The erroneous use of the Haar wavelet transform for stock price prediction

Thesis · March 2017

DOI: 10.13140/RG.2.2.29191.60323

1 author:

Lodewic van Twillert
University of Amsterdam

1 PUBLICATION 0 CITATIONS

SEE PROFILE

Universiteit van Amsterdam

Amsterdam School of Economics

The erroneous use of the Haar wavelet transform for stock price prediction.

Lodewic van Twillert

Supervised by prof. dr. Marcel Worring

H.T.L. van Twillert
10216529

March 17, 2016

Abstract

Hsieh et al. (2011) reported profitable prediction results on the Taiwenese stock market using a Recurrent Neural Network trained using the Artificial Bee Colony algorithm, the ABC-RNN model. Prior to training the model, the discrete shift-variant Haar Wavelet Transformation, or HWT, is applied to the input features and target stock prices for noise reduction. This research presents a critical analysis and replication of the well-cited research paper by Hsieh et al. based on the suspicion that the HWT was used erroneously. Wavelet transformations can be used in time-series analysis to reduce the noise of a signal, or stock prices specifically, but we show that using the Haar wavelet for time series forecasting invalidates the prediction results. Our research criticizes the use of the Haar wavelet by showing that it is both impossible to use in real-time and the sole reason for the reportedly profitable, yet false, 'predictions' shown by Hsieh et al. To further support our claims we show that the proposed ABC-RNN model can even be used to successfully make a profit on generated noise, treated as stock prices. The faulty use of the HWT was potentially reproduced in more recent research, which is left to be reviewed in future research.

Statement of Originality

Contents

1 Introduction **4**

2 Relevant work **5**

3 Haar wavelet **7**
3.1 Haar wavelet decomposition 7
3.2 Wavelet coefficient thresholding 11
3.3 Lack of shift-invariance 12
3.4 Haar wavelet transform for prediction 13
3.5 Alternative wavelet transformation 15

4 Paper description **15**
4.1 Data 16
4.2 Preprocessing 17
4.2.1 Scaling 18
4.2.2 Haar wavelet transformation 19
4.3 Feature selection 21
4.4 RNN 21
4.5 ABC algorithm 22
4.5.1 Initialization phase 23
4.5.2 Onlooker phase 24
4.5.3 Scout phase 25
4.5.4 Employed phase 25
4.5.5 ABC algorithm pseudo code 25
4.6 Evaluation 26
4.6.1 Standard metrics 26
4.6.2 Profit evaluation 27

5 Replication results **28**

6 Noise prediction using the Haar wavelet **34**
6.1 Data generation 34
6.2 Features 35
6.3 Noise prediction results 35

7 Discussion **36**

8 Conclusion **37**

1 Introduction

Predicting stock prices has been a popular subject in economics, econometrics and in machine learning. Researchers may be drawn to this area of research for different reasons, be it simply making large profits, understanding market dynamics, testing novel pattern recognitions models on challenging time series data or to disprove the Efficient Market hypothesis. Reviewing research on stock price prediction comes with an added difficulty since it is a field that suffers from a 'file drawer bias' according to Timmerman and Granger [?]. Researchers who find insignificant effects suffer from an increased difficulty to publish empirical results. On the other hand, researchers who genuinely believe to have found a successful prediction method intuitively have less incentive to publish their methods and may rather sell their prediction strategy to financial institutions.

The Efficient Market Hypothesis, or EMH, states that stock markets are so efficient that any currently available information is reflected in the stock price. It then follows that future stock prices are not predictable using the currently available information. Fama [?] and [?] entertain the thought that the strength of the EMH depends on the considered information set, where using only public market data leads to the EMH in its weakest form. Contradictory to the EMH, there has been plenty of research where profitable prediction models are reported.

This research will focus on the research by Hsieh et al. [?], where a consistently profitable prediction strategy is reported on the Taiwenese stock market, thus presenting contradicting evidence against the EMH. The input data for their proposed model is based only on public market data, contradicting the EMH in its weakest form. While the Taiwense TWII (or TAIEX) stock price lies between 8500 and 4500 Taiwenese dollars during the testing period, a profit of 4299 TWD, including transaction costs, is shown over only 14 trading months in [?]. These successful prediction results are achieved by using a shallow recurrent neural network model trained by the Artificial Bee Colony (ABC) algorithm. While the ABC algorithm is an interesting and valid technique to use, the key modeling step that leads to the reported profits is the Haar wavelet transformation.

The Haar wavelet transform is a discrete shift-variant wavelet transformation that is oftentimes applied in image compression or time-series analysis, excluding prediction. When applied to time series as in [?], the goal is to remove the noise in the data, thus removing the part of the data that is unpredictable by definition. It was already noted in 1997 by Aussem and Murtagh [?] and by Zhang and Dong in 2001 [?], who did similar time-series prediction research, that shift-variant wavelet transformations may lead to issues for use in forecasting. An alternative shift invariant wavelet transform was applied instead. Despite this earlier research, and the warning therein, Hsieh et al. [?] used the Haar wavelet for stock prediction. Because of the successful results presented by Hsieh et al. their published research influenced many other researchers, gauging by the number of citations, some of whom implemented similar methods using the Haar wavelet transformation. We suspect that the use of the Haar wavelet by Hsieh et al. led to effects on the final prediction results that were not discussed.

The goal of this research is to analyze the use of the Haar wavelet transformation in data preprocessing for time-series prediction by [?]. To achieve this goal the Haar wavelet transform is illustrated in the scope of time-series forecasting, while further technical details on wavelet theory may be found in relevant earlier research. In order to confirm any necessarily made assumptions, the research by [?] is replicated and implementation details are given to rule out implementation and interpretation errors.

The remainder of this research starts with a discussion of relevant work, consisting of previous research that made use of discrete wavelet transformations for time series prediction or analysis and more recent work that has cited the work by Hsieh et al. [?]. Then we present some details about the Haar wavelet transform and its properties in an illustratory manner, leaving technical details to be read in cited research. After the Haar wavelet has been explained in the scope of time series we give a detailed description of the paper we replicated [?] to allow the reader to verify the correctness of our interpretations and assumptions needed for replication, after which we present the replication results. Based on the replication results we also apply the ABC-RNN model proposed by Hsieh et al. to generated noise to further support our claims on how the Haar wavelet influences prediction results. Lastly, we discuss our findings in the discussion, criticizing previous research, mentioning our own potential shortcomings and proposing areas of future research before we reach our final conclusion.

2 Relevant work

At the core of this research lies the Haar wavelet transformation, the simplest wavelet transformation also known as the Daubechies 1, Db1, wavelet transformation. The Haar wavelet transform has already been around since 1910, when it was first introduced by Alfred Haar [?]. An overview by Stankovic from 2003 [?] contains detailed information about multiple definitions of the Haar wavelet, its generalizations and its applications at that time. It must be noted that the most prominent use of the Haar wavelet is in pattern recognition and image compression. However, while the Haar wavelet is commonly used in time series analysis, none of the mentioned applications in [?] involve predicting the following value in a sequence, i.e. time-series forecasting.

Chan [?] and Struzik and Siebes [?] used the Haar wavelet transform to compare time series similarity. Using existing time series, the data can be decomposed into multiple resolutions and in turn the correlation between these decomposed sequences are used as a proxy for the similarity of the original time series. Interestingly, Struzik and Siebes explicitly noted that a severe disadvantage of the Haar wavelet transform is the lack of translation invariance. In other words, when the number of observations in the input signal changes then all the coefficients of the wavelet transform need to be recalculated, and in turn this affects the entire transformed signal. Practically this means that if our data consists of all previous stock prices, then 'tomorrow' we will acquire a new data point and all wavelet coefficients will need to be recalculated.

Hsieh et al. [?] mention two papers that they say 'have shown to be particularly useful in analyzing, modeling and predicting the behavior of financial instruments'. However, the first paper that is cited is an overview of wavelets in economic analysis by Ramsey [?] in which only two 'major papers' are mentioned where wavelets were used for financial forecasting. In turn, the first paper mentioned by [?] is unpublished research by Arino in 1998 [?] that is unavailable for further review. The second paper mentioned in the overview by Ramsey is research by Aussem and Murtagh from 1997 [?]. Aussem and Murtagh use a dynamic recurrent neural network, DRNN, with wavelet-transformed input data. It is specifically noted that in order to use wavelets for prediction special care must be taken when selecting the data for the wavelet transform. To be precise, they state that the when the goal is to predict an observation at time $t + 1$ then only information up to and including time t may be used. As such, they propose to use a shift-invariant wavelet transformation that may be updated at every time step to ensure no future information is used, which is realized by the à trous algorithm.

Zhang and Dong [?] and Zhang et al. [?] also used the à trous algorithm to achieve a shift-invariant, or stationary, discrete wavelet transform alternative because of potential issues with discrete wavelets such as the Haar wavelet. They state that Discrete Wavelet Transformation, DWT, has many advantages for compressing signals. But, in time series analysis the DWT based statistical estimators are often affected by the choice of origin because of the lack of translation-invariance, or shift-invariance. While it is noted that the Haar wavelet transformation is potentially problematic for time series prediction, it is unclear what the exact consequences are.

The publication by Hsieh et al. [?] has received over 160 citations, which is quite a lot in this field of research and highlights its relevance. While many of the citations are based on the use of the ABC algorithm, or simply reference a succesful stock prediction model, there are a handful of researchers that have expanded on the model proposed in [?] and use the same Haar wavelet transformation methods.

In one relatively recent article from 2016 that is based on the research by Hsieh et al. [?], Khuat et al. [?] apply the Haar wavelet transform with thresholding to the input data of a one-layer neural network in order to reduce the noise of the input signals. They then aim to predict the stock prices of popular stocks. Naturally they compare predictive power of their model by looking at the prediction error metrics with- and without using the wavelet transform. As to be expected, results improve consistently after applying the transformation. They motivated their choice of the Haar wavelet by mentioning how inexpensive the computational operation is but do not consider the need for alternatives as those used in the articles mentioned above [?, ?, ?]. It is noted in [?] that the Haar wavelet works on time series whose size is a power of two (e.g. 32, 64, 128...) yet Khuat et al. do not this mention this requirement as a potential problem for real-time stock prediction.

Lahmiri [?] used the discrete shift-variant Daubechies3 wavelet to predict daily closing prices of the S&P500 in 2012. However, instead of using wavelet transformations for the purpose of noise reduction, it was used to gather the wavelet coefficients at different decomposition levels directly as input features to a prediction model. Khandelwal et al. [?] proposed a similar model based on the wavelet coefficients without noise reduction to predict forex exchange rates. There exists more similar research that cited [?] but Hsieh et al. seem to be the first to publish a stock prediction model in 2010 utilizing the unadjusted Haar wavelet transform, thus it is likely that they were the ones that originally inspired more recent research mentioned above and therefore we focus our efforts on analyzing their proposed methods.

3 Haar wavelet

The core technique that we look at is the Haar wavelet transformation, applied by Hsieh et al. [?] to reduce the noise in all input and output sequences. Before moving on to replicating the mentioned research, it is vital to understand how the Haar wavelet transformation can be used, and how it can lead to issues when used for time-series prediction.

The Haar Wavelet Transform, or HWT, can be used to analyze signals that are prone to sudden transitions, such as the edges of objects in images, step functions or, arguably, volatile stock prices. On the flip side, transforming smooth signals will rather result in a step-like function. Figure 1 shows the square-shaped nature of the Haar wavelet. From the noisy sinus function a seemingly step-wise and noise-reduced approximation is returned when applying the HWT.

The motivation for using the Haar wavelet transformation is to reduce the noise in a noisy signal such as historical stock prices. We could say that the actual stock price fluctuates around a fundamental stock price, and by using the wavelet transformation we aim to extract the fundamental stock price, see figure 2. The key difference with this type of analysis and prediction is that we can do the analysis in hindsight, meaning that the entire data set is already known at the time of analysis. For prediction, if we aim to predict the price at time t then we should only use all observations up to time $t-1$ to make this prediction.

3.1 Haar wavelet decomposition

The first step when applying the HWT for noise-reduction is the wavelet decomposition. The original signal is split into two series of coefficients, the approximation coefficients and the detail coefficients. The approximation coefficients can be interpreted as localized scaled sums, while the detail coefficients are localized differences. Using the resulting wavelet coefficients we can reconstruct the original signal exactly, using the inverse wavelet transform. In order to reduce the noise of the input signal we use a soft thresholding technique to adjust the detail coefficients, described in the next section. Figures 3 and 4 show the decomposition of the TWII stock prices in 2003 using a three level decomposition and applying soft thresholding to the detail coefficients shown in the bottom right. For technical insight into the Haar wavelet and wavelet theory in general we refer to earlier research [?, ?, ?].

Figure 1: Example of the Haar wavelet applied to a noisy sinus function on the interval [0, 10] with 1000 observations. The red line shows a sinus function with noise added to it. The black line is the Haar wavelet transformation, using universal soft thresholding.

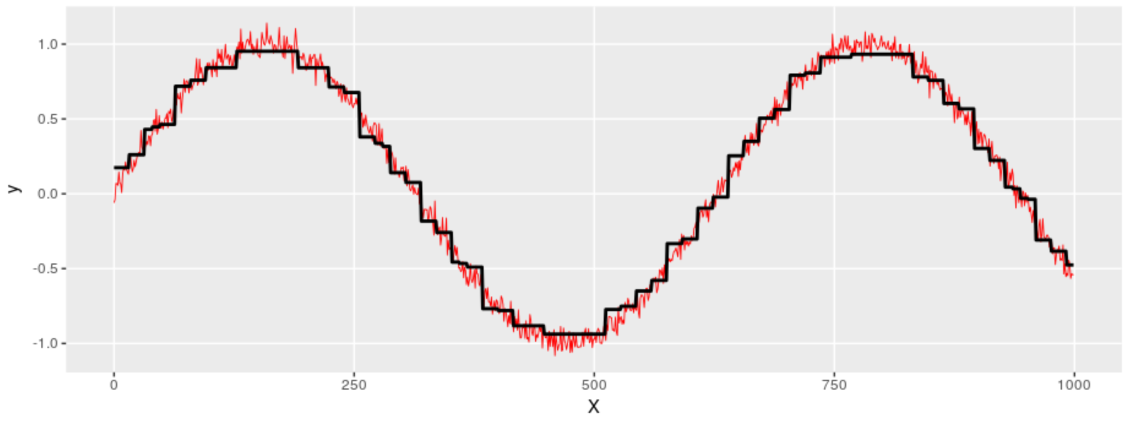


Figure 2: Example of the Haar wavelet applied to stock indices for analysis as shown in WIRED magazine in 2001. The noise is separated from the stock prices in order to compare two US stock indices.

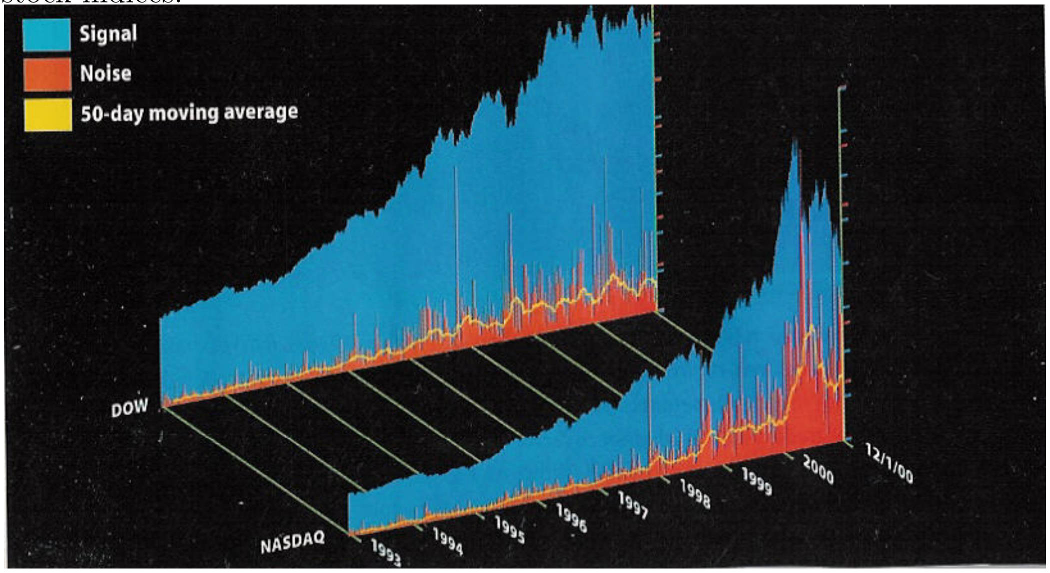


Figure 3: Three level Haar wavelet decomposition of the TWII stock price in 2003. Approximation coefficients on the left, detail coefficients on the right.

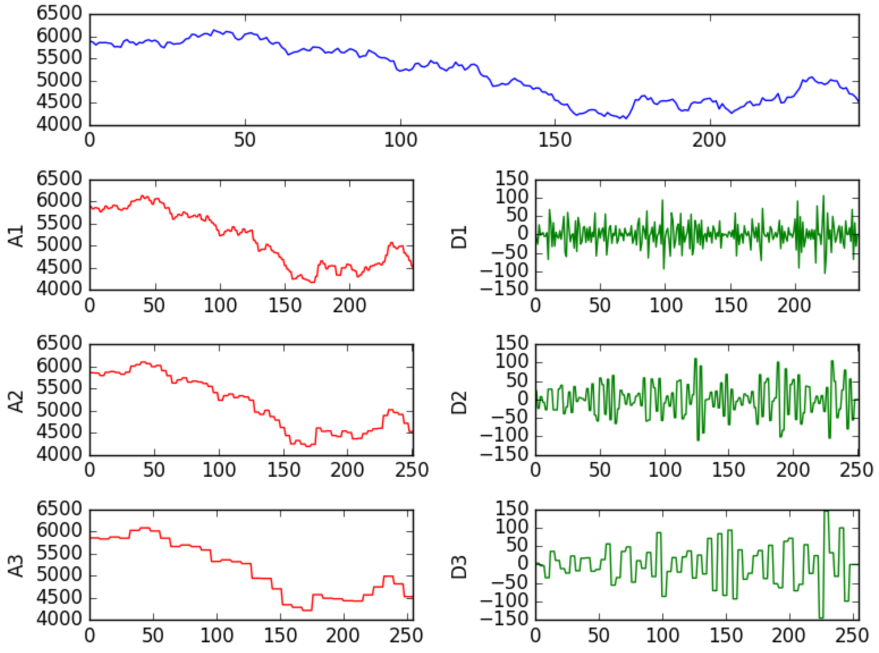
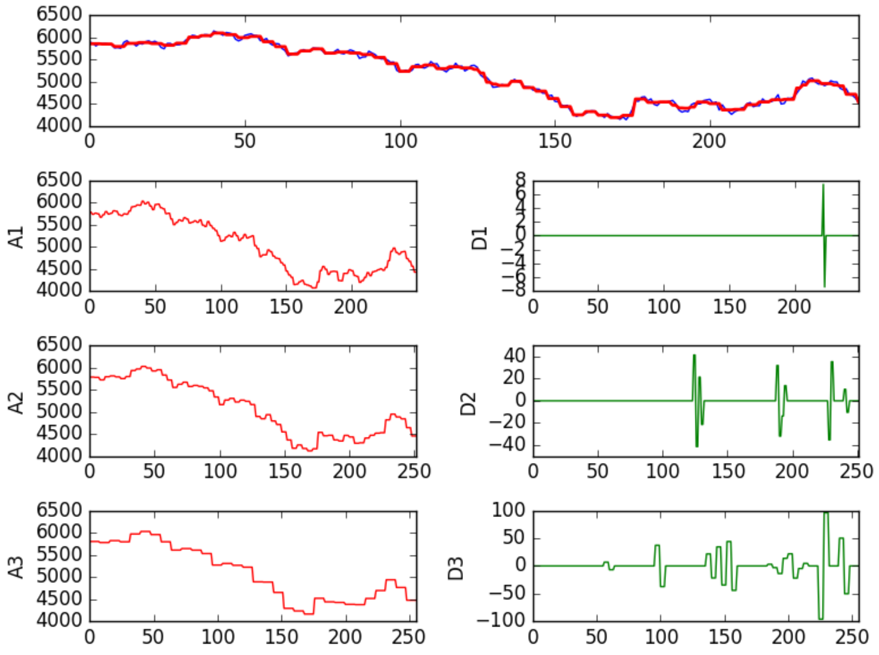


Figure 4: Three level Haar wavelet decomposition and noise-reduced reconstruction of the TWII stock price in 2003 after thresholding. Approximation coefficients on the left, detail coefficients on the right.



To illustrate how the wavelet coefficients are computed we can express the Haar wavelet transformation in matrix form. If we have a sequence of observations, of which the length N is divisible by 2, then the 2×2 Haar matrix can be used to do a step-wise transformation,

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1}$$

First, we transform the input sequence into a sequence of pair-wise vectors,

$$(x_0, x_1, ..., x_N) \rightarrow ((x_0, x_1), (x_2, x_3), ..., (x_{n-1}, x_n))$$

Every two-element pair-wise vector is then multiplied/transformed by H_2 to result in

$$((a_0, d_0), (a_1, d_1), ..., (a_{\frac{n}{2}}, d_{\frac{n}{2}}))$$

where a_i are weighted sums of the vector elements and d_i the weighted differences. These sequences of a_i and d_i are the approximation and detail coefficients respectively mentioned before. A new sequence is composed of all approximation coefficients a_i, and note that this sequence $(a_0, a_1, ..., a_{\frac{n}{2}})$ will be half the length of the original signal. We recursively continue to transform all resulting sequences of approximation coefficients analogously to how we transformed x until the final sequence a is of uneven length. Ideally, and as we shall assume throughout this section, any input signal's length is some power of 2, so that $N = 2^J$ with J any non-negative integer. Then, at the final decomposition level we are left with only one approximation coefficient a and $N - 1$ detail coefficients d.

If the input data has a length $N = 2^J$, for some positive integer J, then we can use a fast Haar transform matrix H_J that computes multiple iterations of H_2 over the data at once so that there is no need for any recursion. For example, the matrix H_4 can be used if the input vector x has 4 elements,

$$H_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix} \tag{2}$$

The resulting vector $c = H_4 x$ contains all the wavelet coefficients such that $c = [a_0, d_1, d_2, d_3]'$. The transformation steps may be generalized to any sequence of even length, and padding of the data is needed for uneven numbers of observations. Note that the first element of c is simply a scaled sum of all observations. In fact,if the coefficients of input signal x with length $N = 2^J$ are defined as $c = H_J x$, then the first element of c is always a scaled sum of all elements. This means that in any case, the first element in the wavelet coefficient vector is always dependent on the entire input signal. Why this dependence is relevant has to do with the problem of shift-variant wavelet transformations outlined in section 3.3.

The inverse wavelet transform is defined by the inverse of H_J, so that $x = H_J^{-1} c$. Since the Haar matrix H_J is an orthogonal matrix, the inverse matrix H_J^{-1} is simply the transposed matrix H_J^T. From the wavelet coefficients we can reconstruct the input signal exactly using H_J^T, but our aim is to filter out the noise of the signal before reconstructing it. In order to reduce the noise of the signal, we must apply thresholding to the wavelet coefficients before the inverse wavelet transform, which will be described in the following section. An illustratory example is presented in section 3.4 making use of the H_4 Haar matrix.

3.2 Wavelet coefficient thresholding

Hsieh et al. [?] mention that the goal of noise reduction through the wavelet transformation is to minimize the RMSE between the signal before and after transformation. However, once the wavelet decomposition coefficients are known we can reconstruct the input signal exactly by the inverse Haar transform matrix. Therefore the RMSE between the signal before and after transformation will equal zero if we simply do not apply any noise reduction, thus minimizing the RMSE. Clearly the goal of reducing the noise of the input signal cannot be to minimize the RMSE, and so we refer to universal noise reduction techniques.

A general approach to extract the underlying signal from the noisy input data is to threshold the Haar wavelet coefficients described in the previous section. Specifically, soft thresholding is proposed by Donoho and Johnstone (1995) [?] by the following thresholding rule given a threshold value t and a wavelet coefficient y_i,

$$\hat{c_i} = sign(c_i) * \max(0, c_i - t) \tag{3}$$

Here c is the sequence of all Haar wavelet coefficients and t is the threshold value that must be chosen beforehand. This is called soft thresholding because we shrink the absolute value of all the coefficients, to a minimum of 0. Hard thresholding would simply set all coefficients that are below the threshold to 0, and leave other coefficients untouched.

The threshold value t in equation 3 must be chosen correctly, if it is chosen too large then we may lose all detailed information that is not necessarily noise and if it is too low then the input signal may still contain noise. The ultimate goal is to replicate the model in [?] but since thresholding details remain unknown we base our implementation assumptions on the universal thresholding value proposed by Donoho and Johnstone [?]. The universal soft thresholding value defined is as follows,

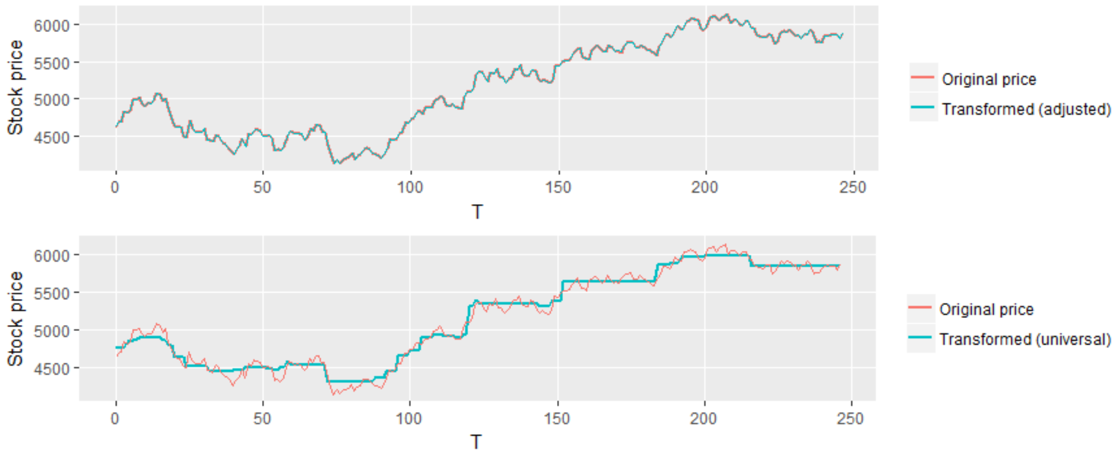
$$t_{universal} = \sqrt{(2 \log N)}\hat{\sigma} \tag{4}$$

Here N is the number of data points and $\hat{\sigma}$ is typically a scaled median absolute deviation (MAD) of the empirical wavelet coefficients. One proposed approach to estimating $\hat{\sigma}$ is to estimate the noise level σ as the MAD of the wavelet coefficients at the maximum decomposition level, divided by 0.6745. (Page 446 in Donoho and Johnstone [?].) For replication we have used the soft thresholding rule with a scaled universal thresholding value $t_{adjusted}$ before reconstructing the true signal using the inverse Haar transformation. Specifically, the final threshold value was chosen to be only one-fifth of the universal thresholding value,

$$t_{adjusted} = \sqrt{(2 \log N)}\hat{\sigma}/5 \tag{5}$$

This value had to be adjusted for the replication in section 5 since in [?] the wavelet transformation was applied twice, which is non-standard and serves no extra purpose over applying it once and choosing the threshold value correctly. Since all coefficients are shrunk after soft thresholding we require a smaller threshold value if we are to apply the noise reduction twice. Using the threshold value from equation 4, and doing the transformation twice, leads to a sequence of local averages where we lost too much of the signal, as shown in the bottom graph in figure 5. The adjusted threshold value leads to some noise reduction in the transformed stock prices, but keeps enough information to use for daily stock prediction, resulting in the data shown in the top graph of figure 5. A more detailed view of the top graph in figure 5 is presented in section 4.2.2, where the data is discussed, since the differences between the original and transformed data is too small to see.

Figure 5: Haar wavelet transformation of the Taiwenese TWII stock prices in 2003 using both the universal threshold and the adjusted threshold. The universal threshold value leads to too much noise reduction.

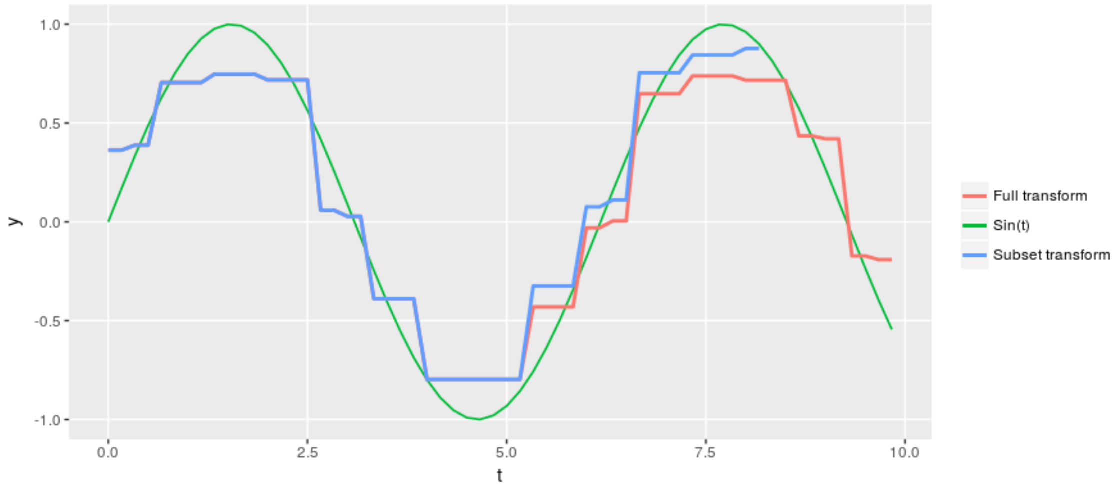


3.3 Lack of shift-invariance

As was mentioned before, the Haar wavelet can be used in time-series analysis when the analyzed signal is completely known. For prediction, we may only use the observations prior to the observation we aim to predict. This means that was we cannot use the Haar wavelet the same way for analysis as for prediction, instead we would intuitively be led to a step-wise wavelet transformation. In the case of daily stock prediction, we should be able to update the wavelet transformation of all past values daily. However, this leads to in-applicability of the Haar wavelet in real-time prediction models since the maximum decomposition level J depends on the length of the time-series to be transformed and, moreover, the input sequence needs to be of even length. This means that the decomposition levels and the thresholding values would change over time, and without any mention of such a non-standard use of the HWT in [?] we assume that the HWT was applied only once for every data subset.

Shift-invariance means that the transformed values do not change when we shift the input signal. The lack of shift-invariance leads to an issue mentioned previously in [**?**,**?**], namely that adding additional observations to the input sequence will affect the entire transformed signal. Figure 6 illustrates what happens if we apply the transformation to a subset of the sequence, compared to applying it to the full sequence. The shown sinus sequence has 60 observations, and the subset chosen for figure 6 is the first 50 observations. The transformations are done using the exact same thresholding value (not depending on all coefficients as in equation 5) and a decomposition level of 5. For illustration we show a simple sinus function and the Haar wavelet transformations based on different subsets of the sequence in figure 6. Since there is a difference between the two transformations, that means that adding observations to the input sequence also affects previous values, a result you'd see with any thresholded shift-variant DWT. By analyzing the noise-reduced transformed signal we may infer information about future value, which is not prediction but rather reverse-engineering of the HWT.

Figure 6: Haar wavelet transformation of the sinus function. The difference between the transform over the full sequence and over a subset of the sequence highlights the issues that arise because of the lack of shift-invariance.



3.4 Haar wavelet transform for prediction

Remember that the Haar wavelet is applied in an attempt to predict stock prices. This means that any features used to predict future prices should not depend on future information, i.e. any feature $f(t)$ used to predict the one-day ahead stock price price $x(t+1)$ may only depend on information up to time t. The problem is that the Haar wavelet must be applied to the whole data set at once, therefore using 'future' values in the time-series to transform the 'current' data. This should only be a problem if the observations beyond time t affect the feature values at times before t, which is exactly what happens when we apply the Haar wavelet with thresholding to an entire sequence. Figure 6 already shows this, but to clarify the arising issues we illustrate how we can infer future values by using the Haar wavelet by means of a simple example.

Suppose that one sequence is $x = [1, 2, 3, 1]'$ then the Haar wavelet coefficients of x are

$$c = H_4 x = \frac{1}{2}[a, d_1, d_2, d_3]'$$

$$c = H_4 x = \frac{1}{2}[7, -1, -\sqrt{2}, 2\sqrt{2}]'$$

To retrieve x from these coefficients we use the inverse of H_4 from equation 2, where $H_4^{-1} = H_4^T$. In order to try to filter noise from x we apply soft threshold from equation 3 to the detail coefficients d_i. Suppose for now that the thresholding value is chosen simply by $t = 0.6$, then

$$\hat{c} = \frac{1}{2}[7, 0, -(\sqrt{2} - 1.2), 2\sqrt{2} - 1.2]'$$

The resulting Haar wavelet transform of x is $\hat{x} \approx [1.67, 1.83, 2.33, 1.17]$.

Now suppose that we have two vectors, $x = [1, 2, 3, 1]'$ and $z = [1, 2, 3, 4]'$, so that the only difference between x and z is the last element. Then $\hat{z}$ can be constructed analogously to $\hat{x}$ and the resulting transformed sequence is $\hat{z} = [1.72, 1.88, 3.12, 3.28]$. It should be seen that $\hat{z}_i > \hat{x}_i$ for all i, highlighting that all transformed values are affected while only the last element in the untransformed sequences differs.

Clearly, the first three elements of x and z do not depend on the last element, either a 1 for x or a 4 for z. Suppose that only the first three elements would be known and the last value is known to be either a 1 or a 4, lets call this y,

$$y = [1, 2, 3, ?]'$$

The last value cannot be predicted with certainty based only on y. However, what if we were to use the first three elements of the transformed y sequence, $\hat{y}$?

$$\hat{y} = [1.72, 1.88, 3.12, ?]'$$

Now, since the first elements between $\hat{x}$ and $\hat{z}$ are different it should immediately be seen that this sequence $\hat{y}$ corresponds to $\hat{z}$ and as such the last value of y must be the same as z, a 4. This is how we can infer the last value of the untransformed y by applying the Haar wavelet transform to all values of y to result in $\hat{y}$.

The glaring problem with the method described above is that in real-time the last element, at time $t = 4$, of this hypothetical sequence y is not known before prediction at time $t = 3$, and so we cannot actually retrieve $\hat{y}$, which is why the HWT cannot be used in the way described above for real-time prediction. Of course, in hindsight using historical data the HWT can still be applied, which is why the HWT is a valid method in analysis of historical data but not for real-time prediction.

We claim that Hsieh et al. [?] have applied the HWT to yearly subsets of their data, including 10 months of training data and the last 2 months of testing data, and used the resulting transformed results as input features. This would mean that the proposed methodology would be flawed, but this erroneous use of the HWT would also be the sole reason for the presented successful predictions in [?]. In order to confirm suspicions and assumptions about the implementation of the HWT we replicate the model in [?], described and discussed in section 4.

3.5 Alternative wavelet transformation

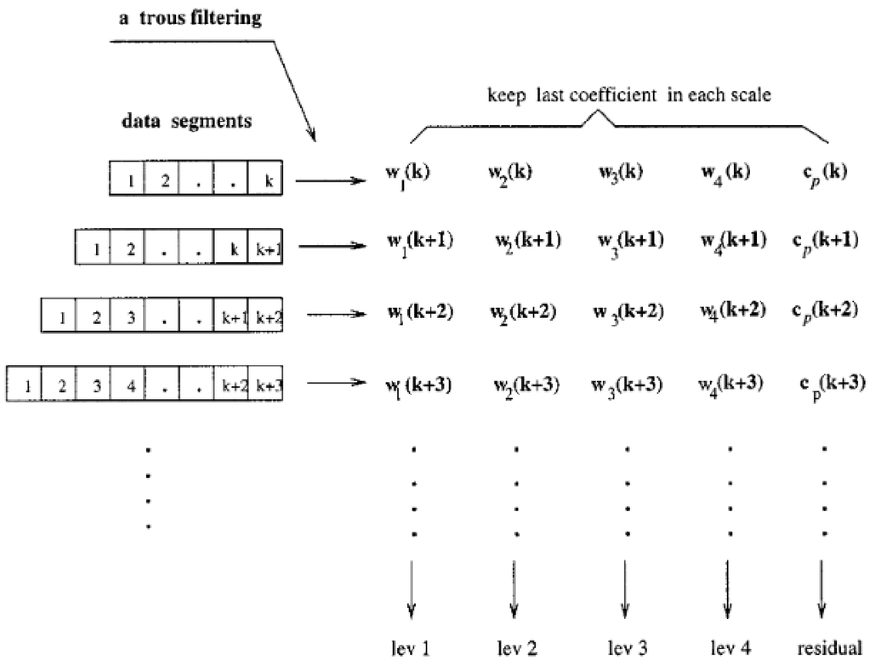
The difference between the wavelet transformations in figure 6 is caused by the lack of translation-, or shift-invariance, a property of most common discrete wavelet transforms, including the Haar wavelet. In turn, this difference represents how we can reverse engineer the HWT after thresholding to infer information of future values. In earlier research by Aussem and Murtagh [?] and Zhang and Dong [?] it is noted that for time-series prediction we require time-invariant wavelet transformations, such as those proposed by Coifman and Donoho [?]. The applied time-invariant wavelet transformation in the previously mentioned papers is realized using the à trous algorithm. Such time-invariant wavelet transformation can be applied over and over again at every time-step as in figure 7 and therefore lend themselves fittingly as part of a real-time signal decomposition. Note that we shall not go deeper into time-invariant wavelet transformation but merely wish to note that such shift-invariant alternatives exist and have been used in the past, while simultaneously noting that the lack of time-invariance of the Haar wavelet poses a major problem for time-series prediction. For further details about such alternative wavelet transformations we refer to [?, ?, ?].

4 Paper description

Hsieh et al. [?] propose a recurrent neural network trained using the Artificial Bee Colony algorithm, called an ABC-RNN, to predict one-day ahead stock prices. In order to achieve a reportedly profitable prediction model the input data, consisting of 14 features common for stock prediction, is first transformed using the HWT in order to extract the underlying market signals and reduce their noise. The input features are then selected using a Step-wise Correlation Regression Selection, SCRS. Eight input features are finally selected to predict one-day ahead transformed stock prices, resulting in reportedly profitable stock price predictions by using a very simple trading strategy. These eight final input features are listed in table 1.

The paper by Hsieh et al. describes all of their methods in a general sense, but assumptions have to be made to replicate the results as some implementation details mostly pertaining to the HWT are missing.

Figure 7: Real-time à trous filtering for time-series decomposition in the prediction model by Zhang and Dong [?]. At every time-step the coefficients of four decomposition levels are calculated, which is only viable because of the time-invariance property of the à trous transform.


a trous filtering
data segments
keep last coefficient in each scale
w1(k) w2(k) w3(k) w4(k) cp(k)
w1(k+1) w2(k+1) w3(k+1) w4(k+1) cp(k+1)
w1(k+2) w2(k+2) w3(k+2) w4(k+2) cp(k+2)
w1(k+3) w2(k+3) w3(k+3) w4(k+3) cp(k+3)
lev 1
lev 2
lev 3
lev 4
residual

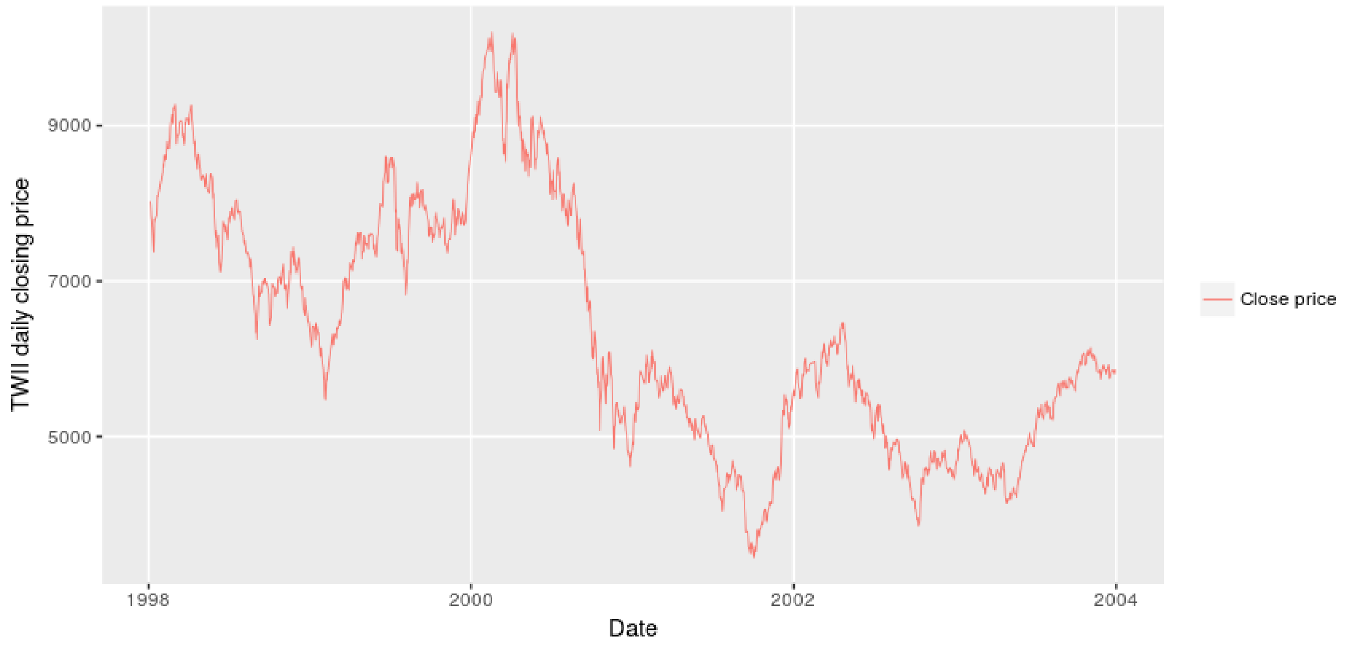
This section is meant to summarize our understanding of the methods by Hsieh et al. and to describe the exact replication methods. Since we claim the invalidity of the results in [?] we find it important to restate the proposed model as to rule out methodological interpretation errors. The steps described in the following sections are the data, preprocessing, scaling, the Haar wavelet transformation, feature selection, the prediction model and the evaluation metrics.

4.1 Data

The data that was used by Hsieh et al. consists of stock index data from 1997 through 2008. The considered stock indices are the Japanese Nikkei, the London FTSE, the US DJIA and the Taiwanese TAIEX/TWII. However, the most impactful evaluation measure that is to be compared in this replication study is the profit evaluation. Other evaluation metrics such as the (R)MSE and MAPE are not as important for stock predictions as they give no information about the accuracy of predicting the direction of the market price. The profit evaluation by Hsieh et al. is only shown for the TAIEX data from 1997 to 2003, therefore this replication study will be focused on the TAIEX data only.

Since stock data is historical data there should be little to no difference between data sources, and since Hsieh et al. do not mention the source of their data the replication data is taken from Yahoo Finance. Yahoo Finance data consists of daily Open, High, Low and Close prices

Figure 8: TWII daily closing prices 1998-2003



along with the date and trading Volume. The aim of the replicated model is to predict the one-day ahead Close price. One issue with the data from Yahoo is that the trading volume data is incomplete. However, none of the considered input features depend on the trading volume. A second issue is that the Yahoo Finance data only goes back to July 1997, therefore we will be missing data for the year 1997 and leave it out.

Figure 8 shows the stock price data we aim to predict. Note that the stock price data represents a noisy time-series, and as such it intuitively makes sense to try to reduce the noise in the data. As per the methods of Hsieh et al. the data is split into yearly subsets. For the years 1997-2003 the first 10 months of the yearly subsets was used as training data, with the last 2 months left as test data. There is no explicit reason for dividing the data into these subsets. Machine learning models like neural networks often require a lot of data so that as many different patterns in the data can be used for training.

A second time period is used from 2002 through 2008 where the yearly subsets consist of 50% training and 50% testing data. However, for these periods the profit evaluation is not shown and therefore not as comparable for replication. The final data used for replication is reduced to TAIEX data from 1998 to 2003, which is enough to replicate the results that highlight the error in the preprocessing methods.

4.2 Preprocessing

The input data, consisting of daily Open, Close, High and Low prices, is first extended by computing financial technical indicators such as the six-day moving average and the demand index. A Haar wavelet transformation is applied twice to all features in an attempt to reduce the noise in the time-series data. After the Haar transformation is applied, the data is scaled into a $[0, 1]$ range to be used as input and target values in the training of a recurrent neural network. Afterwards the data is scaled back to its original scale for the final evaluation metrics and the potential profit is calculated using the untransformed closing prices. However, the exact details of the aforementioned steps are largely unknown. In particular, the scaling of the data and the thresholding of the Haar wavelet coefficients is decided upon through trial-and-error based on educated guesses. This section will describe the details and motivate the assumptions of the data preprocessing steps that were implemented for replication.

As was mentioned earlier, yearly data subsets are taken and split into 10 months of training data and the last 2 months as testing data. There are no trading days in the weekends however, so to ensure that all test periods are of equal length the test period will consist of a static 40 days in the replication, which is roughly two months of trading data. The final profit evaluation on different yearly subsets will yield more comparable results if the testing periods are equal.

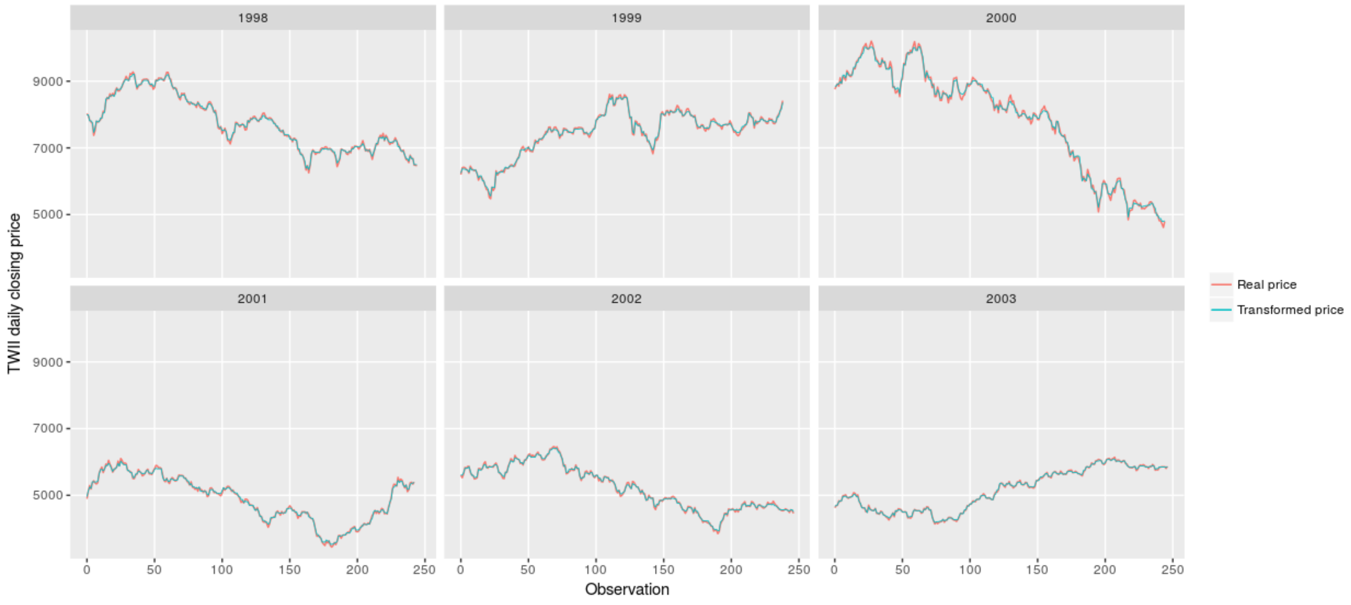
A number of financial technical indicators are computed based on the stock data. Table 1 in [?] provides a description of the considered input features. Some features, such as the moving average, also depend on previous values, so the assumption is made that all features are calculated on the full data set before dividing the data into yearly subsets. Practically this means that the first couple of moving average observations in January depend on data of December in the previous year, as opposed to being missing data.

Subsequent scaling steps rely on the minimum and maximum feature values in the dataset and are therefore applied to each yearly subset individually. The wavelet transformation steps are also applied to the yearly subsets as this assumption leads to replicated results that are most in line with what is presented in [?].

4.2.1 Scaling

The data is scaled differently throughout the reported results by Hsieh et al., and due to the random nature of the ABC-RNN prediction model the prediction results vary wildly depending on the scaling used for the training data even though the profit evaluation is done on unscaled data. To verify the correctness of the replicated implementation we aim to achieve similar values for the evaluation metrics. The MSE of the trained RNN model by Hsieh et al. on the training set reportedly averages to around 0.0002, which implies some down scaling of the data where each feature and target is scaled into a $[0, 1]$ region. After training the model is evaluated again, seemingly on the on the unscaled stock prices, since the RMSE on the test data subsets is reported to lie between 89 and 137. No mention is made of scaling anywhere in [?] so all of the scaling assumptions are confirmed as best as we can by testing which approach best replicates the results.

Figure 9: Yearly subsets of original and transformed daily closing prices of the TWII. The difference between transformed and original prices is hard to see, but this shows how little noise we should actually remove from the data.



For the implemented scaling during the training of the prediction model we assume that every feature sequence (listed in table 1) is denoted f_i with corresponding element $f_i(t)$ at time t. The minimum and maximum value of f_i is denoted f_i^{min} and f_i^{max} respectively. The target value, the one-day ahead stock price, is scaled using this same scaling rule. Features are scaled by the following linear rule,

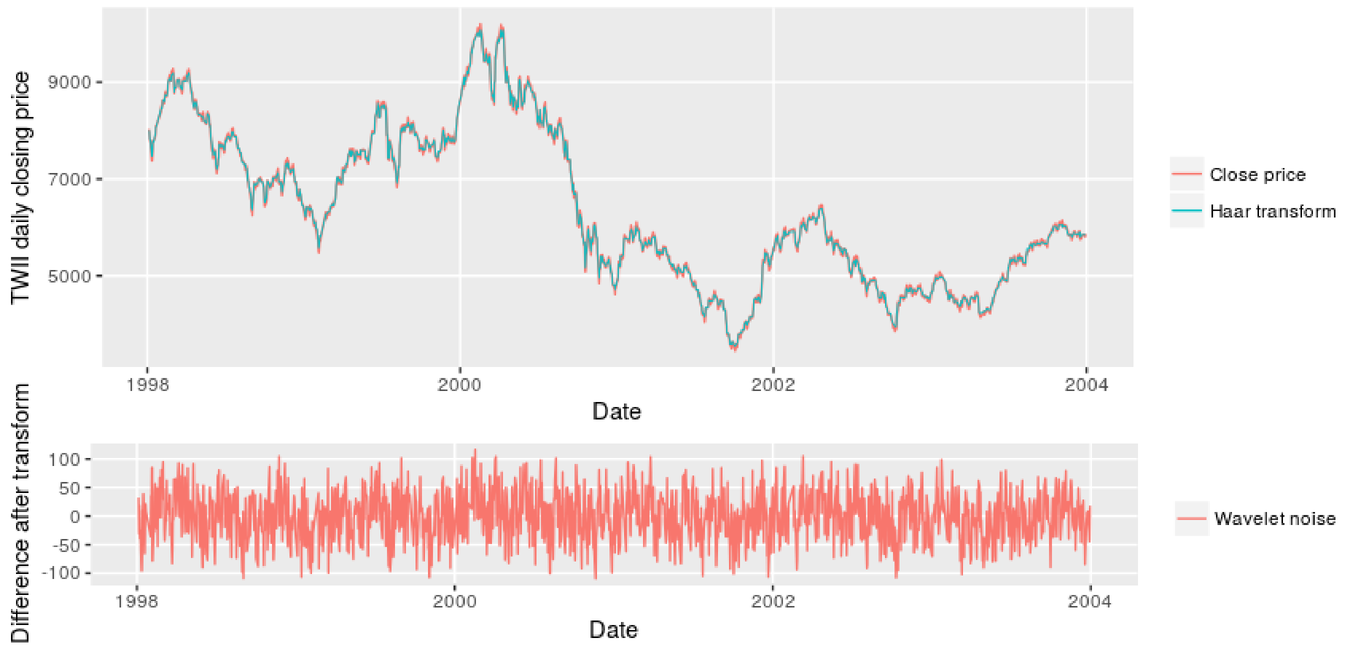
$$f_i(t) = \frac{f_i(t) - f_i^{min}}{f_i^{max} - f_i^{min}} \tag{6}$$

Alternatively, we could simply divide all feature elements by the maximum feature value. However, doing so would not ensure that the scaled feature value range lies between $[0, 1]$ but rather $[\frac{f_i^{min}}{f_i^{max}}, 1]$. Both approaches are valid and we reiterate that many other scaling methods exist. Moreover, we do not deny that different scaling methods may have been used in [?] but the exact scaling implementation does not seem to be crucial for the final results presented in section 5 and as such do not affect our final conclusion.

4.2.2 Haar wavelet transformation

After all input variables are calculated and scaled, the HWT is applied twice to each feature. The Haar wavelet transform is applied to the yearly subsets of input features individually. Details about the thresholding methods and threshold value are not described in the replicated paper, and as such we assume the use of soft thresholding as in equation 3 and a thresholding value based on the universal threshold value in equation 4.

Figure 10: Yearly subsets of original and transformed daily closing prices of the TWII. The difference between transformed and original prices is hard to see in the top figure, but this shows how little noise we should actually remove from the data.



Hsieh et al. [?] give a detailed explanation of general wavelets, leading to a section that is overly complex to the unfamiliar reader while still leaving out crucial information about the actual Haar transform implementation. We assume that the HWT is applied to each yearly subset in advance of model training and testing since there is little room for interpretation. One intuitive alternative interpretation we do not present is to recalculate the Haar transformation on all observations up until time t to predict the price at time $t + 1$. While seemingly sensible, periodic updating of the signal transformation leads to problems described earlier in section 3.3. Furthermore, if a step-wise HWT variant was indeed implemented then surely this would have been described in [?], which is why we assume it was not.

Thresholding must be applied to the wavelet coefficients during the wavelet transformation, to reduce the noise of the input signal, and can be done either according to the universal equation 4 or an adjusted alternative 5. It was already shown in figure 5 that the choice of the threshold value is crucial. While it may look as if there is only little noise-reduction happening using the adjusted threshold value, figure 10 graphs the difference before and after the noise-reduction. Figure 10 still represents the data when applying the HWT to the yearly subsets but these subsets are graphically shown together for a cohesive illustration. For comparison, the average absolute daily price change of the original stock prices between 1998 and 2003 is 85 points.

4.3 Feature selection

After preprocessing the most important input variables are selected using Step-wise Regression Correlation Selection (SRCS). Hsieh et al. used SPSS to apply the SCRS method. From a total of 14 potential input features they select only the set of independent variables that most strongly influence the following-day closing price.

The selected variables are presented so the SCRS feature selection step is skipped in this replication study by directly selecting the input variables proposed by Hsieh et al. For the TAIEX data this leads to 8 input variables, listed below.

Table 1: Input features selected by SCRS feature selection as reported by [?]

ID	Feature description	Notation
1	Close price	C_t
2	Open price	O_t
3	Low price	L_t
4	High price	H_t
5	Demand index	DI_t
6	6-day Moving Average	$MA(6)_t$
7	12-day Exponential Moving Average	$EMA(12)_t$
8	26-day Exponential Moving Average	$EMA(26)_t$

Table 2: Feature formulas

Feature	Formula
Moving Average over period m, MA(m)	$MA(m)_t = \sum_{i=t-m}^{t} C_i/m$
Demand Index, DI	$DI_t = (H_t + L_t + 2C_t)/4$
Exponential Moving average over period m, EMA(m)	$EMA(m)_t = \sum_{i=t-m}^{t} DI_i/m$

4.4 RNN

The input features described in the previous section are fed into a recurrent neural network, RNN. The hidden layer of the RNN consists of merely three neurons with sigmoid activation functions. The neural network parameters consist of weights w and biases b.

Our input data is a time series of eight inputs, which will be fed to the model in chronological order. The input to the hidden neuron at time t is a weighted sum of the values in the input layer at time t and the activation of the hidden neurons in the previous time step, a_{t-1}^h.

$$v_t^h = \sum_{i=1}^{8} w_{input,i}^h x_{t,i} + + \sum_{j=1}^{3} w_{recurrent,j}^h \sigma(v_{t-1}^j) + b^h \tag{7}$$

With $h = 1, 2, 3$ denoting the index of one of the three neurons in the hidden layer. $x_{t,i}$ is one of the 8 inputs in the input layer at time t, multiplied by the neuron-specific weight $w_{input,i}^h$

of the given input x_i for neuron h. The defining property of the recurrent network is the weighted addition of $\sigma(v_{t-1}^j)$, the activations of the hidden neurons at the previous time step. Lastly, per hidden neuron h a bias b^h is added to the received input value.

The received input to the hidden layer is translated to an activation, the outputs of the hidden neurons. The activation function used here is the frequently used sigmoid function,

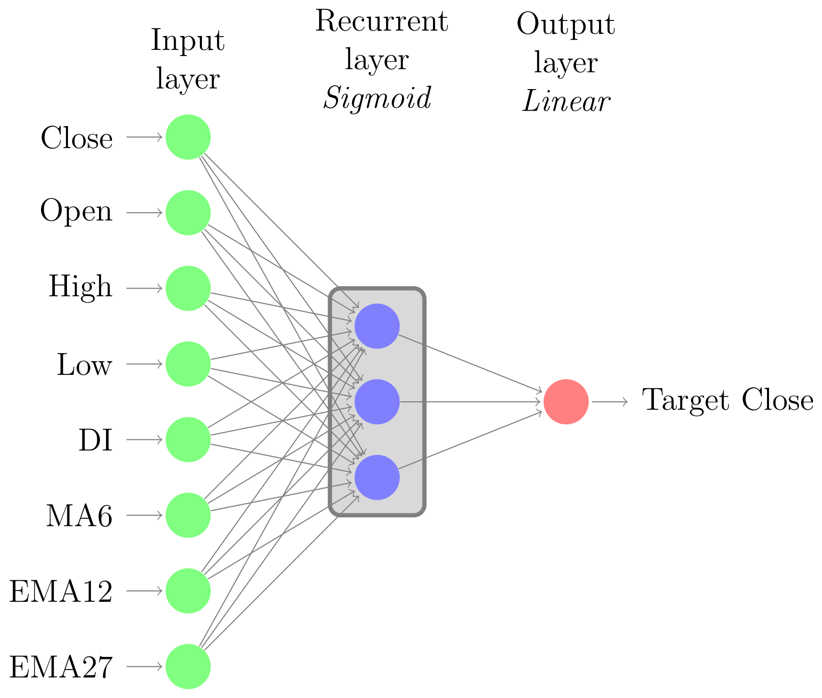
$$\sigma(v_t^h) = \frac{1}{1 + e^{-v_t^h}} \tag{8}$$

Where v_t^h is the activation input from equation 7. A linear combination of the hidden activations and an output bias is calculated to be final activation of the output layer,

$$v_t^{out} = \sum_{j=1}^{3} w_j^{out} \sigma(v_t^j) + b^{out} \tag{9}$$

To train the RNN we must adjust all the parameters so that we minimize the root mean-squared error, RMSE, between the sequences of final outputs v_t^{out} and the target outputs y_t. For a final overview, we must find the optimal combination of RNN parameters listed in table 3.

Figure 11: The RNN architecture used by Hsieh et al. [?]



4.5 ABC algorithm

In order to optimize the parameters of the RNN listed in table 3 there exist many training algorithms. For the replication of [?] we employ their implemented training algorithm, namely

Table 3: Parameters of the RNN and their notation

Parameter	Notation
Input to hidden weights	$w^h_{input,i}$
Recurrent weights	$w^h_{recurrent,j}$
Output weights	w^{out}_j
Hidden bias	b^h
Output bias	b^{out}

the Artificial Bee Colony algorithm. As can be deduced from the name, this training algorithm is based on how bee swarms find the optimal sources of honey to harvest as proposed by [?].

The ABC training algorithm is an iterative process that is initialized randomly and depends on random numbers to update the RNN parameters during training, therefore it practically always converges to a different solution. The ABC algorithm is based on three groups of bees within the swarm: scouting, onlooker and employed bees. Each of the employed bees represents one food source, a solution in the parameter space of the RNN. Onlooker bees choose one of the employed bees based on their success and explore better solutions in the vicinity of their chosen source. If a given source is abandoned, then the corresponding employed bee becomes a scout and finds a brand new food source. Below, the algorithm is described in 4 phases, the initialization, onlooker, scout and employed phase, respectively. Finally the algorithm is summarized in pseudo code at the end of this subsection.

4.5.1 Initialization phase

To apply the ABC algorithm we must choose a number of employed bees, which is equal to the number of available food sources. In this case the total number of food sources is set to $SN = 50$. Each of these sources has their own parameter space and the goal is to idenify the optimal food source, or, in other words, the optimal combination of parameters. So, we are in fact training 50 different networks together. Each RNN, or solution thereof, is represented by the parameters listed in table 3 and is denoted by $X_s = (X_{s,1}, X_{s,2} ..., X_{s,d})$ consisting of all weights and biases, where d is the total number of RNN parameters and $s = 1, 2, ..., SN$. At first, all employed bees are scout bees and find an initial solution by random allocation of the parameters X_s based on uniformly distributed random numbers from a chosen interval, $[-\beta, \beta]$,

$$X_{s,i} \sim UNIF[-\beta, \beta] \tag{10}$$

Where the β parameter is a scale parameter which is set to 3 by trial-and-error as it was mentioned in [?] that we must set this scale, but the actual value of it was not given. The scale parameter affects training efficiency but, with reasonably small values of β, not necessarily the final model performance.

The fitness of each food source/solution s is based on the RMSE between the ABC-RNN output, with parameters X_s, and the target one-day ahead stock prices transformed by the HWT. The goal is to minimize the RMSE between the model output and target values, as such the fitness is defined as follows,

$$fit_s = \frac{1}{RMSE_s} \tag{11}$$

4.5.2 Onlooker phase

Using the random initial allocation of food sources according to equation 10, the scout bees become employed bees and stick to this solution. Onlooker bees then choose one of the food sources based on their fitness, prioritizing those performing the best. The probability of an onlooker bee choosing a given food source,

$$P_s = \frac{fit_s}{\sum_{i=1}^{SN} fit_i} \tag{12}$$

By selecting food sources based on their fitness we attract more onlookers, who search for neighboring solutions, to food sources that are already performing well.

The onlooker bee finds a new solution close to the selected food source by changing only one parameter of the selected solution X_s and some other random solution X_k. The solution of the onlooker bee, denoted V_s, is equal to X_s except for one randomly chosen element. A second, yet different, solution parameter space X_k is selected randomly. The parameter index j is chosen randomly from $[1, SN]$, effectively selecting the parameter to be updated. The chosen parameter is then adjusted by the updating formula for onlooker bees,

$$V_{s,j} = X_{s,j} + u(X_{s,j} - X_{k,j}) \tag{13}$$

where u is a uniformly distributed random number on the interval $[-1, 1]$. If the resulting value of $V_{s,j}$ falls outside the acceptable range of parameter values, namely $[-\beta, \beta]$, then it is set to the closest corresponding extreme value in this range. Note that in the replicated paper the last term in formula 13 (equation 11 in [**?**]) is actually a sum and not a difference, but this does not correspond to the accompanying textual description of it and must be a simple overlooked mistake.

Every onlooker bee has now been allocated a new solution V_s based on its chosen food source X_s and the fitness is calculated according to equation 11. A greedy selection process is used to update the new solutions by simply choosing the best neighboring solution as determined by the onlookers associated with any given initial solution X_s. In most cases there will be some solutions that are not selected at all in the onlooker bee phase and therefore not updated. The employed phase, described later, ensures that all solutions are updated at least once.

4.5.3 Scout phase

If a given food source, or solution, does not improve for a predetermined number of iterations then it is considered to be abandoned. Employed bees corresponding to an abandoned solution, represented by X_s, are changed to scout bees and find a new solution to replace X_s with. This happens by the updating formula for scout bees, depending on the current best solution X_{max} and the current worst solution X_{min},

$$X_{s,j} = X_{min,j} + rand[0,1](X_{max,j} - X_{min,j}) \tag{14}$$

The predetermined number of iterations before a non-improving food source is considered abandoned is given in [?] to be,

$$limit = SN \times d \tag{15}$$

where SN is the total number of food sources and d is the total number of parameters (sum of all weights and biases). Now, if $SN = 50$ and the RNN has 3 hidden recurrent neurons, with 8 inputs, then $d = 40$. This means that $limit = 2000$ in [?].

4.5.4 Employed phase

The employed bees also try to find a better solution themselves during the employed phase. That means that every iteration the current solutions X are updated according to equation 13, making this phase very similar to the onlooker phase. The difference with the onlooker phase is that we do not need the selection probabilities P_s but rather every food source selects itself as the base solution X_s as per the notation in equation 13. This phase happens at the start of each iteration.

4.5.5 ABC algorithm pseudo code

To summarize the steps outlined above we present the pseudo code of the ABC algorithm, specifically for replication of [?]. The input data are 8 time series of features transformed by the HWT, listed in table 1. The target values are the one-day ahead stock prices transformed by the HWT. Finally, the ABC-RNN is trained for 6000 iterations.

Algorithm 1 ABC-RNN training

```
Input: Yearly transformed input data and target values
Fixed hive size of $SN = 50$ neural networks

Initialization phase
Initialize food sources according to eq. 10
Calculate fitness of employed networks according to eq. 11

for Epoch = 1 to 6000 do
    Employed phase
    1. Compute new solutions according to eq. 13
    2. Calculate new fitness
    3. Apply greedy selection between old and new solutions
    4. Calculate selection probabilities $P_h$ according to eq. 12

    Onlooker phase
    1. Roulette wheel selection using $P_h$ to select base employed networks
    2. Compute new solutions according to eq. 13
    3. Calculate fitness of onlooker solutions
    4. Apply greedy selection between onlooker solution and their respective food source

    Scout phase
    if There is an abandoned employed network as determined by limit then
        1. Overwrite abandoned network parameters according to eq. 14
    end if
    Memorize best employed network
end for
```

4.6 Evaluation

4.6.1 Standard metrics

After training the ABC-RNN on each yearly subset we must evaluate the results. To be able to compare them with the results reported by Hsieh et al. we have made assumptions about the scaling of the data in different steps of the training, hyperparameters in the ABC-RNN model and, of course, we assume a specific application of the HWT described in section 4.

The metrics to be compared are the root mean squared error (RMSE), mean absolute percentage error (MAPE) and a profit evaluation. Let us represent the true target value at time t as Y_t and the predicted value as F_t, then the RMSE and MAPE evaluation metrics are calculated as follows,

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(X_t - F_t)^2} \tag{16}$$

$$MAPE = \frac{1}{N}\sum_{t=1}^{N}\left|\frac{X_t - F_t}{X_t}\right| \tag{17}$$

4.6.2 Profit evaluation

The standard RMSE and MAPE metrics given above fail to provide information about the accuracy of predicting the direction of the market. If our prediction of the one-day ahead closing price is greater than the current closing price then we should buy the stock and sell it tomorrow, so predicting the direction of the market is of great importance. So a very general rule would be,

If $F_{t+1} > X_t$ then buy.
If $F_{t+1} < X_t$ then sell.

Hsieh et al. apply a slightly more sophisticated trading strategy as identified by Cheng et al. in 2009 [?]. The trading strategy takes into account the current prediction error and only acts on the next prediction if the current prediction error is sufficiently small. The trading rules are as follows,

Sell rule

$$\text{If } \left|\frac{F_t - X_t}{X_t}\right| \leq \alpha \text{ and } F_{t+1} > X_t \text{ then buy} \tag{18}$$

Buy rule

$$\text{If } \left|\frac{F_t - X_t}{X_t}\right| \leq \alpha \text{ and } F_{t+1} < X_t \text{ then sell} \tag{19}$$

If none of the conditions are met then we simply do nothing with our prediction. The value of α depends on the variability of the daily fluctuations of the market price. This value is chosen by optimizing the profits made in the training period with respect to α using the above trading strategy.

A further extension of these trading rules is to incorporate transactions costs. A standard transaction cost of 0.1425% is used, with an additional transaction cost of 0.3% for selling. The profits of buying and selling are calculated as follows,

Sell profit

$$\Pi_{Sell} = (X_{t+1} - X_t) - |X_{t+1} - X_t| \times 0.4425\% \tag{20}$$

Buy profit

$$\Pi_{Buy} = (X_t - X_{t+1}) - |X_t - X_{t+1}| \times 0.1425\% \tag{21}$$

The total profit is simply the sum of the profits of all transactions in the test period. A maximum of 40 trades can be made during each yearly testing period.

5 Replication results

In this section we will present the final results of replicating the paper by Hsieh et al. [?] and compare our results with their presented results. By replicating the results we may confirm assumptions about the implementation in order to show that the only logical interpretation of the methods described in [?] that lead to their presented results is by using the HWT erroneously. The evaluation metrics compared further down this section are computed based on the unscaled target values.

Since the ABC-RNN is largely based on randomly generated numbers to find an optimal solution the results will vary wildly. Hsieh et al. [?] have only presented the evaluation metrics once for every yearly test set. This means that the results we find are not guaranteed, or expected even, to be centered around the reported results. Especially the MAPE and profit evaluation metrics may vary wildly since they are not optimized directly during training of the ABC-RNN, but the RMSE is. Because of this we mainly look at the replicated RMSE to validate the correctness of our replication, and its underlying assumptions.

To exemplify the variability of the evaluation metrics we replicated the results 50 times, which leads to training the ABC-RNN 300 times for each set of assumptions. We distinguish and test three different scenarios. Most assumptions are equal for all tested scenarios, such as the scaling according to equation 6, soft thresholding according to equation 3 with an adjusted universal threshold value shown in equation 5. What differs however in the first two scenarios however is whether the target output stock prices of the ABC-RNN during training (and evaluation of the MAPE and RMSE) are transformed by the HWT or not. In the third scenario we assume that the HWT has not been used at all, to serve as a baseline of how predictable the Taiwenese stock index is.

Below we present the result of 50 independent replications and compare the RMSE, MAPE and profit evaluations on the yearly test sets for the three scenarios mentioned above. Figures 12 through 14 show the spread of our replication results, compared to those reported by Hsieh et al.

Based on the bottom two graphs of the RMSE replication in figure 12 we can see that the transformation of the target output does not have a clear impact on the final results. The only difference between transformed and untransformed target stock prices is that we do not try to predict the noise in case of the transformed target values, and since the noise is not supposed to be predictable anyway we will not see a large difference between both approaches. The RMSE does not follow a clear distribution, which is why it is not sensible to apply statistical tests to confirm the validity of our assumptions. However, purely based on the RMSE the results without using the HWT at all also appear to be plausible. Of course, we can be sure that the HWT was applied to the input features since this was clearly proposed in [?].

The replication of the MAPE is presented in figure 13 and highlights issues with the test set of the year 2000, where the MAPE and RMSE we found are relatively large. It is unclear where this remarkably large spread comes from, but the unique difference between the test set of the year 2000 and the other years is that the stock prices are consistently trending downwards as can be seen in figure 9. The minimum yearly stock price is also included in the test set which means that, as opposed to the other test sets, the stock prices in the test set are lower than any stock prices seen during training. Whether this is indeed the root cause of the deviating results is unclear. We do not analyze the deviating results of the year 2000 any further because it may very well be that Hsieh et al. found this same spread in their results but we cannot confirm this in any way since they only reported a single observation.

Finally, the profits are shown in figure 14 and show a large variability again. As noted, we do not expect our replication results to be centered around those reported in [**?**]. The bottom two graphs do not show a reason to question the correctness of either scenario, so that we cannot conclude for sure that the HWT was applied to the target output. Of course, the profit evaluation is performed on untransformed data (unlike the RMSE and MAPE) and we do not actually expect large differences in the bottom two graphs. In both cases we have 50 replications of 6 test sets consisting of 40 trading days, a total of 12000 trading opportunities. Whether we apply the HWT to the target output or not leads to a total of 10963 and 11136 performed trades respectively, with a 57% and 58% profit rate of individual trades. For comparison, if we do not apply the HWT at all we achieve a total of 7641 executed trades of which 52% is profitable.

Tables 4 and 5 summarize our findings of the replicated RMSE and profit metrics. We believe that these results replicate the originally reported results for both the RMSE and profit sufficiently, supporting our claim that the HWT was used erroneously. Whether the target values were transformed or not does not impact our final conclusion since that only relies on the application of the HWT for the input features. While the description of the use Haar wavelet in [**?**] is unclear, it does not leave much room for vastly different interpretations. The results may be even closer, or more consistent, if we knew the exact thresholding and scaling methods, or even the threshold values. After a lot of trial and error we have settled upon the presented results and remain convinced that the HWT was applied to yearly subsets with some form of thresholding, which means we are not actually predicting the stock market using the ABC-RNN model but rather reverse engineering the HWT.

Figure 12: The yearly RMSE replicated 50 times, compared to that reported in [?]. Three scenarios are tested with slightly different assumptions.

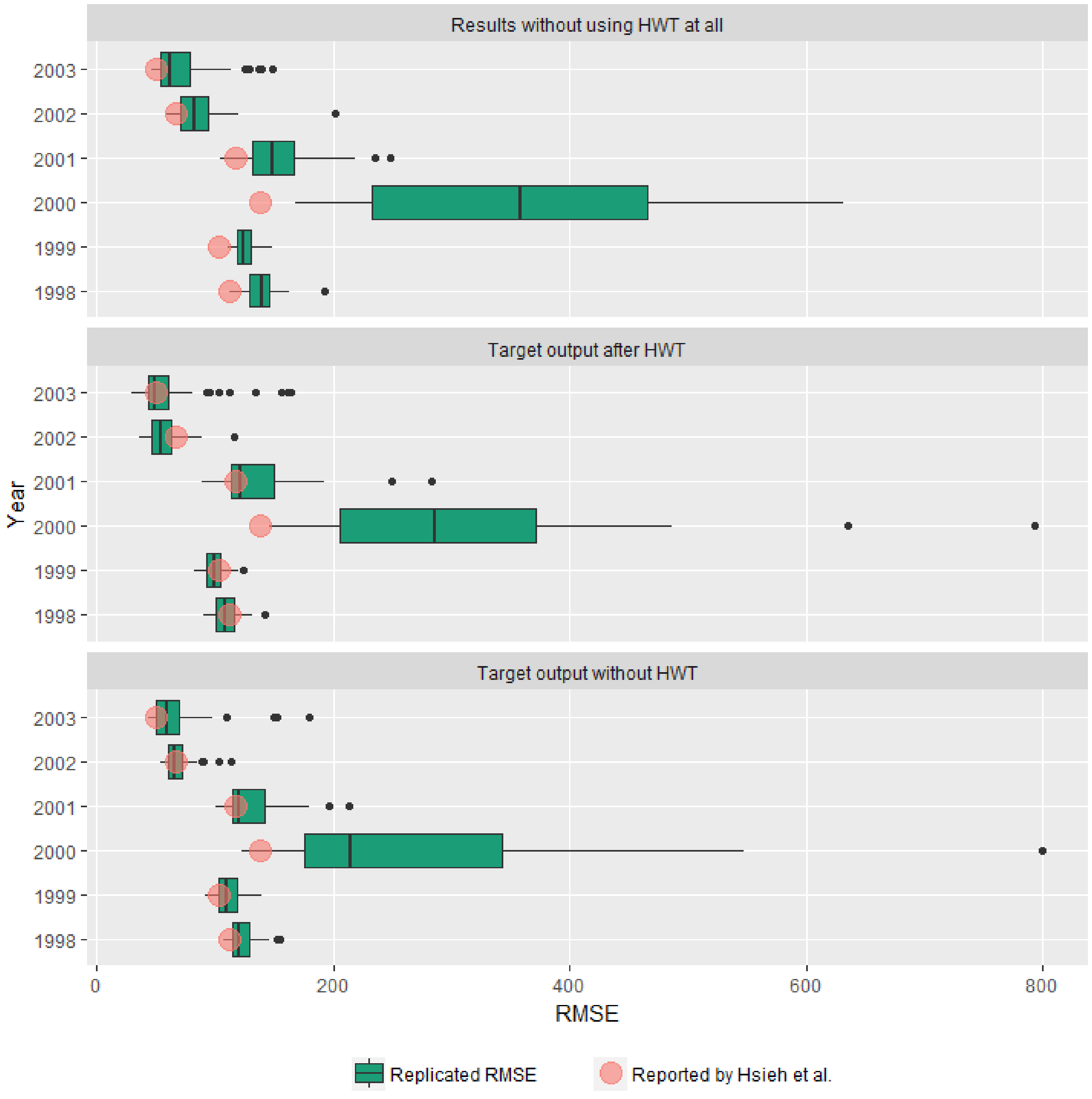


Figure 13: The yearly MAPE replicated 50 times, compared to that reported in [?]. Three scenarios are tested with slightly different assumptions.

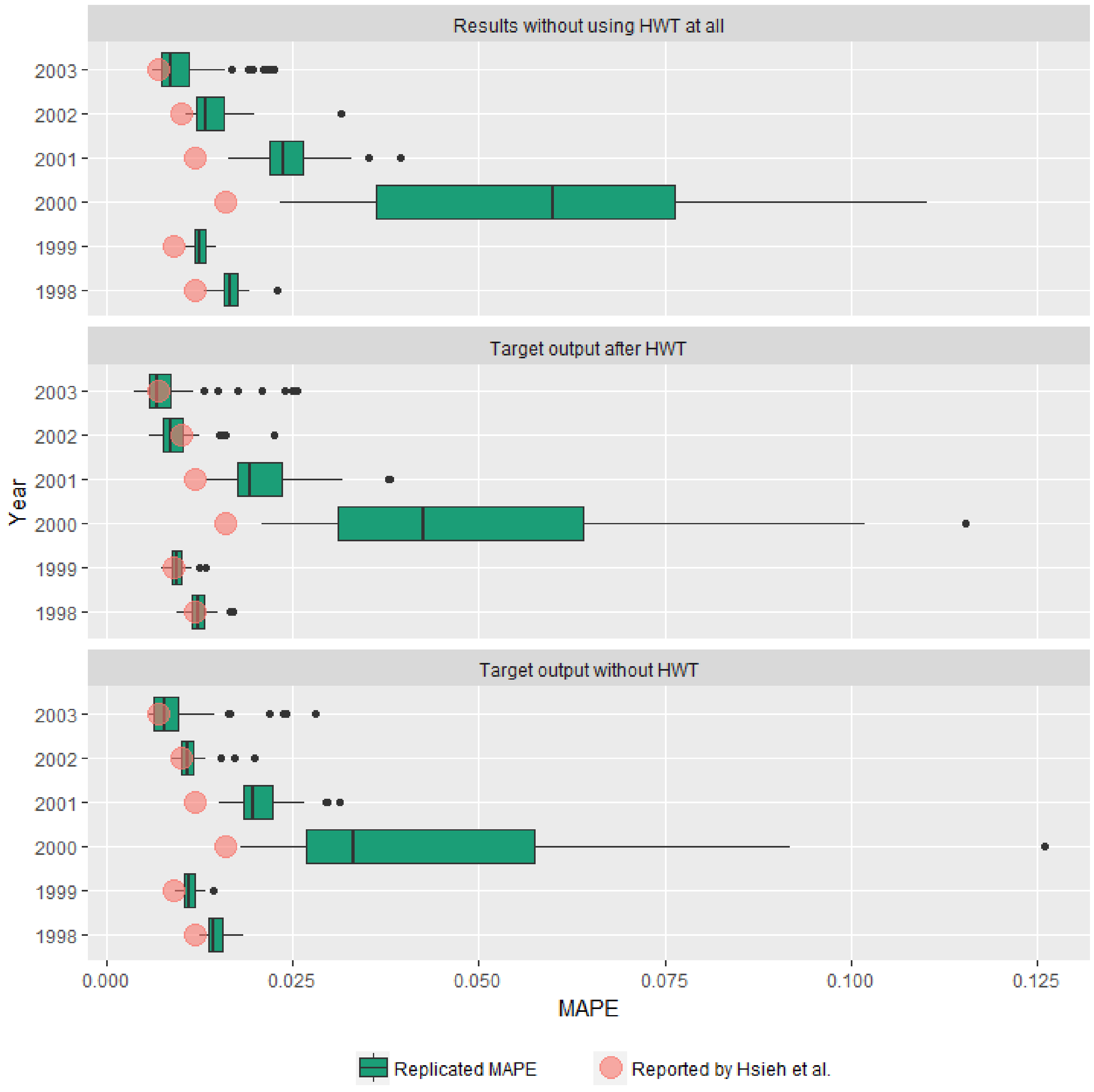


Figure 14: The yearly profit evaluation replicated 50 times, compared to that reported in [?]. Three scenarios are tested with slightly different assumptions.

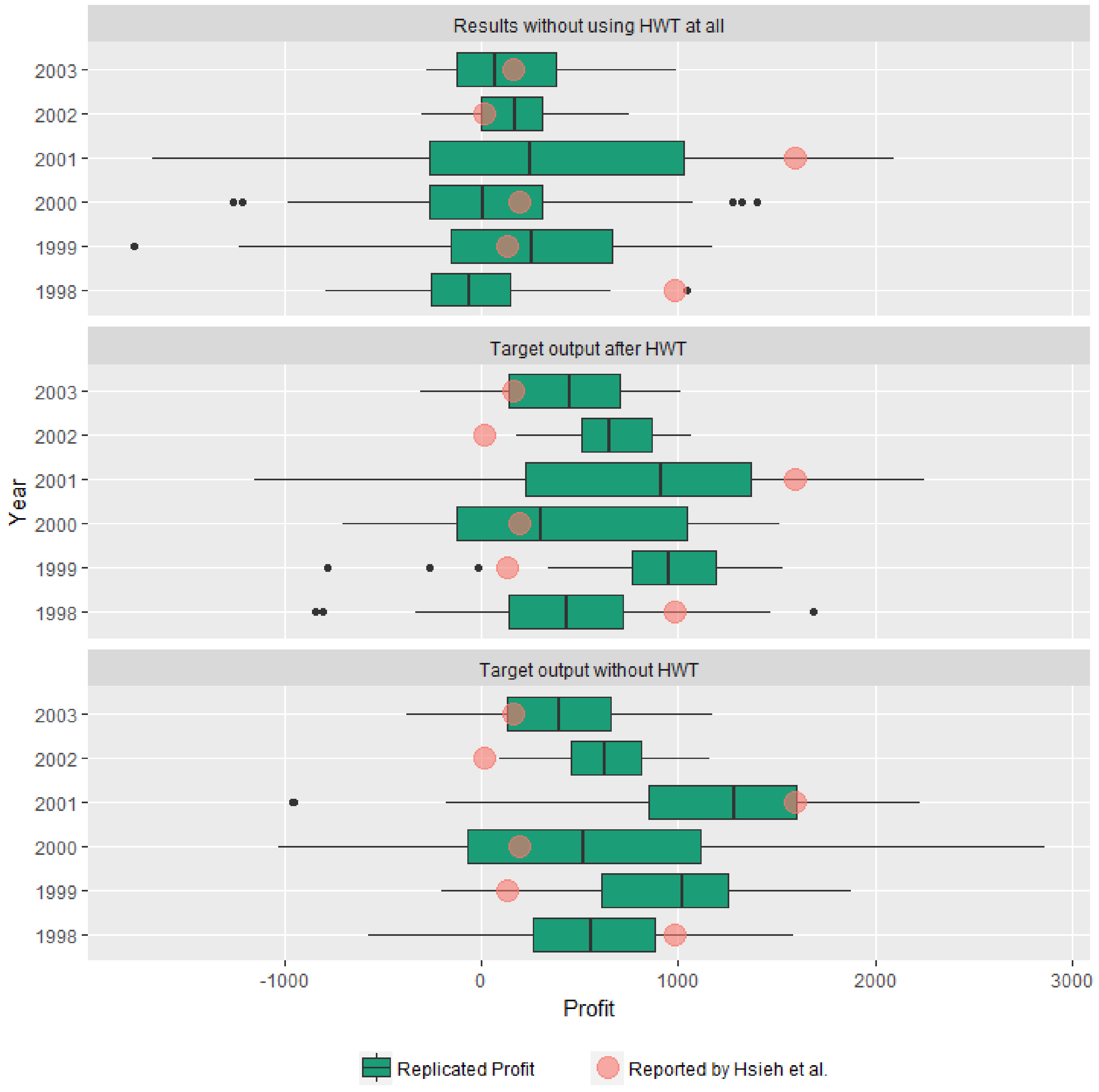


Table 4: RMSE replication results based on 50 replications of each yearly subset. We show the *mean* and the *(sd)* of the replicated RMSE evaluations in white and grey respectively.

Subset	**Reported RMSE**	**Without HWT**	**Untransformed target**	**Transformed target**
1998	113	139	123	110
		(13)	(12)	(12)
1999	103	125	111	99
		(8)	(11)	(9)
2000	138	364	272	299
		(131)	(135)	(129)
2001	118	153	130	136
		(30)	(24)	(37)
2002	67	86	69	57
		(22)	(12)	(15)
2003	50	75	69	63
		(30)	(31)	(36)

Table 5: Profit evaluation results based on 50 replications of each yearly subset. We show the *mean* and the *(sd)* of the replicated profit evaluations in white and grey respectively.

Subset	**Reported profit**	**Without HWT**	**Untransformed target**	**Transformed target**
1998	983	-24	537	449
		(393)	(529)	(507)
1999	132	155	935	921
		(651)	(464)	(435)
2000	197	37	582	405
		(600)	(872)	(650)
2001	1592	319	1159	819
		(859)	(662)	(840)
2002	15	151	639	650
		(227)	(256)	(237)
2003	163	148	398	417
		(319)	(354)	(352)
Accumulated mean profit	3082	785	4250	3661

6 Noise prediction using the Haar wavelet

In the previous section we have shown that we believe that the only sensible interpretation of the proposed model in [?] is to use the HWT erroneously. In section 3.4 we illustrated by example that the last element in a random sequence can be inferred if the HWT is used with thresholding. Of course we assumed that the last element could only be one of two values and we were only interested in a single element, while we apply the same methods to Taiwanese stock price time series with many more elements and a much wider array of possible values. As such we no longer expect to reverse engineer the HWT so that we can infer the exact values, which is why we did not find perfect profits, but instead we can use future information to achieve better results than purely guessing. To conclusively show that the erroneous use of the HWT leads to successful 'prediction' results even when the target data is inherently unpredictable we apply the ABC-RNN model with the HWT to generated noise. The goal is to show that if we treat the generated noise as a price time series, we should be able to make a profit and forecast the sign of the difference between subsequent observations more than 50% of the time.

6.1 Data generation

The noise we generated should be similar to stock data, preferably we would generate unpredictable stock data but we can't do so easily. Daily stock price data consists of open, high, low and closing price and we only predict the daily closing prices. We cannot generate daily open, high and low prices sensibly since they would have to be related to the closing price in some way and there can be no relation between any features besides the HWT to ensure unpredictability. As such, we only generate 250 daily closing prices as follows with a similar mean and variance as the Taiwanese stock prices between 1998 and 2003.

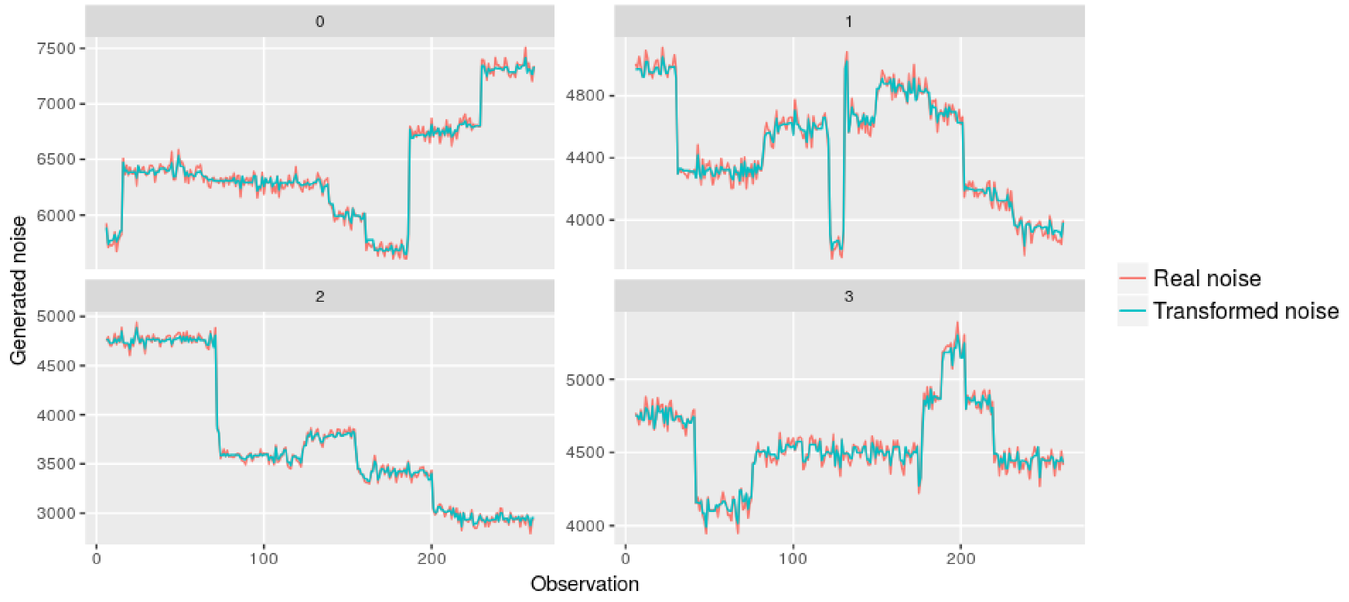
To generate stock price-like noise we also require larger temporal movement, often seen through longer lasting yet unpredictable trends in the stock price movement. Since trends would imply some relation with past observations, but we still want some large random movement, we choose 10 random time steps at which we add an additional large shock. Of course, the goal is not to simulate accurate stock data but rather to produce unpredictable noise with added variability to illustrate our point. As per the description above, the noise is generated as follows,

$$\begin{aligned} & n_0 = \mathcal{N}(5000, 360) \\ & n_t = n_{t-1} + A_t + \mathbb{1}_t B_t \\ & \quad where, \\ & A_t \sim \mathcal{N}(0, 360),\ B_t \sim \mathcal{N}(0, 2500) \end{aligned} \tag{22}$$

with $\mathbb{1}_t \in \{0, 1\}$ an indicator that is only non-zero at 10 randomly chosen time steps.

The resulting noise is then transformed using the HWT with the same soft thresholding rule and threshold value as was used for stock price prediction. Figure 15 shows some of the generated noise samples according to equation 22 before and after the HWT,

Figure 15: Generated noise samples before and after applying the HWT



6.2 Features

Since we have not generated daily open, high or low noise similar to what stock price data would include, we cannot compute all the features presented in table 1. From the example in section 3.4 it could be seen that much of the information about future values is revealed by comparing the observations at any time step before and after the HWT. Therefore we use the noise at time t and a 6-period moving average of the noise, before and after the HWT, to predict the untransformed noise at time $t+1$. The only difference in the ABC-RNN model is the number of inputs, which is 4, all other parameters remain the same.

All features are re-scaled according to equation 5, and since we are not interested in the size of the resulting profits or RMSE but only the sign we do not scale the data back up to their original values before evaluation. Meaning that all of the profits presented in the next segment are relatively small and should only look at the sign.

6.3 Noise prediction results

Figure 16 and table 6 summarize the results of predicting the noise described above. The shown results confirm our claims that using the HWT as proposed in [?] leads to successful but false predictions as future information is revealed when we apply the HWT with (soft) thresholding to reduce noise in the original signal.

Table 6 shows that out of a total of 2000 potential trades we traded 1441 times, since the trading rules in equations 18 and 19 depend on the accuracy of the previous prediction. Of these individual trades, we predicted the direction of the noise sequence correctly 66.07% of the time. In total we achieved a positive accumulated profit on the test sets consisting of 40 observations 86% of the time. Since the step wise direction of a noise sequence cannot be predicted with more than 50% accuracy by definition the methodology must be invalid for prediction models.

Figure 16: Accumulated profits on 50 test sets. A majority of the time we achieve positive profits on generated noise treated as stock prices, supporting the claim that the HWT cannot be used for prediction models as proposed in [?]

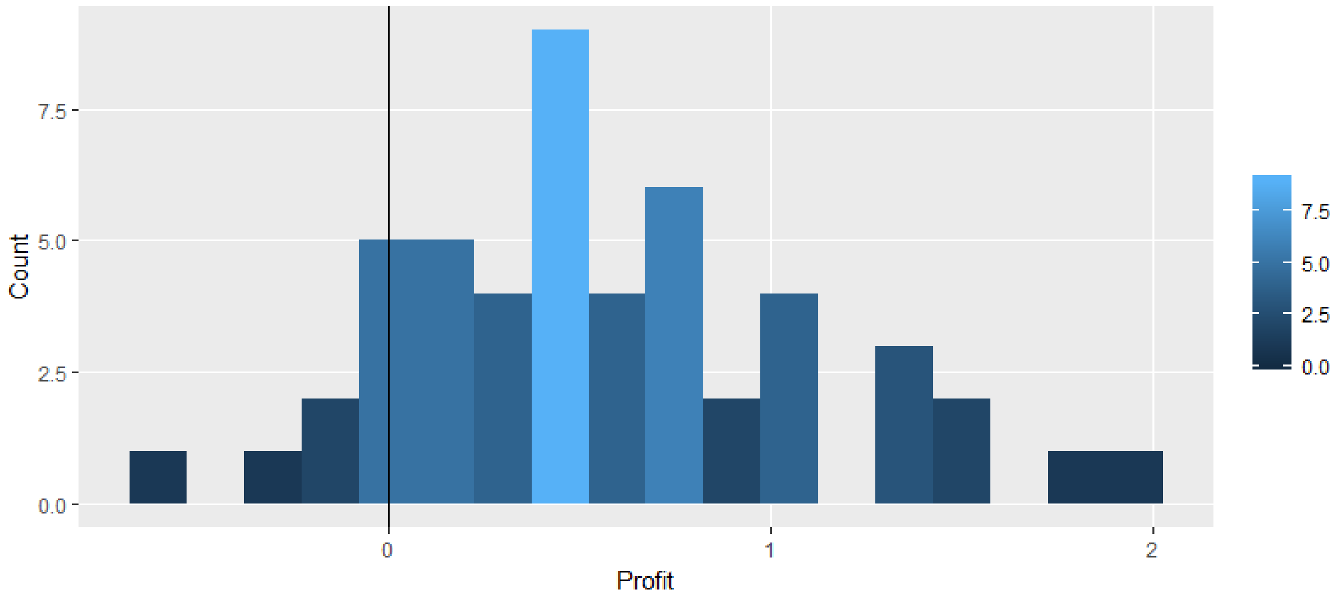


Table 6: Noise prediction on 50 independent test sets with 40 observations each, a total of 2000 potential trades. Trade accuracy is the ratio of the times we make a profit on individual trades, effectively predicting a noise sequence one step ahead.

Total trades	Trade accuracy	Accumulated profit >0
1441	66.07%	86%

7 Discussion

The paper by Hsieh et al. [?] was published in 2011 and cited over 160 times despite containing a methodological error that invalidates all the reported prediction results. The paper was used as a stepping stone for more recent research that was inspired by the use of the HWT for time serires prediction, without there being a past source of research that proposed these methods. Contradictory even, Aussem and Murtagh in 1997 [?], Zhang and Dong in 2001 [?], Zhang et al. also in 2001 [?] and others [?, ?] knew and mentioned why shift-variant discrete wavelet transformations cannot be used without an adaptation for time series forecasting, even referring to it as a well known problem.

The lacking descriptions of the implemented methods in [?] and the insufficient citations on the subject of discrete wavelet transformations may have been the reason that the research was published any way. We have attempted to contact the original authors of [?] but did not manage to do so, thus relying on many assumptions leading to the need of replication. Even after replication we are still unsure of the exact implementation needed to replicate the results but are confident that the only logical interpretation of the proposed methodology is to use the HWT in such a way that we cannot implement it in real-time and, moreover, find invalid prediction results. Sadly, the overlooked error has led to more recent research that potentially implemented similar erroneous methods under the false pretense that it the HWT is a valid technique for prediction purposes.

Lahmiri [?, ?], Khandelwal et al. [?] and Khuat et al. [?], among others, have all used shift-variant wavelet transformations for financial forecasting without using an adaptation such as the à trous algorithm to achieve shift-invariant alternatives. All of these mentioned papers were shortly discussed earlier in the relevant work section, specifically because they are so similar to the research we have replicated. While we do not claim that their results are also invalidated by our findings we do propose that they be further reviewed in future research to confirm the correctness of their methods.

Besides the citations based on the use of the HWT, many citations are also based on the use of the ABC algorithm for RNN optimization. Our results show that while even transformed noise is predictable to some degree, the ABC-RNN leads to wildly varying results which cannot be understood purely based on the results presented in [?]. Furthermore, it is mostly the other evaluation metrics besides the RMSE that show a lack of consistency. For stock price prediction specifically we believe that optimizing the RMSE of a prediction model in order to achieve profitable results is a lackluster approach since a low RMSE does not directly translate to higher profitability. Instead, many other researchers rather classify the direction of daily returns (either up or down) and we believe our results support such alternative approaches that directly deal with the final goal of profitable stock prediction.

8 Conclusion

Using the Haar wavelet transformation to reduce the noise of stock price time series is not new, however proposing such methods for (real-time) stock prediction seems to be first introduced in a published and well-cited paper by Hsieh et al. in 2011 [?]. Through replication we have attempted to confirm assumptions about the model implementation and are confident that the only logical interpretation is that he Haar wavelet transformation was used to the entire daily prices sequence, including training and test sets.

By ways of illustration we have exemplified that the shift-variant nature of the Haar wavelet transformation leads to the possibility of inferring future information by comparing the original sequence and the transformed sequence after applying the HWT with thresholding for noise reduction. This was falsely seen as successful prediction results, but it is rather a case of reverse engineering the HWT. Since some information about the sequence is lost through noise reduction we are not able to reconstruct (or predict) the exact original sequence, but enough information is present to even reconstruct a noise sequence in such a way that is would be profitable if we consider the generated noise as a pricing series.

The replication results do not lend themselves well to statistical testing since the evaluation metrics do not follow a clear distribution and we do not expect our replicated results to be centered around the reported results in [?]. But because our replicated predictions are also consistently profitable with similar sizes of the achieved profits and the RMSE follows the same yearly pattern as the reported results we are confident that we interpreted the use of the HWT proposed in [?] correctly. This does not rule out other assumptions being incorrect, such as the scaling used during training or whether the target values are transformed, although the invalidity of the results in [?] does not rely on these assumptions.

We conclude that the results presented by Hsieh et al. [?] are invalid because of erroneous methods. Their research does not contradict the EMH and cannot be used in real-time stock prediction. Because there are many published articles that rely on the correctness of the methods in [?] we recommend that the relevant papers are reviewed in further research.