mathematics

Article

Deep Neural Network Model for Hurst Exponent: Learning from R/S Analysis

Luca Di Persio [1] and Tamirat Temesgen Dufera [2,*]

[1] Department of Computer Science, University of Verona, 37129 Verona, Italy; luca.dipersio@univr.it
[2] Department of Applied Mathematics, Adama Science and Technology University, Adama P.O. Box 1888, Ethiopia
* Correspondence: tamirat.temesgen@astu.edu.et; Tel.: +25-191-138-2438

Abstract: This paper proposes a deep neural network (DNN) model to estimate the Hurst exponent, a crucial parameter in modelling stock market price movements driven by fractional geometric Brownian motion. We randomly selected 446 indices from the S&P 500 and extracted their price movements over the last 2010 trading days. Using the rescaled range (R/S) analysis and the detrended fluctuation analysis (DFA), we computed the Hurst exponent and related parameters, which serve as the target parameters in the DNN architecture. The DNN model demonstrated remarkable learning capabilities, making accurate predictions even with small sample sizes. This addresses a limitation of R/S analysis, known for biased estimates in such instances. The significance of this model lies in its ability, once trained, to rapidly estimate the Hurst exponent, providing results in a small fraction of a second.

Keywords: fractional Brownian motion; Hurst exponent; deep neural networks; R/S analysis; stock prices; DFA

MSC: 68T07; 60G22; 91G15; 91G20; 60G18; 68T20

check for updates

Citation: Di Persio, L.; Dufera, T.T. Deep Neural Network Model for Hurst Exponent: Learning from R/S Analysis. *Mathematics* **2024**, *12*, 3483. https://doi.org/10.3390/math12223483

Academic Editors: Cristina Raluca Gh. Popescu and Raymond Lee

Received: 4 September 2024
Revised: 1 November 2024
Accepted: 5 November 2024
Published: 7 November 2024

1. Introduction

The Hurst exponent, a crucial parameter in the analysis of stochastic time series, notably in the context of stock prices, plays a pivotal role in detecting and characterising extended memory properties within financial data; see, e.g., [1]. It was first introduced by Hurst in 1951 [2], marking a significant milestone in time series analysis. This exponent's importance lies in its ability to provide insights into financial time series' inherent predictability and persistence. Accurate estimation of the Hurst exponent holds the promise of improved predictions, especially in the volatile and complex realm of stock markets.

The significance of this estimation process becomes evident when it challenges the conventional Efficient Market Hypothesis (EMH). The EMH posits that stock prices follow a random walk, akin to a Brownian motion, implying that future prices cannot be predicted based on past prices. However, when the Hurst exponent indicates a level of predictability, it contradicts this hypothesis and suggests that there are patterns and dependencies within the data that can be exploited for forecasting [3,4].

Annis and Lloyd [5] contributed to estimating the Hurst exponent by deriving the expectation of the adjusted rescaled range for independent normal summands. Lo [6] introduced a long-term memory test in financial time series, particularly in the stock market returns. They applied this test to daily, weekly, monthly, and annual stock return indexes over various periods. Surprisingly, they found no clear evidence of long-range dependence in these indexes once short-term autocorrelation was considered. This led them to suggest that stochastic models of short-range dependence might sufficiently explain stock market behaviour, challenging the prevailing notion of inherent long memory in financial markets.

In [7], Beben investigated long-time correlations in market indexes using the Hurst exponent and detrended fluctuation analysis (DFA). The study revealed that, contrary to random processes, Hurst exponents consistently deviated from the expected value of 0.5 in various cases, indicating the presence of persistent behaviours in the dynamics of the financial market.

Cajueiro and Tabak [8] investigated the efficiency of emerging markets in Latin America and Asia using R/S analysis to assess the Hurst exponent of log-return time series data. Their findings supported that these markets are generally becoming more efficient over time, with some exceptions observed in Brazil, the Philippines, and Thailand.

In their research, Granero et al. [9] assessed the efficacy of R/S analysis and its crucial modifications for detecting long memory in time series data. They also introduced and compared a geometric method with slight modifications. This comparative study enhances our understanding of techniques for identifying long-memory patterns in diverse time series datasets.

In their work, Resta [10] conducted a comprehensive review of effective methods for computing the Hurst exponent. Their study offers insight into the evolving literature on this topic and delves into practical implementations, including patents. This dual perspective enriches our understanding of the computation of the Hurst exponent and its real-world applications.

The authors in [11] introduced a novel approach based on the geometric method (GM2) algorithm and the fractal dimension method initially proposed by Granero et al. [9]. This work extends the toolbox for fractal analysis, offering a more comprehensive and versatile approach to understanding complex data patterns.

Kroha and Skoula [12] introduced an innovative technical indicator known as the moving Hurst, founded on the Hurst exponent. Their research demonstrated that the moving Hurst outperforms the widely used moving average convergence divergence indicator regarding profitability, which relies on moving averages of time series data.

In a recent study, the authors in [13] introduced the Kolmogorov–Smirnov (KS) method for estimating the Hurst exponent. This novel approach was applied to three existing algorithms, including the generalised Hurst exponent and fractal dimension methods. Their findings revealed significant accuracy improvements, particularly for more extended data series, underscoring the KS method's potential to enhance Hurst exponent estimations.

Qian and Rasheed [14] investigated the application of the Hurst exponent as a classification tool for financial data over various periods. Their work utilised backpropagation neural networks, revealing that economic series characterised by a higher Hurst exponent were more accurately predictable than those with Hurst values approaching 0.5. So far, this is the first paper to implement a neural network for Hurst exponent estimation.

Several methods have been developed to estimate the Hurst exponent, including R/S analysis, detrended fluctuation analysis, and Periodogram Regression [10]. Among these, the R/S analysis, pioneered by Mandelbrot and Van Ness [15], remains the most widely used due to its simplicity and effectiveness. However, these estimation techniques are not without their limitations. One notable challenge for the R/S analysis is the issue of data length. Estimating the Hurst exponent using this method with a short time series can yield inadequate results [16,17]. This limitation is particularly pronounced in financial time series, where the complexity and volatility of data can complicate the estimation process. As a result, addressing these limitations and enhancing our ability to accurately estimate the Hurst exponent becomes a critical endeavour in financial modelling and prediction.

Artificial Neural Networks (ANNs) have been known to the scientific community since the 1940s, with their origins often traced back to the research of McCulloch and Pitts in 1943 [18]. However, early research on ANNs saw limited popularity due to the computational limitations of the time. The recent rise in interest and progress in this area can be attributed to the exponential growth in computing power, particularly in terms of data storage and processing speed [19]. These technological advancements have facilitated significant improvements in machine learning and artificial intelligence, making ANNs more feasible for a broader range of applications.

As defined by [20], ANN is an "information-processing system that has certain performance characteristics in common with biological neural networks". It mimics the function of the human brain [19], and is typically structured into layers—input, hidden, and output—where each layer consists of neurons. When there is more than one hidden layer, the architecture is called a DNN [21,22]. DNNs have been particularly influential in fields like computer vision, image processing, pattern recognition, and cybersecurity [23–26], as they automatically learn features from data rather than relying on hand-crafted ones, with deep layers capable of capturing more complex variances [27]. However, DNNs also face challenges such as stability, robustness, and adversarial perturbations, as discussed in [28,29].

Deep neural networks offer a promising avenue to improve the estimation of the Hurst exponent [30,31]. They can effectively handle the challenges posed by short data series by recognising latent features and relationships within the data that might not be apparent to human analysts. Additionally, these models have the potential to adapt and evolve as new data become available, thus enhancing their predictive capabilities in dynamic financial markets. As the intersection of advanced data science and economic analysis continues to grow, the integration of deep learning into Hurst exponent estimation represents a forward-looking approach aimed at overcoming the limitations of traditional methods and further refining our understanding of extended memory properties in financial time series [32].

Often composed of multiple layers of interconnected neurons, these networks can capture complex patterns, latent dependencies, and non-linear trends that traditional statistical methods struggle to grasp. Deep learning techniques have demonstrated significant potential in various financial applications, including stock price prediction, risk assessment, algorithmic trading, and fraud detection. Their adaptability and ability to process vast amounts of data have opened up new horizons for researchers and analysts seeking to gain deeper insights into the dynamics of financial markets and enhance decision-making processes [22,33,34]. However, it is essential to acknowledge that deep neural networks also present challenges related to data quality, interpretability, and overfitting, which require careful consideration in their application to financial time series analysis.

A similar machine learning method is the recurrent neural network method (RNNs) [35]. RNNs are a class of neural networks designed for sequential data processing, leveraging recurrent connections that allow information to persist across time steps. This structure enables RNNs to handle tasks like time series forecasting, natural language processing, and speech recognition, where previous input influences current predictions; see [36,37] and the references therein. However, standard RNNs often face challenges with long-term dependencies due to vanishing or exploding gradient problems during training. To mitigate these issues, architectures such as long short-term memory (LSTM) and gated recurrent units (GRUs) were introduced, offering improved performance in capturing long-range dependencies [38,39].

This study addresses a key challenge in Hurst exponent estimation: accurately estimating the exponent in short financial time series. The primary research question is whether deep neural network models can overcome the limitations of traditional methods such as R/S analysis and detrended fluctuation analysis, which often require longer time series data for reliable estimates. By developing a new DNN model, this research aims to improve the accuracy of Hurst exponent estimation in shorter datasets, advancing the current state of estimation techniques. Our model harnesses the power of deep learning to capture complex patterns and dependencies that conventional statistical methods might miss, offering a more robust and adaptable solution for financial time series analysis.

The primary objective of this study is to address a central problem in the estimation of the Hurst exponent H: the inherent challenge in achieving accurate estimates for short financial time series $\{X_t\}_{t=1}^{N}$, where N is relatively small. Traditional methods, such as rescaled range (R/S) analysis and detrended fluctuation analysis (DFA), often perform poorly for small N due to their dependence on larger sample sizes for consistent statistical behaviour.

We hypothesise that deep neural network (DNN) models can alleviate these limitations by leveraging their capacity to learn complex, non-linear dependencies in data, particularly in short time series. The core research question can be formalised as follows:

1. Research Problem: given a time series $\{X_t\}_{t=1}^{N}$, where N is small, estimate the Hurst exponent $H \in (0,1)$ such that

$$\mathbb{E}[|X_{t+\tau} - X_t|^q] \propto \tau^{Hq}$$

for a lag τ and scaling exponent q. Traditional methods often rely on large N for reliable estimation, whereas this research seeks to improve estimation accuracy for small N.

2. Traditional Methods: Conventional estimators, such as R/S analysis, are typically defined by the rescaled range statistic:

$$R(N)/S(N) \propto N^H,$$

where $R(N)$ is the range of partial sums of the deviations of X_t from its mean, and $S(N)$ is the standard deviation. These methods exhibit increasing variance in estimates as N decreases.

3. Proposed Approach: We propose a DNN-based model $f_\theta(\{X_t\}_{t=1}^{N})$, parameterised by θ, designed to estimate the Hurst exponent H. This model will be trained to minimise the loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{\text{train}}\left[\left(f_\theta(\{X_t\}_{t=1}^{N}) - H_{\text{true}}\right)^2\right],$$

where H_{true} is the ground truth Hurst exponent for the training set, and $f_\theta(\{X_t\}_{t=1}^{N})$ represents the predicted Hurst exponent from the DNN.

4. Advancements: The DNN's ability to approximate non-linear functions of the time series data allows it to capture dependencies and patterns that may not be evident in traditional linear or statistical methods. Precisely, it can model the scaling law:

$$\mathbb{E}[|X_{t+\tau} - X_t|^q] \propto \tau^{Hq}$$

with greater accuracy for small N, thus potentially reducing estimation error as N decreases.

Accordingly, our main aim is to propose an effective novel approach within the financial time series analysis field, specifically by providing a more robust, data-driven solution for Hurst exponent estimation, particularly when dealing with short and noisy datasets that challenge conventional methods.

The subsequent sections of the paper are organised as follows: Section 2 briefly discusses fractional Brownian motion and the Hurst exponent and revisits two popular methods for estimating it. Section 3 delves into the concept of deep neural networks, designing a new architecture tailored explicitly for Hurst exponent estimation. Section 4 presents the results and discussions, Section 5 outlines the further research and future developments, and Section 6 offers the conclusions.

2. Fractional Brownian Motion and Hurst Exponent

This section will briefly review the concepts related to fractional Brownian motion (fBM) and Hurst exponents. The information presented here is based on various references, including [15,40–43].

Fractional Brownian motion (fBm) is a stochastic process that is defined on a probability space denoted as $(\Omega, \mathcal{F}, \mathbb{P})$. In this notation, $\mathcal{F}$ represents a σ-field, a collection of subsets of the sample space Ω, and $\mathbb{P}$ is a probability measure.

The fBm process is characterised by the Hurst exponent, which provides valuable information about its properties. It is typically denoted as H, a measure of the long-term memory or persistence of the time series generated by fBm. The value of H ranges between 0 and 1.

Definition 1. *A fBm (B_t^H) with a Hurst exponent $H \in (0,1)$ is a continuous and centered Gaussian process satisfying the following conditions:*

$$\mathbb{E}(B_t^H) = 0 \quad \forall t \in \mathbb{R},$$
$$\mathbb{E}[B_t^H B_s^H] = \frac{1}{2}(|t|^{2H} + |s|^{2H} - |t-s|^{2H}).$$

The fractional Brownian motion B^H has the following properties:

1. $B_0^H = 0$,
2. $\mathbb{V}[B_t^H] = \mathbb{E}((B_t^H)^2) = t^{2H}$,
3. Let $\Delta B_{t,s}^H = B_t^H - B_s^H$, be fractional Bm increment, then
$$\mathbb{E}(\Delta B_{t,s}^H) = 0,$$
and,
$$\mathbb{E}((\Delta B_{t,s}^H)^2) = |t-s|^{2H}.$$
Hence, $B_{t+s}^H - B_s^H$ has the same distribution as B_t^H, for any $s, t \in \mathbb{R}$,
$$B_{t+s}^H - B_s^H \sim B_t^H.$$
It has homogeneous increments (stationary increments).
4. In general, increments of fBm are not independent like those of classical Bm. See the covariance of two non-overlapping increments:
$$\mathbb{E}(\Delta B_{t,s}^H \Delta B_{s,0}^H) = \frac{1}{2}[t^{2H} - s^{2H} - (t-s)^{2H}].$$
Consequently, for $H \neq \frac{1}{2}$, the increments of fBm are correlated.
$$\mathbb{E}(B_t^{\frac{1}{2}} B_s^{\frac{1}{2}}) = \frac{1}{2}[|t| + |s| - |t-s|] = \min(s,t)$$
5. It is a self-similar process, i.e., the process $a^{-H} B_{at}^H$ has the same law as B_t^H.

Remark 1. *Thus, B_t^H is an incremental stationary continuous Gaussian process, which means that B_t^H follows a normal distribution with mean zero and variance t^{2H}. When $H = \frac{1}{2}$, the fractional Brownian motion reduces to the standard Brownian motion.*

The Hurst exponent plays a crucial role in characterising the behaviour of a stochastic process. When $H < \frac{1}{2}$, the process is known as anti-persistent, with future increments tending to move in the opposite direction of past increments. Conversely, when $H > \frac{1}{2}$, the process is termed persistent, indicating that future increments tend to continue in the same direction as past increments. When $H = \frac{1}{2}$, the process exhibits no correlation between increments, making them independent of the past. Understanding the value of H is essential in identifying the stochastic process's long-range dependence and self-similarity properties.

The auto-covariance function of the stationary process of discrete increments ΔB^H with increments of length 1 and a lag size τ can be approximated as

$$\gamma_H(\tau) = \frac{1}{2}[(\tau+1)^{2H} - 2\tau^{2H} + (\tau-1)^{2H}].$$

Interestingly, this approximation corresponds to the second derivative of the function $f(\tau) = \tau^{2H}$, indicating that for large values of τ, the auto-covariance function behaves similarly to $f''(\tau) = 2H(2H-1)\tau^{2H-2}$. Considering the theory of infinite sums, we find that the infinite sum of the latter function exists only for exponents smaller than minus one, implying that the sum exists for $H < \frac{1}{2}$ but is unbounded for $H > \frac{1}{2}$. Consequently, we can classify the antipersistent fBm as having intermediate memory, while the persistent one exhibits long memory. Figures 1 and 2 illustrate these facts.

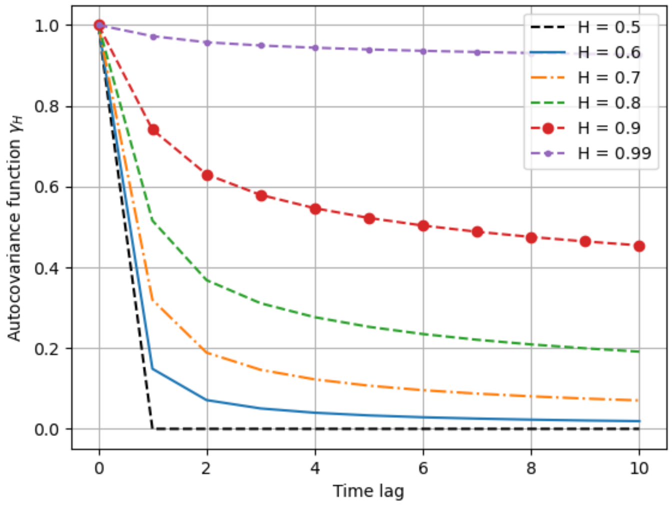

Figure 1. Auto-covariance function of fBm for the case of persistence.

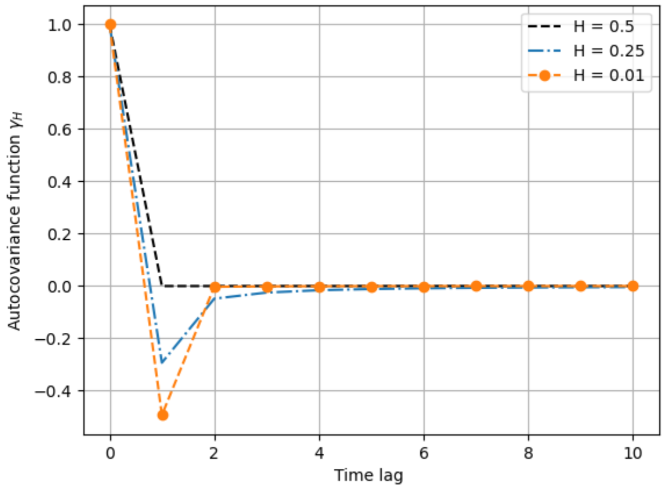

Figure 2. Auto-covariance function of fBm for the case of anti-persistence.

2.1. The Rescaled Range Analysis—R/S Analysis Method

The R/S analysis entails calculating the range of partial sums of deviations of time series segments from their means, which are then rescaled by their standard deviations. We adopt the algorithm described in [12,14] and their respective references to determine the Hurst exponent. Given a stochastic time series denoted as X_i, where $i = 1, \dots, N$ represents the data points corresponding to the underlying asset, the R/S algorithm (Algorithm 1) follows the steps below:

1. Divide the time series into $d = 2^k (k = 0, 1, \dots)$ adjacent sub-series of the same length n, so that $2^k n = N$.
2. Compute the mean and standard deviation for each sub-series, $\mu_m = \frac{1}{n}\sum_{i=1}^{n} X_i,$ and $S_m = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu_m)^2}\; m = 1, \dots d.$
3. Normalise the data by subtracting the sample mean:

$$Z_{r,m} = X_{r,m} - \mu_m, \quad r = 1, 2, \dots, n;$$

4. Create the cumulative time series $Y_{r,m}$:

$$Y_{r,m} = \sum_{i=1}^{r} Z_i, \quad r = 1, 2, \ldots, n.$$

5. Compute the range time series R:

$$R_m = \max(Y_{1,m}, Y_{2,m}, \ldots, Y_{i,m}) - \min(Y_{1,m}, Y_{2,m}, \ldots, Y_{i,m}).$$

6. Compute the rescaled range time series by dividing R_m by S_m: R_m/S_m. Averaging over the whole set of sub-samples d, the mean value of the rescaled range for sub-series of length n is then

$$(R/S)_n = \frac{1}{d}\sum_{m=1}^{d}\frac{R_m}{S_m},$$

Note that $(R/S)_n$ is averaged over the regions $[X_1, X_n]$, $[X_{n+1}, X_{2n}]$ until $[X_{(m-1)n+1}, X_{mn}]$.

7. The Hurst exponent H is estimated by fitting

$$\mathbb{E}[(R/S)_n] = Cn^H, \tag{1}$$

to the data.

Then, the parameter H could be determined by a log–log plot of the last relation (1). Taking the logarithm of (1) both sides,

$$\log(\mathbb{E}[(R/S)_n]) = H\log(n) + \log C.$$

Algorithm 1 R/S analysis algorithm.

```
Require: Stochastic time series data, $X$.
 1: $N$ = the length of $X$
 2: $K$ = number of possible divisors of N
 3: Empty lists to store the means of $R/S$ and $n$ sizes of sub-series, say $meanRS = [\,], ns = [\,]$
 4: for all $k$ in $K$ do
 5:     divide $X$ into $d$ sub-series
 6:     $n$ - the size of each sub-series
 7:     append $ns$ by $n$
 8:     Create a matrix $R$ where the rows be the sub-series, $d \times n$
 9:     $meanSub, stdSub$- calculate the mean and standard deviation of each sub-series, the
    rows of $R$
10:     Create a zero matrix $Z$ of the same size as $R$ to be a updated by normalised data
11:     for $i$ in $(0, 1, \ldots, d)$ do
12:         $Z[i] = R[i] - meanSub[i]$
13:     end for
14:     $Y$ the cumulative time series along the row of $Z$
15:     Find the range $Rn = \max(Y) - \min(Y)$ along the rows,
16:     Rescale the range by dividing with standard deviation $RS = Rn/stdSub$
17:     append $meanRS$ by the average of $RS$
18: end for
19: $x = \log(ns)$ and $y = \log(meanRS)$
20: Apply regression on $y = Hx + C$
        return $H, C, y$.
```

2.2. Multifractal Detrended Fluctuation Analysis (MFDFA)

Here, we summarise the definition of MFDFA based on the following literature [44–55].

Definition 2 ([53])**.** *Given a time series $x(t)$(in time or space t) with N data points, discretised as $X_i, i = 1, 2, \ldots, N$, conduct the following:*

1. *Integrate the series*
$$Y_i = \sum_{k=1}^{i} (X_k - \langle X \rangle) \tag{2}$$
where the value $\langle X \rangle = \frac{1}{N} \sum_{i=1}^{N} X_i$ is the average of the time series. The process Y_i is the original time series' cumulative sum or profile.
2. *Partition the new time series Y into non-overlapping boxes of equal size s, therefore obtaining $N_s = int(N/s)$ segments. Given the total length of the data is not always a multiple of the segment's length s, discard the last points of the data.*
3. *Considering the same data, apply the same procedure but now discard these instead of the first data points. One now has $2N_s$ boxes of the time series.*
4. *Within each box, fit a polynomial y_v of the order m*
5. *Calculate the variance in the difference of the data to the polynomial fit. Compute the root-mean-square deviation from the local trend (local fluctuation),*
$$F(v,s) = \sqrt{\frac{1}{s} \sum_{i=1}^{s} (Y_{(v-1)s+i} - y_{(v-1)s+i})^2}. \tag{3}$$
for $v = 1, 2, \ldots, N_s$, where $y_{(v-1)s+i}$ is the polynomial fitting for the segment $Y_{(v-1)s+i}$ of length s, fitted via least-squares. The order of the polynomial y_v can be freely chosen, giving rise to the denotes $(MF)DFA1, (MF)DFA2, \ldots, (MF)DFAm$, dependent on the chosen degree m of the polynomial.
6. *Now $F(v,s)$ is a function of each variance of each data segment and of the different $s-$length segments chosen. Define the $q-th$ order fluctuation function by averaging over the N_s variances of the segments of size s*
$$F_q(s) = \left\{ \frac{1}{N_s} \sum_{i=1}^{N_s} [F(v,s)]^{q/2} \right\}^{1/q}.$$
7. *Repeat step 2–6 for various box sizes s (different scales) to provide a relationship between $F_q(s)$ and s.*

Make the log–log plot $\log s - \log F_q(s)$. *A straight line of slope α in the log–log plot indicates a statistical self-affinity of the form*
$$F_q(s) \propto s^{h(q)} \tag{4}$$
where $h(q)$ is the generalised Hurst exponent or self-similarity exponent, which depends on q if the data are multifractal and relate directly to the Hurst index. The generalised Hurst exponent $h(q)$ is obtained by finding the slope of the $F_q(s)$ curve in the log–log plots.

Remark 2.

- *If the data are monofractal, $h(q) = H$ is independent of q and is simply the Hurst index H.*
- *If the data are multifractal, the dependence on q can be understood by studying the multifractal scaling exponent $\tau(q)$, given by*
$$\tau(q) = qh(q) - 1,$$
which depends on the generalised Hurst exponent $h(q)$.

3. Deep Neural Networks

Similar to the approach outlined in [56] and the references, we employ a deep neural network architecture depicted in the schematic diagram in Figure 3.

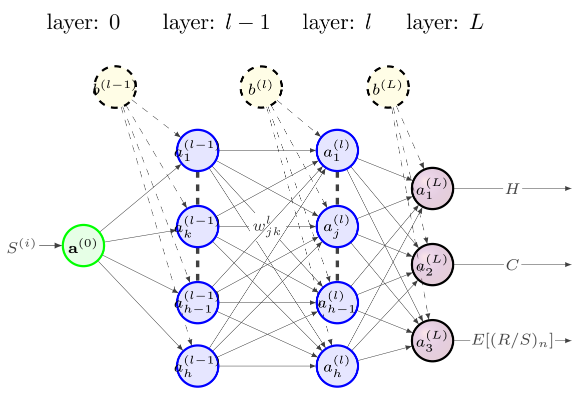

Figure 3. The schematic diagram of DNN.

This network comprises an input layer with neurons corresponding to stochastic time series data, specifically stock market prices over a given period. The output layer consists of three neurons, corresponding to the Hurst exponent (H), the constant (C) in the R/S analysis equation $\log(\mathbb{E}[(R/S)_n]) = H\log(n) + \log(C)$, and the mean $\mathbb{E}[(R/S)_n]$. Additionally, we incorporate $L-1$ hidden layers, with the number of neurons in each hidden layer determined based on the model's performance.

The feedforward network operates as follows: for each stochastic time series denoted as $S^{(i)}$ (in our case, representing stock prices over a range of trading times), the learning process begins by initialising $\mathbf{a}^{(0)} = S^{(i)}$. We label the nodes on layer $l-1$ as k and nodes on layer l as j. Then, the value entering the j^{th} node of the layer l is given by the following:

$$z_j^l := \sum_{k=1}^{n_l} w_{jk}^l a_k^{l-1} + b_j^l, \tag{5}$$

where n_l denotes the number of nodes in layer l. The matrix form of Equation (5) is as follows:

$$\mathbf{z}^l = W^l \mathbf{a}^{l-1} + \mathbf{b}^l,$$

where the matrix W^l contains all the multiplicative parameters, i.e., the weights w_{jk}^l, and $\mathbf{b}^l$ represents the bias. The values of Equation (5) are then passed to the next hidden layer through an appropriate activation function, denoted by σ^l. Therefore, the values for the next layer are expressed as follows:

$$\mathbf{a}^l = \sigma^l(\mathbf{z}^l) = \sigma^l(W^l\mathbf{a}^{l-1} + \mathbf{b}^l).$$

For the m different stochastic time series (learning samples), $1 \le i \le m$, and for each layers, $1 \le l \le L$, we have the following:

$$\mathbf{z}^{l(i)} = W^l\mathbf{a}^{l-1(i)} + \mathbf{b}^l, \text{ and } \mathbf{a}^{l(i)} = \sigma^l(\mathbf{z}^{l(i)}).$$

Now, by arranging $\boldsymbol{S} = [S^{(1)}, \ldots, S^{(m)}]$ as a matrix, and similarly organising the $\mathbf{z}^{l(i)}$'s and the $\mathbf{a}^{l(i)}$'s into the matrices Z^l and A^l, respectively, we obtain the following matrix form for $1 \le l \le L$:

$$A^0 = \boldsymbol{S}, \quad Z^l = W^L A^{l-1} + \mathbf{b}^l, \text{ and } A^l = \sigma^l(Z^l).$$

An optimisation technique will be applied to minimise the parameters using the loss function, defined as the mean squared error between the network output and the target.

3.1. Data

It is well known that, in general, the performance of deep neural networks becomes more accurate with increased training data points. Moreover, estimating the Hurst exponent requires long-time-series stochastic data. For instance, R/S analysis requires a minimum length of 100 periods. Accordingly, we considered the adjusted closing prices of 446 stocks randomly selected from the S&P 500 index in training the deep neural network models. This dataset spans 2010 trading days, from 3 January 2012 to 27 December 2019. Table 1 shows a partial view of the data and Table 2 summarizes the statistical information of the data.

Table 1. The partial view of the input data for the training. The nearest 106 days' close prices of 103 randomly selected tickers.

Date	A	AAL	AAPL	ABT	…	XYL	YUM	ZBH	ZBRA
3 January 2012	23.41	4.83	12.40	21.06	…	21.34	33.17	47.16	35.72
4 January 2012	23.22	4.74	12.47	20.98	…	22.04	33.39	46.52	35.45
5 January 2012	23.74	5.16	12.61	20.93	…	21.73	33.65	47.07	35.40
6 January 2012	24.00	5.28	12.74	20.74	…	21.36	33.89	47.19	35.11
9 January 2012	24.63	5.39	12.72	20.74	…	21.45	33.76	47.54	34.95
⋮	⋮	⋮	⋮	⋮	…	⋮	⋮	⋮	⋮
20 December 2019	82.44	28.93	67.80	79.80	…	74.48	92.55	140.76	252.49
23 December 2019	82.50	29.19	68.91	80.43	…	74.80	91.84	141.62	256.73
24 December 2019	82.54	29.07	68.97	80.37	…	74.51	92.32	141.28	254.33
26 December 2019	82.69	29.56	70.34	80.37	…	74.47	93.66	140.97	254.42
27 December 2019	82.66	28.34	70.32	80.48	…	74.69	93.76	140.70	256.00

Table 2. The summary of the statistical description for selected stock price returns.

	AAPL	SYY	TER	VZ	WBA	WDC	WELL	XYL
count	2009	2009	2009	2009	2009	2009	2009	2009
mean	0.0010	0.0007	0.0010	0.0005	0.0005	0.0007	0.0005	0.0007
std	0.0161	0.0108	0.0193	0.0104	0.0156	0.0240	0.0123	0.0143
min	−0.1236	−0.0944	−0.1639	−0.0468	−0.1434	−0.1818	−0.0799	−0.1026
25%	−0.0066	−0.0042	−0.0089	−0.0057	−0.0071	−0.0106	−0.0057	−0.0067
50%	0.0009	0.0006	0.0014	0.0007	0.0005	0.0006	0.0014	0.0008
75%	0.0096	0.0060	0.0105	0.0063	0.0084	0.0131	0.0075	0.0090
max	0.0887	0.0983	0.2048	0.0768	0.1179	0.2094	0.0574	0.1244

Figure 4 shows the plots of some of the selected stock prices from the input data and their corresponding price changes (returns).

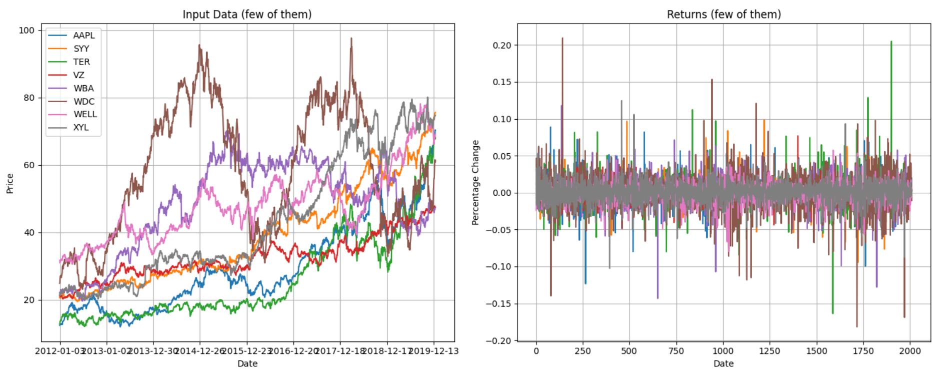

Figure 4. The plot of the price of some parts of the input data and their corresponding price changes.

It is well known that financial models often assume standard Brownian motion when modelling financial theories. Capital market theory has primarily been based on this assumption. However, empirical evidence frequently suggests that this assumption may not hold. This is clearly illustrated in Figure 5, which compares the distribution of returns with the standard normal distribution, and in Figure 6, which presents the Q–Q plots. These figures reveal heavy tails and skewness, indicating that returns exhibit leptokurtosis and may follow a different distribution, such as a fat-tailed distribution. This discrepancy highlights the need for more robust models that can account for these deviations from normality.

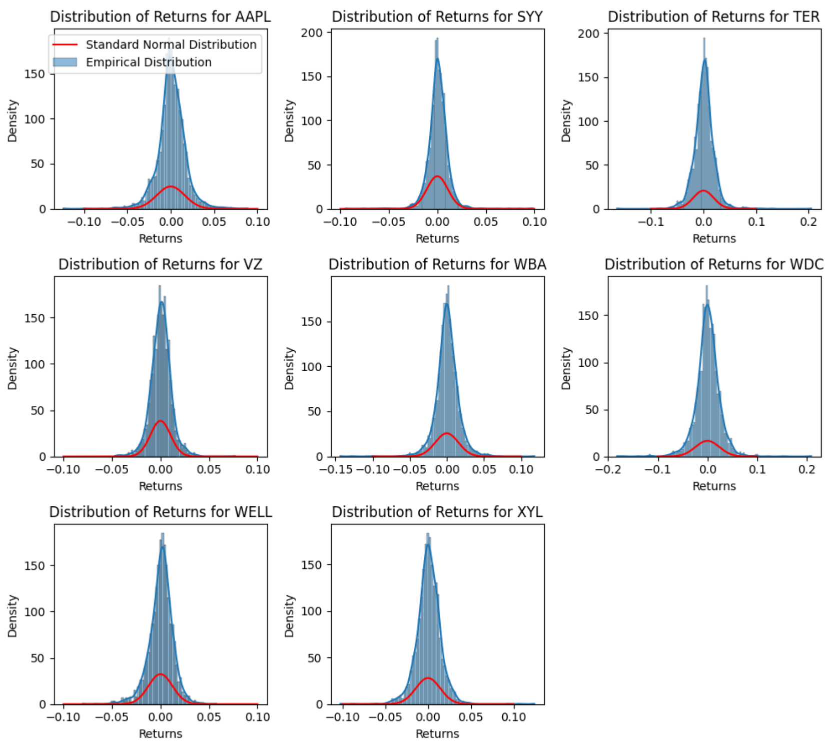

Figure 5. Comparing the distributions of the returns with standard normal distribution.

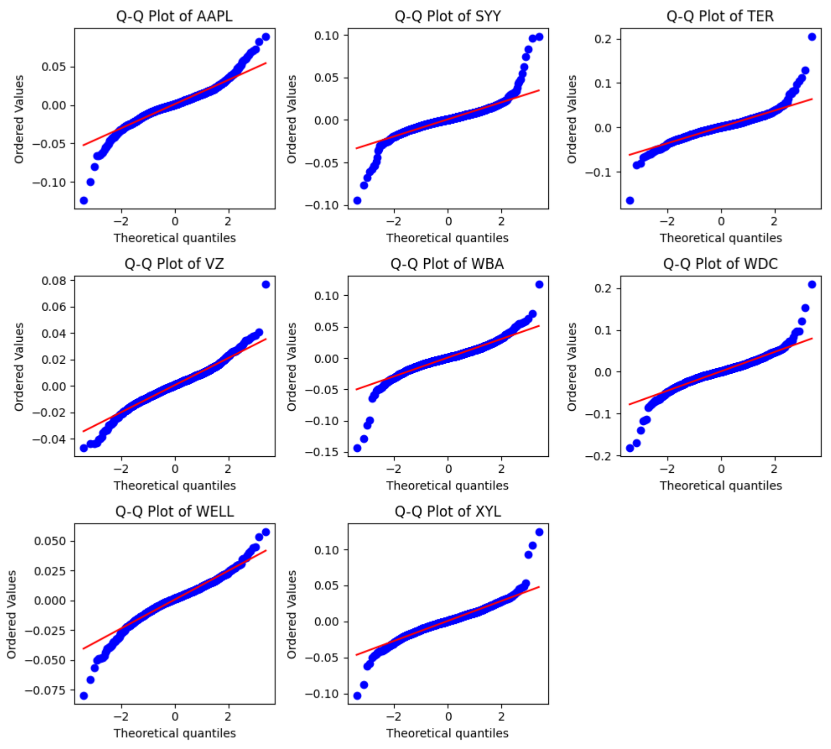

Figure 6. The Q–Q plots of returns on stock price.

Table 3 presents the results of the Shapiro–Wilk and Anderson–Darling tests for various stock tickers to assess the normality of their returns. The stocks show significant deviations from normality, as low p-values in the Shapiro–Wilk test and high Anderson–Darling statistics indicate. These results suggest that the returns for these stocks do not follow a normal distribution, often exhibiting heavy tails or skewness.

Table 3. Shapiro-Wilk and Anderson-Darling Test Results for Stock Returns Normality Assessment.

Ticker	Shapiro-Wilk Test	Anderson-Darling Test
AAPL	Statistic = 0.946, p-value = 0.000	Statistic = 20.106
SYY	Statistic = 0.862, p-value = 0.000	Statistic = 35.788
TER	Statistic = 0.919, p-value = 0.000	Statistic = 19.959
VZ	Statistic = 0.975, p-value = 0.000	Statistic = 8.060
WBA	Statistic = 0.919, p-value = 0.000	Statistic = 22.437
WDC	Statistic = 0.931, p-value = 0.000	Statistic = 18.935
WELL	Statistic = 0.965, p-value = 0.000	Statistic = 14.163
XYL	Statistic = 0.944, p-value = 0.000	Statistic = 12.521

3.2. DNN Based on R/S Analysis

Using the R/S analysis, we calculated the Hurst exponents (H), the constants (C), and $\mathbb{E}[(R/S)_n]$ for each of these stocks, as detailed in Table 4. These computed values were employed as the target variables.

Table 4. The partial view of computed H, C and $\mathbb{E}[(R/S)_n]$ using the R/S analysis for the S&P500 stock prices over 2010 days.

Tickers	H	C	$\mathbb{E}[(R/S)_n]$
A	0.49	0.99	16.56
AAL	0.60	0.72	26.01
AAPL	0.54	0.87	21.29
ABT	0.50	0.95	17.91
ACGL	0.57	0.78	21.82
⋮	⋮	⋮	⋮
IP	0.54	0.88	20.33
IPG	0.48	1.04	16.97
IRM	0.50	1.03	18.87
ISRG	0.51	0.92	18.53
⋮	⋮	⋮	⋮
XOM	0.48	1.06	17.12
XYL	0.51	0.94	18.42
YUM	0.48	1.02	16.32
ZBH	0.53	0.87	19.90
ZBRA	0.54	0.85	20.56

In the context of supervised learning and our deep neural network models, we considered three distinct approaches that differ in their selection of target variables:

1. The first model, which we denote as DNN-HCRS, employs three values as targets: the Hurst exponent, H, coefficient, C, and expected value of the rescaled range, $\mathbb{E}[(R/S)_n]$.
2. The second model, denoted by DNN-HC, focuses on two values as targets: the Hurst exponent, H, and coefficient, C.
3. The third model, which we denote by DNN-H, uses only one value as a target: the Hurst exponent H.

In all three cases, the input data comprises the returns on adjusted closing prices of the 2010 days for a total of 446 indices, as detailed in Table 4. Here we considered the three slightly different models with the expectation that different target data might result in different outcomes. Hence, the best one will be selected based on accuracy, computational time, and other advantages. For the deep neural architect, we followed the state-of-the-art by effectively utilising TensorFlow [57] and Keras [58] libraries, where we optimally selected the best model using KerasTuner [59] for each case.

3.2.1. DNN-HCRS

The first model utilises a deep neural network, where the target variables consist of three values, H, C, and $\mathbb{E}[(R/S)]$, computed from the R/S analysis. These variables correspond to the columns listed in Table 4. Among several optimised model architectures, one is summarised in Table 5. The model employs a ReLU activation function and undergoes training for 1500 epochs. The learning curve depicted in Figure 7 shows that both the training loss and validation loss reach a point of stability and demonstrate a good fit.

Table 5. The optimised choice for layers and neurons for the model DNN-HCRS.

Layer (Type)	Output Shape	Param #
dense (Dense)	(None, 50)	100,500
dense_1 (Dense)	(None, 74)	13,974
dense_2 (Dense)	(None, 198)	54,450
dense_3 (Dense)	(None, 48)	9552
dense_4 (Dense)	(None, 3)	147

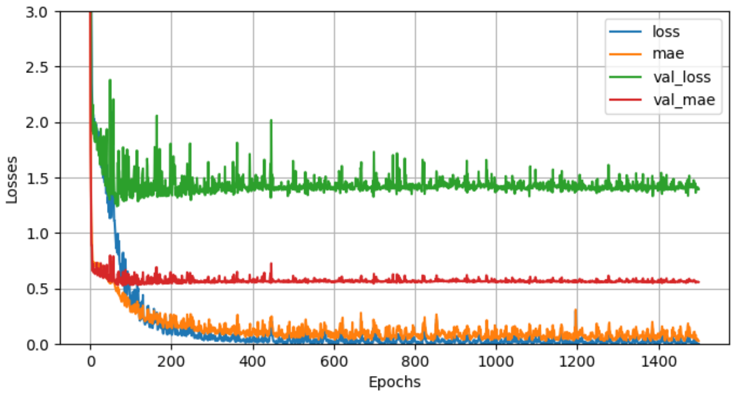

Figure 7. Learning curves for the first model DNN-HCRS. The mean training loss (loss) and accuracy (mae) were measured over each epoch, and the mean validation loss (val_loss) and accuracy (val_mea) were measured at the end of each epoch.

The comparison between the estimated Hurst exponent and the test data is depicted in Figure 8. The figure demonstrates that the model effectively learned or estimated the Hurst parameter for almost all the indices in the test dataset with few exceptions. Moreover, we calculated the mean absolute error (0.043), the standard deviation of the error (0.03), the minimum error (0.002), and the maximum error (0.123), which are summarised in Table 9. The anomaly suggests further investigation of the properties of the time series for the specified time periods.

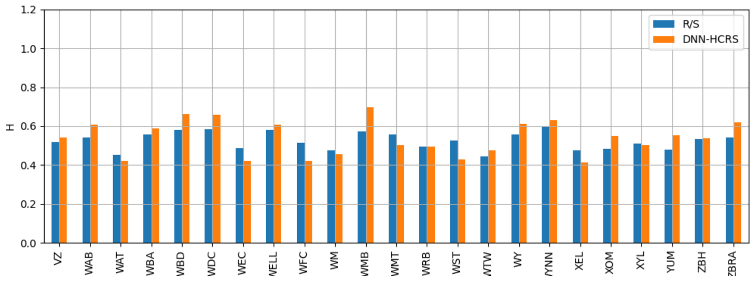

Figure 8. A plot for comparing the predicted and computed H for model DNN-HCRS.

3.2.2. DNN-HC

The second deep neural network model employs the two values, H and C, computed from the R/S analysis, as target variables. These variables correspond to the first two columns in Table 4. One of several optimised model architectures is detailed in Table 6.

Table 6. The optimised choice for layers and neurons for the model DNN-CH.

Layer (Type)	Output Shape	Param #
dense (Dense)	(None, 26)	52,260
dense_1 (Dense)	(None, 282)	7614
dense_2 (Dense)	(None, 126)	35,658
dense_3 (Dense)	(None, 14)	1778
dense_4 (Dense)	(None, 2)	30

The model utilises a ReLU activation function and undergoes training for 1500 epochs. The learning curve depicted in Figure 9 exhibits a good fit. The illustration shows that the training and validation accuracy exhibit a consistent upward trend throughout the training session, whereas the training loss and validation loss consistently decrease. Figure 10 compares the predicted Hurst exponent and the test data. Similar to the first model, we calculated the mean absolute error (0.045), the standard deviation of the error (0.037), the minimum error (0.002), and the maximum error (0.133), which are summarised in Table 9. The outcome is almost the same as that of the first model, with a slight difference.

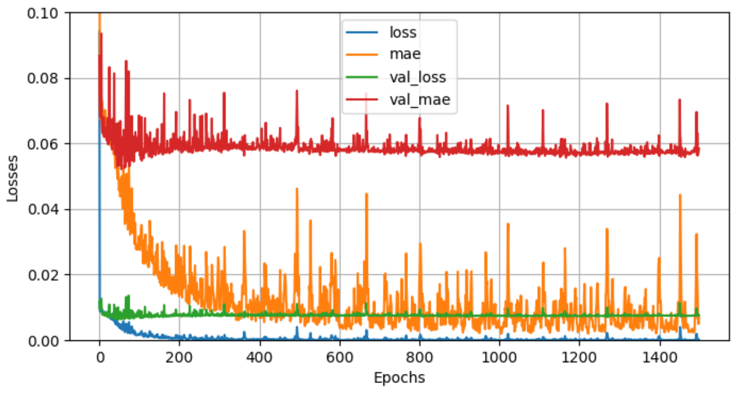

Figure 9. Learning curves for model DNN-HC. The mean training loss (loss) and accuracy (MAE) were measured over each epoch, and the mean validation loss (val_loss) and accuracy (val_mea) were measured at the end of each epoch.

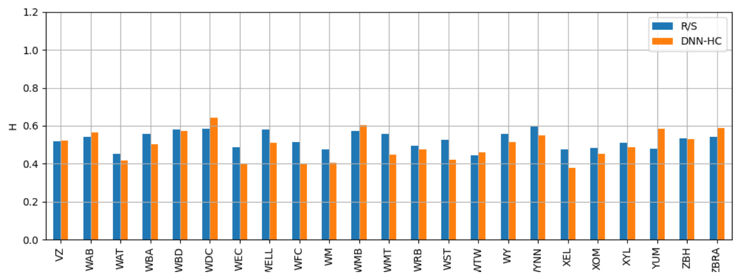

Figure 10. A plot for comparing the predicted and computed H for the model DNN-HC.

3.2.3. DNN-H

The third deep neural network model exclusively utilises the H value, calculated from the R/S analysis, as the target variable. This variable corresponds to the first column listed in Table 1. Among several optimised model architectures, one is summarised in Table 7. The model incorporates a ReLU activation function and undergoes training for 1500 epochs. The learning curve depicted in Figure 11 demonstrates a good fit. The illustration shows that the training and validation accuracy consistently show upward trends throughout the training session. In contrast, the training loss and validation loss consistently decrease. Moreover, the metrics show far better measurement compared to the two-model architecture.

Table 7. The optimised choice for layers and neurons for the model DNN-H.

Layer (Type)	Output Shape	Param#
dense (Dense)	(None, 26)	52,260
dense_1 (Dense)	(None, 222)	5994
dense_2 (Dense)	(None, 66)	14,718
dense_2 (Dense)	(None, 1)	67

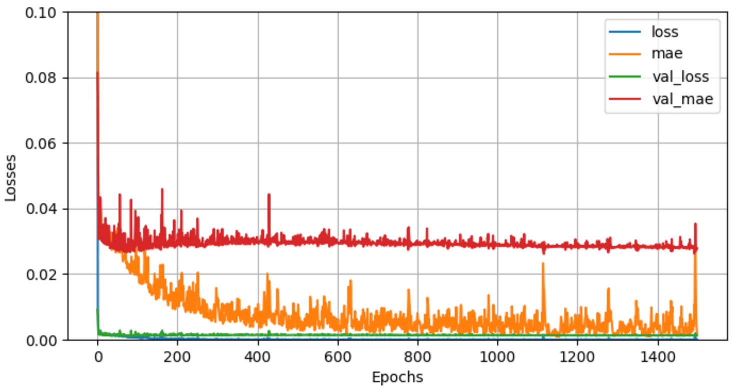

Figure 11. Learning curves for model DNN-H. The mean training loss (loss) and accuracy (mae) were measured over each epoch, and the mean validation loss (val_loss) and accuracy (val_mea) were measured at the end of each epoch.

The predicted Hurst exponent and the test data are compared in Figure 12. Additionally, the calculated mean absolute error, standard deviation of the error, maximum error, and minimum error shown in Table 9 are 0.043, 0.032, 0.001, and 0.108, respectively, demonstrating a slight improvement.

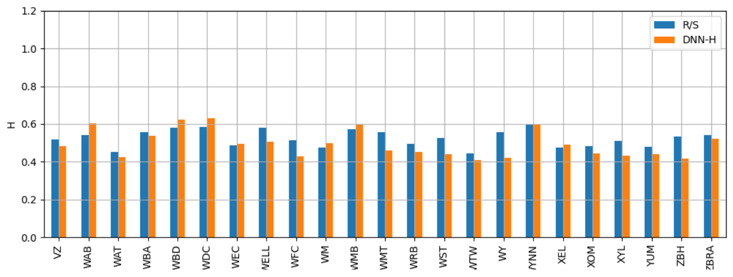

Figure 12. A plot for comparing the predicted and computed H for model DNN-H.

3.2.4. Comparison of the Three Models

All the figures, including Figures 7, 9 and 11, illustrate the models' performance. The learning curve of the third model architect indicated in Figure 11 shows the best fit. Furthermore, the comparison between the predicted Hurst exponent and the test data is depicted in Figures 8, 10 and 12, revealing the models' estimation capabilities. It is important to note that we examine 446 distinct stochastic time series, each representing the closing price movements for the nearest 2010 days. When evaluating the performance of the three models in predicting the Hurst exponent, relying solely on a single numerical metric such as the l_2 norm may not be entirely accurate. Instead, we assessed how closely each model aligns with the computed values through graphical comparisons.

Figure 13 illustrates that all three models successfully learned the Hurst exponent from the R/S analysis, with slightly different performances for different tickers. Furthermore, as outlined in Tables 8 and 9, we analysed statistical descriptors and examined the corresponding errors depicted in Figure 14. Across all these measures, it becomes evident that there is no significant difference between the three approaches in terms of the accuracy of the estimation of H. However, we observe that the estimation of H for some tickers has significant errors across all models. These discrepancies may arise from the time series properties of the data, which require further investigation; in Section 3.4, possible explanations based on other statistical measures, volatility, sharp ratio, and skewness are used for this.

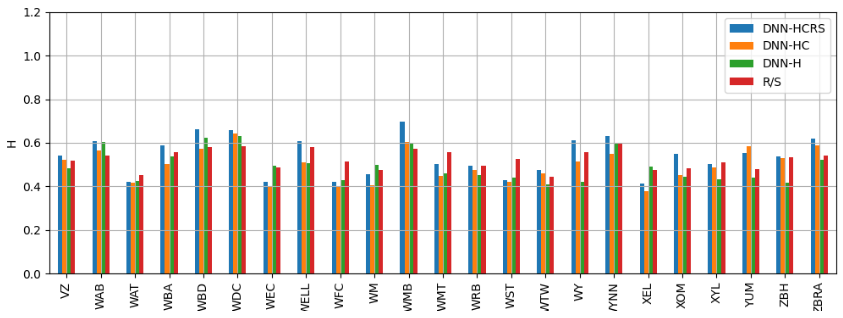

Figure 13. The comparison of the three models with the computed H using the R/S analysis.

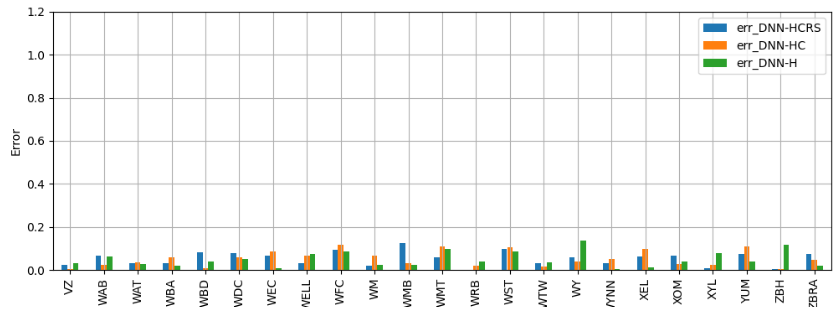

Figure 14. A plot showing the absolute error between the H computed with the new methods and R/S analysis.

Table 8. Statistical description of the three models' prediction compared to the computed Hurst exponents.

Index	DNN-HCRS	DNN-HC	DNN-H	R/S
count	23	23	23	23
mean	0.539	0.497	0.494	0.525
std	0.089	0.075	0.072	0.045
min	0.414	0.379	0.408	0.444
25%	0.465	0.435	0.437	0.484
50%	0.542	0.501	0.484	0.526
75%	0.611	0.557	0.531	0.559
max	0.696	0.642	0.632	0.600

Table 9. The mean absolute error, standard deviation of the error, minimum error, and maximum error of the estimated Hurst exponent compared to the target for the three models.

Error	DNN-HCRS	DNN-HC	DNN-H
mean	0.043	0.045	0.043
std	0.030	0.037	0.032
min	0.002	0.002	0.001
max	0.123	0.133	0.108

Let H_i be the target Hurst exponent of the ith index; $i = 1, 2, \ldots, 23$, and the corresponding predicted value is denoted by $\hat{H}_i$. Then, the absolute error is given by

$$e_i = |H_i - \hat{H}_i|.$$

The mean absolute error is given by

$$\bar{e} = \frac{1}{23}\sum_{i=1}^{23} e_i.$$

The standard deviation of the error is computed by

$$\mathrm{SDE} = \sqrt{\frac{1}{23}\sum_{i=1}^{23}(e_i - \bar{e})^2},$$

where the minimum and maximum are given by

$$\min(e_1, e_2, \ldots, e_{23}) \quad \text{and} \quad \max(e_1, e_2, \ldots, e_{23}),$$

respectively. The computed values for the corresponding three models are indicated in Table 8.

3.3. DNN Model Based on DFA(DNN-DFA)

Figure 15 depicts a comparison of the two well-known methods, the R/S analysis and the DFA. The data described above in the estimation using these two algorithms have minor differences. Here, the estimations based on R/S analysis are slightly higher than the ones based on DFA.

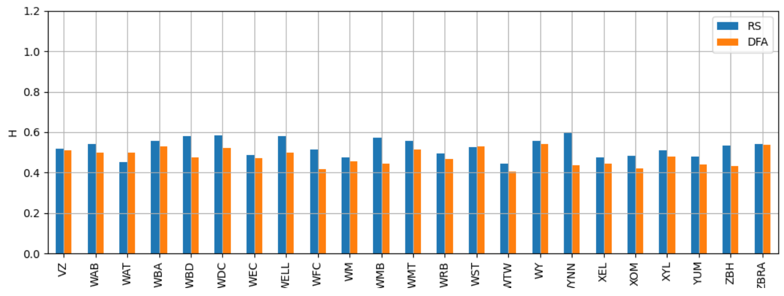

Figure 15. The comparison of Hurst exponent estimation using the R/S analysis and the DFA.

Now, we investigate the model by changing only the computation of the target. In this case, the target is derived from the DFA algorithm and computed using the Python code provided in [53]. An advantage of using DFA instead of R/S scale analysis is that the former can be implemented on a short time series. On the other hand, the latter necessitates a time series data length of at least 100. Regarding model evaluation, we found the same result as those of the previous models; see Figure 16.

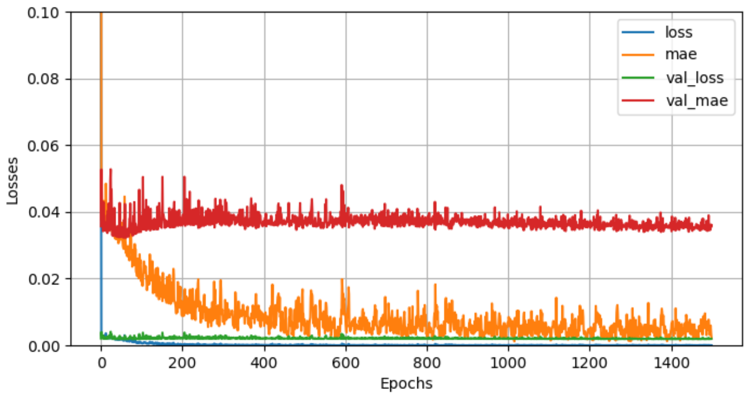

Figure 16. A plot showing the evaluation of the DNN-FDA based model.

The abilities to estimate the Hurst exponent can be compared in Figure 17. The model predicted the H for most of the index with few exceptions. Figure 18 shows the comparison of Hurst exponent estimates using DNN, where the input data are derived from both R/S and DFA. The error comparison in Figure 19 and more experiments indicated that the deep neural network based on R/S performs better than the DFA model for this specific data and code implementation.

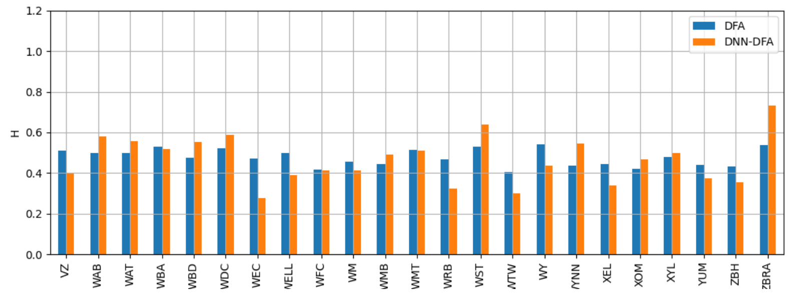

Figure 17. A plot demonstrating how the DNN-FDA model accurately estimated the Hurst exponent.

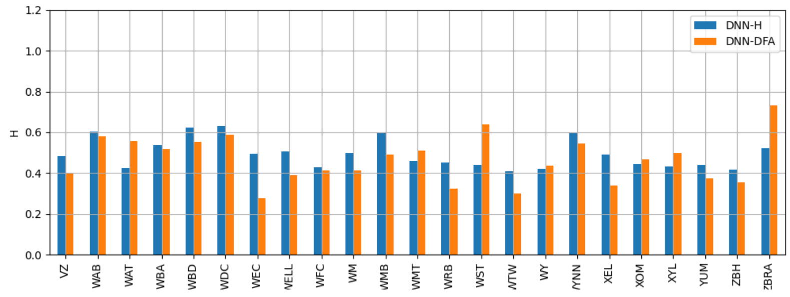

Figure 18. A comparison of Hurst exponent estimates using DNN, where the input data are derived from both R/S and DFA.

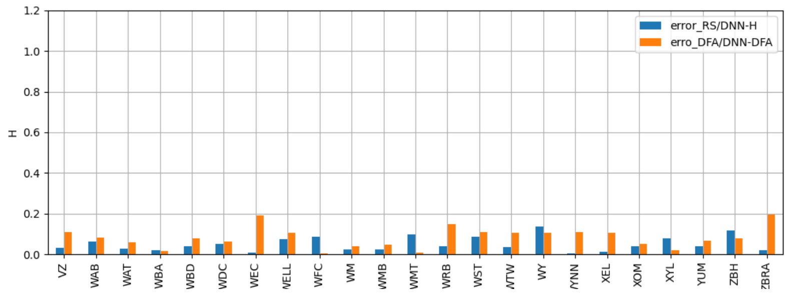

Figure 19. The comparison of the error between DNN-H learned from R/S and DFA.

3.4. Analysing Volatility, Sharpe Ratio, and Skewness Against the Hurst Exponent

Table 10 shows the correlation between the four estimation methods (i.e., the three DNN approaches and the R/S analysis) and the volatility, Sharpe ratio, and skewness. The results confirm that the three newly proposed DNN methods are highly correlated. Moreover, there is a strong correlation between the new techniques and the R/S analysis.

Table 10. A correlation table for the four methods of Hurst estimation, volatility, Sharpe ratio and skewness.

	DNN -HCRS	DNN -HC	DNN -H	R/S	Volatility	Sharpe Ratio	Skewness
DNN-HCRS	1.00	0.93	0.95	0.78	0.74	−0.73	−0.19
DNN-HC	0.93	1.00	0.95	0.74	0.79	−0.67	−0.31
DNN-H	0.95	0.95	1.00	0.76	0.76	−0.72	−0.19
R/S	0.78	0.74	0.76	1.00	0.68	−0.52	0.16
Volatility	0.74	0.79	0.76	0.68	1.00	−0.59	−0.21
Sharpe Ratio	−0.73	−0.67	−0.72	−0.52	−0.59	1.00	0.27
Skewness	−0.19	−0.31	−0.19	0.16	−0.21	0.27	1.00

Examining the volatility against the Hurst exponent can help identify whether high or low volatility is associated with more persistent or mean-reverting behaviour. High volatility might be related to higher H values if the market shows persistent trends or lower H values if the market is highly random. Specifically, in the stock price data considered in this paper, volatility is highly correlated with the Hurst exponent, with a stronger correlation observed when H is estimated via the DNN models.

The Sharpe ratio measures the risk-adjusted return. The formula is as follows:

$$\text{harpe ratio} = \frac{\text{return} - \text{risk-free rate}}{\text{standard deviation of excess return}}.$$

It is expected that higher H values might be associated with higher Sharpe ratios if persistent trends lead to better risk-adjusted returns. However, the correlation table (Table 10) shows a strong negative correlation between the Hurst estimation and the Sharpe ratio for the data under consideration.

Skewness indicates the symmetry of the return distribution. We expect that higher skewness might correlate with lower H values in a mean-reverting market or higher H values in a trending market.

The observation from the comparison in Figure 20 indicates that the estimation based on the deep neural network is slightly higher for some stock prices, such as WDC and WMB. For these tickers, volatility is higher, the Sharpe ratio is smaller, and skewness is negative. On the other hand, for tickers like WFC and WST, the estimation based on the deep neural network is slightly lower compared to that observed in the R/S analysis. In these cases, volatility is lower, the Sharpe ratio is higher, and skewness is positive. This suggests that the new model corrects missing information from the R/S analysis.

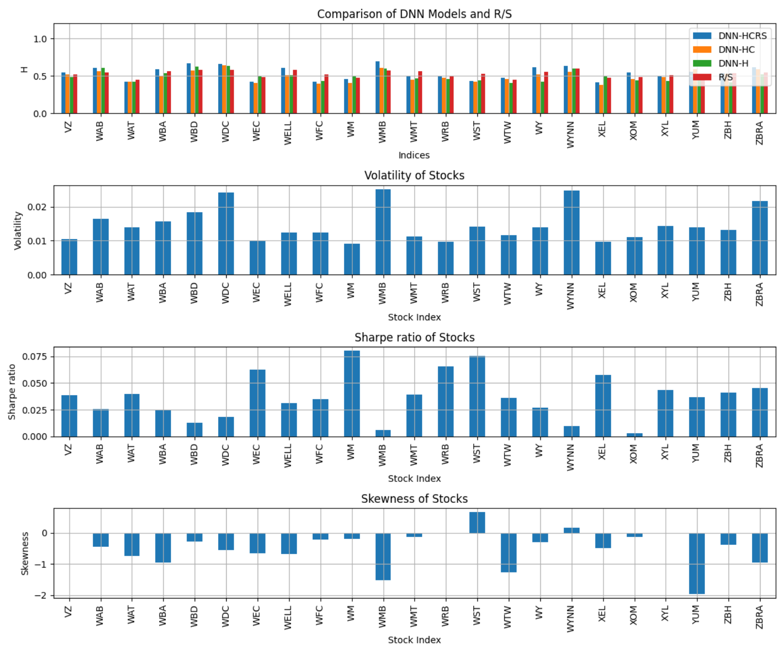

Figure 20. Comparision of the volatility, Sharpe ratio, and the skewness against the Hurst exponent.

4. Results and Discussions

In this research, we develope a novel deep neural network model capable of estimating the Hurst exponent by leveraging one of the popular estimation methods: R/S analysis and DFA. This approach addresses a limitation of R/S analysis, which tends to yield inaccurate and biased estimates when applied to shorter time series data. We experimented with three target values incorporating the Hurst exponent, denoted as H, and the constant C calculated using the R/S analysis.

Empirical evidence within the realm of stochastic time series, particularly in the context of stock prices, has revealed deviations from the classical assumption of a random walk. This departure challenges the foundational underpinnings of many financial models and can potentially lead to erroneous conclusions. Therefore, accurately estimating the Hurst exponent becomes paramount in addressing this challenge. The outcomes of this research illuminate a pathway towards harnessing advanced machine learning tools, specifically deep learning techniques, to tackle this issue. We hope these findings will inspire further research, experimentation, and the development of related software tools.

5. Further Research and Future Developments

It is essential to acknowledge certain limitations inherent in our approach. We did not explore alternative neural network architectures, such as deep Recurrent Neural Networks (deep RNNs) and Convolutional Neural Networks (CNNs), which could potentially offer valuable insights and improve predictive performance [34]. In what follows, we would like to discuss the current limitations of the presented approach, along with possible enhancements and future research we have already started to draft.

While the presented DNN architecture has shown promising results, it is worth mentioning that further generalisation can be achieved by incorporating stochastic elements into the model. In particular, in the deterministic approach outlined in the paper, the Hurst exponent H is computed as a point estimate derived from the R/S analysis or DFA. However, to better account for the stochastic nature of financial data, we aim at modelling the Hurst exponent as a random variable, $\mathcal{H}$, that follows a particular probability distribution. Such probabilistic interpretation allows us to express the Hurst exponent not as a fixed value but as a distribution that reflects the variability in the estimation due to noisy data or limited sample size. The distribution can be parameterised and updated as more data become available. Accordingly, the stochastic generalisation will require modifying the loss function of the DNN to account for the probabilistic nature of the output. Instead of minimising a point estimate of H, we aim to minimise the expected loss over the distribution of H. Therefore, for a given time series, X_t, we define the loss function as follows:

$$L(\theta) = \mathbb{E}_{\mathcal{H}}\left[(f_\theta(X_t) - \mathcal{H})^2\right],$$

where $f_\theta(X_t)$ is the predicted Hurst exponent by the neural network, and $\mathcal{H}$ is the random Hurst exponent following the distribution $\mathbb{P}(H|\mu_H, \sigma_H)$. Moreover, we are considering further generalising such an approach by coupling the estimation of H with stochastic volatility models. Indeed, we can consider a stock price $S(t)$ modelled as follows under fBm:

$$dS(t) = \mu S(t)dt + \sigma S(t)dB_H(t),$$

where $B_H(t)$ is a fractional Brownian motion with Hurst exponent H, and the volatility σ is no longer constant but can be modelled as a stochastic process that depends on H itself,

$$\sigma_H(t) = \sigma_0 e^{-\lambda t} + \sigma_1 H\,,$$

so that we can reflect the dependence of volatility on the persistence or anti-persistence of the time series, as captured by H. In particular, we know that the process exhibits persistence for $H > 0.5$, meaning that volatility trends are more likely to continue. In contrast, the process exhibits mean-reverting behaviour for $H < 0.5$, indicating that smaller ones follow large fluctuations. Accordingly, by incorporating the stochastic volatility model into the DNN framework, we can improve the estimation of H by accounting for the impact of volatility changes over time. In particular, the DNN can be trained to estimate H and adjust for the changing volatility structure, making the model more adaptable to real-world financial data.

A further field of investigation will consider the estimation problem for H as formulated within a stochastic control framework. Specifically, let $\mathcal{U}(t)$ represent the control variables of the DNN model that influence the prediction of H. Then, the objective is to minimise the expected error in the prediction of H by solving the following stochastic optimisation problem,

$$\min_{\mathcal{U}(t)} \mathbb{E}\left[\int_0^T (f_\theta(X_t) - \mathcal{H})^2 dt\right],$$

subject to the dynamics of the time series modelled by fractional Brownian motion. This stochastic control approach introduces feedback mechanisms that adjust the model parameters dynamically as new data become available, thereby refining the estimation of H in real time.

From a more oriented ML approach, let us underline once more that even if the application of deep learning methods in the estimation of the Hurst exponent has already shown considerable promise, as demonstrated in the current work, it is worth studying the possible ML/NNs-based alternatives. The introduction of such architectures, including Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNNs), has the potential to lead us towards deeper insights and enhanced predictive accuracy. Indeed, these models have distinct advantages in handling temporal dependencies and extracting features from high-dimensional data, which could significantly advance the estimation of the Hurst exponent, particularly in the presence of complex, dynamic financial time series.

To be more precise, let us underline that RNNs are well suited for tasks involving sequential data, as they are designed to capture temporal dependencies by introducing cyclical connections in the network architecture. Moreover, RNNs enable the network to maintain a hidden state h_t evolving over time to capture information from previous inputs. This latter characteristic could be considered to enhance our current approach, taking care of the fact that RNNs suffer difficulties capturing long-term dependencies due to the vanishing gradient problem. Indeed, when gradients are propagated back through time during training, they may either vanish or explode, making it difficult for the model to learn dependencies across distant time steps. This limitation is particularly critical in estimating the Hurst exponent, where long-term memory plays a central role in characterising the self-similarity and persistence of time series. Hence, we plan to merge the previous approach, applying an LSTM layer to mitigate the vanishing gradient problem by introducing memory cells that explicitly control what information to store, update, and discard through a system of gates, specifically according to the standard structure, input, forget and output gate(s), implying the following evolution of the memory cell C_t

$$\begin{aligned} f_t &= \sigma(W_f[h_{t-1}, X_t] + b_f) \\ i_t &= \sigma(W_i[h_{t-1}, X_t] + b_i) \\ C_t &= f_t \cdot C_{t-1} + i_t \cdot \tanh(W_C[h_{t-1}, X_t] + b_C) \\ o_t &= \sigma(W_o[h_{t-1}, X_t] + b_o) \\ h_t &= o_t \cdot \tanh(C_t) \end{aligned}$$

where f_t, resp. i_t, and resp. o_t are the forget, resp. the input, and resp. the output gate, with C_t representing the memory cell at time t. Therefore, since the gates f_t, i_t, and o_t control how information flows in and out of the memory cell, this will allow us to maintain long-term dependencies over extended time intervals. Accordingly, we will obtain a particularly well-suited structure to estimate the Hurst exponent, especially when dealing with financial time series that exhibit long-range dependencies, at the same time improving our solution(s) in modelling complex temporal correlations in stock prices while better capturing the persistence or anti-persistence characteristics that are central to the Hurst exponent. Then, we plan to move forward to obtain further extensions based on stacked LSTMs, which will enhance the model's capacity to learn hierarchical and multi-scale patterns in the data since the stacked architecture will enable the network to learn increasingly abstract representations of the input data, potentially capturing the multi-scale nature of financial time series.

Furthermore, especially in this case, it is worth emphasising how the following does indeed characterize our plans for further development, which are nevertheless still in a rather primitive phase; we aim at exploiting Convolutional Neural Networks (CNNs), attention mechanisms and Transformer Models. In particular, even if CNNs are traditionally associated with image processing tasks, recently, they have been used in modelling time series data as well, mainly because they excel at identifying local patterns through their convolutional layers, which apply learnable filters to the input data. Hence, in the context of financial time series, CNNs can be used to capture local trends and short-term dependencies, complementing the ability of RNNs and LSTMs to model longer-term relationships. Indeed, since the central operation in a CNN is the convolution, where a filter $w \in \mathbb{R}^k$ is applied to the input time series $X = \{X_1, X_2, \dots, X_T\}$ to produce a feature map F,

$$F_t = \sum_{i=1}^{k} w_i X_{t+i-1} \,,$$

with the filter sliding over the input data, by learning multiple filters, the CNN can extract various features, passed through non-linear activation functions, e.g., the ReLU one, and downsampled through pooling layers to reduce dimensionality while retaining the most salient information. Hence, in estimating the Hurst exponent, CNNs can be particularly useful for detecting localised patterns of persistence or anti-persistence. For example, in a stock price series, short-term fluctuations may exhibit distinct behaviours, such as rapid price reversals or bursts of volatility, which provide crucial information about the overall memory properties of the process. By applying convolutional filters, CNNs can automatically learn to detect such patterns without requiring hand-crafted features.

Therefore, a combined approach, where CNNs are used in conjunction with LSTMs or RNNs, may offer the most powerful architecture for estimating the Hurst exponent. Indeed, in such hybrid models, the CNN layers can be used to extract local features from the input time series, which are then fed into an LSTM layer to capture long-term dependencies. Finally, with regard to the attention Mechanism and Transformer Model approaches we mentioned before, we underline that, even if initially developed for natural language processing tasks, attention mechanisms allow the model to focus selectively on relevant parts of the input sequence, effectively learning a weighted representation of the entire sequence based on the importance of each time step. While in the Transformer architecture, the attention mechanism replaces the need for recurrent connections by allowing the model to attend to all time steps simultaneously, and we think that this aspect could be advantageous in financially based time series analysis, where specific time steps may exert a more decisive influence on the prediction than others. Indeed, the attention mechanism computes a set of attention weights α_{ij} that determine the importance of the input at time j when predicting the output at time i,

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}$$

$$e_{ij} = \text{score}(h_i, h_j)$$

where e_{ij} is a learned compatibility function between the hidden states h_i and h_j; then, we will try to exploit the Transformer model's self-attention mechanism to improve the model's ability to capture both local and global dependencies in the time series data, which is critical for accurately estimating the Hurst exponent. Moreover, unlike RNNs or LSTMs, the Transformer model does not require sequential processing, allowing it to scale efficiently to long time series. This crucial aspect makes it an ideal candidate for estimating the Hurst exponent in high-frequency financial data.

6. Conclusions

In conclusion, this study delved into estimating the Hurst exponent for a diverse financial time series dataset. We evaluated three distinct models, each designed to predict the exponent using a combination of variables derived from the R/S analysis.

Our findings underscore the significance of employing neural network methodologies, which notably improve predictive performance compared to that of existing methods. This suggests that a more comprehensive approach, encompassing multiple variables and a well-structured neural network architecture, can improve predictive capabilities when estimating the Hurst exponent.

This research holds implications for a deeper understanding of long-term memory properties within financial data, with potential applications in risk assessment, algorithmic trading, and various facets of economic analysis. Future work may refine the model and explore its applicability in real-world financial forecasting scenarios. Efficient estimation of the Hurst exponent using DNN has significant practical implications for the finance industry, particularly in stock price prediction and options pricing. The Hurst exponent quantifies the degree of long-term memory in a time series and plays a crucial role in understanding market trends and volatility. By leveraging DNNs to estimate the Hurst exponent more accurately, especially for shorter financial time series, traders and analysts can better forecast future stock price movements and assess market inefficiencies. This improved estimation technique can enhance the precision of predictive models, aiding in developing more effective trading strategies. In options pricing, where volatility estimation is critical, incorporating a reliable Hurst exponent can refine models like the Black–Scholes framework, leading to better risk management and more accurate pricing of derivative instruments. Ultimately, efficient DNN-based estimation of the Hurst exponent provides a powerful tool for market analysis, helping practitioners make more informed decisions in dynamic financial environments.

The efficient estimation of the Hurst exponent H using deep neural networks (DNNs) has significant applications in financial modelling, particularly for stock price prediction and options pricing. Let us summarise the claimed impact by considering the following examples.

6.1. Stock Price Prediction

The Hurst exponent, H, derived from fractional Brownian motion (fBm), characterises the persistence or anti-persistence of a time series. A time series $X(t)$ with $H > 0.5$ indicates persistent (trending) behaviour, while $H < 0.5$ indicates anti-persistent (mean-reverting) behaviour. For $H = 0.5$, the series behaves as a random walk, implying no long-term memory. Given a stock price time series $S(t)$, where

$$S(t) = S_0 \exp(\mu t + \sigma W_H(t)),$$

$W_H(t)$ represents a fractional Brownian motion with Hurst exponent H, the exponent H influences the autocorrelation structure of the returns $\log(S(t))$. Accurate estimation of H through DNNs allows for more precise modelling of the underlying stochastic process, which is key to forecasting future stock price movements $S(t + \Delta t)$. Moreover, by estimating H for shorter time series, DNN-based techniques overcome the limitations of traditional estimators, enabling traders to capture market inefficiencies, which can be modelled as deviations from $H = 0.5$. It is worth mentioning that such inefficiencies can be exploited for more precise trend prediction and strategy formulation.

6.2. Options Pricing

In options pricing, the underlying asset's volatility is a critical parameter. The classical Black–Scholes model assumes constant volatility and no long-term memory in the price process. However, when the underlying asset follows a process with long-term memory (as indicated by $H \neq 0.5$), the volatility σ must be adjusted accordingly. A modified pricing model can incorporate H as follows:

$$C(S,t) = S_0\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2),$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution, and d_1 and d_2 are given by the following:

$$d_1 = \frac{\log\left(\frac{S_0}{K}\right) + \left(r + \frac{\sigma_H^2}{2}\right)(T-t)}{\sigma_H\sqrt{T-t}}, \quad d_2 = d_1 - \sigma_H\sqrt{T-t},$$

where σ_H represents the volatility adjusted for the Hurst exponent H. Efficient estimation of H can improve the accuracy of σ_H, leading to more precise options pricing and enhanced risk management.

Author Contributions: Conceptualization, L.D.P. and T.T.D.; methodology, L.D.P. and T.T.D.; software, T.T.D.; validation, L.D.P. and T.T.D.; formal analysis, L.D.P. and T.T.D.; investigation, L.D.P. and T.T.D.; resources, L.D.P. and T.T.D.; data curation, T.T.D.; writing—original draft preparation, L.D.P. and T.T.D.; writing—review and editing, L.D.P. and T.T.D.; visualization, T.T.D.; supervision, L.D.P.; project administration, L.D.P. and T.T.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used in this research can be found in an open database: https://finance.yahoo.com/.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tzouras, S.; Anagnostopoulos, C.; McCoy, E. Financial time series modeling using the Hurst exponent. *Phys. Stat. Mech. Appl.* **2015**, *425*, 50–68. [CrossRef]
2. Hurst, H.E. Long-term storage capacity of reservoirs. *Trans. Am. Soc. Civ. Eng.* **1951**, *116*, 770–799. [CrossRef]
3. Lo, A.W. *Efficient Markets Hypothesis*; Palgrave Macmillan Ltd.: London, UK, 2007.
4. Dufera, T.T. Fractional Brownian motion in option pricing and dynamic delta hedging: Experimental simulations. *N. Am. J. Econ. Financ.* **2024**, *69*, 102017. [CrossRef]
5. Annis, A.; Lloyd, E. The expected value of the adjusted rescaled Hurst range of independent normal summands. *Biometrika* **1976**, *63*, 111–116. [CrossRef]
6. Lo, A.W. Long-term memory in stock market prices. *Econom. J. Econom. Soc.* **1991**, *59*, 1279–1313. [CrossRef]
7. Beben, M.; Orłowski, A. Correlations in financial time series: Established versus emerging markets. *Eur. Phys. J. Condens. Matter Complex Syst.* **2001**, *20*, 527–530. [CrossRef]
8. Cajueiro, D.O.; Tabak, B.M. The Hurst exponent over time: Testing the assertion that emerging markets are becoming more efficient. *Phys. Stat. Mech. Appl.* **2004**, *336*, 521–537. [CrossRef]
9. Granero, M.S.; Segovia, J.T.; Pérez, J.G. Some comments on Hurst exponent and the long memory processes on capital markets. *Phys. Stat. Mech. Appl.* **2008**, *387*, 5543–5551. [CrossRef]
10. Resta, M. Hurst exponent and its applications in time-series analysis. *Recent Patents Comput. Sci.* **2012**, *5*, 211–219. [CrossRef]
11. Fernández-Martínez, M.; Sánchez-Granero, M.; Segovia, J.T.; Román-Sánchez, I. An accurate algorithm to calculate the Hurst exponent of self-similar processes. *Phys. Lett.* **2014**, *378*, 2355–2362. [CrossRef]
12. Kroha, P.; Skoula, M. Hurst Exponent and Trading Signals Derived from Market Time Series. In Proceedings of the ICEIS 2018—20th International Conference on Enterprise Information Systems, Madeira, Portugal, 21–24 March 2018; pp. 371–378.
13. Gómez-Águila, A.; Trinidad-Segovia, J.; Sánchez-Granero, M. Improvement in Hurst exponent estimation and its application to financial markets. *Financ. Innov.* **2022**, *8*, 86. [CrossRef]
14. Qian, B.; Rasheed, K. Hurst exponent and financial market predictability. In Proceedings of the IASTED Conference on Financial Engineering and Applications, In Proceedings of the IASTED International Conference, Cambridge, MA, USA, 17–19 February 2004; pp. 203–209.
15. Mandelbrot, B.B.; Van Ness, J.W. Fractional Brownian motions, fractional noises and applications. *SIAM Rev.* **1968**, *10*, 422–437. [CrossRef]
16. Couillard, M.; Davison, M. A comment on measuring the Hurst exponent of financial time series. *Phys. Stat. Mech. Appl.* **2005**, *348*, 404–418. [CrossRef]

17. Davies, R.B.; Harte, D.S. Tests for Hurst effect. *Biometrika* **1987**, *74*, 95–101. [CrossRef]
18. McCulloch, W.S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **1943**, *5*, 115–133. [CrossRef]
19. Basheer, I.; Hajmeer, M. Artificial neural networks: Fundamentals, computing, design, and application. *J. Microbiol. Methods* **2000**, *43*, 3–31. [CrossRef]
20. Yadav, N.; Yadav, A.; Kumar, M. *An Introduction to Neural Network Methods for Differential Equations*; Springer: Berlin/Heidelberg, Germany, 2015.
21. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef]
22. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1.
23. Dong, S.; Wang, P.; Abbas, K. A survey on deep learning and its applications. *Comput. Sci. Rev.* **2021**, *40*, 100379. [CrossRef]
24. Dixit, P.; Silakari, S. Deep learning algorithms for cybersecurity applications: A technological and status review. *Comput. Sci. Rev.* **2021**, *39*, 100317. [CrossRef]
25. Minaee, S.; Boykov, Y.Y.; Porikli, F.; Plaza, A.J.; Kehtarnavaz, N.; Terzopoulos, D. Image segmentation using deep learning: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 3523–3542. [CrossRef]
26. Li, Z.; Li, H.; Meng, L. Model compression for deep neural networks: A survey. *Computers* **2023**, *12*, 60. [CrossRef]
27. Bruna, J.; Dec, L. *Mathematics of Deep Learning*; Courant Institute of Mathematical Science: New York, NY, USA, 2018.
28. Haber, E.; Ruthotto, L. Stable architectures for deep neural networks. *Inverse Probl.* **2017**, *34*, 014004. [CrossRef]
29. Zheng, Z.; Hong, P. Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; pp. 7924–7933.
30. Walczak, S. An empirical analysis of data requirements for financial forecasting with neural networks. *J. Manag. Inf. Syst.* **2001**, *17*, 203–222.
31. Hæke, C.; Helmenstein, C. Neural networks in the capital markets: An application to index forecasting. *Comput. Econ.* **1996**, *9*, 37–50. [CrossRef]
32. Nguyen, T.; Nguyen, T.; Nguyen, B.M.; Nguyen, G. Efficient Time-Series Forecasting Using Neural Network and Opposition-Based Coral Reefs Optimization. *Int. J. Comput. Intell. Syst.* **2019**, *12*, 1144–1161. [CrossRef]
33. Calin, O. *Deep Learning Architectures*; Springer: Berlin/Heidelberg, Germany, 2020.
34. Di Persio, L.; Honchar, O. Artificial neural networks architectures for stock price prediction: Comparisons and applications. *Int. J. Circuits Syst. Signal Process.* **2016**, *10*, 403–413.
35. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning internal representations by error propagation. In *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*; Rumelhart, D.E., Mcclelland, J.L., Eds.; The MIT Press: Cambridge, MA, USA, 1986; Volume 1.
36. Rodriguez, P.; Wiles, J.; Elman, J.L. A recurrent neural network that learns to count. *Connect. Sci.* **1999**, *11*, 5–40. [CrossRef]
37. Medsker, L.R.; Jain, L. Recurrent neural networks. *Des. Appl.* **2001**, *5*, 2.
38. Hochreiter, S. Long Short-term Memory. In *Neural Computation*; MIT Press: Cambridge, MA, USA, 1997.
39. Cho, K. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
40. Rostek, S. *Option Pricing in Fractional Brownian Markets*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2009; Volume 622.
41. Elliott, R.; Van Der Hoek, J. Fractional Brownian Motion and Financial Modelling. In *Mathematical Finance*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 140–151.
42. Hu, Y.; Øksendal, B. Fractional white noise calculus and applications to finance. *Infin. Dimens. Anal. Quantum Probab. Relat. Top.* **2003**, *6*, 1–32. [CrossRef]
43. Biagini, F.; Hu, Y.; Oksendal, B.; Zhang, T. *Stochastic Calculus for Fractional Brownian Motion and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2008.
44. Peng, C.K.; Buldyrev, S.V.; Havlin, S.; Simons, M.; Stanley, H.E.; Goldberger, A.L. Mosaic organization of DNA nucleotides. *Phys. Rev. E* **1994**, *49*, 1685–1689. [CrossRef] [PubMed]
45. Peng, C.K.; Havlin, S.; Stanley, H.E.; Goldberger, A.L. Quantification of scaling exponents and crossover phenomena in nonstationary heartbeat time series. *Chaos Interdiscip. J. Nonlinear Sci.* **1995**, *5*, 82–87. [CrossRef] [PubMed]
46. Hu, K.; Ivanov, P.C.; Chen, Z.; Carpena, P.; Stanley, H.E. Effect of trends on detrended fluctuation analysis. *Phys. Rev. E* **2001**, *64*, 011114. [CrossRef] [PubMed]
47. Kantelhardt, J.W.; Zschiegner, S.A.; Koscielny-Bunde, E.; Havlin, S.; Bunde, A.; Stanley, H.E. Multifractal detrended fluctuation analysis of nonstationary time series. *Phys. Stat. Mech. Its Appl.* **2002**, *316*, 87–114. [CrossRef]
48. Zunino, L.; Tabak, B.M.; Figliola, A.; Pérez, D.G.; Garavaglia, M.; Rosso, O.A. A multifractal approach for stock market inefficiency. *Phys. Stat. Mech. Its Appl.* **2008**, *387*, 6558–6566. [CrossRef]
49. Zunino, L.; Figliola, A.; Tabak, B.M.; Pérez, D.G.; Garavaglia, M.; Rosso, O.A. Multifractal structure in Latin-American market indices. *Chaos Solitons Fractals* **2009**, *41*, 2331–2340. [CrossRef]
50. Hardstone, R.; Poil, S.S.; Schiavone, G.; Jansen, R.; Nikulin, V.V.; Mansvelder, H.D.; Linkenkaer-Hansen, K. Detrended fluctuation analysis: A scale-free view on neuronal oscillations. *Front. Physiol.* **2012**, *3*, 450. [CrossRef]

51. Grech, D.; Pamuła, G. Multifractality of nonlinear transformations with application in finances. *Acta Phys. Pol. A* **2013**, *123*, 529–537. [CrossRef]
52. Ihlen, E.A. Introduction to multifractal detrended fluctuation analysis in Matlab. *Front. Physiol.* **2012**, *3*, 141. [CrossRef]
53. Rydin Gorjão, L.; Hassan, G.; Kurths, J.; Witthaut, D. MFDFA: Efficient multifractal detrended fluctuation analysis in python. *Comput. Phys. Commun.* **2022**, *273*, 108254. [CrossRef]
54. Drożdż, S.; Kowalski, R.; Oswiecimka, P.; Rak, R.; Gebarowski, R. Dynamical variety of shapes in financial multifractality. *Complexity* **2018**, *2018*, 1–13. [CrossRef]
55. Lee, M.; Song, J.W.; Kim, S.; Chang, W. Asymmetric market efficiency using the index-based asymmetric-MFDFA. *Phys. Stat. Mech. Appl.* **2018**, *512*, 1278–1294. [CrossRef]
56. Dufera, T.T. Deep neural network for system of ordinary differential equations: Vectorized algorithm and simulation. *Mach. Learn. Appl.* **2021**, *5*, 100058. [CrossRef]
57. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: https://www.tensorflow.org (accessed on 10 June 2024).
58. Chollet, F. Keras. 2015. Available online: https://keras.io (accessed on 10 June 2024).
59. O'Malley, T.; Bursztein, E.; Long, J.; Chollet, F.; Jin, H.; Invernizzi, L. KerasTuner. 2019. Available online: https://github.com/keras-team/keras-tuner (accessed on 10 June 2024).