Expert Systems With Applications 169 (2021) 114474

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A fractal interpolation approach to improve neural network predictions for difficult time series data

Sebastian Raubitzek *, Thomas Neubauer

Vienna University of Technology, Institute of Information Systems Engineering, Information & Software Engineering Research Division (194-01), Favoritenstrasse 9-11/188, 1040 Vienna, Austria

ARTICLE INFO

Keywords:
Fractal dimension
Hurst exponent
Chaos
Hyperchaos
Lyapunov exponent
Time series data
Time series prediction
Deep learning
Machine learning
LSTM
Time series analysis
R/S analysis
Linear interpolation

ABSTRACT

Deep Learning methods, such as *Long Short-Term Memory* (*LSTM*) neural networks prove capable of predicting real-life time series data. Crucial for this technique to work is a sufficient amount of data. This can either be very long time-series data or fine-grained time-series data. If the data is insufficient, in terms of length or complexity, LSTM approaches perform poorly. We propose a fractal interpolation approach to generate a more fine-grained time series out of insufficient data sets. The interpolation is dynamically adapted to the time series using a time-dependent complexity feature so that the complexity properties of the interpolated time series are related to that of the original one. Also we perform a linear interpolation with the same number of interpolation points to compare results.

This paper shows that predictions of fractal interpolated and linear interpolated time series clearly outperform the ones of the original data for a test fit on unknown data. Though predictions of linear and fractal interpolated time series data perform very similar, the fractal interpolated ones outperform the linear interpolated ones on difficult time series data. Also, though the complexities of sub-intervals are tailored to match the one from the original data, the interpolated time series shows a much higher degree of persistency and (in terms of the *Hurst exponent*) a much higher degree of long term memory.

1. Introduction

In recent years many researchers have employed artificial intelligence, i.e., machine learning and deep learning, to make predictions and analysis based on historical data, Hey et al. (2020). There are many reasons and applications for making predictions, such as future population estimates, predicting epileptic seizures, estimating future stock market prices, etc. . The outcomes of these studies are promising and manifold, e.g., in solid-state physics, first-principle calculations can be sped up (Schmidt et al., 2019), or one can employ machine learning to predict solar radiation (Voyant et al., 2017). In biology applications are genomics, proteomics and evolution (Larrañaga et al., 2005). In medicine neural networks can be used to identify atrial fibrillation and congestive heart failure, Agliari et al. (2020). When it comes to agriculture, machine learning can be used to predict yields and give estimates on the nitrogen status or other soil parameters, Chlingaryan et al. (2018).

All of these approaches depend on the available data, meaning that the complexity of the data and the actual amount of data is crucial. The two main data-specific[1] reasons for machine learning to perform poorly are:

(i) An insufficient amount of data. For time series data this means that the data is whether not fine-grained enough or not long enough.
(ii) Random data, meaning that despite there is a sufficient amount of data available, the inherent noise (randomness) in the data prevents the employed model from making good predictions.

One way to produce more fine-grained data is to interpolate data sets. Karaca et al. (2019) uses a linear interpolation to increase the fine-grainedness of the data under study. In most cases, real-life data is non-linear and originates from complex systems with many unknown interactions; therefore, a linear interpolation approach does not depicture the complexity of the system. An approach taking into account the complexity of real-life systems when doing data interpolation is to use fractal interpolation (Manousopoulos et al., 2008). In contrast to traditional interpolation methods, which are based on elementary functions,

* Corresponding author.
E-mail addresses: sebastian.raubitzek@tuwien.ac.at (S. Raubitzek), thomas.neubauer@tuwien.ac.at (T. Neubauer).

[1] Of course there are other reason for machine learning to perform poorly such as a bad choice of hyperparameters or the wrong training approach, but here we focus on data-specific reasons because of the approach presented in Section 3.

https://doi.org/10.1016/j.eswa.2020.114474
Received 17 July 2020; Received in revised form 18 August 2020; Accepted 6 December 2020
Available online 13 December 2020

such as polynomials, fractal interpolation is based on iterated functions systems. Since iterated functions systems are capable of generating fractal and multi-fractal structures, the inherent complexity of fractal interpolated data is closer to real-life data than that of traditionally interpolated data.

Overall, the complexity of the data under study has an impact on the quality of forecasts. Usually, a complexity measure, such as the Hurst exponent (Hurst et al., 1965), the fractal dimension (Feder, 1988) or an entropy measure is employed for this task. For example, Vörös and Jankovičová (2002) use the local Hölder exponent to add a complexity feature to the time series to be forecast to improve predictions. Karaca et al. (2019) use the Hurst exponent, Rényi entropy and Shannon entropy to improve forecasts of financial markets and cryptocurrencies.

The aim of this research therefore is to identify to what extent a combination of fractal interpolation and complexity measures can be used to improve predictions using a basic Long Short Term Memory (LSTM) neural network implementation (Hochreiter & Schmidhuber, 1997).

We apply a fractal and a linear interpolation method to four data sets to increase the fine-grainedness of the data. Also, the fractal interpolation was tailored to match the original data's complexity using the Hurst exponent. The four data sets are chosen to be two popular machine learning test data sets (which are used in many tutorials) and two difficult real-life data sets. Afterward, a LSTM neural network is trained on one part of the data and then fitted on an unknown part of the data. This is done for each data set.

As far as the authors know, fractal interpolation has not been used in a combination with long short term memory neural networks yet. Thus, this is a novel approach, though several applications of machine learning approaches together with ideas from chaos theory exist. As done here, ideas from chaos theory (the spectrum of Lyapunov exponents and complexity measures) are further employed to analyze time series data and draw conclusions about predictability and accuracy, just as done in Diaconescu (2008), Ni et al. (2011) and Yakuwa et al. (2003).

In Section 2 the four data sets are described in detail. In Section 3, the fractal interpolation and the corresponding implementation is described. In Section 4 we briefly show the applied linear interpolation. Section 5 discusses the corresponding complexities of all original data sets and the corresponding interpolated ones. Section 6 describes the applied neural network, and its properties with respect to each data set. Section 7 shows the results of the test fits and discusses the predictions with respect to the complexity of the data. In Section 8 we sum up the findings of this research and give an outlook on future applications.

2. Datasets

For this research, four different data sets were used. Two of them are popular test data sets which are widely used in machine learning tutorials. The other two data sets are chosen to be real-life data sets. Both real-life data sets are comparatively short (57 data points) annual yield time series data. Also, we consider them to be difficult time series data because of the complexity of the underlying interactions, since they result from environmental/agricultural systems (Sakai, 2001).

1. **Shampoo sales:** This is a popular data set to test machine learning predictions and can be found on various tutorials. It is a comparatively short data set and features only 36 observations, i.e. three years of monthly sales of a shampoo product. The original data set is credited to Makridakis et al. (1984).
2. **Airline passengers:** This is also a well-known test data set for machine learning applications and is part of many tutorials. It is known to work very well with basic LSTM neural network implementations. This is a problem where, given a year and a month, the task is to predict the number of international airline passengers in units of 1,000. The data ranges from January 1949 to December 1960 or 12 years, with 144 observations. This data set can be downloaded at kaggle.com
3. **Annual wheat yields Austria**: This is a data set of the annual yields of wheat in Austria ranging from 1961 to 2017 with a total of 57 data points. This data set can be downloaded at http://www.fao.org/faostat/.
4. **Annual maize yields Austria**: This is a data set of the annual yields of maize in Austria ranging from 1961 to 2018 with a total of 58 data points. This data set can be downloaded at http://www.fao.org/faostat/.

3. Fractal interpolation of time series data

For the fractal interpolation, we employ a method described in Manousopoulos et al. (2008). We give a summary of the method and refer to the original paper for further reading.

Traditional interpolation approaches are based on elementary functions such as polynomials. In contrast, fractal interpolation is based on iterated functions systems. An iterated functions system is defined as a complete metric space X with a distance function h and a finite set of contractive mappings, $\{w_n : X \to X \text{ for } n = 1, 2, \dots, N\}$ (Mazel & Hayes, 1992). For further reading on iterated functions systems we refer to Barnsley et al. (1985).

Therefore, the fractal interpolated functions are reproducing data with a complexity closer to real-life data than, e.g., polynomials.

A time series is given as a set of data points as $\{(u_m, v_m) \in \mathbb{R}^2 : m = 0, 1, \dots, M$. The interpolation is then applied to a subset of those data points, i.e. the interpolation points $\{(x_i, y_i) \in \mathbb{R}^2 : i = 0, 1, \dots, N\}$. Both sets are linearly ordered with respect to their abscissa, i.e. : $u_0 < u_1 < \dots < u_M$ and $x_0 < x_1 < \dots < x_N$. Therefore the set of interpolation points partition the set of data points into interpolation intervals. In our application the interpolation intervals are chosen to be equidistantly. The more interpolation points are used the better the interpolation fits the original data, but more interpolation points result in a smaller compression ratio, (the ratio of the information of the original data and the information of the interpolated data) since more information is needed to describe the interpolation function.

$\{\mathbb{R}^2; w_n, n = 1, 2, \dots, N\}$ is an iterated functions system (IFS) with affine transformations

$$w_n \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_n & 0 \\ c_n & s_n \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_n \\ e_n \end{bmatrix} \tag{1}$$

which is constrained to satisfy

$$w_n \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_{n-1} \\ y_{n-1} \end{bmatrix} \quad \text{and} \quad w_n \begin{bmatrix} x_N \\ y_N \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{2}$$

for every $n = 1, 2, \dots, N$. As done in Manousopoulos et al. (2008), solving these equations yields

$$\begin{aligned} a_n &= \frac{x_n - x_{n-1}}{x_N - x_0} \quad , \\ d_n &= \frac{x_N x_{n-1} - x_0 x_n}{x_N - x_0} \quad , \\ c_n &= \frac{y_n - y_{n-1}}{x_N - x_0} - s_n \frac{y_N - y_0}{x_N - x_0} \quad , \\ e_n &= \frac{x_N y_{n-1} - x_0 y_n}{x_N - x_0} - s_n \frac{x_N y_0 - x_0 y_N}{x_N - x_0} \quad . \end{aligned} \tag{3}$$

Here, the real numbers a_n, d_n, c_n, e_n are determined by the interpolation points and s_n is a free parameter referred to as *vertical scaling factor*. For the IFS to be hyperbolic with respect to an appropriate metric, s_n is bounded by $|s_n| < 1$. The parameter s_n will later be used to manipulate the complexity of the interpolated curve.

3.1. Fractal interpolation applied

The following procedure was applied to every time series to find a fractal interpolation that reproduces real-life complex time series data:

1. Divide time series into m sub-sets of size l

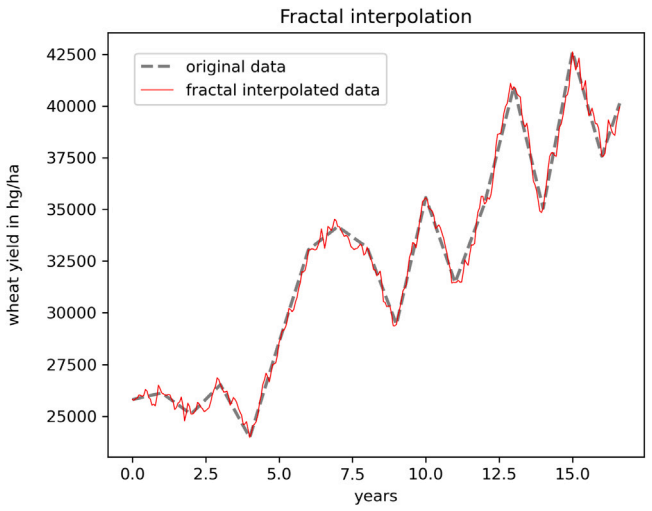

Fig. 1. Original and fractal interpolated wheat yields data. Only a fraction of the whole data is shown so that the fractal interpolation can clearly be seen.

2. For each sub-set i calculate the corresponding Hurst exponent H_i
3. For each subset i, the following routine is performed k times, where k was previously set to 200 for each dataset:
   (a) Use the fractal interpolation method from Section 3 with a random parameter s_n, where s_n was set constant for the whole sub-set.
   (b) Calculate the Hurst exponent $H_{i,\text{int,new}}$ for the interpolated time series.
   (c) If there was $H_{i,\text{int,old}}$ set beforehand, compare it to $H_{i,\text{int,new}}$. If $H_{i,\text{int,new}}$ is closer to H_i keep s_n, the corresponding fractal interpolation and $H_{i,\text{int,old}}$ to $H_{i,\text{int,new}}$.

Remarks:

- The Hurst exponent was calculated using *R/S Analysis* (Hurst et al., 1965).
- The number of iterations k was set to 200 for each data set. We found that after 200 iterations the complexity of the interpolated data was not getting significantly closer to the original data.
- No threshold was set for the Hurst exponent of the interpolated time-series to match the one from the original time series, since for some sub-intervals several thresholds that have been tried were not achieved.

Each data set was divided into subsets of a length of ten data points, where the first and the last one always match with the previous and following subset, respectively. For each set of two data points a total of 19 interpolation points were found, i.e. between each two original data points 17 additional data points were added to achieve a more fine-grained data set. This number was determined by randomly testing several different numbers of interpolation points. Since the aim of this work is to give a first example to legitimate fractal interpolation to be used together with neural networks, further work has to be done on determining the best number of interpolation points for data sets with different complexities.

In Fig. 1 the fractal interpolation is shown for the wheat yields data set.

4. Linear interpolation

Also, a linear interpolation was performed, such that the original data $\{(u_m, v_m) \in \mathbb{R}^2\} : m = 0, 1, \dots, M$ is interpolated using a linear fit $y_i = a_m x_i + b_m$, to get all interpolation points $\{(x_i, y_i) \in \mathbb{R}^2 : i = 0, 1, \dots, N\}$ for each interval $[u_m, u_{m+1}]$. The corresponding coefficients a_m and b_m are then calculated using

$$a_m = \frac{v_{m+1} - v_m}{u_{m+1} - u_m} \quad \text{and} \quad b_m = v_m - a_m u_m, \quad \forall \quad m = 0, 1, \dots, M-1 \quad . \tag{4}$$

5. Complexity of the data

The complexity properties of the original and the interpolated time series are compared for each data set. Three complexity measures are used to compare the chaotic properties of the data, i.e. the Hurst exponent, the fractal dimension of a time series and the spectrum of Lyapunov exponents.

5.1. The Hurst exponent (R/S analysis)

Following Di Matteo (2007) and Hurst et al. (1965): The rescaled range analysis (*R/S analysis*) is a sensitive method to reveal long-run correlations in random processes. This information is then given as one parameter: the *Hurst exponent* "H".

Given a time series $X(t)$ defined at discrete time intervals $t = v, 2v, 3, \dots, kv$. The average over a period τ (an entire multiple of v) is defined as

$$\langle X \rangle_\tau = \frac{v}{\tau} \sum_{k=1}^{\tau/v} X(kv) \quad . \tag{5}$$

The difference between maximal and minimal values of $X(t)$ in the interval $[v, T]$ is referred to as the range R,

$$R(\tau) = \max[X(t)]_{v \le t \le \tau} - \min[X(t)]_{v \le t \le \tau} \quad . \tag{6}$$

The *Hurst exponent* H is defined using the scaling property of the ratio

$$\frac{R(\tau)}{S(\tau)} \propto \left(\frac{\tau}{v}\right)^H \quad , \tag{7}$$

where $S(\tau)$ is the standard deviation:

$$S(\tau) = \sqrt{\frac{v}{\tau} \sum_{k=1}^{\tau/v} \left[X(kv) - \langle X \rangle_\tau\right]^2} \quad . \tag{8}$$

Proven by Hurst Hurst et al. (1965) and Feller Feller (1971) the asymptotic behavior for any independent random process with finite variance is given as

$$\frac{R(\tau)}{S(\tau)} = \left(\frac{\pi}{2v}\tau\right)^{\frac{1}{2}} \quad , \tag{9}$$

this implies $H = \frac{1}{2}$, which is only true for completely random processes. For real-life data, $H \neq \frac{1}{2}$, because real-life processes usually feature long-term correlations. For these processes, the scaling is modified, and R/S is asymptotically given by (7).

The value of the Hurst exponent can be $0 < H < 1$. A value of $H < 0.5$ indicates anti-persistency, meaning that a low value follows a high value, and a low value by a high value, thus fluctuating, but not random. Values very close to 0 are distinctive for a strong anti-persistency, and values close to 0.5 are distinctive for a weak anti-persistent behavior. On the other hand, a value of $H > 0.5$ indicates persistent behavior, and respectively strong persistency for values close to 1. According to Selvaratnam and Kirley (2006) a time series with $H \neq 0.5$ can theoretically be forecast.

For R/S analysis and to calculate the Hurst exponent we used the algorithm provided by the python package `nolds`.

5.2. Fractal dimension of a time-series

The *fractal dimension* of a time series is a measure of signal complexity. This idea can be understood as the time series (as a two-dimensional plot) lying on a grid of equal spacing, and checking the number of grid boxes necessary for covering it. Therefore, the fractal dimension is the ratio of the boxes covering the time series and the overall area of the plot. This process is referred to as box-counting. It has also been characterized as a measure of the space-filling capacity of a time series that tells how a fractal scales differently from the space it is embedded in; a fractal dimension does not have to be an integer. Also the fractal dimension is closely related to the Hurst exponent (Feder, 1988). The fractal dimension of a self-affine time series can have values $1 < D < 2$. A value close to 1 is typical for time series data with very few fluctuations, i.e. a very simple and not complex signal. A value close to 2 is typical for very complex signals. A value of ≈ 1.5 is an indicator for a random signal.

We used the algorithm developed by Higuchi (1988) to calculate the fractal dimension of a time series. An already existing python implementation from the package `hfda` was used for calculation.

5.3. The spectrum of Lyapunov exponents:

The *spectrum of Lyapunov exponents* measures the system's dependency on initial conditions. Referring to (experimental) time series data, it also is a measure of predictability (Eckmann et al., 1987). The first Lyapunov exponent of the spectrum is also referred to as the largest Lyapunov exponent. A positive largest (the first one) Lyapunov exponent is a strong indicator for chaos (Henriques et al., 2020); therefore, it is often sufficient to only calculate the largest Lyapunov exponent. Systems with more than one positive Lyapunov exponent are referred to as *hyperchaotic* (Zengrong et al., 1999). The algorithm developed by Eckmann et al. (1987) was used to calculate the spectrum of Lyapunov exponents.

We used the algorithm provided by the python package `nolds` to calculate the spectrum of Lyapunov exponents. This algorithm is an implementation of the algorithm developed by Eckmann et al. (1987), and yields the first four Lyapunov exponents.

5.4. Discussion

All calculated complexity properties for all time series data can be found in Table 1. Regarding the complexities of the original and all interpolated time series (Table 1), when comparing the Hurst exponent and the fractal dimension of the original and the interpolated time series, though the complexities of the sub-intervals are tailored to match the complexity of the original time series for the fractal interpolation, for all data sets all interpolated time series have a higher Hurst exponent and a lower fractal dimension. This results from the fact that both concepts can be linked using R/S analysis by $D_f \approx 2 - H$,[2] Feder (1988), where D_f is the fractal dimension and H is the Hurst exponent. Both measures therefore show that all interpolated time series are more persistent ones, or in terms of the Hurst exponent, data with more long term memory. It is expected that time series with these characteristics, i.e. a lower fractal dimension and a higher Hurst exponent can be forecast with higher accuracy.

Regarding the difference between the linear and the fractal interpolated time series data one observes that the linear interpolated time series data have a lower fractal dimension and a lower Hurst exponent than the fractal interpolated data, which is contradictory. But, again, given that we calculated those measures with completely different algorithms, this again proofs the previous statement regarding the link between Hurst exponent and fractal dimension. (Diaconescu, 2008; Ni et al., 2011).

All interpolated time series data show a total of two positive Lyapunov exponents, therefore indicating very chaotic behavior. Also, it is very interesting that for the linear interpolated data the largest (first) Lyapunov exponent is always larger than 2, and therefore indicating an even more chaotic behavior than the fractal interpolated time series data with a maximal largest Hurst exponent of 0.0971 for the shampoo sales time series. Therefore we conclude that the linear interpolated time series are less predictable compared to the fractal interpolated ones (Eckmann et al., 1987).

Table 1
Complexity measurements/properties.

Data	Hurst exponent	Fractal Dimension	Lyapunov spectrum
Shampoo sales	0.5894	1.7637	−0.0262 −0.0545 −0.1633 −0.1738
Shampoo sales fractal interpolated	0.9252	1.2063	0.0971 0.0102 −0.0784 −0.2158
Shampoo sales linear interpolated	0.9009	1.0982	2.9902 0.9656 −0.2968 −2.0357
Airline passengers	0.3996	1.7658	0.0160 0.0103 −0.0242 −0.0750
Airline passengers fractal interpolated	0.9521	1.2401	0.0926 0.0162 −0.0746 −0.2317
Airline passengers linear interpolated	0.9298	1.033	2.5814 0.7693 −0.3465 −1.8434
Wheat yields	0.9063	0.7145	−0.0156 −0.0379 −0.0286 −0.2699
Wheat yields fractal interpolated	0.8982	1.2269	0.0886 0.0099 −0.0589 −0.2374
Wheat yields linear interpolated	0.8758	1.0921	2.5700 0.8839 −0.3784 −2.0337
Maize yields	0.8591	1.7556	−0.0260 −0.0263 −0.0969 -0.2620
Maize yields fractal interpolated	0.8983	1.2050	0.0868 0.0061 −0.0671 −0.2182
Maize yields linear interpolated	0.8809	1.0941	2.7797 0.9551 −0.3495 −2.0198

[2] Here it is important to note that only R/S analysis links the fractal dimension and the Hurst exponent as $D_f = 2 - H$. Since different algorithms were used to calculate the fractal dimension and the Hurst exponent this identity is only approximately true.

6. LSTM neural networks predictions

A long short term memory (LSTM) neural network (Hochreiter & Schmidhuber, 1997) was applied to analyze the 4 data sets. Each time series was split into two parts, one for training the data set and one to test the performance on previously unknown data, whereas the training part makes up two thirds and the unknown part the last third of the data, chronologically.

6.1. Data preprocessing

Since the performance of neural networks can be improved with proper data preparation the following techniques were applied to all or specific data sets.

First, the data $X(t)$ defined at discrete time intervals $t = v, 2v, 3v, \ldots, kv$, was scaled so that $X(t) \in [-1, 1], \forall t$. This was done for all 4 data sets. Second, the performance was increased when the data was stationary, i.e. a linear fit was subtracted at each time step from the data. This was done for all data sets except for the wheat yield data, since it worsened the results of the predictions.

6.2. LSTM neural network

Fits on unknown data of the original and the fractal interpolated time series data was done using a long short term memory (*LSTM*) neural network (Hochreiter & Schmidhuber, 1997; Hua et al., 2019).

LSTMs are a category of recurrent neural networks (*RNNs*). RNNs are capable of using feedback or *recurrent* connections to cope with time series data. Though in principle designed for time series analysis and prediction, a standard RNN lacks the problem that the influence of a given input on the neural network either decays or blows up exponentially when passing through recurrent connections. LSTMs are specifically designed to solve this problem and to also learn inherent long-term dependencies. LSTMs feature a component called *memory block* to enhance their capability to model long-term dependencies. This memory block is a recurrently connected subnet containing two functional modules, i.e. the memory cell and the corresponding gates. The task of the memory cell is to remember the temporal state of the neural network. The gates on the other hand are responsible for controlling the information flow and consist of multiplicative units. There are three types of gates: Input gates, output gates and forget gates. The input gates control the information flow into the cell, where the forget gates control how information remains in the memory cell. The output gate controls how much information is used for the output activation and therefore decides what will be returned to the rest of the neural network.

A basic LSTM neural network architecture with one dense-layer and different numbers of hidden layers was used. For the dense-layer no regularizers were used and no other alterations were applied. The algorithm was optimized by trial and error, observing the training loss-curve and the overall performance. Using `adam` as an optimizer we observed decaying learning rates for all data sets. For the loss function we used `mean_squared_error`. The batch size was set to 1, and verbose was set to 2.

An existing `Keras 2.3.1` implementation was used for this research. The following list describes in detail the architecture of the LSTM-layer:

- `activation =''tanh''`
- `recurrent_activation=''sigmoid''`
- `use_bias=True`
- `kernel_initializer=''glorot_uniform''`
- `recurrent_initializer=''orthogonal''`
- `bias_initializer=''zeros''`
- `unit_forget_bias=True`
- `kernel_regularizer=None`
- `recurrent_regularizer=None`
- `bias_regularizer=None`
- `activity_regularizer=None`
- `kernel_constraint=None`
- `recurrent_constraint=None`
- `bias_constraint=None`
- `dropout=0.0`
- `recurrent_dropout=0.0`
- `implementation=2`
- `return_sequences=False`
- `return_state=False`
- `go_backwards=False`
- `stateful=False`
- `time_major=False`
- `unroll=False`

Table 2
LSTM architecture.

Data	Epochs	Hidden Layers	Number of input data points
Shampoo sales	89	2	7
Shampoo sales fractal interpolated	13	25	7
Shampoo sales linear interpolated	13	25	7
Airline passengers	50	10	15
Airline passengers fractal interpolated	2	35	20
Airline passengers linear interpolated	2	35	20
Wheat yields	120	10	2
Wheat yields fractal interpolated	15	25	100
Wheat yields linear interpolated	15	25	100
Maize yields	100	10	2
Maize yields fractal interpolated	5	30	70
Maize yields linear interpolated	5	30	70

In Table 2 the varying parameters for the neural network for each data set can be found.

6.3. Plots

All test fits, train fits, time series data and interpolated time series are depicted in Figs. 2, 3, 4, 5, 6, 7, 8, 9, 10,11, 12 and 13. The dashed purple line separates the training data from the unknown data.

6.4. Error analysis

All errors from Table 3 were calculated using a root-mean-square error (RMSE):

$$E_{RMSE} = \left(\frac{1}{S-1} \sum_{t=1}^{S} \left[\hat{X}(t) - X(t) \right]^2 \right) \quad , \tag{10}$$

where $X(t)$ is the original data and $\hat{X}(t)$ is the LSTM fit. S is the number of fitted data points. Also, to make the errors comparable with each other, all data was normalized so that all $X(t), \hat{X}(t) \in [-1, 1]$ when the RMSE was calculated. The errors for all train fits and test fits can be found in Table 3.

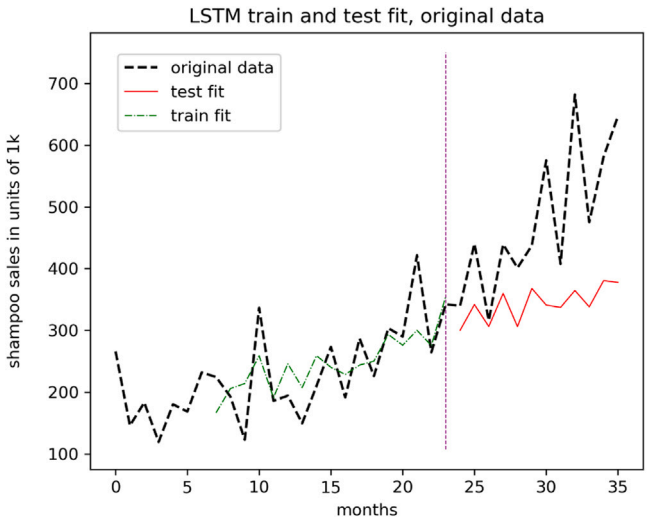

Fig. 2. Shampoo sales.

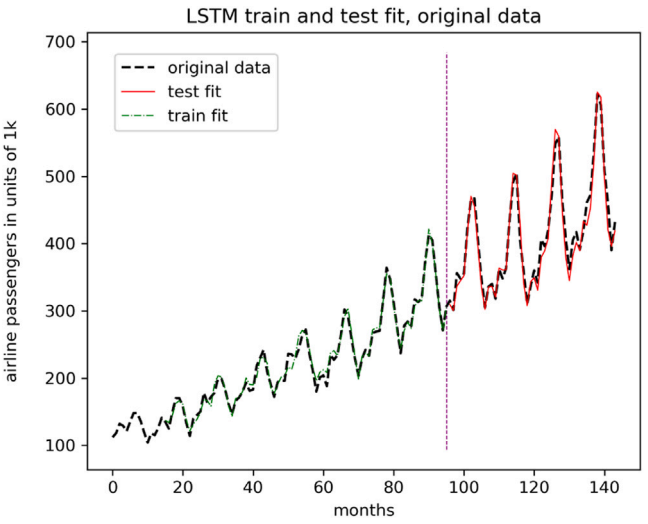

Fig. 5. Airline passengers.

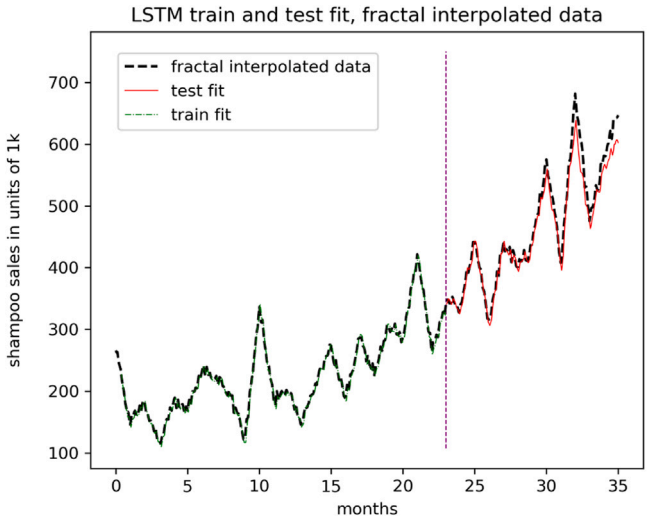

Fig. 3. Shampoo sales, fractal interpolated.

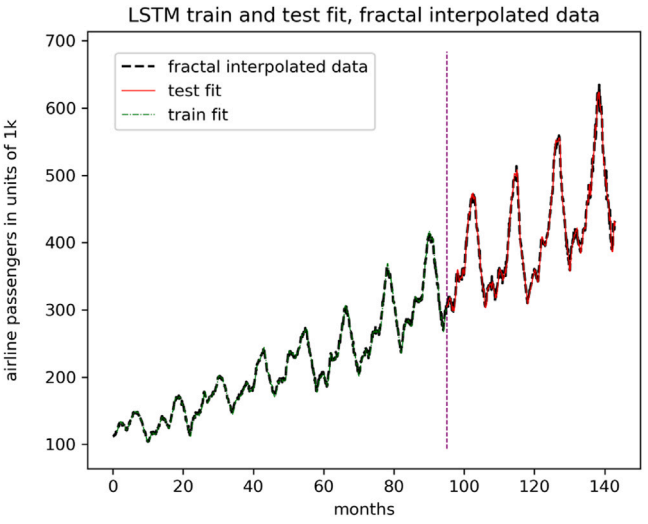

Fig. 6. Airline passengers fractal interpolated.

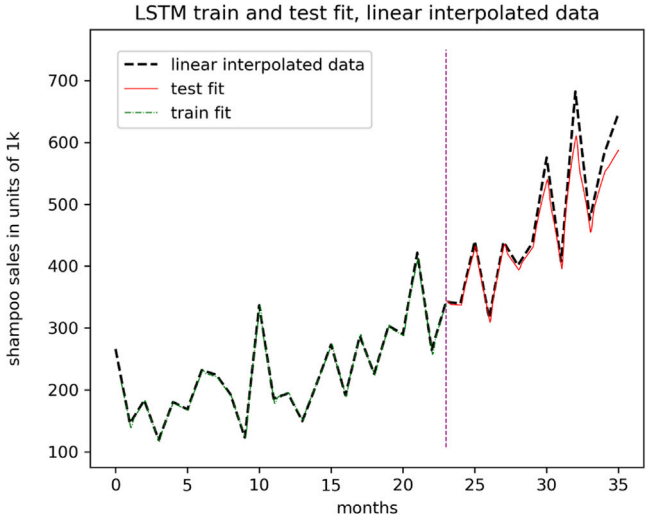

Fig. 4. Shampoo sales, linear interpolated.

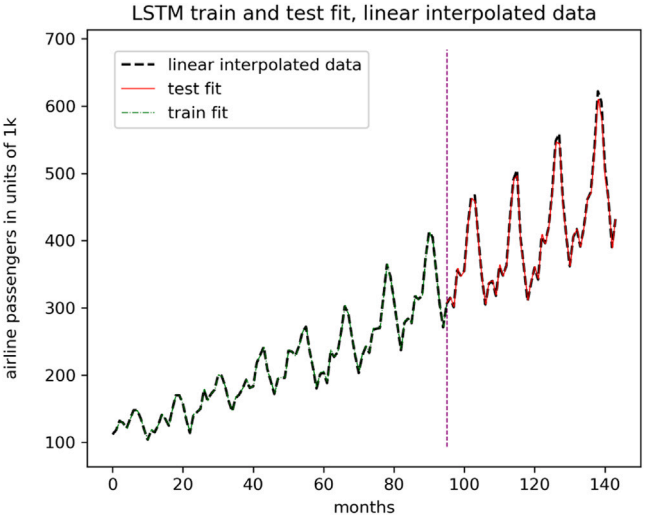

Fig. 7. Airline passengers fractal interpolated.

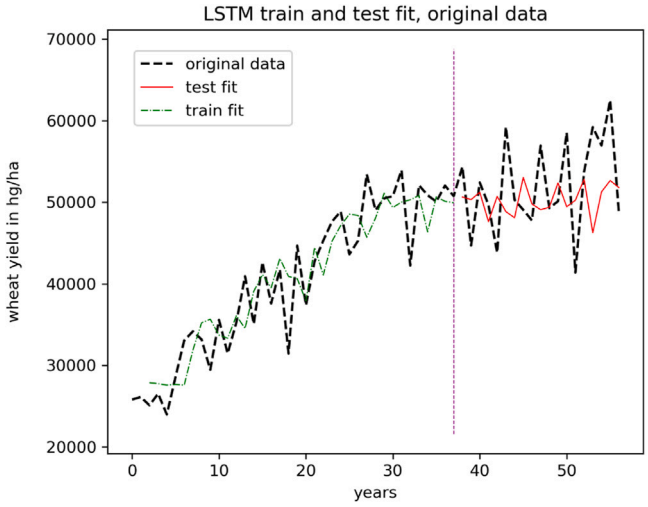

Fig. 8. Austrian wheat yields.

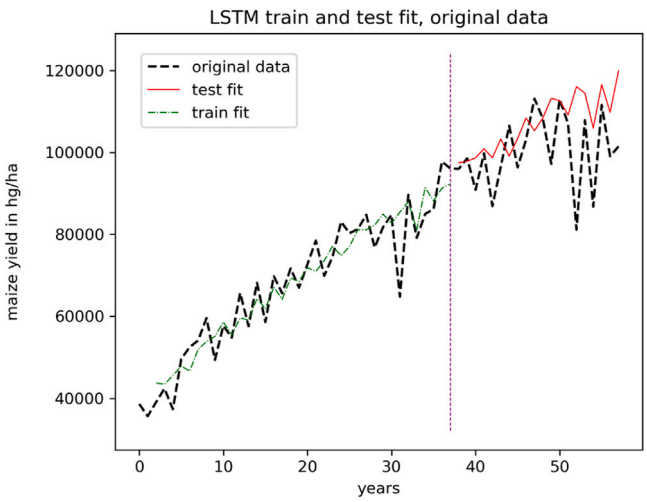

Fig. 11. Austrian maize yields.

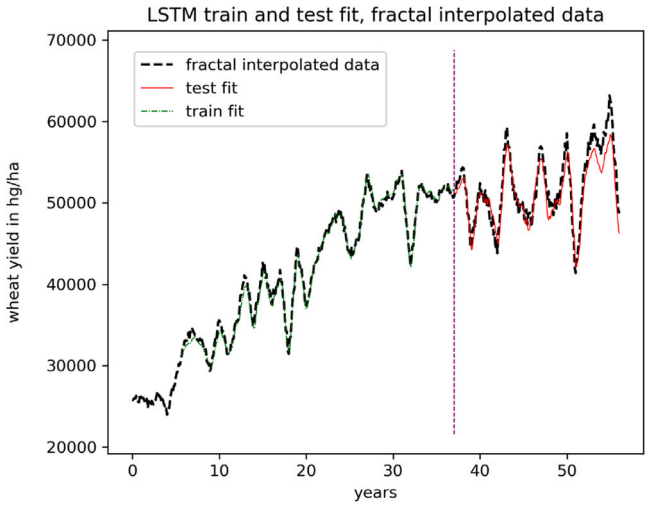

Fig. 9. Austrian wheat yields, fractal interpolated.

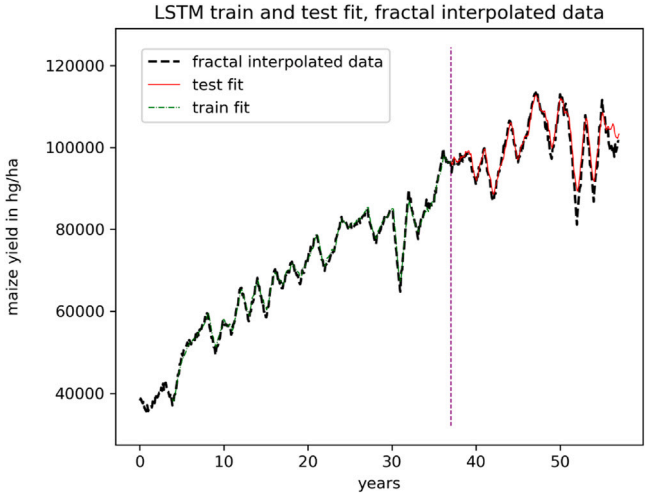

Fig. 12. Austrian maize yields, fractal interpolated.

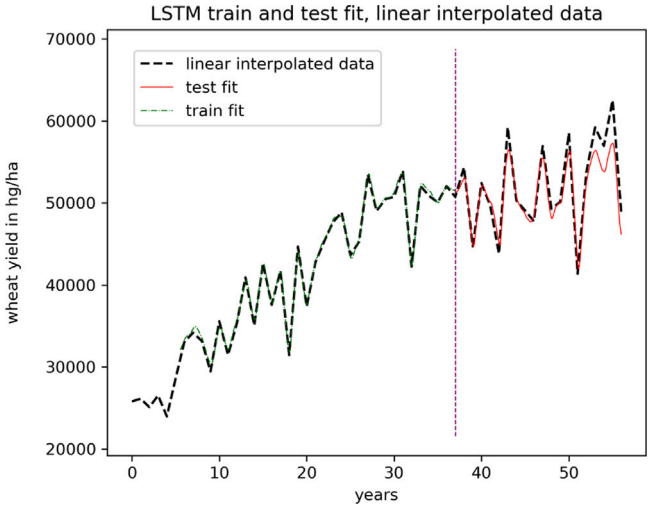

Fig. 10. Austrian wheat yields, linear interpolated.

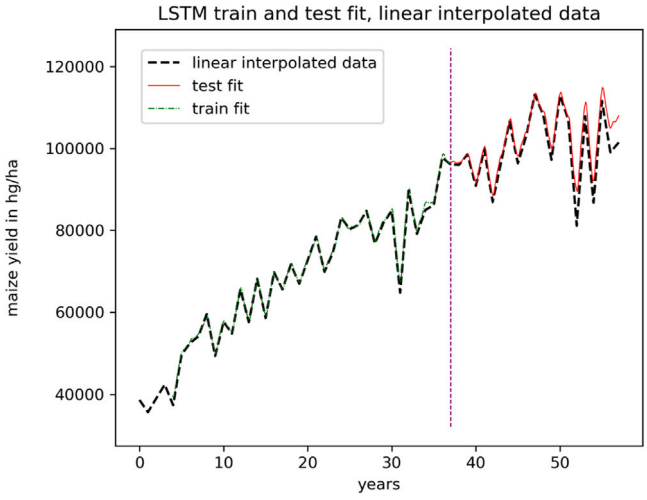

Fig. 13. Austrian maize yields, fractal interpolated.

Table 3
RMSE for each test run.

Data	Train data error	Test data error
Shampoo sales	0.1860	0.5839
Shampoo sales fractal interpolated	0.0236	0.0654
Shampoo sales linear interpolated	0.0132	0.0899
Airline passengers	0.0307	0.0566
Airline passengers fractal interpolated	0.0120	0.0237
Airline passengers linear interpolated	0.0058	0.0185
Wheat yields	0.1957	0.3298
Wheat yields fractal interpolated	0.0320	0.0849
Wheat yields linear interpolated	0.0212	0.0892
Maize yields	0.1382	0.3062
Maize yields fractal interpolated	0.0248	0.0558
Maize yields linear interpolated	0.0177	0.0756

7. Results and discussion

The results show that the accuracy of a LSTM neural network on difficult time series data can significantly be improved using a fractal or a linear interpolation method.

Regarding the errors of the predictions (Table 3), this is exactly what we observed. the errors on the training data. For all data sets, the interpolated approaches outperformed the regular ones. On the training data the linear interpolated approach performed best for all data sets. For the unknown data the fractal interpolated approach performed best except for the airline passengers data set, the author's guess is that this is because LSTM is already performing very well on this data set. Thus, we recommend a fractal interpolation approach for difficult data sets, and a linear interpolation approach to improve already good predictions.

Another characteristic of the interpolated times series data is that the corresponding spectrum of Lyapunov exponents suggests hyper-chaotic behavior (Zengrong et al., 1999), i.e. at least two positive Lyapunov exponents. In contrast, the original time series data have a maximum of two positive Lyapunov exponents for the airline passenger data set and the remaining three have zero positive Lyapunov exponents. The spectrum of Lyapunov exponents can also be interpreted as an indicator for a system to have multiple degrees of freedom (Bailin, 1989). Considering the two positive Lyapunov exponents of each interpolated time series, the conclusion is that the fractal interpolation method gives it sort of a pseudo-deterministic chaotic structure, just as would be expected from fine grained real-life time series data. Another observation regarding the Lyapunov exponents is that the lowest errors for fits on unknown data could be achieved on data with two positive Lyapunov exponents, i.e. the original airline passenger time series and all interpolated ones. Here, the RMSE is ≈ 0.06 or below.

8. Conclusion and outlook

Four different time series data were tested for a possible application of fractal or linear interpolation to improve predictions using a LSTM neural network. As was shown using the Hurst exponent and the fractal dimension, the interpolated time series data show a higher degree of persistency than the original data. The fits of the LSTM on all interpolated time series data outperforms the LSTM on the original data when it comes to the test, i.e. the unknown data. This was exactly what was expected for data with higher persistency, i.e. a higher Hurst exponent and a lower fractal dimension. For this very basic LSTM-implementation (since a lot more could be done to further improve the algorithm) the results are very encouraging, and can help to speed up calculations, since the number of epochs was reduced drastically. Another possible application would be to use fractal interpolation in research for time series prediction where one has to deal with difficult, i.e. random or very short time series data as in agriculture. Therefore the two yield data sets were chosen, and it was shown that the performance of the algorithm on unknown test data could significantly be improved using a fractal interpolation to alter the original data set. Research on the optimal number of interpolation points still has to be done. Also, other neural network architectures, e.g. gated recurrent neural networks, should be considered for future research.

Given that the lowest errors were achieved on data sets that have two positive Lyapunov exponents, one should examine the connection between the spectrum of Lyapunov exponents and the performance of a LSTM neural network. A possible use-case for this is multivariate time series prediction. Here, for some tasks, one is confronted with a multitude of different data to choose from. The spectrum of Lyapunov exponents can here be employed to discard data in order to enhance performance and speed up the calculations.

Though a lot of work still needs to be done in this sector, the results seem promising, and the fractal interpolation approach together with complexity measures to monitor the prediction process should be considered when dealing with difficult-to-predict time series data and neural networks.

CRediT authorship contribution statement

Sebastian Raubitzek: Conceptualization, Methodology, Software, Data curation, Writing - original draft, Visualization, Validation, Formal analysis. **Thomas Neubauer:** Supervision, Validation, Writing - review & editing, Project administration, Funding acquisition, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors acknowledge the funding of the project "DiLaAg – Digitalization and Innovation Laboratory in Agricultural Sciences", by the private foundation "Forum Morgen, Austria", the federal state of Lower Austria and by Project AI4CROPR, No. 877158 funded by the FFG, Austria.

References

Agliari, E., Barra, A., Barra, O. A., Fachechi, A., Franceschi Vento, L., & Moretti, L. (2020). Detecting cardiac pathologies via machine learning on heart-rate variability time series and related markers. *Scientific Reports*, *10*(1), 8845.

Bailin, H. (1989). *Elementary symbolic dynamics and chaos in dissipative systems*. World Scientific.

Barnsley, M. F., Demko, S., & Powell, M. J. D. (1985). Iterated function systems and the global construction of fractals. *Proceedings of the Royal Society of London, Series A (Mathematical and Physical Sciences)*, *399*(1817), 243–275, Publisher: Royal Society.

Chlingaryan, A., Sukkarieh, S., & Whelan, B. (2018). Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review. *Computers and Electronics in Agriculture*, *151*, 61–69.

Di Matteo, T. (2007). Multi-scaling in finance. *Quantitative Finance*, *7*(1), 21–36.

Diaconescu, E. (2008). The use of NARX neural networks to predict chaotic time series. *WSEAS Transactions on Computer research*, *3*(3), 182–191.

Eckmann, J.-P., Kamphorst, S., & Ciliberto, S. (1987). Liapunov exponents from time series. *Physical Review A*, *34*, 4971–4979.

Feder, J. (1988). *Physics of solids and liquids*, *Fractals*. Springer US.

Feller, W. (1971). *An Introduction to probability theory and its applications*. Hoboken, New Jersey: Wiley-Blackwell.

Henriques, T., Ribeiro, M., Teixeira, A., Castro, L., Antunes, L., & Costa Santos, C. (2020). Nonlinear methods most applied to heart-rate time series: A review. *Entropy*, *22*, 309.

Hey, T., Butler, K., Jackson, S., & Thiyagalingam, J. (2020). Machine learning and big scientific data. *Philosophical Transactions of the Royal Society of London A (Mathematical and Physical Sciences)*, *378*(2166), Article 20190054, Publisher: Royal Society.

Higuchi, T. (1988). Approach to an irregular time series on the basis of the fractal theory. *Physica D: Nonlinear Phenomena*, *31*(2), 277–283.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735–1780.

Hua, Y., Zhao, Z., Li, R., Chen, X., Liu, Z., & Zhang, H. (2019). Deep learning with long short-term memory for time series prediction. *IEEE Communications Magazine*, *57*(6), 114–119.

Hurst, H., Black, R., & Sinaika, Y. (1965). *Long-term storage in reservoirs: An experimental study*. London: Constable.

Karaca, Y., Zhang, Y.-D., & Muhammad, K. (2019). A novel framework of rescaled range fractal analysis and entropy-based indicators: Forecasting modelling for stock market indices. *Expert Systems with Applications*.

Larrañaga, P., Calvo, B., Sanatana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J. A., Armañanzas, R., Santafé, G., Pérez, A., & Robles, V. (2005). Machine learning in bioinformatics. *Briefings in Bioinformatics*, *7*(1), 86–112.

Makridakis, S., Wheelwright, S., & Hyndman, R. (1984). Forecasting: Methods and applications. *The Journal of the Operational Research Society*, *35*, Journal Abbreviation: The Journal of the Operational Research Society.

Manousopoulos, P., Drakopoulos, V., & Theoharis, T. (2008). Curve fitting by fractal interpolation. In M. L. Gavrilova, & C. J. K. Tan (Eds.), *Transactions on computational science (vol. I)* (pp. 85–103). Berlin, Heidelberg: Springer Berlin Heidelberg.

Mazel, D., & Hayes, M. (1992). Using iterated function systems to model discrete sequences. *IEEE Transactions on Signal Processing*, *40*, 1724–1734.

Ni, L.-P., Ni, Z.-W., & Gao, Y.-Z. (2011). Stock trend prediction based on fractal feature selection and support vector machine. *Expert Systems with Applications*, *38*.

Sakai, K. (2001). *Nonlinear dynamics and chaos in agricultural systems* (1st ed.). Amsterdam: Elsevier Science.

Schmidt, J., Marques, M. R. G., Botti, S., & Marques, M. A. L. (2019). Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, *5*(1), 83.

Selvaratnam, S., & Kirley, M. (2006). *Lecture notes in computer science : Advances in artificial intelligence*: *vol. 4304, no. II*, *Predicting stock market time series using evolutionary artificial neural networks with hurst exponent input windows*.

Vörös, Z., & Jankovičová, D. (2002). Neural network prediction of geomagnetic activity: a method using local Hölder exponents. *Nonlinear Processes in Geophysics*, *9*(5/6), 425–433.

Voyant, C., Notton, G., Kalogirou, S., Nivet, M.-L., Paoli, C., Motte, F., & Fouilloy, A. (2017). Machine learning methods for solar radiation forecasting: A review. *Renewable Energy*, *105*, 569–582.

Yakuwa, F., Dote, Y., Yoneyama, M., & Uzurabashi, S. (2003). Novel time series analysis & prediction of stock trading using fractal theory and time delayed neural net- work. In *IEEE SMC'03 2003 IEEE international conference on systems, man and cybernetics. conference theme - system security and assurance cat. no.03CH37483 (vol. 1)*.

Zengrong, L., Liqun, C., & Ling, Y. (1999). On properties of hyperchaos: Case study. *Acta Mechanica Sinica*, *15*(4), 366–370.