—Basic theory and numerical computation technique,” in *Quantum Finance*. Singapore: Springer, 2020, doi: 10.1007/978-981-32-9796-8_5.

[2] S. Focardi, F. J. Fabozzi, and D. Mazza, “Quantum option pricing and quantum finance,” *The Journal of Derivatives*, vol. 28, no. 1, pp. 79–98, 2020, doi: 10.3905/jod.2020.1.111.

[3] P. Yue and X. Li, ”Research on stock index futures arbitrage strategy based on convolutional neural network,” *Advances in Applied Mathematics*, vol. 10, no. 2, pp. 557-567, 2021. [Online]. Available: https://doi.org/10.12677/aam.2021.102061.

[4] O. B. Sezer and A. M. Ozbayoglu, ”Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach,” *Applied Soft Computing*, vol. 70, pp. 525-538, 2018. [Online]. Available: https://doi.org/10.1016/j.asoc.2018.04.024.

[5] N. Chen, ”Exploring the development and application of LSTM variants,” *Applied and Computational Engineering Research*, vol. 53, 2024. [Online]. Available: https://doi.org/10.54254/2755-2721/53/20241288

[6] S. Hochreiter and J. Schmidhuber, ”Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[7] J. Hu, X. Wang, Y. Zhang, D. Zhang, M. Zhang, and J. Xue, ”Time series prediction method based on variant LSTM recurrent neural network,” *Neural Processing Letters*, vol. 52, pp. 1485–1500, 2020. [Online]. Available: https://doi.org/10.1007/s11063-020-10319-3

[8] I. D. Mienye, T. G. Swart, and G. Obaido, ”Recurrent neural networks: A comprehensive review of architectures, variants, and applications,” *Information*, vol. 15, no. 9, p. 517, 2024. [Online]. Available: https://doi.org/10.3390/info15090517

[9] R. C. Staudemeyer and E. Rothstein Morris, ”Understanding long short-term memory recurrent neural networks – A tutorial-like introduction,” *arXiv preprint*, 2019. [Online]. Available: https://arxiv.org/abs/1909.09586

[10] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014. [Online]. Available: https://arxiv.org/abs/1409.0473

[11] X. He and L. Liao, “Neural collaborative filtering,” in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[12] A. Vaswani *et al.*, “Attention is all you need,” in *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.

[13] S. Wang, Q. Liu, and L. Zhang, “Linformer: Self-attention with linear complexity,” in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, 2020, pp. 4551–4561. [Online]. Available: https://proceedings.mlr.press/v119/wang20d.html

[14] L. Zhang and Y. Xu, “A comprehensive review of attention mechanisms in financial prediction,” *J. Financ. Eng.*, vol. 6, no. 2, pp. 142–158, Jun. 2019.

[15] B. N. Lehmann, “Fads, martingales, and market efficiency,” *The Quarterly Journal of Economics*, vol. 105, no. 1, pp. 1–28, 1990.

[16] N. Jegadeesh, “Evidence of predictable behavior of security returns,” *The Journal of Finance*, vol. 45, no. 3, pp. 881–898, 1990.

[17] T. Fischer and C. Krauss, “Deep learning with long short-term memory networks for financial market predictions,” *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[18] Quantum Financial Forecast Center (QFFC). [Online]. Available: https://qffc.uic.edu.cn/home