Wavelet Based Financial Forecast Ensemble Featuring Hybrid Quantum-Classical LSTM Model

PETER BIGICA, XIAODI WANG
Department of Mathematics and Computer Science
Western Connecticut State University
181 White Street, Danbury, Connecticut
UNITED STATES

Abstract: One of the most sought-after goals in the financial world is a reliable method by which investors can predict a stock price movement consistently. Advancements in stock prediction via the use of machine learning have improved the accuracy of such predictions and yielded better ideas about value investments in the stock market. However, with the addition of the state-of-art tool M-band wavelet transform as a preprocessing step, we can denoise our raw data set (prior stock prices) and refine it to make the forecast even more accurate. Following this preprocessing step, multiple machine learning techniques are deployed to construct a robust, non-parametric hybrid forecasting model. To demonstrate the novelty of our algorithm, we present a case study on stock price prediction employing a discrete 4-band wavelet transform integrated with a hybrid machine learning ensemble. Our results underscore the potential and importance of such ensemble methods in refining the accuracy and reliability of financial forecasts. Furthermore, in theory, quantum computing can further optimize these algorithms, potentially leading to more precise stock price forecasts. In particular, our ensemble will feature a hybrid quantum-classical LSTM Model to demonstrate the potential of both wavelets and quantum computing in stock forecasting.

Key-Words: Machine Learning, M-Band Wavelet Transform, Neural Network Ensemble, Quantum Computing, Stock forecasting, Long Short-Term Memory (LSTM) networks

Received: December 16, 2023. Revised: July 15, 2024. Accepted: August 9, 2024. Published: September 18, 2024.

1 Introduction

The financial realm is characterized by its volatile nature, with stock price movements often being described as "random". As such, one of the most sought-after goals in the financial world is finding a consistent, reliable method by which these movements can be anticipated. Over the years as investors and analysts have attempted to find such a solution, technological advancements (particularly in machine learning) have provided new techniques by which stock prices can be forecasted. These advances have not only elevated the precision of stock predictions but have allowed for the creation of new algorithms to be created to in- increase the precision of market predictions further, [1].

However, while these techniques continue to advance and boundaries are continually being pushed, handling financial data and trying to make an accurate/consistent prediction remains a daunting task. All predictions must be based on historical data which can often be limited in its sample size, is virtually always non-linear, has no clear trends, and can have high volatility. In other words, financial data can have a lot of "noise" which makes the task of forecasting financial movements a difficult one, [2].

To combat such volatility in the quest to develop a more reliable model, we introduce the M-Band wavelet. As a state-of-art preprocessing tool, the M-band wavelet transform has the profound ability to denoise a raw dataset, [3]. It refines and purifies the data, setting the stage for even more accurate forecasting than what was previously achievable.

The M-band wavelet transform can separate data into different frequency components, specifically 1 low-frequency component and M high-frequency components, and more often the "noise" embedded in the raw data that makes stock forecasting so difficult is contained within the high-frequency components of the data, [4]. Once the Wavelet Transform and denoise procedure have been applied, we are left with a "clean" data set that better reflects the true trends within the data, and ultimately allows us to use machine learning techniques to achieve a better prediction.

To test the validity of this method, several different techniques and neural networks will be employed through a financial ensemble to not only see the effectiveness of different neural networks but also to test the validity of the Wavelet denoise algorithm applied to the data. An ensemble forecast will give us more results, and should the results align more confidence as to what financial decision to make in the future, [5].

The metric of testing error will be accomplished with a Root Mean Squared Error (RMSE).

In this research, the following techniques that will be used include Autoregressive Integrated Moving Average (ARIMA), Long Short Term Neural Network (LSTM), Support Vector Regression (SVR), and Recurrent Neural Network (RNN).

Lastly, we will continue our exploration into constructing a more reliable forecasting model by venturing into the realm of quantum computing. Quantum computing is an emerging advancement that offers several advantages over traditional methods including but not limited to more computational power, more efficient optimization, and the potential for new algorithms to make even better predictions using machine learning methods, [6].

The goal of this research is to find a more reliable method for investors to make smarter financial decisions by employing alternative machine learning methods. We will set out to prove the potential of wavelets in financial forecasting, while also demonstrating the power of utilizing an ensemble forecast as opposed to standard single-method forecasts when making financial decisions to help create better judgment. Additionally, we will show the potential of quantum mechanics in financial forecasting as we get closer to a new era of computing

2 Literature Review

Stock forecasting (for our purposes the area of trying to predict future stock prices via machine learning) has been an area of intense research in the last several decades. After 50 years of advances in the field, neural networks have gained significant attention for their capacity to model complex non-linear relationships predominant in stock markets.

Historically, stock market predictions relied heavily on statistical methods like the ARIMA and GARCH models, [7]. As deep/machine learning methods have developed leading into the 21st century, stock forecasters started utilizing neural networks, with initial experiments focusing primarily on Feedforward Neural Networks (FNN) and Recurrent Neural Networks (RNN). As techniques developed, more sophisticated networks such as Long Short-Term Memory (LSTM) units and Convolutional Neural Networks (CNN) have become predominant in the modern day, [5].

Advantages of neural networks over statistical methods include:

- Modeling Non-linearity data sets
- Better proficiency with working with time series data
- Integration of multiple diverse data sets to train models

Despite their advances, neural networks still contain challenges:

- Overfitting of data sets making models less generalized
- Interpretability and acting like a ”black box” which complicates relying on models for financial decisions
- The required use of significant computational resources, which can limit their ability to perform real-time predictions,

In this paper, our contribution is to overcome these challenges by creating a forecasting ensemble consisting of various machine learning algorithms all based on a Wavelet Transform to denoise prior stock data and in turn produce better, more accurate forecasts that we can study and evaluate.

3 Primaries

3.1 M-Band Wavelet

An Orthogonal M-Band Discrete Wavelet Transform (DWT) is totally determined by M sets of filters in a filter bank with certain properties, [4]. In any such filter bank, there are one low pass filter α, and $M-1$ high pass filters $\beta^{(j)}$ for $j = 1,2,....,M-1$ with N vanishing moments. These filters satisfy the following conditions:

$$\sum_{i=1}^{n} \alpha_i = \sqrt{M} \tag{1}$$

$$\sum_{i=1}^{n} i^K \beta_i^{(j)} = 0 \text{ for } k = 1,2,\ldots,N-1 \text{ and } j = 1,\ldots,M-1 \tag{2}$$

$$||\alpha|| = ||\beta^{(j)}|| = 1 \text{ for } j = 1,...,M-1 \tag{3}$$

$$< \alpha, \beta^{(j)} >= 0 \text{ for } j = 1,...,M-1 \tag{4}$$

$$\langle \beta^{(i)}, \beta^{(j)} \rangle = 0 \text{ for } i,j = 1,\ldots,M-1 \text{ and } i \neq j \tag{5}$$

In this research, the wavelet transform was performed with the following filter banks:
$\alpha = [-0.06737176, 0.09419511, 0.40580489, 0.56737176, 0.56737176, 0.40580489, 0.09419511, -0.06737176]$,
$\beta = [-0.09419511, 0.06737176, 0.56737176, 0.40580489, -0.40580489, -0.56737176, -0.06737176, 0.0941951]$,
$\gamma = [-0.09419511, -0.06737176, 0.56737176, -0.4058048, -0.4058048, -0.5673717, -0.06737176, -0.0941951]$,
$\delta = [-0.06737176, -0.09419511, 0.40580489, -0.56737176, 0.56737176, -0.40580489, 0.09419511, 0.06737176]$

If the signal we are working with (S) is contained within $\mathbb{R}^{4k}(k \in \mathbf{N}, k \geq 2)$, we can create a $4k \times 4k$ Wavelet transform matrix T_1 by shifting and wrapping around the filters (as shown in figure 1).

$$\begin{bmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 \\
\alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\
\beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 \\
\beta_5 & \beta_6 & \beta_7 & \beta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 \\
\gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 \\
\gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\
\delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 & \delta_7 & \delta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 & \delta_7 & \delta_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 & \delta_7 & \delta_8 \\
\delta_5 & \delta_6 & \delta_7 & \delta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_1 & \delta_2 & \delta_3 & \delta_4
\end{bmatrix}$$

Fig. 1: An example of 16x16 Wavelet Transform matrix

Let $C_1, C_2, \ldots, C_{Mk}$ be the column vectors of T_1^T. Then, $C_1^T, C_2^T, \ldots, C_{Mk}^T$ are row vectors of T_1. Since T_1 is an orthogonal matrix, $\{C_1, C_2, \ldots, C_{Mk}\}$ forms an orthonormal basis of $\mathbb{R}^{Mk}$. Therefore, the components of $\tilde{S}_1$ are coordinates of S under this wavelet basis, and hence $\|T_1 S\| = \|\tilde{S}_1\|$. The components of $\tilde{S}_1$ are also called the wavelet coefficients of S. Moreover, the M-Band DWT of S decomposes S into M different frequency components with $a^{(1)}$ being the lowest frequency component (or trend) and $d_1^{(1)}, \ldots, d_{M-1}^{(1)}$ being the higher frequency components (or fluctuations) of S. If necessary, and if k is divisible M, we can apply DWT to $a^{(1)}$ using a $k \times k$ DWT matrix T_2 such that:

$$T_2 a^{(1)} = \left[a^{(2)}, d_1^{(2)}, \ldots, d_{M-1}^{(2)}\right]^T \triangleq \tilde{S}_2, \qquad (6)$$

where

$$a^{(2)} = \left[a_1^{(2)}, a_2^{(2)}, a_3^{(2)}, \ldots, a_{\frac{k}{M}}^{(2)}\right]^T, \qquad (7)$$

and

$$d_i^{(2)} = \left[d_{i,1}^{(2)}, d_{i,2}^{(2)}, d_{i,3}^{(2)}, \ldots, d_{i,\frac{k}{M}}^{(2)}\right]^T, \quad i = 1, \ldots, M-1, \qquad (8)$$

The M-Band DWT of $a^{(1)}$ decomposes $a^{(1)}$ into M different frequency components with $a^{(2)}$ being the lowest frequency and $d_i^{(2)}$ $(i = 1, \ldots, M-1)$ being higher frequency components of $a^{(1)}$.

Let $T = \begin{bmatrix} T_2 & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix} T_1$, where the lower corner $\mathbf{0}$ is an $(M-1)k \times k$ zero matrix, the upper corner $\mathbf{0}$ is a $k \times (M-1)k$ zero matrix, and I is an $(M-1)k \times (M-1)k$ identity matrix. Then,

$$TS = \left[a^{(2)}, d_1^{(2)}, \ldots, d_{M-1}^{(2)}, d_1^{(1)}, \ldots, d_{M-1}^{(1)}\right]^T \triangleq \tilde{S}_2, \qquad (9)$$

and $\tilde{S}_2$ is the second level wavelet coefficients of S. Since $\{C_1, C_2, \ldots, C_{Mk}\}$ is an orthonormal basis of $\mathbb{R}^{Mk}$

$$S = s_1 C_1 + s_2 C_2 + \ldots + s_n C_n, \qquad (10)$$

where $n = Mk$ and $s_i = C_i^T S = \langle C_i, S \rangle$, the inner product of C_i and S for $i = 1, 2, \ldots, n$. Therefore,

$$s_i = \begin{cases} a_i^{(1)} & \text{for } i = 1, 2, \ldots, k \\ d_{1,i}^{(1)} & \text{for } i = (k+1), k+2, \ldots, 2k \\ \vdots & \\ d_{(M-1),i}^{(1)} & \text{for } i = (M-1)k+1, \ldots, Mk \end{cases} \qquad (11)$$

Let

$$\begin{aligned} A^{(1)} &= a_1^{(1)} C_1 + \ldots + a_k^{(1)} C_k, D_i^{(1)} \\ &= d_{i,1}^{(1)} C_{ik+1} + \ldots + d_{i,(i+1)k}^{(1)} C_{(i+1)k} \end{aligned} \qquad (12)$$

for $i = 1, \ldots, M-1$. Then $A^{(1)}$ is corresponding to $a^{(1)}$ and $(D_i^{(1)}$ is corresponding to $d_i^{(1)})$ for $i = 1, \ldots, M-1$.If we let

$$V^{(1)} = \text{span}\{C_1, \ldots, C_k\} \qquad (13)$$

and

$$W_i^{(1)} = \text{span}\{C_{ik+1}, \ldots, C_{(i+1)k}\} \qquad (14)$$

for $i = 1, \ldots, M-1$, then $V^{(1)}, W_1^{(1)}, \ldots,$ and $W_{(M-1)}^{(1)}$ are orthogonal subspaces of $\mathbb{R}^{Mk}$ and therefore it can be represented as the following direct sum,

$$\mathbb{R}^{Mk} = V^{(1)} \oplus W_1^{(1)} \oplus \ldots \oplus W_{(M-1)}^{(1)}. \quad (15)$$

So, for any $S \in \mathbb{R}^{Mk}$, S can be written uniquely as

$$S = A^{(1)} + D_1^{(1)} + \ldots + D_{M-1}^{(1)}, \quad (16)$$

where $A^{(1)} = \text{Proj}_{V^{(1)}} S$ is the orthogonal projection of S onto $V^{(1)}$, $D_i^{(1)} = \text{Proj}_{W_i^{(1)}} S$ is the orthogonal projection of S onto $W_i^{(1)}$, for $i = 1, \ldots, M-1$.

If we apply the second level DWT to S, then subspaces $V^{(2)}, W_1^{(2)}, \ldots, W_1^{(2)}, W_2^{(1)}, \ldots,$ and $W_{M-1}^{(1)}$ are orthogonal to each other and therefore,

$$\mathbb{R}^{Mk} = V^{(2)} \oplus W_1^{(2)} \oplus \ldots \oplus W_{(M-1)}^{(2)} \oplus W_1^{(1)} \oplus \ldots \oplus W_{(M-1)}^{(1)}, \quad (17)$$

$$V^{(1)} = V^{(2)} \oplus W_1^{(2)} \oplus \ldots \oplus W_{(M-1)}^{(2)}, \quad (18)$$

$$S = A^{(2)} + D_1^{(2)} + \ldots + D_{M-1}^{(2)} + D_1^{(1)} + \ldots + D_{M-1}^{(1)}, \quad (19)$$

$$A^{(1)} = A^{(2)} + D_1^{(2)} + \ldots + D_{M-1}^{(2)} \quad (20)$$

3.2 Autoregressive Integrated Moving Average (ARIMA)

The ARIMA model, primarily based on linear relationships, has been a cornerstone in the field of time series analysis for several decades, celebrated for its simplicity and ability to handle non-stationary data. Its strengths lie in its capacity to model and forecast based on both prior data and prior errors, while incorporating autoregressive and moving average components, [7]. However, despite its strengths, ARIMA models do possess limitations especially when dealing with nonlinear data sets which more often than not are to be expected in real-world scenarios, [8].

ARIMA models primarily have been designed for making predictions involving time-series-based data. This model is typically configured using 3 values denoted (p,d,q), where 'p' signifies the order of the model, 'd' indicates the degree of differencing, and 'q' represents the number of lagged forecast errors in the model, [8].

We can see how ARIMA is derived from AR and MA models from the below. A pure auto-regressive model (AR) is one where Y_t depends only on its lag.

$$Y_t = \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + + \beta_p Y_{t-p} \quad (21)$$

where β_n is the coefficient of the lag that the model estimates.

On the other hand, a pure moving average (MA) model is one where Y_t depends only on the lagged forecast errors

$$Y_t = \phi_1 \varepsilon_{t-1} + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + \cdots + \phi_q \varepsilon_{t-q} \quad (22)$$

Now, the ARIMA model can be derived from both of these equations combined with an integrated (I) term which controls the differencing in the model. The ARIMA model is then given by:

$$Y_t' = \alpha + \beta_1 Y_{t-1}' + \cdots + \beta_p Y_{t-p}' + \varepsilon_t + \phi_1 \varepsilon_{t-1} + \cdots + \phi_q \varepsilon_{t-q} \quad (23)$$

3.3 Long Short Term Neural Network (LSTM)

Long Short-Term Memory (LSTM) networks, an advancement Recurrent Neural Networks (RNNs), have been an important point in research involving long-range modeling of time series data. Introduced by German computer scientists Hochreiter and Schmidhuber in 1995, LSTMs were designed to overcome the predominant challenges faced by RNN models at the time, more specifically the vanishing and exploding gradient problems, [3]. In RNNs, during the back-propagation process, gradients are calculated to update the network's weights. However, when dealing with larger data sets, these gradients can become extremely small (vanish) or very large (explode). These issues made it difficult for RNNs to learn and retain information over longer intervals, constraining their prowess in tasks in many tasks, [9].

LSTMs, with their enhanced architecture, effectively address these challenges which limited RNN models. Their architecture is comprised of three types of gates: the input, forget, and output gates. Each of these gates is responsible for regulating the flow of information in the network. This flow allows LSTMs to decide what information to store, discard, or out-put at each time step. These gates and their ability to determine what information should be kept or discarded at each step in the sequence, allow the network to maintain a more stable gradient over time and solve the problems which RNN suffers from, [1]. It also has the added benefit of being able to maintain information over longer intervals and thus capture intricate patterns and dependencies in time series data. Over time, various modifications to the standard LSTM model have led to new architectures, such as the Gated Recurrent Units (GRUs).

As mentioned, the LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate.

The cell remembers values over arbitrary time intervals, and the three gates regulate the flow of information into and out of the cell.

Let's denote:

- x_t as the input at time step t.
- h_t as the hidden state at time step t.
- c_t as the cell state at time step t.
- f_t as the forget gate's activation.
- i_t as the input gate's activation.
- o_t as the output gate's activation.
- σ as the sigmoid activation function.

LSTM can then be derived from the following set of equations. Additionally, the equation for the root mean square error loss function which will be used to test the accuracy of the model is given below:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{24}$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{25}$$
$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{26}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{27}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{28}$$
$$h_t = o_t \odot \tanh(c_t) \tag{29}$$
$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - h_t)^2} \tag{30}$$

where:

- W_f, W_i, W_c, and W_o are weight matrices.
- b_f, b_i, b_c, and b_o are bias vectors.
- $\odot$ represents element-wise multiplication.
- N represents the number of observations
- y_t and h_t are the target and predicted outputs at time t

The LSTM's ability to forget, learn, and output information gives it the capacity to model time series and sequences with long-range dependencies and avoid the vanishing and exploding gradient problems of traditional RNNs.

- The *forget gate* (f_t) decides what information from the cell state should be thrown away or kept.
- The *input gate* (i_t) updates the cell state with new information.
- The *output gate* (o_t) determines what the next hidden state should be.

3.4 Support Vector Regression (SVR)

Support Vector Regression (SVR), an extension of the widely acclaimed Support Vector Machines (SVM) used for regression tasks, is known for its ability to handle high-dimensional data. SVR operates by mapping input data into a higher-dimensional space where it seeks to find a hyperplane that best fits the data. The central idea is to identify a function that, for a given tolerance of error, has the smallest possible deviation from the actual training data while maintaining a maximal margin, [3]. SVR typically uses an ε-insensitive loss function that allows errors falling within a defined threshold (ε) and it penalizes errors outside this range much more heavily.

Over the years, SVR's application has spanned across diverse domains, from finance to environmental modeling. One of the defining features of SVR is its utilization of kernel functions, such as polynomial, radial basis function (RBF), sigmoid, and wavelets which allow it to model complex, non-linear relationships in data by transforming it into a space where linear regression techniques become applicable, [3]. The flexibility to choose and craft kernels grants SVR a versatile edge over many traditional regression models. While SVR's robustness and efficacy in high-dimensional spaces are notable, challenges like tuning hyperparameters and potential computational inefficiencies in very large datasets are areas of current research.

Given a dataset: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ the SVR optimization problem can be formulated as:

$$\text{minimize: } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{31}$$

$$\text{subject to: } \begin{cases} y_i - \langle w, x_i \rangle - b & \leq \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i & \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{cases} \tag{32}$$

Where:

- w is the weight vector.
- b is the bias term.
- ξ_i and ξ_i^* are slack variables.
- C is a regularization parameter.

3.5 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) have gathered substantial attention in the realm of deep learning due to their ability to process sequential data, making them an ideal choice for tasks involving time series. RNNs are designed to maintain a memory of past incidents in their internal state, allowing them to exhibit

temporal dynamic behavior and recognize patterns across time steps. Unlike traditional feed-forward neural networks, RNNs possess loops, enabling the propagation of information across sequences. This inherent feature makes them distinctively powerful in handling tasks where context from earlier steps is vital for understanding subsequent data points, [1].

However, while RNNs at the time were revolutionary, they unfortunately came with some detrimental challenges. One significant limitation is RNN's difficulty in learning long-term dependencies due to the infamous vanishing and exploding gradient problems. This challenge is still within RNN models, however, it did lead to the development of more sophisticated architectures like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), both of which were designed to overcome the shortcomings of RNNs.

RNN can be defined as such:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{33}$$
$$y_t = W_{hy}h_t + b_y \tag{34}$$

where:

- x_t is the input at time step t.
- h_t is the hidden state at time step t.
- y_t is the output at time step t.
- W_{xh}, W_{hh}, and W_{hy} are weight matrices.
- b_h and b_y are bias vectors.
- σ is the activation function.

While RNNs are powerful for modeling sequential data, they have challenges such as the vanishing and exploding gradient problems. These challenges make it difficult to learn long-range dependencies in sequences. Advanced RNN architectures, like the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU), were developed to overcome these limitations.

3.6 Quantum Hybrid Quantum Classical Model

The integration of quantum computing with classical machine learning models is one such area of research that has emerged over recent years with the slow but steady creation of new tools to put quantum computing (or rather quantum simulation) in the hands of users. Quantum computing promises unprecedented advances, allowing uses to explore new areas that classical computers could not. For our purpose, quantum computing has the potential to be combined with current classical machine learning algorithms in order to enhance their predictions abilities, [9].

Hybrid quantum-classical LSTM models aim to intertwine the time series forecasting strengths of LSTMs with quantum-enhanced data processing techniques such as quantum gates, superstition, and entanglement, [5]. Preliminary studies into this hybrid model have shown potential advantages including improved accuracy and more efficient processing speeds. Quantum algorithms, such as the Quantum Approximate Optimization Algorithm (QAOA) or Quantum Support Vector Machines (QSVM), have shown the potential in solving optimization problems more efficiently, [10]. Integrating these methods with classical LSTMs can potentially lead to faster model training and enhanced prediction accuracy. However, the creation of these hybrid models remains a challenge due to quantum computing still being young and the lack of infrastructure. Additionally, the complexity involved in effectively combining quantum and classical processes adds yet another layer of challenge for working with quantum computing. Continued ad-advancements in quantum hardware and algorithms will likely pave the way for more robust and scalable hybrid models in financial forecasting and find use in other fields as well

In the quantum version of the LSTM cell, quantum gates, and qubits replace classical gates and bits. The core memory cell would comprise qubits, and quantum gates would handle the gating mechanisms:

- The *input*, *forget*, and *output* gates could be implemented using quantum gates that control the flow of quantum information.
- Entanglement could potentially be used to capture and maintain long-range dependencies in the sequence.
- Quantum computations result in a superposition of states. A measurement collapses this superposition to classical bits. The outcome of the quantum LSTM cell would be interfaced with classical neural network layers:
- Quantum states (from qubits) of the memory cell would be measured and converted to classical information.
- This classical information would then be processed by traditional LSTM layers or other neural network architectures.

4 Experiment Design and Procedure

4.1 Wavelet Transform

For this experiment, in order to denoise our prior data set (in this case, AMZN will be the stock we will try to forecast and will use prior closing price data) a 4 Band wavelet will be applied as such:

$$S^* = TS = \begin{bmatrix} a \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} \tag{35}$$

where

$$a = \begin{bmatrix} < T_1, S > \\ < T_2, S > \\ \vdots \\ < T_{\frac{n}{4}-1}, S > \\ < T_{\frac{n}{4}}, S > \end{bmatrix} \tag{36}$$

And

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} < T_{\frac{n}{4}+1}, S > \\ < T_{\frac{n}{4}+2}, S > \\ \vdots \\ < T_n, S > \end{bmatrix} \tag{37}$$

where $T = 4$ Band Wavelet Transform Matrix, S = Prior Stock Closing Data, a = Approximation Portion (low frequency), d_i = detail portions for $i = 1, 2, 3$ and T_i is $i^{(th)}$ row of T. Additionally, n = Number of days (in the form $n = 4k$ where $k \in \mathbb{N}$). In this research, 256 days of prior data were used so $k = 84$, and hence the wavelet transform matrix was of size 256×256. Thresholding was then accomplished by redefining d_j^i using the following hard threshold to prevent over-smoothing of the data:

$$\lambda = \sigma\sqrt{2\log(N)} \tag{38}$$

where σ = Standard deviation of noise from the wavelet coefficients and N = number of elements in the band. After applying the threshold, we achieve $\hat{S}$ from S^*. Let T^T be the transform of the wavelet matrix, we can use this to reconstruct our smoothed data from the wavelet data and finally get our smoothed usable data, $\tilde{S}$:

$$T^T \cdot \hat{S} = \tilde{S} \approx S \tag{39}$$

Figure 3 shows the denoised AMZN data that will be used to train our various models in our ensemble can be seen compared to the original AMZN in Figure 2.

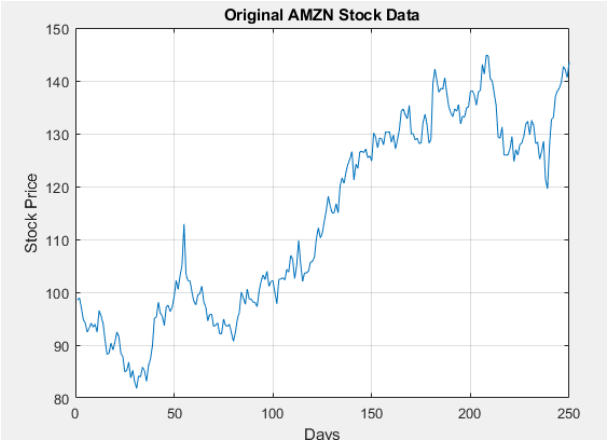

Fig. 2: Original AMZN stock data

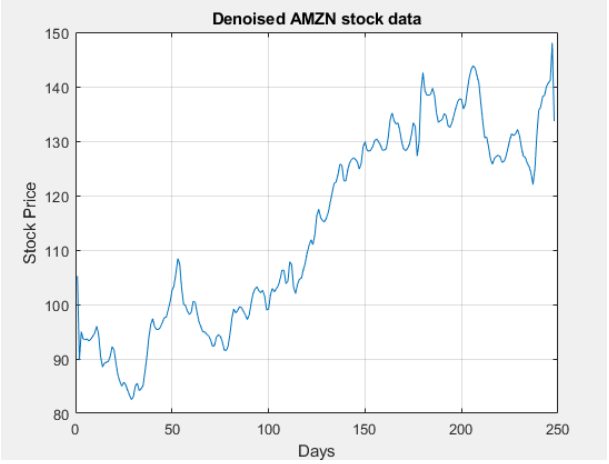

Fig. 3: Denoised AMZN stock data

As we can see, the denoised data retains the shape of the original data and the vital time points and does not over smoothing the data.

4.2 Long Short Term Neural Network (LSTM)

For the LSTM model, the model was created in Python and trained both from 248 days of original data and wavelet denoised data to compare the two, then a 10-day prediction was output by the model using each respective dataset. The network was constructed using TensorFlow's Keras API with the following layers:

1. Conv1D layer with 64 filters and a kernel size of 2. This convolutional layer helps to extract features from the sequence data, which can be beneficial in identifying trends or patterns over time.
2. Two LSTM layers with 60 units each, designed to capture long-term dependencies in sequence data. The first LSTM returns sequences to provide a three-dimensional array as input to the next LSTM layer.

3. Two Dense layers with 30 and 4 units respectively, using ReLU activation. These are fully connected layers that help in learning non-linear combinations of the features.
4. A final Dense layer outputs a single value, representing the model's prediction for the next time step.

In total, this model is comprised of 6 layers and relies on the Adam optimizer as the loss function in order to minimize the difference between the predicted and actual values in training. The Adam optimizer was chosen for its adaptive learning rate, which is very effective in handling the noisy gradients of financial time series data.

The dataset was segmented into 80% training, 10% validation, and 10% testing sets, in order to create an effective learning curve. The table (Table 1) below features a comparison of RMSE values for 1- 1-day-historical, 5-day- historical, and 10-day-historical values for AMZN Stock, both with and without the Wavelet Denoising.

Below, the training of the LSTM model using original and denoised data can be seen in Fig. 4 and Fig.5, alongside the respective original and denoised predictions in Fig. 6 and Fig. 7:

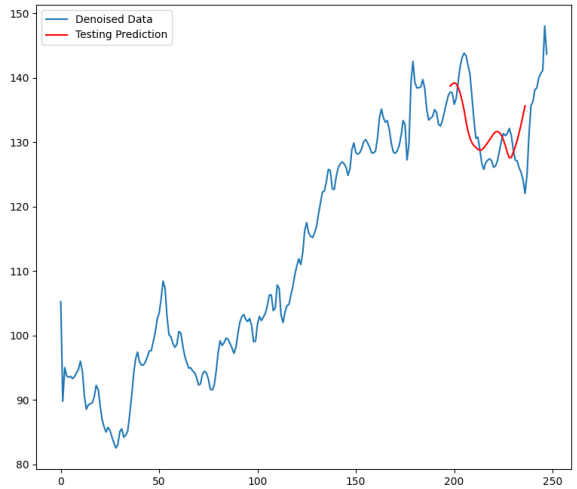

Fig. 4: LSTM training with original data

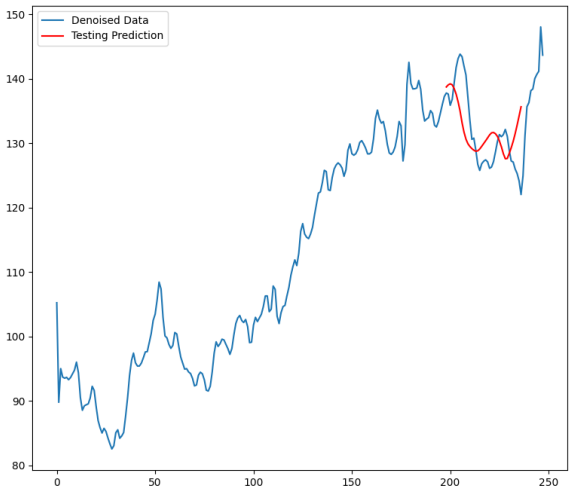

Fig. 5: LSTM training with denoised data

Lastly, the 10-day predictions using each data set:

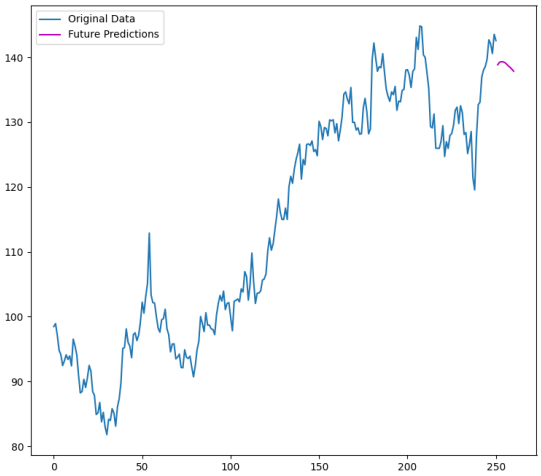

Fig. 6: LSTM prediction with original data

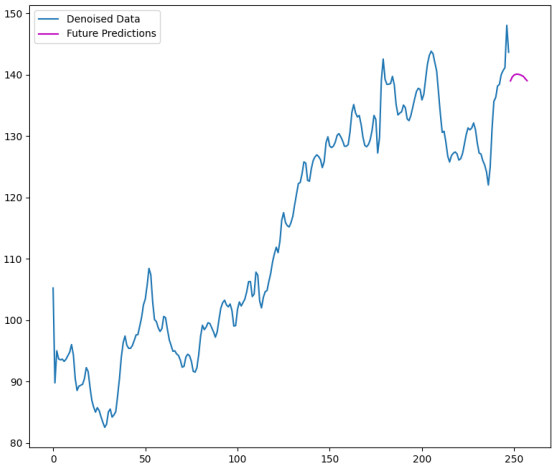

Fig. 7: LSTM prediction with denoised data

Table 1: LSTM comparison of Predicted Stock Values

Day	Original Value	Denoised Value
1	138.34	138.35
2	138.73	138.90
3	138.90	139.16
4	138.95	139.24
5	138.89	139.22
6	138.65	139.11
7	138.37	138.93
8	138.16	138.71
9	137.82	138.26
10	138.47	137.86
RMSE Values		
Original	6.27	
Denoised	5.94	

As we can see in Table 1, our LSTM model forecasts AMZN prices staying the same with a minor downward trend over the next 10 days. This would suggest to investors a good strategy would be to hold over the next coming days. Additionally, we can also see our model had a lower RMSE value when trained with denoised data vs original data, which suggests that our wavelet denoising had a positive impact on the result.

4.3 Recurrent Neural Network (RNN)

For the RNN model, utilizing Python a 2-layer model was created and trained both with original and denoised data to compare the results. The model is comprised of a SimpleRNN layer and a Dense layer which utilizes a ReLU activation function in order to mitigate the vanishing gradient problem which occurs when training neural networks. Likewise, the 'adam' optimizer is used once again. The figures below feature a comparison of RMSE values and predicted stock price between the original and denoised data in Fig. 8 and Fig. 9:

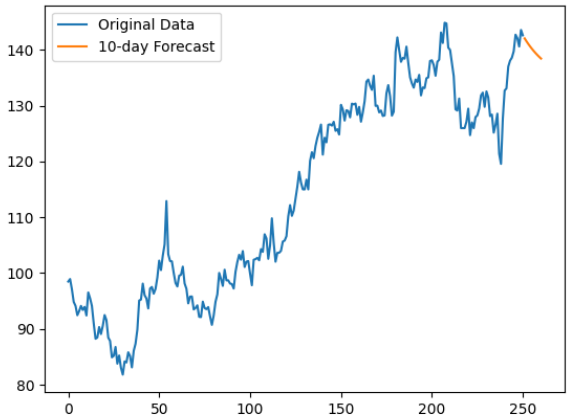

Fig. 8: RNN prediction with original data

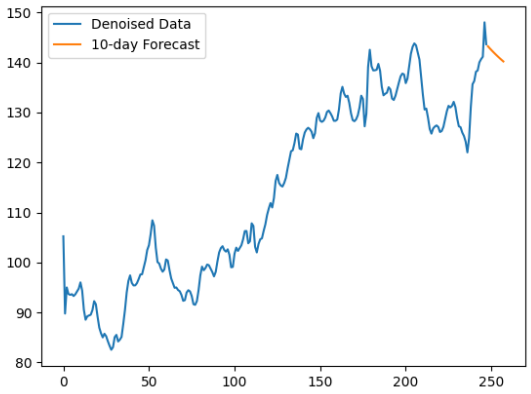

Fig. 9: RNN prediction with denoised data

Table 2: RNN comparison of Predicted Values

Day	Original Value	Denoised Value
1	142.04	142.87
2	141.53	142.51
3	141.05	142.15
4	140.60	141.81
5	140.19	141.47
6	139.79	141.15
7	139.42	140.83
8	139.07	140.53
9	138.74	140.23
10	138.43	138.43
RMSE Values		
Original	.0389	
Denoised	.0297	

As we can see in Table 2, our RNN model likewise forecasts AMZN prices staying the same with a subtle downward trend over the next 10 days. This would suggest to investors a good strategy would be to hold over the next coming days, or potentially sell before a price decrease occurs. Additionally, we can also see our model had a lower RMSE value when trained with denoised data vs original data, which suggests that our wavelet denoising had a positive impact on the result.

4.4 Support Vector Regression (SVR)

The SVR model is likewise trained once with denoised data and once with original data for comparison. The model employs the RBF kernel, which is defined as such:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \qquad (40)$$

This kernel was chosen due to its ability to handle nonlinear relationships.

The hyperparameters in this model were carefully chosen to balance model complexity with generalization capability, thereby minimizing overfitting, a

critical consideration for financial time-series predictions. Below the stock prediction is given both with the original and denoised data along with their RMSE values in Fig. 10 and Fig. 11:

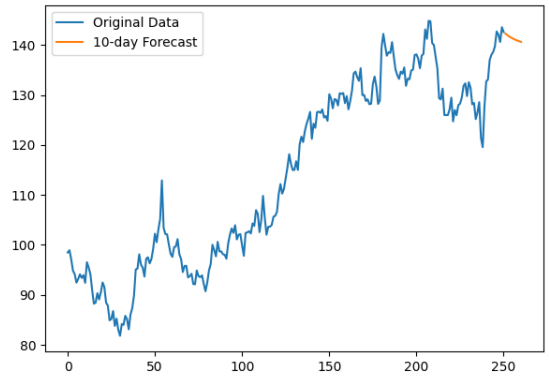

Fig. 10: SVR prediction with original data

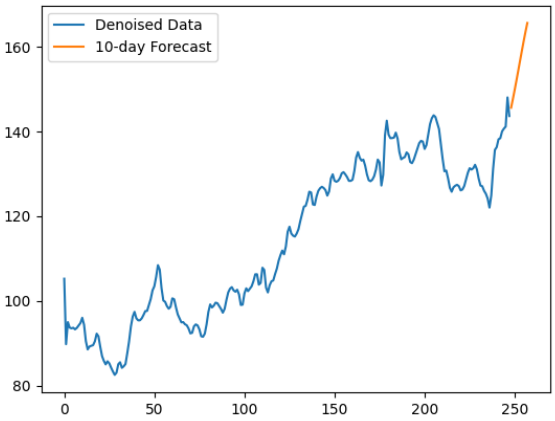

Fig. 11: SVR prediction with denoised data

Table 3: SVR comparison of Predicted Stock Values

Day	Original Value	Denoised Value
1	142.00	145.62
2	141.76	149.87
3	141.53	152.17
4	141.34	154.45
5	141.16	156.79
6	141.00	159.12
7	140.86	161.40
8	140.73	163.58
9	140.62	165.63
10	140.59	165.70
RMSE Values		
Original	3.87	
Denoised	2.54	

As we can see in Table 3, our SVR model forecasts a drastic increase in AMZN prices over the next 10 days. This would suggest to investors a good strategy would be to buy before the next 10 days occur, to capitalize on the "bullish" behavior of the market to come. Additionally, we can also see our model had a lower RMSE value when trained with de-noised data vs original data, which suggests that our wavelet de-noising had a positive impact on the result.

4.5 Autoregressive Integrated Moving Average (ARIMA)

For the ARIMA model, the model was created in Python with the model's parameters (p,d,q) being chosen from the Partial Auto-correlation Function (PACF) as seen in Figure 12 and Auto-correlation Function (ACF) as seen in Figure 13 based on the prior stock data seen below:

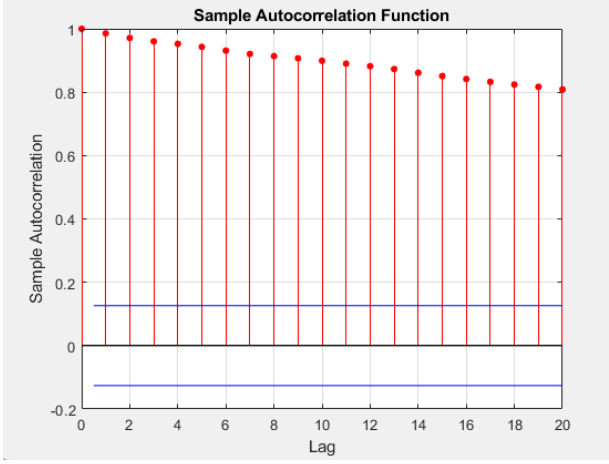

Fig. 12: Auto-correlation Function

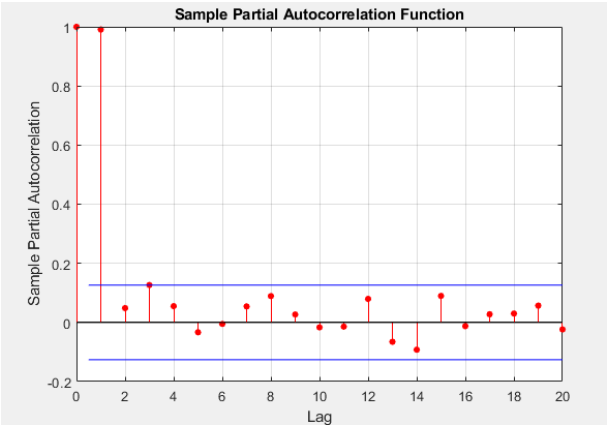

Fig. 13: Partial Auto-correlation Function

Lastly, a day 10 forecast was made using both de-noised (Figure 15) and original stock data (Figure 14):

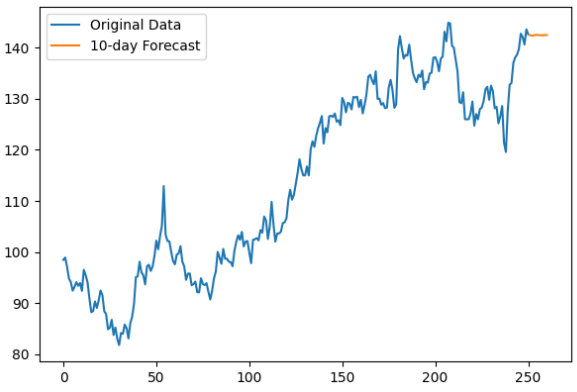

Fig. 14: ARIMA prediction with original data

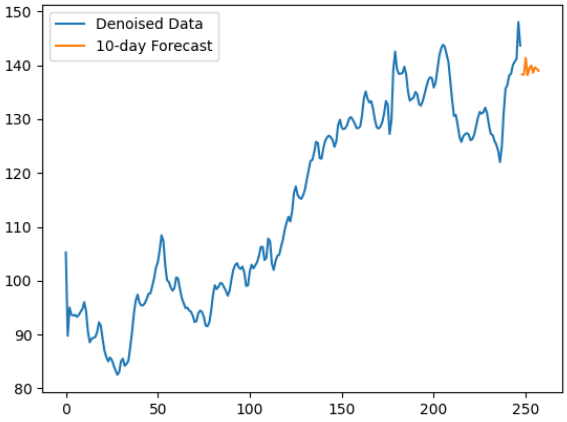

Fig. 15: ARIMA prediction with denoised data

Table 4: ARIMA comparison of Predicted Stock Values

Day	Original Data	Denoised Data
1	142.32	139.30
2	142.39	141.37
3	142.47	138.17
4	142.48	139.49
5	142.42	139.97
6	142.39	138.63
7	142.42	139.62
8	142.44	139.34
9	142.45	138.98
10	142.46	139.04
RMSE Values		
Original	4.35	
Denoised	4.02	

As we can see in Table 4, our ARIMA model forecasts AMZN prices staying the same over the next 10 days with minor volatility. This would suggest to investors a good strategy would be to hold over the next coming days. Additionally, we can also see our model had a lower RMSE value when trained with denoised data vs original data, which suggests that our wavelet denoising had a positive impact on the result.

4.6 Quantum Hybrid Classical LSTM Method

For the final model in our ensemble, an LSTM model that is equipped with a quantum layer via the use of the Pinneylane package in Python was deployed and was trained with our wavelet denoised stock price data to likewise give a 10-day stock prediction. Using Pinneylane, the quantum layer simulates 4 qubits (the equivalent of a bit in classical computing). Angle embedding within the quantum circuit was used to map classical data (closed stock prices) to a quantum state, by which an entangle layer was used within the circuit to simulate quantum entanglement. The quantum circuit's output was then fed back into the classical neural network, specifically an LSTM model which contains a convolutional layer, 2 LSTM layers, a dropout layer, and 3 dense layers (9 layers in total)

The goal of this model was to leverage the benefits of quantum computing, with the use of wavelets, to process data in ways that might be more efficient or effective than classical-only approaches. Quantum computing can potentially identify patterns in data that classical algorithms might miss since it has additional tools that classical methods do not such as superposition and entanglement, [9]. For our purposes, the qubits we are utilizing can exist in multiple states simultaneously and can entangle with each other, allowing our model to not only analyze data sets more efficiently but also evaluate numerous future possibilities for our forecast which classical computing could not, [10]. These benefits allow quantum methods to be an ideal candidate as a preprocessing step.

The training of the model using wavelet de-noised data is as seen below in Figure 16:

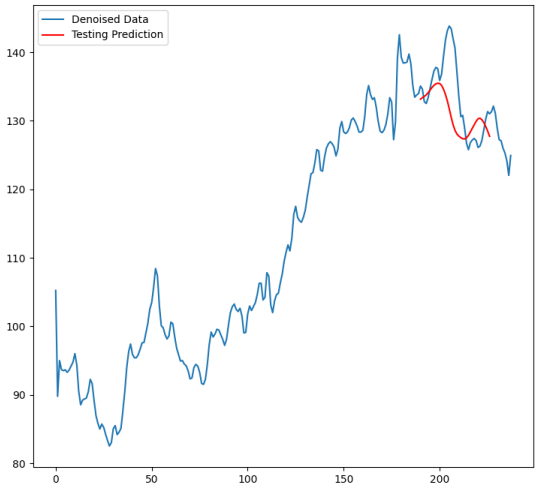

Fig. 16: Hybrid Quantum Classical LSTM model training

Lastly, the final prediction of our ensemble is given in Figure 17:

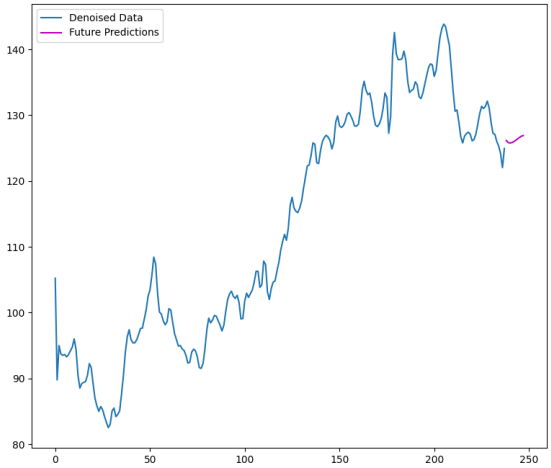

Fig. 17: Hybrid Quantum Classical LSTM model prediction

Table 5: Predicted stock values and RMSE score for the Hybrid Quantum Classical Model

Day	Predicted Value
1	126.15
2	125.83
3	125.76
4	125.84
5	125.98
6	126.19
7	126.40
8	126.61
9	126.80
10	126.90
RMSE: 12.82	

As we can see from Table 5, the hybrid quantum-classical model was in line with our other models as it suggested AMZN stock price would remain unchanged with little movement. These results paired with our other models, in addition to the low RMSE value, show that this is a quality forecast and gives a glimpse into the power of quantum computing in financial forecasts.

5 Conclusion

The results from our newly developed ensemble model demonstrate the power of wavelet, machine learning, and quantum computing in several ways. Firstly, the different models show different lenses by which the future of AMZN stock can be interpreted. The majority of the models predict that the price of AMZN stock over the next 10 days will either stay the same or mildly decrease, suggesting to a potential investor that holding may be a good financial move. We can also see across the board that using the de-noised data aided in the training of our models, as across the board RMSE values were lower when models utilized de-noised data vs raw data, demonstrating the power of the wavelet de-noising algorithm.

Additionally, the hybrid quantum-classical model was able to capture trends from prior data well and give a prediction that was in line with the other models. This demonstrates the potential for such hybrid quantum-classical models and shows that there is potential for them in the field. We believe that further research into quantum computing can prove to be fruitful as a new emerging method to make financial forecasts.

Overall, the results of this research show that wavelet de-noising is a useful tool in financial forecasts, being able to make our predictions more reliable and efficiently. While training each model, we can see an overall decrease in RMSE values when training with wavelet denoised data as opposed to original data. This means that not only did each model capture the history and trends of the data better, but it also gave us more confidence in the quality of the prediction leading to better financial decisions. This clearly indicates that wavelets show great promise in financial forecasting and contribute to making a more sound financial decision.

Additionally, our ensemble forecast clearly shows the usefulness of such models when making financial decisions. Employing multiple methods for stock forecasting helps to eliminate the aforementioned “black box” issue. Utilizing multiple techniques yields a more diverse range of results which can aid in making financial decisions. In our case, when each model yields similar results about the future of a stock price, it can give investors more confidence in their decision.

In the future, we can extend our current methods by combining them into a hybrid model, as opposed to separate models. Additionally, there is a large amount of potential for tweaks to the quantum model such as using other packages, adjusting the number of qubits, etc. We believe the results of this research can also be improved by better selecting a wide variety of stocks to train our models vs using only one.

Acknowledgment:

- Dr. Xiaodi Wang, PhD, Western Connecticut State University for his time and support
- The WCSU Department of Mathematics for making this research possible
- The WCSU Student Government Association for their constant support for this research
- The WCSU Foundation for their contributions to this research

Disclosure:

During the preparation of this work, the author(s) used generative AI to assist in generating initial drafts and refining the language of sections related to the explanation of machine learning techniques and stock forecasting methods. After using this tool, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

References:

[1] M. Goldani, "Comparative analysis on forecasting methods and how to choose a suitable one: case study in financial time series," *Journal of Mathematics and Modeling in Finance*, pp. 35–59, 2023.

[2] S. Selvaraj, N. Vijayalakshmi, S. S. Kumar, and G. D. Kumar, "Maximizing Profit Prediction: Forecasting Future Trends with LSTM Algorithm and compared with Loss function and Mean error code using Python," *Ushus Journal of Business Management*, vol. 22, no. 4, pp. 15–28, 2023.

[3] A. H. Rahimyar, H. Q. Nguyen, and X. Wang, "Stock forecasting using M-band wavelet-based SVR and RNN-LSTMs models," in *2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, 2019, pp. 234–240.

[4] V. H. Shah, "Machine learning techniques for stock prediction," *Foundations of Machine Learning| Spring*, vol. 1, no. 1, pp. 6–12, 2007.

[5] Y. Yang and J. Wang, "Forecasting wavelet neural hybrid network with financial ensemble empirical mode decomposition and MCID evaluation," *Expert Systems with Applications*, vol. 166, pp. 114097, 2021.

[6] V. Novykov, C. Bilson, A. Gepp, G. Harris, and B. J. Vanstone, "Deep learning applications in investment portfolio management: a systematic literature review," *Journal of Accounting Literature*, 2023.

[7] J. Fattah, L. Ezzine, Z. Aman, H. El Moussami, and A. Lachhab, "Forecasting of demand using ARIMA model," *International Journal of Engineering Business Management*, vol. 10, pp. 1847979018808673, 2018.

[8] L. Rubio, A. Palacio Pinedo, A. Mejía Castaño, and F. Ramos, "Forecasting volatility by using wavelet transform, ARIMA and GARCH models," *Eurasian Economic Review*, pp. 1–28, 2023.

[9] E. Paquet and F. Soleymani, "QuantumLeap: Hybrid quantum neural network for financial predictions," *Expert Systems with Applications*, vol. 195, pp. 116583, 2022.

[10] S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang, "Quantum long short-term memory," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 8622–8626.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

- Peter Bigica carried out all experiments and modeling
- Xiaodi Wang was responsible for advising and contributing the wavelet filter banks.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

- The Western Connecticut State University Student Government Association
- The Western Connecticut State University Foundation.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.