entropy

MDPI

Article

Taming the Chaos in Neural Network Time Series Predictions

Sebastian Raubitzek * and Thomas Neubauer

Information and Software Engineering Group, Institute of Information Systems Engineering, Faculty of Informatics, TU Wien, Favoritenstrasse 9-11/194, 1040 Vienna, Austria; thomas.neubauer@tuwien.ac.at
* Correspondence: sebastian.raubitzek@tuwien.ac.at

Abstract: Machine learning methods, such as Long Short-Term Memory (LSTM) neural networks can predict real-life time series data. Here, we present a new approach to predict time series data combining interpolation techniques, randomly parameterized LSTM neural networks and measures of signal complexity, which we will refer to as complexity measures throughout this research. First, we interpolate the time series data under study. Next, we predict the time series data using an ensemble of randomly parameterized LSTM neural networks. Finally, we filter the ensemble prediction based on the original data complexity to improve the predictability, i.e., we keep only predictions with a complexity close to that of the training data. We test the proposed approach on five different univariate time series data. We use linear and fractal interpolation to increase the amount of data. We tested five different complexity measures for the ensemble filters for time series data, i.e., the Hurst exponent, Shannon's entropy, Fisher's information, SVD entropy, and the spectrum of Lyapunov exponents. Our results show that the interpolated predictions consistently outperformed the non-interpolated ones. The best ensemble predictions always beat a baseline prediction based on a neural network with only a single hidden LSTM, gated recurrent unit (GRU) or simple recurrent neural network (RNN) layer. The complexity filters can reduce the error of a random ensemble prediction by a factor of 10. Further, because we use randomly parameterized neural networks, no hyperparameter tuning is required. We prove this method useful for real-time time series prediction because the optimization of hyperparameters, which is usually very costly and time-intensive, can be circumvented with the presented approach.

Keywords: Hurst exponent; chaos; Lyapunov exponents; neural networks; time series prediction; deep learning; machine learning; LSTM; R/S analysis

check for updates

Citation: Raubitzek, S.; Neubauer, T. taming the Chaos in Neural Network Time Series Predictions. *Entropy* **2021**, *23*, 1424. https://doi.org/10.3390/e23111424

Academic Editor: Boris Ryabko

Received: 27 September 2021
Accepted: 26 October 2021
Published: 28 October 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Machine learning and neural networks are today's state of the art when it comes to predicting time series data. Applications feature various research areas and tasks, such as future population estimates, predicting epileptic seizures [1], or estimating future stock market prices.

All machine learning approaches depend on the quality and quantity of the available data, i.e., their complexity or randomness and the actual amount of data, and the algorithm's right parameterization. The three main reasons for machine learning approaches to perform poorly are:

1. An insufficient amount of data, i.e., the data are not fine-grained or long enough;
2. Random data, with its blueprint, the Brownian motion (cf. [2]);
3. Bad parameterization of the algorithm.

A way to enrich a dataset and increase its fine-grainededness is to use interpolation. In the simplest case, this is done using a linear interpolation [3]. Complex real-life data are generally considered non-linear and originate from systems where the interactions and corresponding agents are not known in detail. Therefore, linear interpolation approaches do not depict the complexity and non-linearity of real-life systems. One approach to

consider the complexity of the data under study is fractal interpolation [4]. The challenge is to find the right interpolation technique for the task at hand, i.e., what interpolation technique is best suited for improving machine learning time series predictions?

There are many ways to measure the complexity or inherent information of time series data. This article will refer to these measures of signal complexity as *complexity measures* since they measure how complex the data under study are. Given that, the question is how can the complexity of the data under study be taken into account to improve predictions?

Finding the right set of hyperparameters for any given machine learning algorithm, especially for neural networks, can be a tedious and time-consuming task. In addition, there is usually more than one set of hyperparameters with high performance for a given dataset. Here, the hypothesis is that differently parameterized algorithms, again especially neural networks, capture different aspects of the data. However, is there a way to take into account different aspects of the data to reduce the effort of finding the right set of parameters?

We present an approach that addresses those three issues. The overall hypothesis here is that neural network time series predictions can be improved by considering the complexity of the data under study. First, we interpolate the data under study using a fractal interpolation adapted to the data complexity. Second, we use randomly parameterized long short-term memory (LSTM [5]) neural networks to make ensemble predictions. Third, we use complexity properties of the data for filtering the ensemble predictions to improve their accuracy. Using the presented ideas, we can reduce the error of a random long short term memory (LSTM [5]) neural network ensemble prediction on average by a factor of 10 by filtering the results depending on their complexity. Further, we set a baseline by using an LSTM, a GRU, [6], and a recurrent neural network, [7], with only one hidden layer to make predictions of the time series data under study. The best ensemble predictions always outperformed these baseline predictions. The discussed filter methods can be applied to any ensemble predictions and therefore be used to improve existing ensemble approaches.

We show that the fractal interpolation approach, which considers the complexity of the data under study, is the preferred method to improve neural network time series predictions. Therefore, one should consider the discussed interpolation techniques when facing an insufficient amount of data.

Given that we used a randomly parameterized neural network implementation, this approach is a more generalized one and can be applied to any univariate time series dataset without further optimization, i.e., we can circumvent the optimization procedure. In addition, since the ensemble consists of several independent neural networks, this approach can, in principle, be parallelized infinitely to speed up the calculations.

We further show how different filters perform for different datasets and give recommendations on which filters to employ to improve ensemble predictions.

The remainder of the paper is organized as follows. Section 2 gives an overview on related research. In Section 3, we describe the used methodology and discuss how the different techniques fit together. Section 4 presents all used datasets and Section 5 the corresponding interpolation techniques. In Section 6, we discuss the used complexity measures. Section 7 shows the neural network implementation and the ensemble scheme. The error metrics are discussed in Section 8. Section 9 describes the used ensemble (-complexity) filters. In Section 10, we present LSTM, GRU and recurrent neural network predictions as a baseline to compare our ensemble predictions to. Section 11 discusses the results. We further conclude our findings in Section 12. In addition, we added an Appendix to collect peripheral plots and data to keep the main text focused.

2. Related Work

Machine learning and neural networks are the state of the art when it comes to time series predictions. Tools, such as support vector regression, are used for a variety of applications, as shown in [8]. Here, the often studied problems are the forecasting of financial markets [9,10] and predicting electrical load forecasting [11]. Other applications

are to predict solar radiation [12] with, e.g., artificial neural networks (ANNs). In general, we expect neural networks to perform best for various types of data [13]. Popular types of neural networks used for predictions are recurrent neural networks (RNNs), as described in [14,15], or long short term memory neural networks (LSTM) [5]. LSTM neural networks are a category of RNNs [16] and are a class of artificial neural networks. RNNs are capable of using recurrent connections, i.e., a directed graph along a temporal sequence. Though, in theory, this is supposed to be very effective when analyzing time series, RNNs suffer from vanishing or exploding gradients when trained with Gradient Descent. LSTMs are designed to solve this issue and also to learn inherent long-term dependencies, and as such, are capable of predicting time series data. For these reasons, we chose to use LSTM neural networks in this study.

A feature specific for LSTMs is the so-called *memory block*, which enhances the LSTM's capability to learn and model long-term dependencies. This memory block is a subnet of the actual neural network, which is recurrently connected. Two functional modules are part of this subnet, the memory cell and corresponding gates. The memory cell serves to remember the neural network's temporal state, and the corresponding gates control the information flow and are multiplicative units. Three types of gates are used, input gates, output gates, and forget gates. The input gates, just like the name says, control how much information goes in the cell, and the output gates control the returned information, i.e., fuel the output activation. The forget gates, on the other hand, are responsible for containing information in the cell. All these mechanics of the LSTM are to serve the task at hand as best as possible.

In addition, ensemble methods, i.e., combinations of different neural networks or machine learning algorithms (or both), prove useful for predicting time series data, as done in [17] to forecast exchange rates. Here, the traditional approach is to *keep-the-best* (KTB). State-of-the-art is to use different neural network architectures to capture different aspects of the data, e.g., [18], where an ensemble of LSTM neural networks (in conjunction with other non-linear methods) is used to forecast wind speeds. Further, in [19], ensemble predictions could be improved when adding noise to the data under study, somehow similar to the noise added using fractal interpolation in this research.

When it comes to interpolation techniques to improve machine learning applications, one is tempted to use a linear interpolation as done in [3]. However, just as the name says, a linear interpolation is only a linear fit between some data points. One approach to consider the complexity of the data under study is fractal interpolation [4]. Traditional interpolation methods are based on elementary functions, such as polynomials. Fractal interpolation, in contrast, is based on iterated function systems. Iterated function systems can generate fractal and multi-fractal structures, therefore preserving the inherent complexity of the original data.

Measuring the complexity/information/randomness (i.e., non-linear properties) of given time series data can be done in many ways. One example is the Hurst exponent [20], which was found to be a measure for long-term memory of time series data. Other information measures use all sorts of entropy measures that can be applied to time series data, such as Shannon's entropy, as used in [21] to analyze astronomical time series data. See Section 6 for all employed complexity measures.

There are very few approaches combining complexity measures and machine learning yet. In [22], the local Hölder exponent is used as an additional complexity feature to the time series to improve predictions. In [3], the Hurst exponent, Rényi entropy and Shannon's entropy are employed to improve forecasts of financial markets and cryptocurrencies. In [23], ideas from fuzzy logic and fractal theory improve neural networks time series predictions. Further, in [24], a fractal interpolation taking into account the complexity of the data under study is used to improve LSTM time series predictions. As such, this research is based on the findings of [24].

3. Methodology

We apply a fractal interpolation method and a linear interpolation method to five datasets to increase the data fine-grainededness. The fractal interpolation was tailored to match the original data complexity using the Hurst exponent. Afterward, random LSTM neural networks are trained and used to make predictions, resulting in 500 random predictions for each dataset. These random predictions are then filtered using Lyapunov exponents, Fisher information and the Hurst exponent, and two entropy measures to reduce the number of random predictions. Here, the hypothesis is that the predicted data have to have the same complexity properties as the original dataset. Therefore, good predictions can be differentiated from bad ones by their complexity properties.

As far as the authors know, a combination of fractal interpolation, complexity measures as filters, and random ensemble predictions in this way has not been presented yet.

We developed a pipeline connecting interpolation techniques, neural networks, ensemble predictions, and filters based on complexity measures for this research. The pipeline is depicted in Figure 1.

First, we generated several different fractal-interpolated and linear-interpolated time series data, differing in the number of interpolation points (the number of new data points between two original data points), i.e., $\{1, 3, 5, 7, 9, 11, 13, 15, 17\}$ and split them into a training dataset and a validation dataset. (Initially, we tested if it is necessary to split the data first and interpolate them later to prevent information to leak from the train data to the test data. However, that did not make any difference in the predictions, though it made the whole pipeline easier to handle. This information leak is also suppressed as the interpolation is done sequentially, i.e., for separated subintervals.) Next, we generated 500 randomly parameterized long short-term memory (LSTM) neural networks and trained them with the training dataset. Then, each of these neural networks produces a prediction to be compared with the validation dataset. Next, we filter these 500 predictions based on their complexity, i.e., we keep only those predictions with a complexity (e.g., a Hurst exponent) close to that of the training dataset. The remaining predictions are then averaged to produce an ensemble prediction.

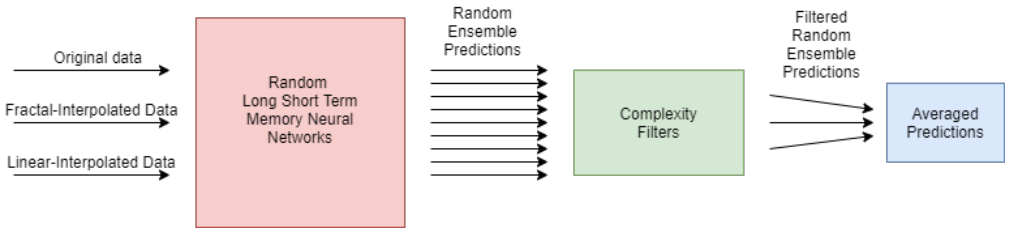


Figure 1. Schematic depiction of the developed pipeline. The whole pipeline is applied to three different sorts of data for each time series. First, the original non-interpolated data, second, the fractal-interpolated data, and third, the linear-interpolated.

4. Datasets

For this research, we tested five different datasets. All of them are real-life datasets, and some are widely used for time series analysis tutorials. All of them are contributed to [25] and are part of the *Time Series Data Library*. They differ in their number of data points and their complexity (see Section 6).

1. **Monthly international airline passengers:** January 1949 to December 1960, 144 data points, given in units of 1000. Source: Time Series Data Library, [25];
2. **Monthly car sales in Quebec:** January 1960 to December 1968, 108 data points. Source: Time Series Data Library [25];
3. **Monthly mean air temperature in Nottingham Castle:** January 1920 to December 1939, given in degrees Fahrenheit, 240 data points. Source: Time Series Data Library [25];

4. **Perrin Freres monthly champagne sales:** January 1964 to September 1972, 105 data points. Source: Time Series Data Library [25];
5. **CFE specialty monthly writing paper sales.** This dataset spans 12 years and 3 months, 147 data points. Source: Time Series Data Library [25].

5. Applied Interpolation Techniques

We apply two different interpolation techniques to generate more data points for the discussed datasets: Fractal interpolation and linear interpolation. We used the interpolation techniques to generate new interpolated datasets differing in the number of interpolation points, i.e., new data points between every two original data points. The interpolations were done for the following numbers $N_I = \{1, 3, 5, 7, 9, 11, 13, 15, 17\}$.

5.1. Fractal Interpolation of Time Series Data

For the fractal interpolation, we employ a method developed in [24]. With the actual interpolation described in [4]. Therefore, we only give a summary of this method and refer to the sources for further reading.

In contrast to traditional interpolation approaches based on polynomials, fractal interpolation is based on iterated function systems. Iterated function systems are defined as a complete metric space X with a corresponding distance function h and a finite set of contractive mappings, $\{w_n : X \to X \text{ for } n = 1, 2, \ldots, N\}$ [26]. For further reading on iterated function systems, we refer to [27].

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

An iterated function system is given as $\{\mathbb{R}^2; w_n, n = 1, 2, \ldots, N\}$ with the corresponding 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 satisfy

$$w_n \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} x_{n-1} \\ y_{n-1} \end{bmatrix} \quad \text{and} \quad w_n \begin{bmatrix} x_N \\ y_N \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix}, \tag{2}$$

for every $n = 1, 2, \ldots, N$. Solving these equations yields

$$a_n = \frac{x_n - x_{n-1}}{x_N - x_0} \quad , \tag{3}$$

$$d_n = \frac{x_N x_{n-1} - x_0 x_n}{x_N - x_0} \quad , \tag{4}$$

$$c_n = \frac{y_n - y_{n-1}}{x_N - x_0} - s_n \frac{y_N - y_0}{x_N - x_0} \quad , \tag{5}$$

$$e_n = \frac{x_N y_{n-1} - x_0 y_n}{x_N - x_0} - s_n \frac{x_N y_0 - x_0 y_N}{x_N - x_0} \quad . \tag{6}$$

The interpolation points determine the real numbers a_n, d_n, c_n, e_n and the *vertical scaling factor* s_n is a free parameter. s_n is bounded by $|s_n| < 1$ so that the IFS is hyperbolic with respect to an appropriate metric. Later on, s_n is the parameter used to ensure the IFS fits the original data the way we want it.

5.2. Fractal Interpolation Applied

The following procedure, from [24], 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;
2. For each sub-set i, calculate the corresponding Hurst exponent H_i;
3. For each subset i, the following routine is performed $k = 500$ times:
   (a) Use the fractal interpolation method from Section 5.1 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, and the corresponding fractal interpolation and $H_{i,\text{int,old}}$ is set to $H_{i,\text{int,new}}$.

Remarks:

- The Hurst exponent was calculated using *R/S Analysis* [20];
- The number of iterations k was set to 500 for each dataset;
- 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 could not be reached.

In Figure 2, the fractal interpolation is shown for the monthly international airline passengers dataset.

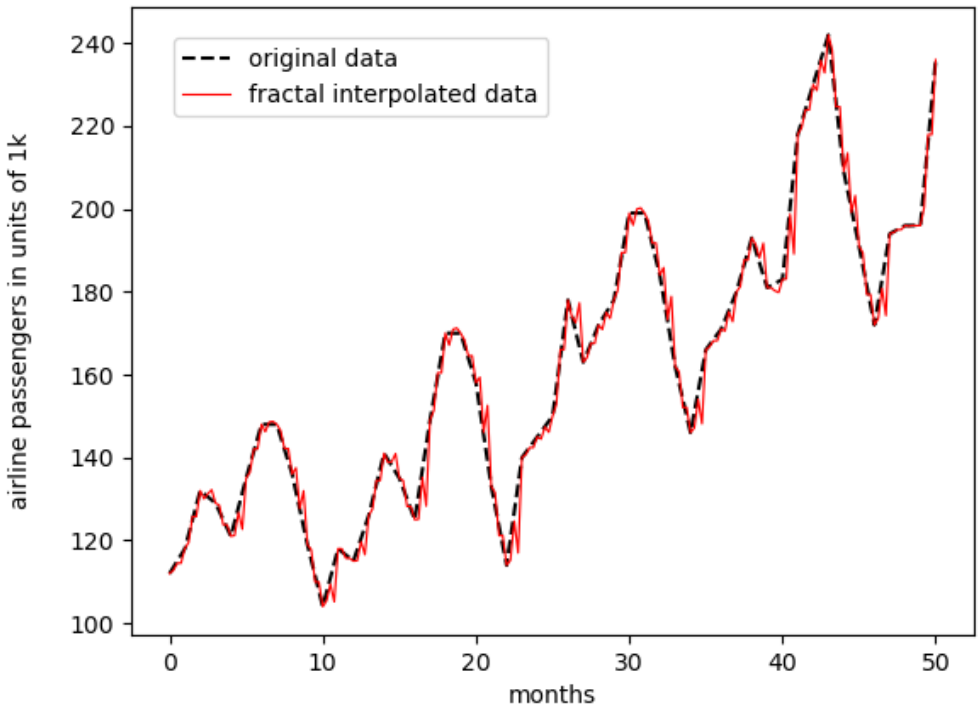


Figure 2. Fractal-interpolated monthly airline passengers data, first 50 data points.

5.3. Linear Interpolation

The second method that was used to increase the number of data points is linear interpolation. Therefore, the original data $\{(u_m, v_m) \in \mathbb{R}^2\} : m = 0, 1, \ldots, M$ is interpolated using a linear fit $y_i = a_m x_i + b_m$, to obtain the interpolation points

$\{(x_i, y_i) \in \mathbb{R}^2 : i = 0, 1, \ldots, N\}$. This was done for each interval $[u_m, u_{m+1}]$. The coefficients a_m and b_m are calculated using

$$\begin{aligned} a_m &= \frac{v_{m+1} - v_m}{u_{m+1} - u_m} \quad \text{and} \quad b_m = v_m - a_m u_m \,, \\ \forall\, m &= 0, 1, \ldots, M-1 \quad . \end{aligned} \tag{7}$$

6. Measuring the Complexity of the Data

First, we used the Hurst exponent to adjust the fractal interpolation (see Section 5.1) to be as close as possible to the original data in terms of its Hurst exponent.

Second, we compared the complexities, i.e., the complexity measures presented in this section, with the original complexity as a function of the interpolation points and the method used, i.e., fractal or linear interpolation.

Third, we used the complexity measures as a filter to improve the random ensembles' accuracy which is discussed in Section 9.

The results presented in this section are not all complexity measures considered in this research. Before using the listed complexity/information/entropy measures, we performed an evaluation of additional complexity measures, including *detrended fluctuation analysis* (DFA) [28], some algorithms to calculate the *fractal dimension* [29] of a time series and the *generalized Hurst exponent* [30]. The complexity measures did not make it into the results because they performed poorly in the initial tests. We mention them here to be useful for future time series analysis approaches.

The *fractal dimension* of time series data was excluded because of its similarity to the *Hurst exponent* and *R/S analysis* [29].

In the following, we give an overview of the used complexity measures:

6.1. The Hurst Exponent (R/S Analysis)

The Hurst exponent is a measure of long term memory in time series data and it is calculated using *R/S Analysis* [20]. We only outline the main aspects of R/S analysis, for an in-depth treatment of the topic we refer to [20,31].

The rescaled range analysis (*R/S analysis*) is a technique to analyze long-run correlations in signals, and yields one parameter, the *Hurst exponent* "H".

Given a signal $[x_1, x_2, \ldots, x_n]$, the average over a period τ (a sub-interval of the signal, i.e., $1 \leq \tau \leq n$), with a corresponding k as $1 \leq k \leq n$ and elements i in this interval such that $k \leq i \leq k + \tau$.

$$\langle x \rangle_{\tau,k} = \frac{1}{\tau} \sum_{j=k}^{k+\tau} x_j \quad . \tag{8}$$

The accumulated departure $\delta x(i, \tau, k)$ over a period $i \in 1, 2, \ldots, \tau$ is:

$$\delta x(i, \tau, k) = \sum_{j=k}^{i} \Big(x_j - \langle x \rangle_{\tau,k} \Big) \tag{9}$$

The difference between maximal and minimal values of all x_i in the interval $[k, k + \tau]$ is referred to as the range R of this interval τ:

$$\begin{aligned} R(\tau, k) &= \max[\delta x(i, \tau, k)] - \min[\delta x(i, \tau, k)] \,, \\ &\text{satisfying } k \leq i \leq k + \tau \,. \end{aligned} \tag{10}$$

We further use the standard deviation for each sub-interval:

$$S(\tau, k) = \sqrt{\frac{1}{\tau} \sum_{i=k}^{k+\tau} \Big[x_i - \langle x \rangle_{\tau,k} \Big]^2} \quad . \tag{11}$$

The next step is then to average the range and the standard deviation over all possible k as

$$R(\tau) = \frac{\sum_k R(\tau,k)}{\text{number of different } k\text{s}}$$
$$\text{and} \tag{12}$$
$$S(\tau) = \frac{\sum_k S(\tau,k)}{\text{number of different } k\text{s}},$$

where $1 \leq k \leq n$ and $k \leq i \leq k+\tau$. The *Hurst exponent* H is then defined using the scaling properties as

$$\frac{R(\tau)}{S(\tau)} \propto \tau^H \quad . \tag{13}$$

The asymptotic behavior, [20] and [32], for any independent random process with finite variance is given as

$$\frac{R(\tau)}{S(\tau)} = \left(\frac{\pi}{2v}\tau\right)^{\frac{1}{2}} \quad , \tag{14}$$

where v is the equidistant interval between the data points of the signal x_i. This implies $H = \frac{1}{2}$, however, only for completely random processes. For real-life data, $H \neq \frac{1}{2}$, because real-life processes usually feature long-term correlations. The value of H is set to $0 < H < 1$. A value $H < 0.5$ indicates anti-persistency, meaning that low values follow on high values and vice versa, thus heavily fluctuating, but not totally random. Values close to 0 prove strong anti-persistency. Contrarily, values of $H > 0.5$ indicate persistent behavior and strong persistence for values close to 1. It is also important to note that time series with $H \neq 0.5$ can theoretically be forecast [33].

For *R/S analysis* and to calculate the Hurst exponent, we used the algorithm provided by the python package `nolds`, https://pypi.org/project/nolds/, accessed on 20 October 2021 [34].

6.2. The Lyapunov Exponents' Spectrum

The spectrum of Lyapunov exponents is a measure for a system's predictability depending on initial conditions. When it comes to experimental time series data, as there are not different trajectories for different initial conditions available, we still interpret it as a measure for predictability [35]. As the calculation and the corresponding algorithm to calculate Lyapunov exponents of experimental data are too complex to be presented here, we refer to [35] for an in-depth discussion of the topic.

The employed algorithm from the python package `nolds`[34] yields the first 4 exponents of the spectrum. When referring to the largest Lyapunov exponent, we thereby mean the first one of the spectrum. In general, a positive Lyapunov exponent is a strong indicator for chaos [36]. Therefore, in most cases, it is sufficient to calculate the first Lyapunov exponent of the spectrum, hence the largest Lyapunov exponent. Systems possessing several positive Lyapunov exponents are referred to as hyperchaotic [37].

6.3. Fisher's Information

Fisher's information can be interpreted as the amount of information extracted from a set of measurements, i.e., the quality of the measurements [38]. Additionally, it can be interpreted as a measure of order or disorder of a system or phenomenon. It can also be used to investigate non-stationary and complex signals, hence ideally suited for this study.

The discrete version of Fisher's information is suitable for univariate time series analysis, given as a signal $[x_1, x_2, \ldots, x_n]$.

First, we construct embedding vectors as:

$$\vec{y}(i) = \left[x_i, x_{i+\tau}, \ldots, x_{i+(d_E-1)*\tau}\right] \quad , \tag{15}$$

with time delay τ and an embedding dimension d_{E}. The embedding space, as a matrix, then is:

$$Y = [\vec{y}(1), \vec{y}(2), \ldots, \vec{y}(N - (d_{\mathrm{E}} - 1)\tau)]^{\mathrm{T}}. \tag{16}$$

On this matrix, we perform a single value decomposition, [39], yielding M singular values σ_i with the corresponding normalized singular values:

$$\bar{\sigma}_i = \frac{\sigma_i}{\sum_{j=1}^{M} \sigma_j} \quad . \tag{17}$$

Thus, we find Fisher's Information as

$$I_{\mathrm{Fisher}} = \sum_{i=1}^{M-1} \frac{[\bar{\sigma}_{i+1} - \bar{\sigma}_i]^2}{\bar{\sigma}_i}. \tag{18}$$

Here, the implementation from the python package `neurokit`, https://neurokit.readthedocs.io/en/latest/, accessed on 28 October 2021 [40] was used. This implementation requires two parameters, first the time delay, which was found using the calculation of the average mutual information from [41] and the embedding dimension which was set to 3 for all time series data, a possible alternative for detecting the embedding dimension would be to use the *False Nearest Neighbours* algorithm [42] to determine the embedding dimension. The reason for this is that Takens' theorem [43] guarantees that a correct embedding space for systems consisting of m ordinary coupled differential equations is $2m + 1$ dimensional. For real-life data, however, we cannot employ a proper model of differential equations and, therefore, stick with 3. Further, one can find a discussion on why we chose 3 and the actual estimated embedding dimensions in Appendix F. In addition, since we define the interpolated and non-interpolated time series data to be of same origin, it makes sense to keep the embedding dimension d_{E} constant for different interpolations but not the time delay τ.

6.4. SVD Entropy

SVD entropy (Single Value Decomposition) is an entropy measure based on the correlation matrix of a time series and a corresponding single value decomposition. It is known to be applicable to stock market data as a prediction criterion as done in [44,45].

SVD entropy is calculated by constructing an embedding space for a signal $[x_1, x_2, \ldots, x_n]$ with delay vectors as [46]:

$$\vec{y}(i) = \left[x_i, x_{i+\tau}, \ldots, x_{i+(d_{\mathrm{E}}-1)*\tau}\right] \quad , \tag{19}$$

with the corresponding time delay τ and the embedding dimension d_{E}. We construct the embedding space as the matrix:

$$Y = [\vec{y}(1), \vec{y}(2), \ldots, \vec{y}(N - (d_E - 1)\tau)]^{\mathrm{T}}. \tag{20}$$

For this matrix, a single value decomposition, [39], is then performed to get M singular values $\sigma_1, \ldots, \sigma_M$ known as the singular spectrum. Further, we find the corresponding spectrum of normalized singular values as:

$$\bar{\sigma}_i = \frac{\sigma_i}{\sum_{j=1}^{M} \sigma_j}. \tag{21}$$

Using the formula for Shannon's entropy then yields SVD entropy as:

$$H_{\mathrm{SVD}} = \sum_{i=1}^{M} \bar{\sigma}_i \log_2 \bar{\sigma}_i \tag{22}$$

We used the implementation from the python package `neurokit` [40]. For this algorithm, just as for the algorithm for Fisher's information, we have to find two additional parameters, i.e., the embedding dimension and the time delay. Both were found the same way as above for Fisher's Information.

6.5. Shannon's Entropy

Given a signal $[x_1, x_2, \ldots, x_n]$, we then find the probability to occur for each value denoted as $P(x_1), \ldots, P(x_n)$, thus, we formulate Shannon's entropy [47] as:

$$H_{\text{Shannon}} = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \quad . \tag{23}$$

Giving units as bits, the base of the logarithm is set to 2. Applications include astronomy [21], where it is used to identify periodic variability, or in finance [48], as to measure the diversity of portfolios or to estimate risks. Shannon's entropy is a measure for the uncertainty of a (random) process/signal.

6.6. Initial Complexity

We applied the five featured complexity measures to the original datasets. The results can be found in Table 1. We briefly discuss the complexities for each measure separately, as an in-depth discussion with regards to their predictability can be found in Section 11:

- **The Hurst exponent:** The most persistent dataset, with a Hurst exponent of 0.7988, is the dataset of monthly car sales in Quebec. According to [33], we expected that time series data with a very high Hurst exponent can be predicted with higher accuracy than ones with a value close to 0.5, as it is considered more random. The datasets under study are three persistent ones, i.e., with a Hurst exponent larger than 0.5. Contrary to that, we obtained two anti-persistent ones with a Hurst exponent below 0.5;
- **The largest Lyapunov exponent:** All largest Lyapunov exponents of all time series data under study are positive, just as we would expect from chaotic or complex real-life data. The dataset with the highest value is the monthly car sales data in the Quebec dataset. As the Lyapunov exponents of experimental time series data serve as a measure for predictability, we suggest this dataset, therefore, to be forecast with the least accuracy;
- **Fisher's information:** The dataset with the highest value of Fisher's information is the monthly international airline passengers dataset. The lowest value can be found for the Perrin Freres monthly champagne sales dataset. [38]: It is expected that Fisher's information behaves contrary to entropy measures since it is a measure for order/quality, which we observed only for SVD entropy. This means that the Fisher's information value for the monthly international airline passengers dataset is the highest, and the corresponding value for the SVD entropy is the lowest among all datasets. The reason for this may be that, just like Fisher's information, SVD entropy is based on a Single Value Decomposition [39]. Considering Shannon's entropy, which is also an entropy measure but not one based on Single Value decomposition, its behavior differs;
- **SVD entropy:** The largest value of SVD entropy is possessed by the Perrin Freres monthly champagne sales dataset, which has, just as expected, the lowest value of Fisher's information. We expect, as it has a very high SVD entropy value, the Perrin Freres champagne sales dataset not to be predicted with high accuracy;
- **Shannon's entropy:** Shannon's entropy is based on the frequency of occurrence of a specific value. As we deal with non-integer-valued complex datasets, we expect Shannon's entropy not to be of much use. Shannon's entropy's highest value is found for the monthly mean temperature in Nottingham Castle dataset. Since reoccurring

temperature distributions possess a higher regularity than, e.g., airline passengers, this explains the corresponding value.

Table 1. Complexities of the original datasets.

	Hurst Exponent	Largest Lyapunov Exponent	Fisher's Information	SVDEntropy	Shannon's Entropy
Monthly international airline passengers	0.4233	0.0213	0.7854	0.3788	6.8036
Monthly car sales in Quebec	0.7988	0.0329	0.5965	0.5904	6.7549
Monthly mean air temperature in Nottingham Castle	0.4676	0.0069	0.6617	0.5235	7.0606
Perrin Freres monthly champagne sales	0.7063	0.0125	0.3377	0.8082	6.6762
CFE specialty monthly writing paper sales	0.6830	0.0111	0.5723	0.6138	7.0721

6.7. Complexity Plots

We compared the complexities of all time series data under study, i.e., for different interpolation techniques and different numbers of interpolation points.

The results are shown in Figures 3–5 for the monthly international airline passengers dataset, the plots for all the other time series data can be found in Appendix A. Note that, in each of the plots, the blue line, i.e., the complexity of the non-interpolated time series is not a plot depending on the number of interpolation points, but a constant, since there is only one dataset, the one with zero interpolation points, and plotted as a line for reference. In addition, those original complexities are contained in Table 1.

For this study, we observe the following behavior:

- The Hurst exponent of the fractal and the linear interpolated datasets behave very similar to each other for the monthly international airline passengers dataset, see Figure 3. We observe similar behavior for the other datasets as well, see Appendix A. Though the Hurst exponent is initially lower for the fractal-interpolated data for some datasets, the Hurst exponent does not differ significantly between fractal and linear interpolated time series data. In addition, adding more interpolation points increases the Hurst exponent and makes the datasets more persistent;
- The Largest Lyapunov exponents of the fractal-interpolated data are much closer to the original data than the ones for the linear-interpolated data; see Figure 4. We observe the same behavior for all datasets; see Appendix A;
- Fisher's information for the fractal-interpolated dataset is closer to that of the original dataset (see Figure 3). We observe the same behavior for all datasets, as can be seen in Appendix A;
- Just as expected, SVD entropy behaves contrary to Fisher's information. In addition, the SVD entropy of the fractal interpolated time series is closer to that of the non-interpolated time series; see Figure 5. The same behavior and, specifically, the behavior contrary to that of Fisher's information can be observed for all datasets under study; see Appendix A;
- Shannon's entropy increases. This can be explained as follows: As more data points are added, the probability of hitting the same value increases. However, this is just what Shannon's entropy measures. For small numbers of interpolation points, Shannon's

entropy of the fractal interpolated time series data is closer to the original complexity than the linear interpolated time series data. For large numbers of interpolation points, Shannon's entropy performs very similarly, not to say overlaps, for the fractal- and linear-interpolated time series data. This behavior can be observed for all datasets, see Figure 4 and Appendix A.

Summing up our findings of the complexity analysis above, we find that:

- The fractal interpolation captures the original data complexity better, compared to the linear interpolation. We observe a significant difference in their behavior when studying SVD entropy, Fisher's information, and the largest Lyapunov exponent. This is especially true for the largest Lyapunov exponent, where the behavior completely differs. The largest Lyapunov exponent of the fractal interpolated time series data stays mostly constant or behaves linearly. The largest Lyapunov exponent of the linear-interpolated data behaves approximately like a sigmoid function, and for some datasets even decreases again for large numbers of interpolation points.
- Both Shannon's entropy and the Hurst exponent seem not suitable for differentiating between fractal- and linear-interpolated time series data.

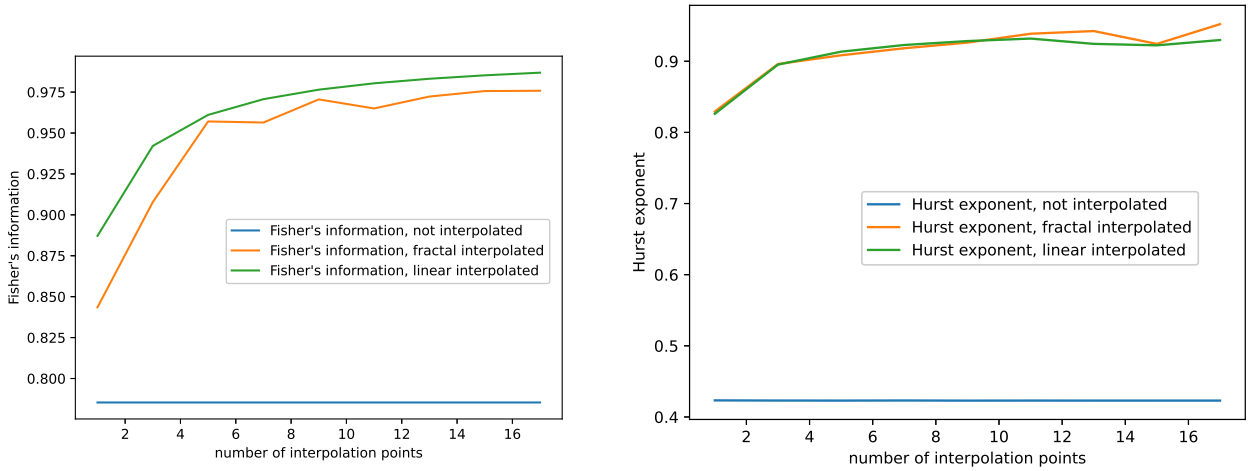


Figure 3. Plots for Fisher's information and the Hurst exponent depending on the number of interpolation points for the non-interpolated, the fractal-interpolated and the linear-interpolated data. Monthly international airline passengers dataset.

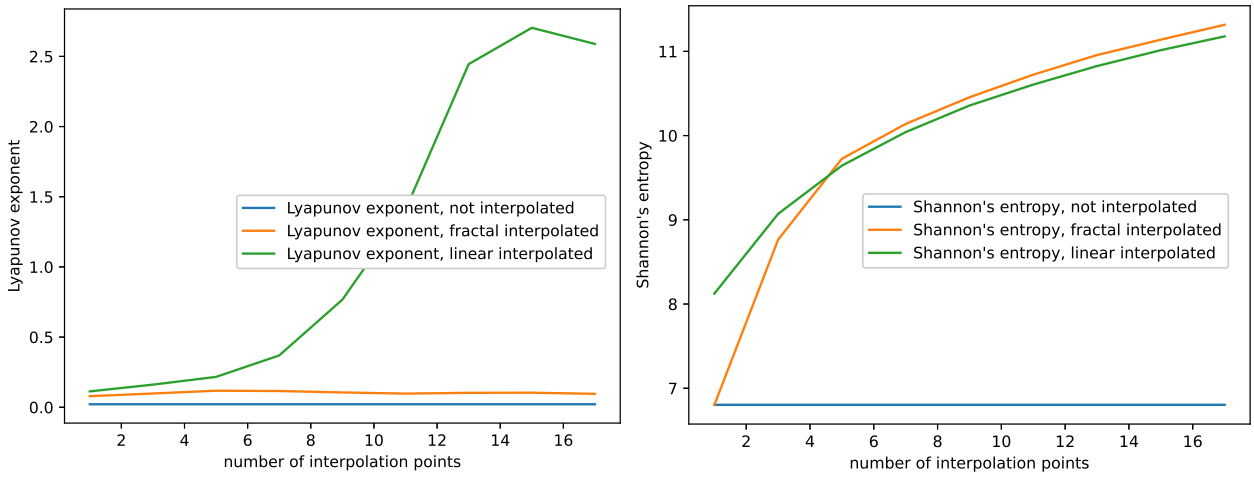


Figure 4. Plots for the Largest Lyapunov exponent and Shannon's entropy depending on the number of interpolation points for the non-interpolated, the fractal-interpolated and the linear-interpolated data. Monthly international airline passengers dataset.

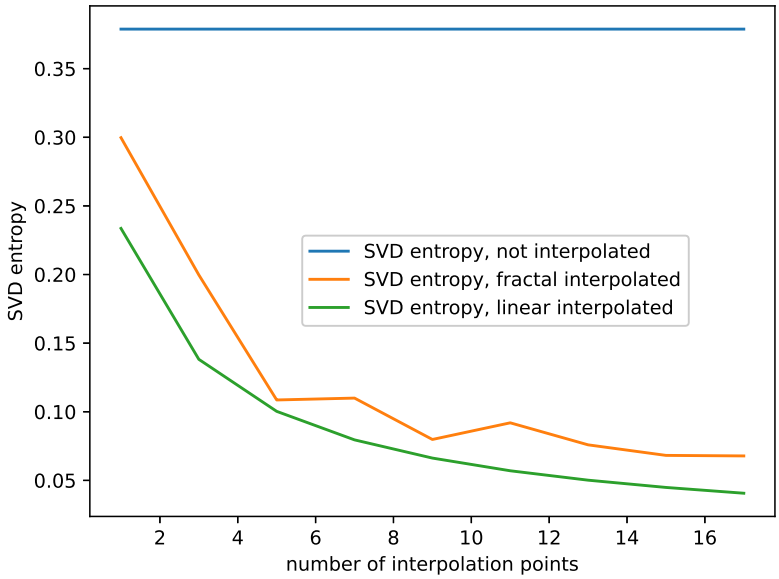


Figure 5. Plot for the SVD entropy depending on the number of interpolation points, for the non-interpolated, the fractal-interpolated and the linear-interpolated data. Monthly international airline passengers dataset.

7. LSTM Ensemble Predictions

For predicting all time series data, we employed random ensembles of different long short term memory (LSTM) [5] neural networks.

Our approach is to not optimize the neural networks but to generate many of them, in our case 500, and use the averaged results to obtain the final prediction.

For all neural network tasks, we used an existing `keras 2.3.1` implementation.

7.1. Data Preprocessing

Two basic concepts of data preprocessing were applied to all datasets before the ensemble predictions.

First, the data $X(t)$ defined at discrete time intervals v, thus $t = v, 2v, 3, \ldots kv$, were scaled so that $X(t) \in [0, 1]$, $\forall t$. This was done for all datasets. Second, the data were made stationary by detrending them using a linear fit.

All datasets were split so that the first 70% were used as a training dataset and the remaining 30% to validate the results.

7.2. Random Ensemble Architecture

As previously mentioned, we used a random ensemble of LSTM neural networks. Each neural network was generated at random and consists of a minimum of 1 `LSTM` layer and 1 `Dense` layer and a maximum of 5 `LSTM` layers and 1 `Dense` layer.

Further, for all activation functions (and the recurrent activation function) of the `LSTM` layers, `hard_sigmoid` was used and `relu` for the `Dense` layer. The reason for this is that, at first, `relu` for all layers was used and we sometimes experienced very large results that corrupted the whole ensemble. Since `hard_sigmoid` is bound by $[0, 1]$ changing the activation function to `hard_sigmoid` solved this problem. Here, the authors' opinion is that the shown results can be improved by an activation function, specifically targeting the problems of random ensembles.

Overall, no regularizers, constraints or Drop out criteria have been used for the `LSTM` and `Dense` layers.

For the initialization, we used `glorot_uniform` for all LSTM layers, `orthogonal` as the recurrent initializer and `glorot_uniform` for the Dense layer.

For the LSTM layer, we also used `use_bias=True`, with `bias_initializer="zeros"` and no constraint or regularizer.

The optimizer was set to `rmsprop` and, for the loss, we used `mean_squared_error`. The output layer always returned only one result, i.e., the next time step. Further, we randomly varied many parameters for the neural network:

- **LSTM layers**: min 1, max 5;
- **Size of the input layer**: min 1, max depending on the size of the data, i.e., length of the training data -1;
- **Epochs**: min 1, max 30;
- **Neurons**: For both LSTM and Dense layers, min 1, max 30;
- **Batchsize**: The batchsize was chosen randomly from $\{1, 8, 32, 64, 128\}$.

For each prediction, 500 of these networks were created, trained, and generated a prediction for the unknown data.

8. Error Analysis

For each ensemble prediction, i.e., consisting of several different predictions labeled with i for each time step t, we calculated the mean and the standard deviation as

$$\hat{X}(t) = \frac{1}{N_p}\sum_{i=1}^{N_p}\hat{X}_i(t)\,, \quad \sigma(t) = \sqrt{\frac{1}{N_p}\sum_{i=1}^{N_p}(\hat{X}_i(t) - \hat{X}(t))^2}\,, \tag{24}$$

where $\hat{X}_i(t)$ is a single observation, $\hat{X}(t)$ is the averaged observation for a single time step, $\sigma(t)$ is the corresponding standard deviation and N_p is the number of different predictions for each time step.

Next, to compare it to the validation dataset, we calculated the root-mean-square error (RMSE) as

$$E_{RMSE} = \left(\frac{1}{N_t}\sum_{t=1}^{N_t}\left[\hat{X}(t) - X(t)\right]^2\right) \quad , \tag{25}$$

where $X(t)$ are the data points of the validation dataset and N_t is the number of validation data points. Using error propagation, the corresponding error of the root-mean-square error was calculated as

$$\Delta E_{RMSE} = \sqrt{\left(\frac{\partial E_{RMSE}}{\partial \hat{X}(1)}\right)^2\sigma^2(1) + \left(\frac{\partial E_{RMSE}}{\partial \hat{X}(2)}\right)^2\sigma^2(2) + \cdots}\,, \tag{26}$$

thus yielding:

$$\Delta E_{RMSE} = \sqrt{\frac{\sum_{t=1}^{N_t}\left[\hat{X}(t) - X(t)\right]^2\sigma^2(t)}{N_t * \sum_{t=1}^{N_t}\left[\hat{X}(t) - X(t)\right]^2}} \quad . \tag{27}$$

9. Complexity Filters

For each time series data, we produced statistics consisting of 500 random predictions for the non-interpolated data, for the fractal-interpolated data, and the linear-interpolated data. Averaging these predictions yields the ensemble results which can be found in Tables A5 and A6. Since these averaged predictions are tainted with very bad predictions (because, after all, these are non-optimized, randomly parameterized neural networks), we propose that good predictions need to have a similar complexity as the original (fractal or linear-interpolated, respectively) time series. Therefore, we employed several filters to keep those predictions with complexities close to the original (fractal or linear-interpolated, respectively) time series. Therefore, five different filters were applied to all ensemble predictions so that only 1% of all predictions remained, i.e., 5 of 500 predictions. For an in-depth discussion of the employed complexity measures, see Section 6.

1. **Hurst exponent filter:** The Hurst exponent of each prediction of the ensemble was calculated and compared to the complexity of the corresponding training dataset: (The complexity of the fractal interpolated training data was compared with the complexity of the fractal-interpolated predictions, the same is true with linear-interpolated

prediction and training datasets and non-interpolated training and test datasets.) (Note that this is crucial for real-time predictions as there is no validation dataset.)

$$\Delta H = \left| \hat{H} - H \right| \quad , \tag{28}$$

where $\hat{H}$ is the Hurst exponent of a prediction and H is the Hurst exponent of the training dataset. Afterward, only the ensemble predictions with Hurst exponents similar to the training dataset, i.e., with low ΔH are kept, and the others are discarded;

2. **Lyapunov exponents filter:** The first 4 Lyapunov exponents of each prediction of the ensemble were calculated and compared with those of the training dataset.

$$\Delta L = \sum_{i=1}^{4} \left| \hat{L}_i - L_i \right| \tag{29}$$

where $\hat{L}_i$ is the ith Lyapunov exponent of a prediction and L_i is the ith Lyapunov exponent of the training dataset. Thus, only the predictions with the lowest ΔL are kept, and the others are discarded;

3. **Fisher's information filter:** Fisher's information of each prediction of the ensemble was calculated and compared to Fisher's information of the training dataset:

$$\Delta I_{\text{Fisher}} = \left| \hat{I}_{\text{Fisher}} - I_{\text{Fisher}} \right| \quad , \tag{30}$$

where $\hat{I}_{\text{Fisher}}$ is Fisher's information of a prediction and I_{Fisher} is Fisher's information of the training dataset. Afterward, only the predictions of the ensemble with Hurst exponents similar to the training dataset, i.e., with low ΔI_{Fisher} are kept and the others are discarded;

4. **SVD entropy filter:** The SVD entropy of each prediction of the ensemble was calculated and compared to the SVD entropy of the training dataset:

$$\Delta H_{\text{SVD}} = \left| \hat{H}_{\text{SVD}} - H_{\text{SVD}} \right| \quad , \tag{31}$$

where $\hat{H}_{\text{SVD}}$ is the SVD entropy of a prediction and H_{SVD} is the SVD entropy of the training dataset. Afterward, only the predictions of the ensemble with a SVD entropy similar to the training dataset, i.e., with low ΔH_{SVD}, are kept and the others are discarded;

5. **Shannon's entropy filter:** Shannon's entropy of each prediction of the ensemble was calculated and compared to the Shannon's entropy of the training dataset:

$$\Delta H_{\text{Shannon}} = \left| \hat{H}_{\text{Shannon}} - H_{\text{Shannon}} \right| \quad , \tag{32}$$

where $\hat{H}_{\text{Shannon}}$ is Shannon's entropy of a prediction and H_{Shannon} is Shannon's entropy of the training dataset. Afterward, only the predictions of the ensemble with Shannon's entropy similar to the training dataset, i.e., with low $\Delta H_{\text{Shannon}}$, are kept and the others are discarded.

In addition, all mentioned filters were applied in combination with each other, e.g., first, the Hurst exponent filter and afterward the Fisher's information filter, to yield a remaining 1% of the ensemble. Therefore, the first filter reduces the whole ensemble to only 10%, i.e., 50 predictions, and the second filter reduces the remaining predictions to 10%, i.e., five predictions, thus 1%.

Figure 6 depicts the idea of the complexity filters. The left image shows the whole ensemble without any filtering, and the right side shows the filtered ensemble prediction. In this particular case, a filter combining SVD entropy and Lyapunov exponents was used to improve the ensemble predictions.

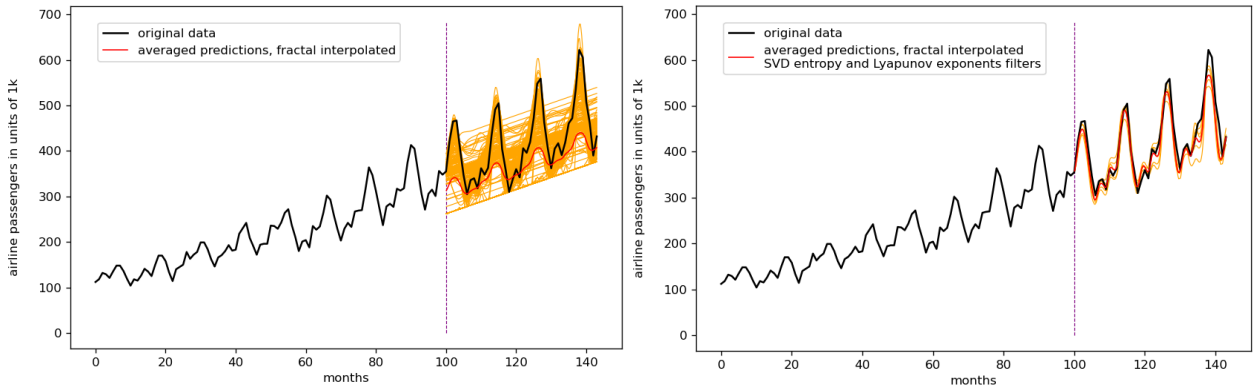


Figure 6. Plots for both the unfiltered ensemble predictions (left side) and the filtered ensemble prediction using a consequent application of, first, an SVD entropy filter, and second, a filter based on Lyapunov exponents to improve the prediction, 6 interpolation points. The orange lines are all the predictions constituting the ensemble, the red lines are the averaged predictions.

10. Baseline Predictions

For a baseline comparison, we use three types of recurrent neural networks. First, an LSTM neural network, second, a gated recurrent unit neural network (GRU) and, third, a simple recurrent neural network (RNN). All three employed types of neurons are reasonable tools for predicting time series data; the interested reader is referred to [6] for GRU and to [7] for simple RNN architectures for predicting time series data. We used a neural network with 1 hidden layer and 30 neurons, i.e., the RNN-neurons: LSTM, GRU, RNN. We used 20 input nodes consisting of consecutive time steps. The neural network was trained with a total of 50 epochs for each dataset. A batch size of 2 was used and `verbose` was set to 2 as well.

For the activation function (and the recurrent activation function) of the `LSTM`, the `GRU` and the `SimpleRNN` layer `hard_sigmoid` was used and `relu` for the `Dense` layer.

Overall, no regularizers, constraints or Drop out criteria have been used for the `LSTM` and `Dense` layers.

For the initialization, we used `glorot_uniform` for the LSTM layer, `orthogonal` as the recurrent initializer and `glorot_uniform` for the Dense layer.

For the LSTM layer, we also used `use_bias=True`, with `bias_initializer="zeros"` and no constraint or regularizer.

The optimizer was set to `rmsprop` and, for the loss, we used `mean_squared_error`. The output layer always returned only one result, i.e., the next time step.

These baseline predictions provide a reasonable guess for the accuracy of a LSTM, GRU or RNN prediction of the time series data under study.

All plots for the baseline predictions can be found in Appendix D, and here, we only give the accuracies for the test fit, the train fit and the single step-by-step prediction. These accuracies are shown in Tables 2–4. The accuracies such as the one for the ensemble predictions were calculated for linear-detrended and normalized (in the interval $[0, 1]$) data.

Table 2. Baseline RMSE for all datasets, LSTM.

Dataset	Train Error	Test Error	Single Step Error
Monthly international airline passengers	0.04987	0.08960	0.11902
Monthly car sales in Quebec	0.09735	0.11494	0.12461
Monthly mean air temperature in Nottingham Castle	0.06874	0.06193	0.05931
Perrin Freres monthly champagne sales	0.07971	0.07008	0.08556
CFE specialty monthly writing paper sales	0.07084	0.22353	0.21495

Table 3. Baseline RMSE for all datasets, GRU.

Dataset	Train Error	Test Error	Single Step Error
Monthly international airline passengers	0.04534	0.07946	0.10356
Monthly car sales in Quebec	0.09930	0.11275	0.11607
Monthly mean air temperature in Nottingham Castle	0.07048	0.06572	0.06852
Perrin Freres monthly champagne sales	0.06704	0.05916	0.07136
CFE specialty monthly writing paper sales	0.09083	0.22973	0.23296

Table 4. Baseline RMSE for all datasets, RNN.

Dataset	Train Error	Test Error	Single Step Error
Monthly international airline passengers	0.05606	0.08672	0.10566
Monthly car sales in Quebec	0.10161	0.12748	0.12075
Monthly mean air temperature in Nottingham Castle	0.07467	0.07008	0.06588
Perrin Freres monthly champagne sales	0.08581	0.07362	0.07812
CFE specialty monthly writing paper sales	0.07195	0.22121	0.21316

11. Results and Discussion

We linear- and fractal-interpolated five different time series data. Afterward, we did a random ensemble prediction for each, consisting of 500 different predictions for each interpolation technique (and non-interpolated time series data). The results of these random ensembles can be found in Appendix E in Tables A5 and A6. We further filtered these predictions using complexity filters (see Section 9) to finally reduce the number of ensemble predictions from 500 to 5, i.e., to 1%. The best five results for all time series data and each interpolation technique, regarding the RMSE and the corresponding error (see Section 8) are shown in Table 5 for the monthly international airline passengers dataset. Tables A1–A4, which feature the results for all other datasets, can be found in Appendix B. The corresponding plots for the three best predictions of each time series data can be found in Appendix C. We highlighted the overall best three results as bold entries.

The results show that the interpolated approaches always outperformed the non-interpolated ones when it comes to the lowest RMSEs. Further, the ensemble predictions could significantly be improved using a combination of interpolation techniques and complexity filters.

11.1. Interpolation Techniques

Regarding the different interpolation techniques of the overall three best results for all time series data, i.e., a total of 15 predictions, we find 9 fractal-interpolated predictions and 6 linear-interpolated predictions. Though the linear-interpolated results outperformed the fractal-interpolated ones in some cases, we conclude that fractal interpolation is a better way to improve LSTM neural network time series predictions. The reason for this is:

Taking into account the results shown in Figure 7 and Table 5, though the RMSE of the linear-interpolated result is lower (best result, lowest RMSE) than that of the second and third best ones (the fractal-interpolated results), the corresponding error of the RMSE is higher. Taking a closer look at the different ensemble predictions of Figure 7, we can see that the quality of the single predictions for the linear interpolated case is lower in terms of how close the actual curve data are to the different ensemble predictions. Therefore, the authors guess that this advantage of the linear-interpolated results vanishes when the statistic, i.e., the number of different ensemble predictions, increases. This behavior can be found for the monthly international airline passenger dataset, the monthly car sales in Quebec dataset, and the CFE specialty monthly writing paper sales dataset.

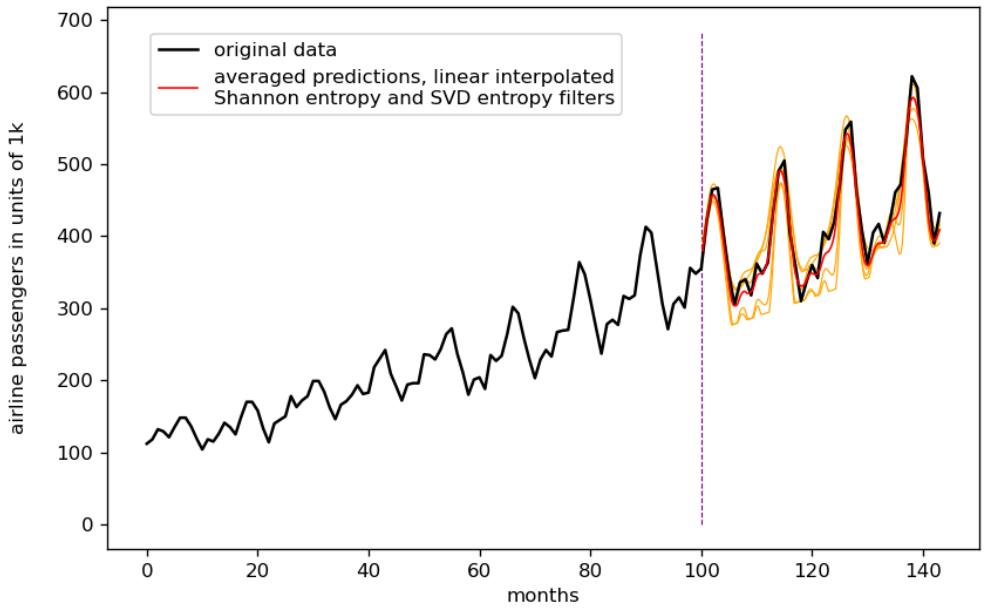


Figure 7. Best result monthly airline passengers dataset. The orange lines show the remaining ensemble predictions after filtering, the red line is the averaged ensemble prediction. Linear-interpolated, three interpolation points, Shannon entropy and SVD entropy filter, error: 0.03542 ± 0.00625.

Table 5. Error table for the monthly airline passengers dataset. The bold results are the three best ones for this dataset.

Interpolation Technique	# of Interpolation Points	Filter	Error
non-interpolated	-	fisher svd	0.04122 ± 0.00349
non-interpolated	-	svd	0.04122 ± 0.00349
non-interpolated	-	svd shannon	0.04122 ± 0.00349
non-interpolated	-	fisher	0.04166 ± 0.00271
non-interpolated	-	fisher shannon	0.04166 ± 0.00271
fractal-interpolated	**1**	**fisher hurst**	**0.03597 ± 0.00429**
fractal-interpolated	**1**	**svd hurst**	**0.03597 ± 0.00429**
fractal-interpolated	5	hurst fisher	0.03980 ± 0.00465
fractal-interpolated	5	hurst svd	0.03980 ± 0.00465
fractal-interpolated	5	shannon	0.04050 ± 0.00633
linear-interpolated	**3**	**shannon svd**	**0.03542 ± 0.00625**
linear-interpolated	3	shannon fisher	0.03804 ± 0.00672
linear-interpolated	5	fisher	0.04002 ± 0.00357
linear-interpolated	5	fisher shannon	0.04002 ± 0.00357
linear-interpolated	5	svd fisher	0.04002 ± 0.00357

11.2. Complexity Filters

Of these 75 best results for all interpolation techniques and different data, only 13 are single filtered predictions. A significant 62 are double-filtered predictions (i.e., two different complexity filters were applied). Not a single unfiltered prediction made it into the top 75 results. We, therefore, suggest always using two different complexity filters for filtering ensembles.

When it comes to the specific filters used, we cannot find a pattern within the 15 best results, as only the combinations SVD entropy & Hurst exponent and Lyapunov exponents & Hurst exponent occur more than once, i.e., each occurred only two times. Examining the 75 best results, though, we get a different picture. Here, we find 7 occurrences of the combination Shannon's entropy & Fisher's information followed by 6 occurrences of Shannon's entropy & SVD entropy. Further, taking into account that SVD entropy and Fisher's information behave similarly (as both are based on SVD, see Section 6), we find that 57 of the best 75 results contain at least one SVD-based, i.e., Single-Value-Decomposition-based, complexity measure. Therefore, we suggest using an SVD-based complexity measure in combination with the Hurst exponent Shannon's entropy. The authors' recommendation is that the best combination is SVD entropy & Hurst exponent.

11.3. Remarks and Summary

Summing up the results of this research, we draw the following conclusions:

- Random ensemble predictions can significantly be improved using fractal and linear interpolation techniques. The authors recommend using a fractal interpolation approach as the shown results feature a more stable behavior than those for the linear interpolation;
- Random ensemble predictions can significantly be improved using complexity filters to reduce the number of predictions in an ensemble. Taking into account the unfiltered and non-interpolated results shown in Tables A5 and A6 and comparing them to the best results, shown in Tables 5 and A1–A4, we see that the RMSE was reduced by a factor of ≈ 10 on average;
- The best results of the random ensemble, i.e., the single step-by-step predictions always outperformed the baseline predictions, Tables 2–4, and Appendix D. Here, we note that the given baseline predictions are probably not the best results that can be achieved with an optimized LSTM neural network but are still reasonable results and serve as baseline to show the quality of the ensemble predictions;
- Though the unfiltered results (Tables A5 and A6) suggest a trend and a minimum for the errors depending on the number of interpolation points, this trend vanishes when applying complexity filters. Therefore, we could not find a trend for the number of interpolation points for any interpolation technique and any complexity filters.

Though this research used univariate time series data for analysis, our approach can be extended to arbitrary dimensional multivariate time series data. We expect multivariate prediction approaches to benefit from this research greatly. For multivariate time series, different features may have different complexity properties. Thus, one may employ tools such as effective transfer entropy, [49], which is a complexity measure specifically dealing with multivariate time series data, or other complexity measures fit to deal with multivariate problems. Further criteria for the best fit based on correlations present in multivariate data may be found regarding the fractal interpolation for multivariate time series data.

The limitations of the presented approach hide in the parameter range of the neural network implementation. Though we can set arbitrary ranges to parameterize the neural network, computation costs can be reduced significantly if a good range for a specific dataset is known or can be guessed.

Further research of the presented framework may include switching the LSTM layers to feed-forward neural network layers or simple recurrent neural network (RNN, i.e., non-LSTM layers) layers. Here, one can adopt the ideas of time-delayed recurrent neural networks, [50], or time-delayed feed-forward neural networks, [51]. For both approaches, one can choose the input of the neural network to match the embedding of the time series, i.e., use the estimated time-delay and embedding dimension as done for a phase space reconstruction of a univariate time series data (see Appendix F), as done in [52].

12. Conclusions

This research aimed to improve neural network time series predictions using complexity measures and interpolation techniques. We presented a new approach not existing in the recent literature and tested it on five different univariate time series.

First, we interpolated the time series data under study using fractal and linear interpolation. Second, we generated randomly parameterized LSTM (one may understand the randomly parameterized LSTM neural network as a sort of a brute-force neural network approach). Neural networks and a step-by-step approach predicted the data under study, i.e., each dataset, and consequently, a non-interpolated dataset, a linear-interpolated dataset, and a fractal-interpolated dataset. Lastly, we filtered these random ensemble predictions based on their complexity, i.e., we kept only the forecasts with complexity close to the original complexity of the data. By applying the filters to the randomly parameterized LSTM ensemble, we reduced the error of the randomly parameterized ensemble by a factor of 10. The best filtered ensemble predictions consistently outperformed a single LSTM prediction, which we use as a reasonable baseline.

Filtering ensembles based on their complexities has not been done before and should be considered for future ensemble predictions to reduce the costs of optimizing neural networks.

In terms of interpolation techniques, we found that fractal interpolation works best under the given circumstances. For the complexity filters, we identified a combination of a Single-Value-Decomposition-based, e.g., SVD entropy or Fisher's information, and another complexity measure, e.g., the Hurst exponent, to perform best.

We conclude that interpolation techniques generating new data with a complexity close to that of the original data are best suited for improving the quality of a forecast. We expect the presented methods to be further exploited when predicting complex real-life time series data, such as environmental, agricultural, or financial data. This is because researchers are often confronted with a meager amount of data, and the given time series properties/complexity may change over time. Thus, new sets of hyper-parameters may have to be found. Using a random neural network ensemble and then filtering the predictions with respect to the complexity of older data circumvents this problem.

Author Contributions: Conceptualization, S.R.; Data curation, S.R.; Funding acquisition, T.N.; Investigation, S.R.; Methodology, S.R.; Project administration, T.N.; Resources, T.N.; Software, S.R.; Supervision, T.N.; Validation, S.R.; Visualization, S.R.; Writing—original draft, S.R.; Writing—review and editing, S.R. and T.N. All authors have read and agreed to the published version of the manuscript.

Funding: Open Access Funding by TU Wien; The state of Lower Austria: Forum Morgen-01112018; FFG Project AI4Cropr, No. 877158.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All used data sets are part of the *time series data library*, which is cited and linked in the discussion of the data sets.

Acknowledgments: The authors acknowledge the funding of the project "DiLaAg —Digitalization and Innovation Laboratory in Agricultural Sciences", by the private foundation "Forum Morgen", the Federal State of Lower Austria, by the FFG; Project AI4Cropr, No. 877158 and by TU Wien Bibliothek for financial support through its Open Access Funding Program.

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

Appendix A. Complexity Plots for All Datasets

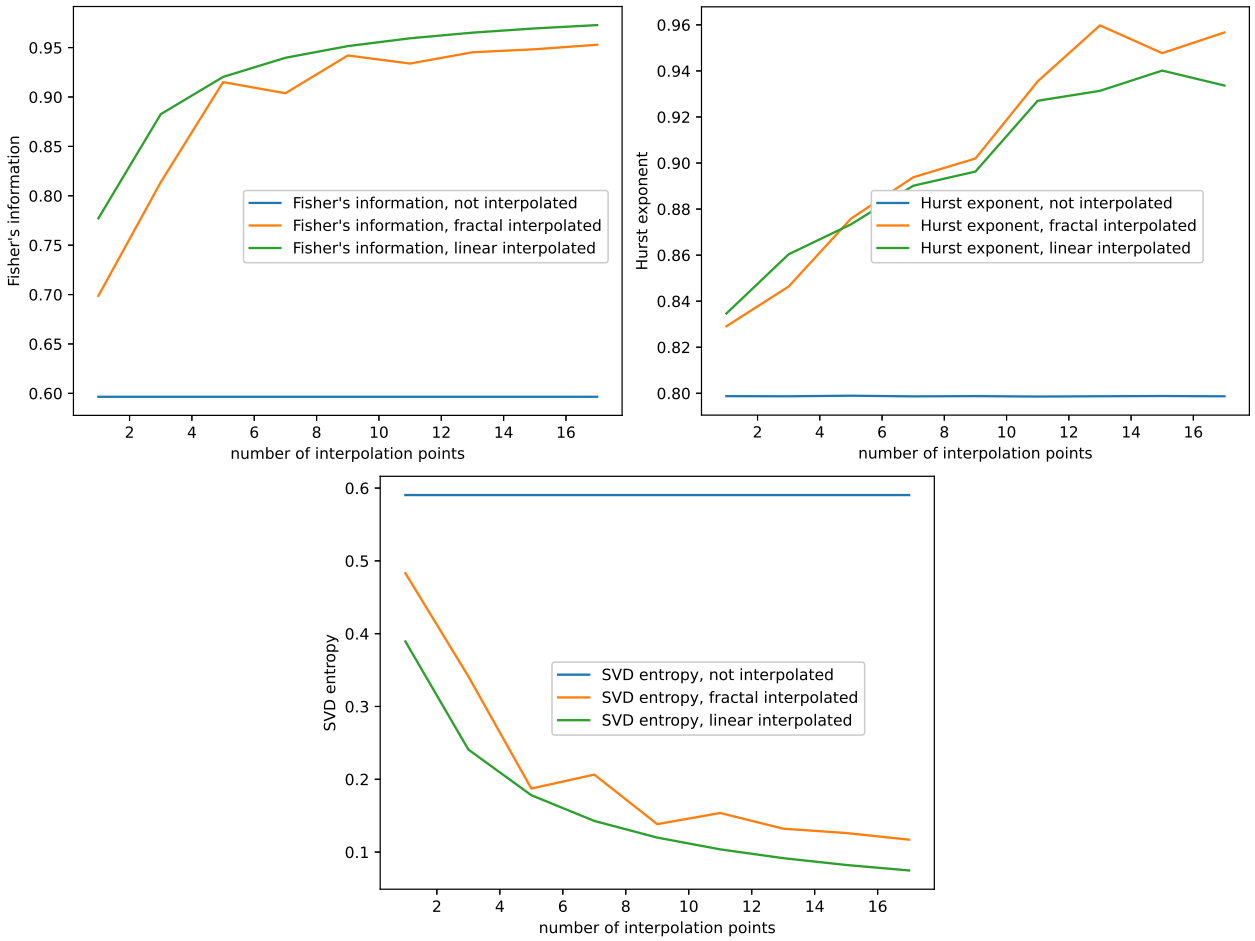


Figure A1. Plots for Fisher's information, the Hurst exponent and SVD entropy depending on the number of interpolation points for the non-interpolated, the fractal-interpolated and the linear-interpolated data, monthly mean temperature in Nottingham castle dataset.

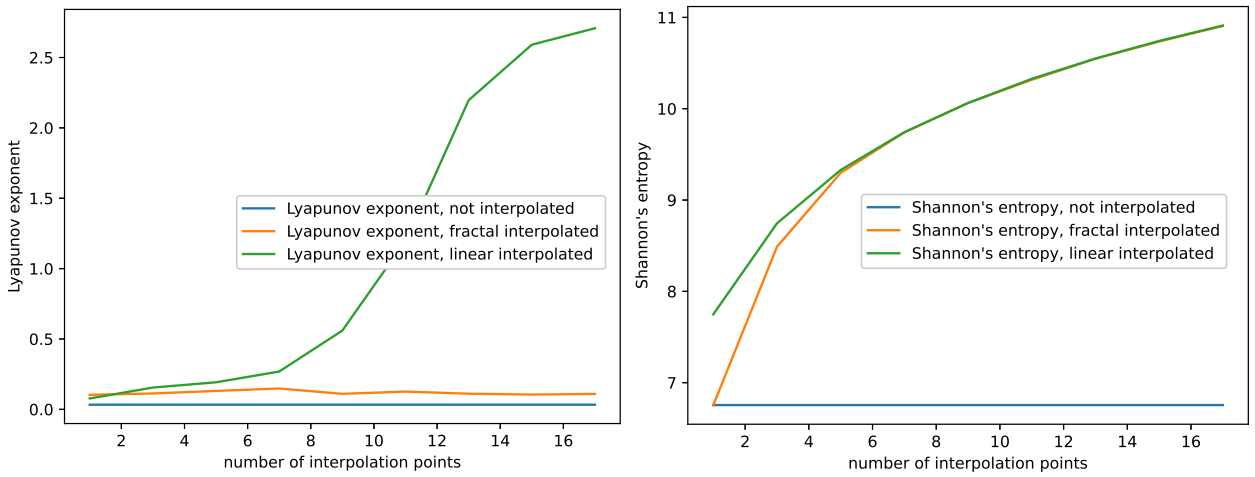


Figure A2. Plots for the Largest Lyapunov exponent and Shannon's entropy depending on the number of interpolation points for the non-interpolated, the fractal-interpolated and the linear-interpolated data, monthly car sales in Quebec dataset.

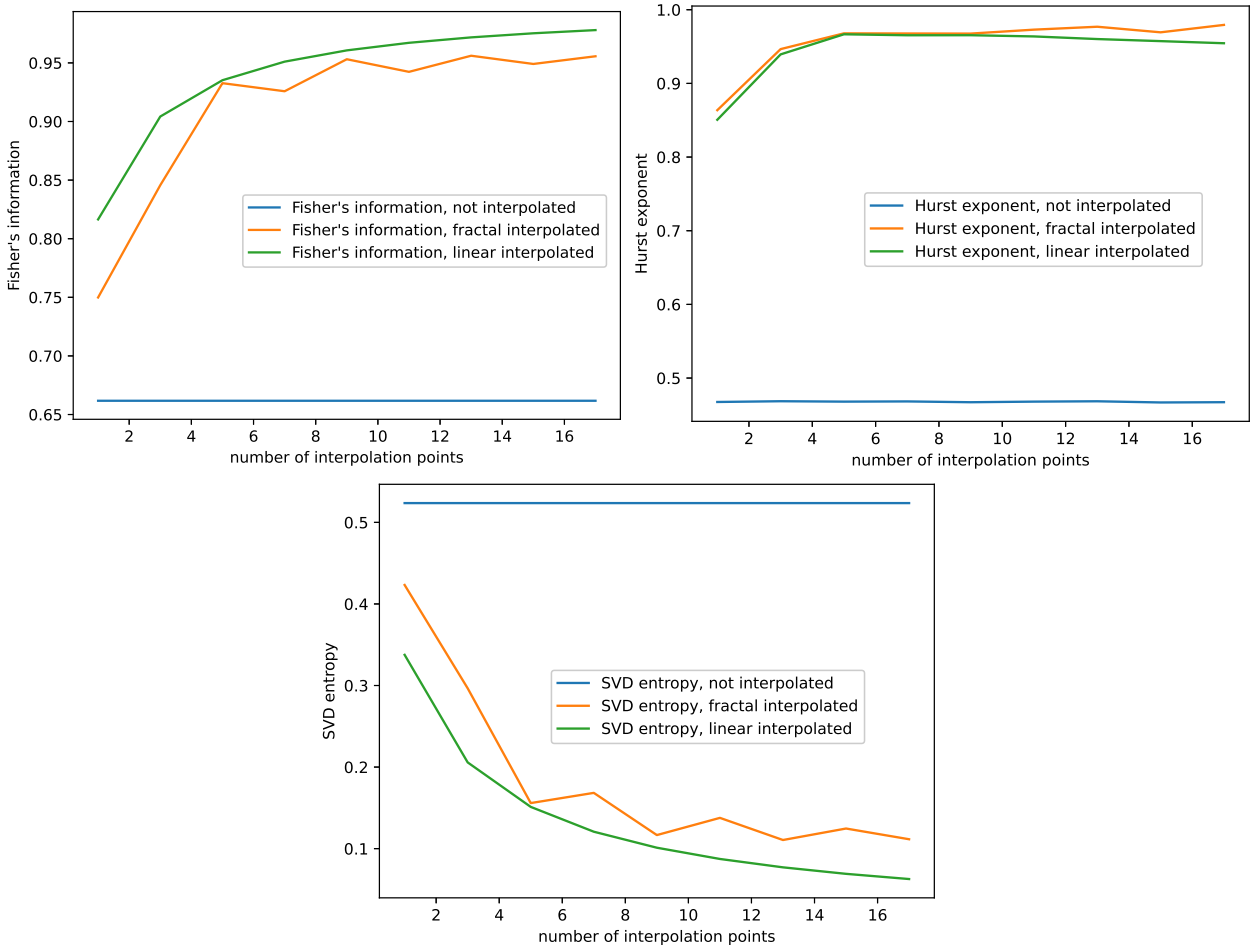


Figure A3. Plots for Fisher's information, the Hurst exponent and SVD entropy depending on the number of interpolation points for the non-interpolated, the fractal-interpolated and the linear-interpolated data, monthly mean temperature in Nottingham castle dataset.

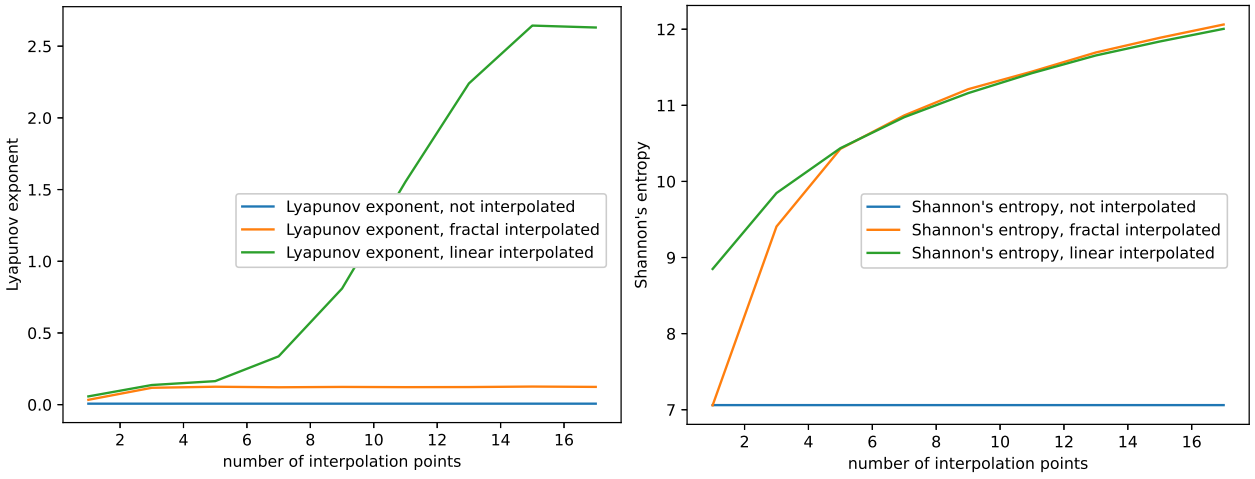


Figure A4. Plots for the Largest Lyapunov exponent and Shannon's entropy depending on the number of interpolation points for the non-interpolated, the fractal-interpolated and the linear-interpolated data, monthly mean temperature in Nottingham castle dataset.

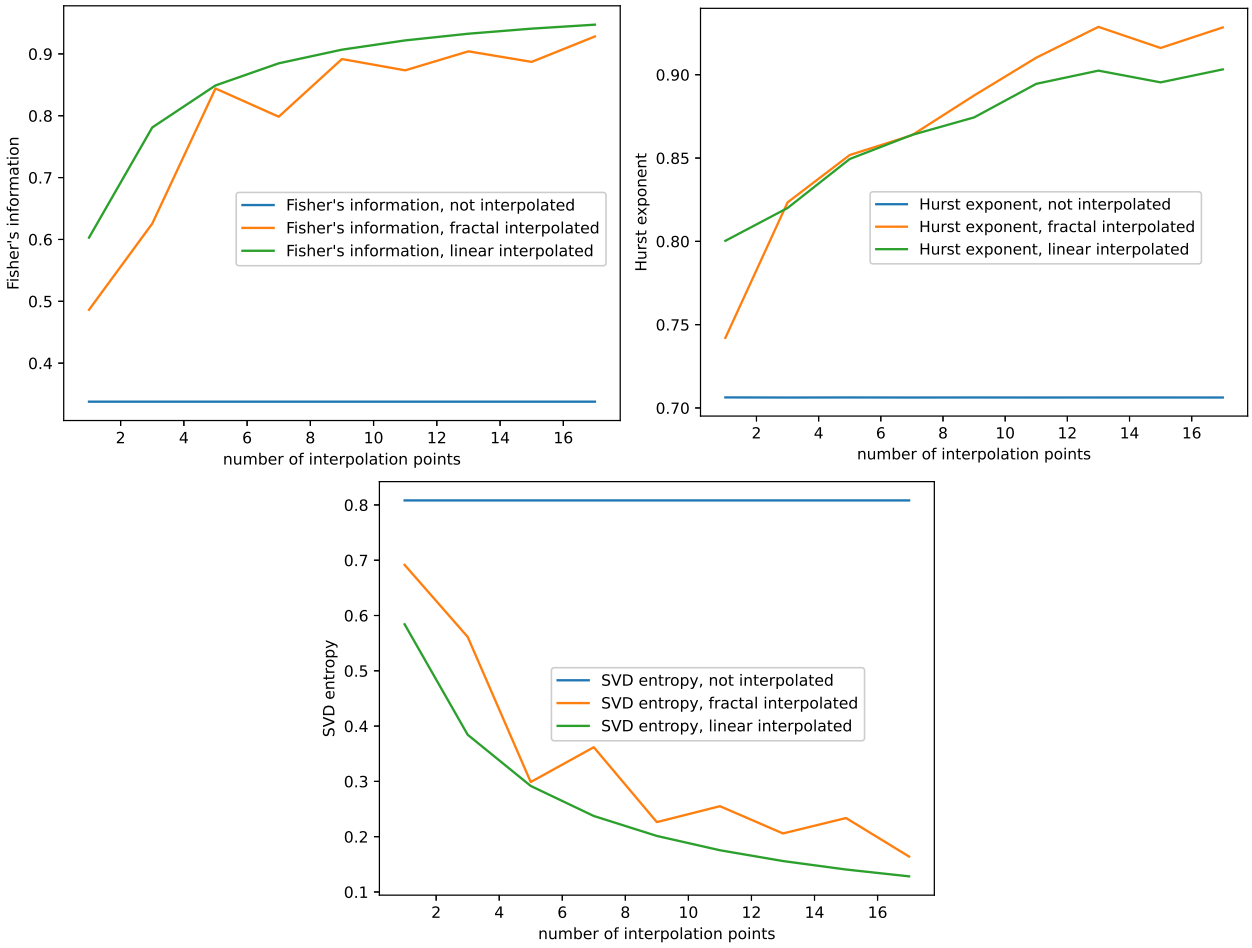


Figure A5. Plots for Fisher's information, the Hurst exponent and SVD entropy depending on the number of interpolation points for the non-interpolated, the fractal-interpolated and the linear-interpolated data, Perrin Freres monthly champagne sales dataset.

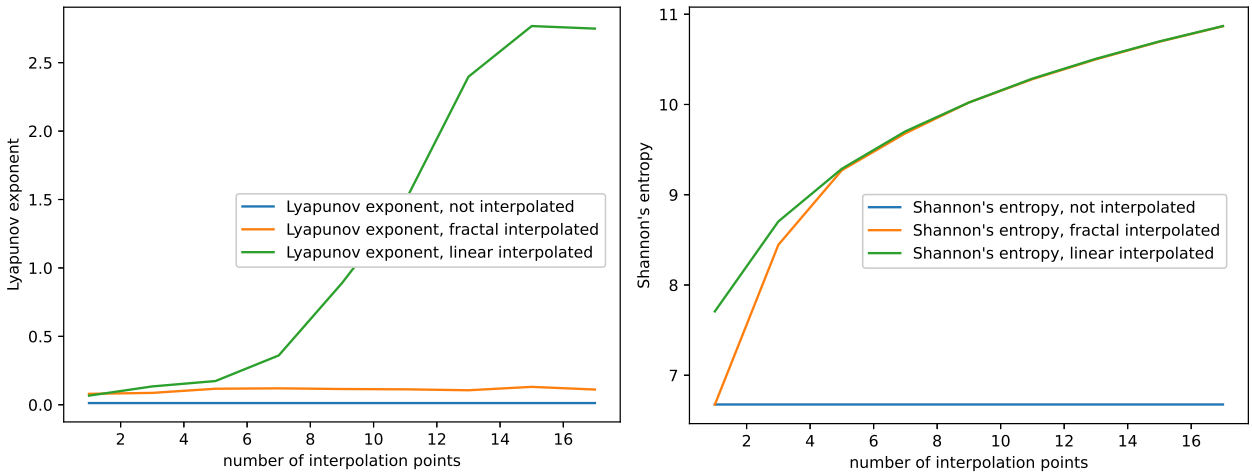


Figure A6. Plots for the Largest Lyapunov exponent and Shannon's entropy depending on the number of interpolation points for the non-interpolated, the fractal-interpolated and the linear-interpolated data, Perrin Freres monthly champagne sales dataset.

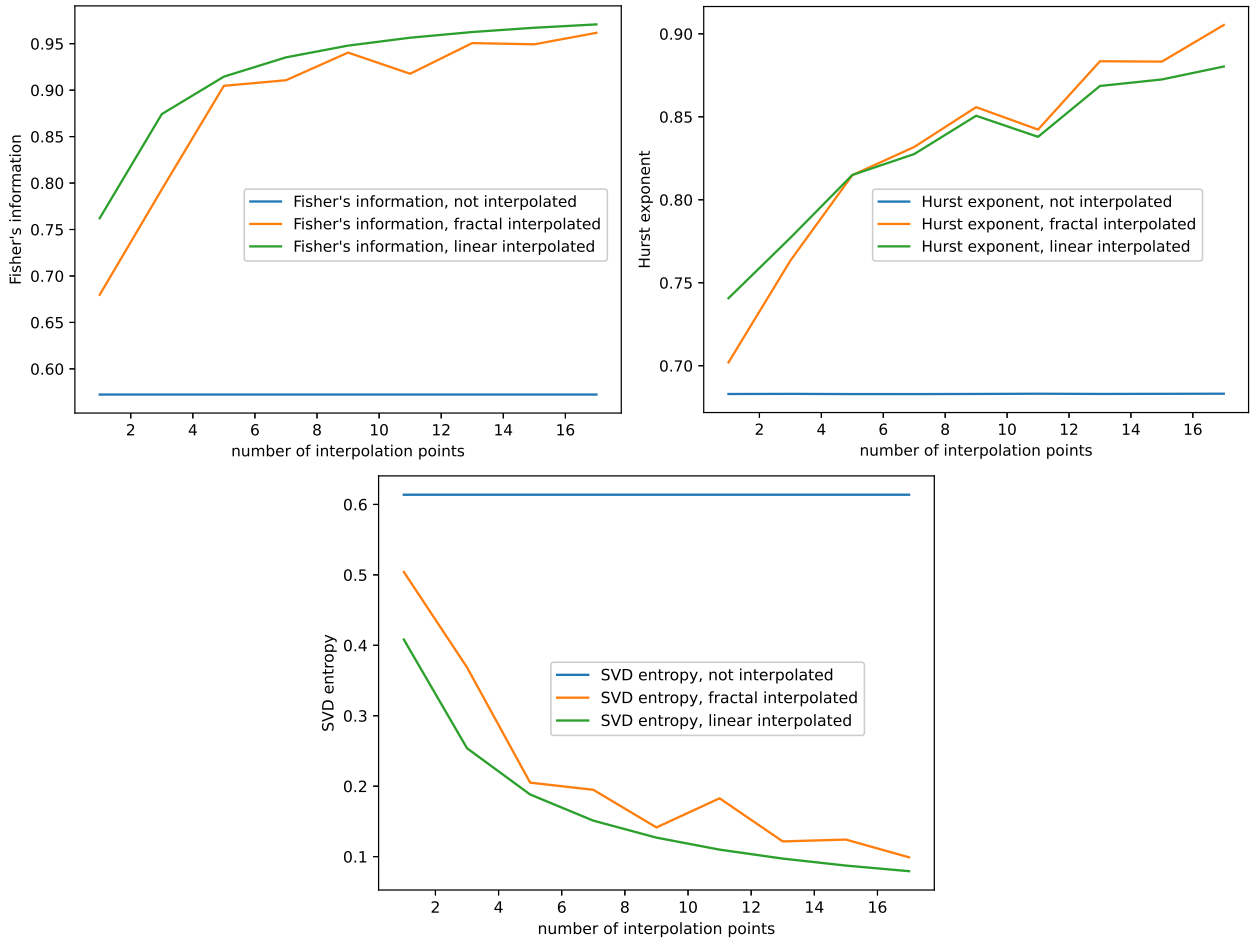


Figure A7. Plots for Fisher's information, the Hurst exponent and SVD entropy depending on the number of interpolation points for the non-interpolated, the fractal-interpolated and the linear-interpolated data, CFE specialty monthly writing paper sales dataset.

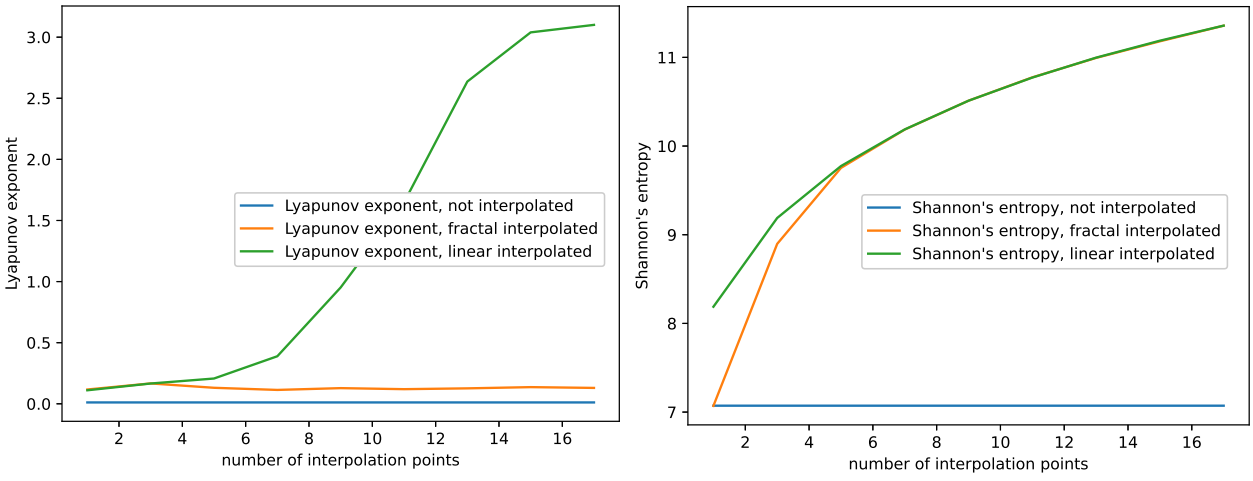


Figure A8. Plots for the Largest Lyapunov exponent and Shannon's entropy depending on the number of interpolation points for the non-interpolated, the fractal-interpolated and the linear-interpolated data, CFE specialty monthly writing paper sales dataset.

Appendix B. Error Tables

Table A1. Error table for the monthly car sales in Quebec dataset. The bold results are the three best ones for this dataset.

Interpolation Technique	# of Interpolation Points	Filter	Error
non-interpolated	-	shannon fisher	0.08814 ± 0.01496
non-interpolated	-	shannon svd	0.08814 ± 0.01496
non-interpolated	-	fisher hurst	0.09098 ± 0.01511
non-interpolated	-	svd hurst	0.09098 ± 0.01511
non-interpolated	-	fisher	0.09932 ± 0.00602
fractal-interpolated	**13**	**shannon fisher**	**0.08099 ± 0.00961**
fractal-interpolated	**13**	**shannon svd**	**0.08099 ± 0.00961**
fractal-interpolated	9	lyap hurst	0.08585 ± 0.01546
fractal-interpolated	17	fisher lyap	0.08659 ± 0.01869
fractal-interpolated	17	svd lyap	0.08659 ± 0.01869
linear-interpolated	**11**	**lyap hurst**	**0.07567 ± 0.03563**
linear-interpolated	7	fisher	0.08500 ± 0.02254
linear-interpolated	7	fisher svd	0.08500 ± 0.02254
linear-interpolated	7	fisher shannon	0.08500 ± 0.02254
linear-interpolated	5	svd	0.08500 ± 0.02254

Table A2. Error table for the monthly mean air temperature in Nottingham dataset. The bold results are the three best ones for this dataset.

Interpolation Technique	# of Interpolation Points	Filter	Error
non-interpolated	-	shannon fisher	0.05728 ± 0.00418
non-interpolated	-	fisher svd	0.05877 ± 0.01496
non-interpolated	-	svd	0.05877 ± 0.01496
non-interpolated	-	svd shannon	0.05877 ± 0.01496
non-interpolated	-	shannon fisher	0.05901 ± 0.00263
fractal-interpolated	**1**	**shannon svd**	**0.05724 ± 0.00495**
fractal-interpolated	7	shannon hurst	0.05873 ± 0.00684
fractal-interpolated	5	shannon lyap	0.05943 ± 0.01648
fractal-interpolated	7	fisher hurst	0.05946 ± 0.00519
fractal-interpolated	3	hurst	0.05998 ± 0.00544
linear-interpolated	**3**	**lyap hurst**	**0.05625 ± 0.00632**
linear-interpolated	**7**	**lyap fisher**	**0.05635 ± 0.00481**
linear-interpolated	3	hurst	0.05742 ± 0.00623
linear-interpolated	7	lyap svd	0.05786 ± 0.00511
linear-interpolated	3	svd lyap	0.05862 ± 0.00416

Table A3. Error table for the Perrin Freres monthly champagne sales dataset. The bold results are the three best ones for this dataset.

Interpolation Technique	# of Interpolation Points	Filter	Error
non-interpolated	-	shannon fisher	0.06383 ± 0.02706
non-interpolated	-	shannon svd	0.06383 ± 0.02706
non-interpolated	-	hurst fisher	0.07245 ± 0.01571
non-interpolated	-	hurst svd	0.07387 ± 0.01695
non-interpolated	-	hurst lyap	0.07403 ± 0.01740
fractal-interpolated	**13**	**fisher hurst**	**0.04968 ± 0.02155**
fractal-interpolated	**13**	**svd hurst**	**0.04968 ± 0.02155**
fractal-interpolated	**11**	**shannon hurst**	**0.05001 ± 0.01416**
fractal-interpolated	17	hurst lyap	0.05166 ± 0.01066
fractal-interpolated	13	hurst	0.05386 ± 0.0154
linear-interpolated	17	hurst	0.05449 ± 0.0280
linear-interpolated	17	hurst shannon	0.05449 ± 0.0280
linear-interpolated	9	fisher	0.05730 ± 0.03250
linear-interpolated	9	fisher shannon	0.05730 ± 0.03250
linear-interpolated	9	svd fisher	0.05730 ± 0.03250

Table A4. Error table for the CFE specialty monthly writing paper sales dataset. The bold results are the three best ones for this dataset.

Interpolation Technique	# of Interpolation Points	Filter	Error
non-interpolated	-	shannon fisher	0.18996 ± 0.00957
non-interpolated	-	shannon svd	0.19200 ± 0.01041
non-interpolated	-	hurst fisher	0.19314 ± 0.01057
non-interpolated	-	hurst svd	0.19314 ± 0.01057
non-interpolated	-	fisher hurst	0.19328 ± 0.01021
fractal-interpolated	**5**	**fisher lyap**	**0.17685 ± 0.00601**
fractal-interpolated	5	svd lyap	0.17685 ± 0.00601
fractal-interpolated	5	lyap hurst	0.18138 ± 0.00939
fractal-interpolated	1	hurst fisher	0.18332 ± 0.00751
fractal-interpolated	1	hurst svd	0.18332 ± 0.00751
linear-interpolated	**7**	**hurst fisher**	**0.17651 ± 0.01096**
linear-interpolated	**7**	**hurst svd**	**0.17651 ± 0.01096**
linear-interpolated	15	shannon lyap	0.18026 ± 0.00973
linear-interpolated	3	shannon hurst	0.18149 ± 0.01623
linear-interpolated	7	fisher lyap	0.18201 ± 0.00619

Appendix C. Prediction Plots

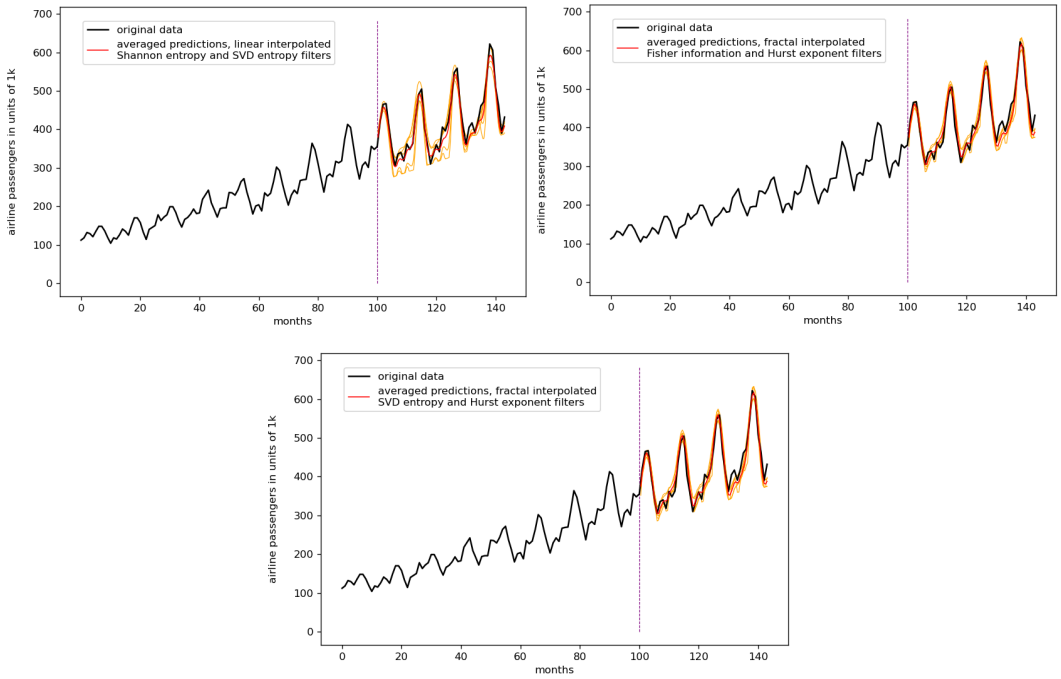


Figure A9. Best results monthly airline passengers dataset. The orange lines show the remaining ensemble predictions after filtering, the red line is the averaged ensemble prediction. Left to right: linear-interpolated, 3 interpolation points, Shannon entropy and SVD entropy filter, error: 0.03542 ± 0.00625; fractal-interpolated, 1 interpolation point, Fisher's information and Hurst exponent filter, error: 0.03597 ± 0.00429; fractal-interpolated, 1 interpolation point, SVD entropy and Hurst exponent filter, error: 0.03597 ± 0.00429.

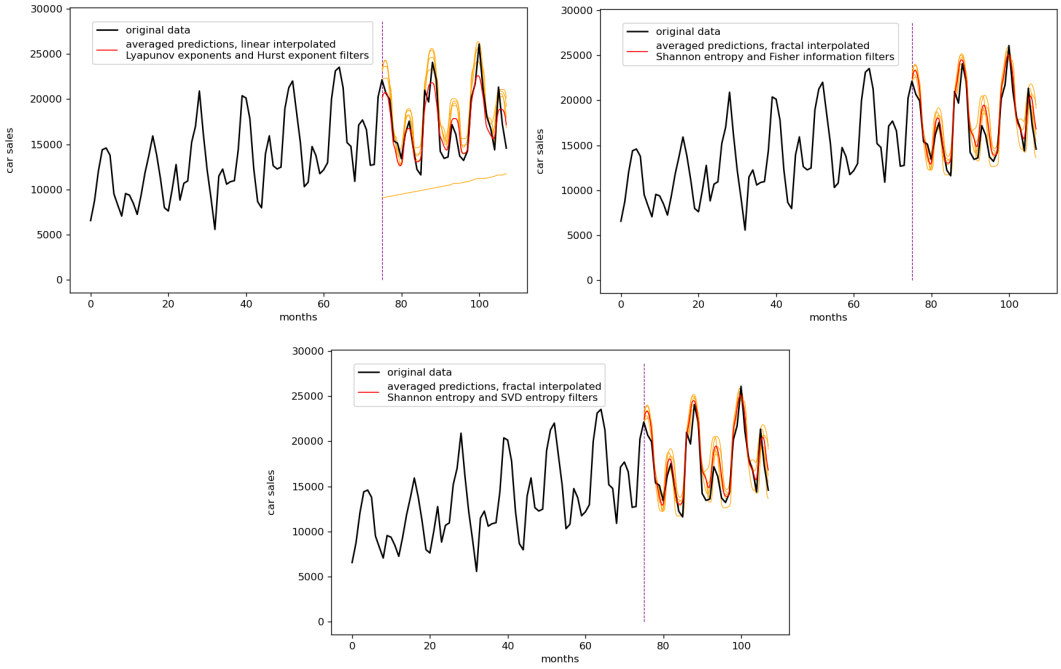


Figure A10. Best results monthly car sales in Quebec dataset. The orange lines show the remaining ensemble predictions after filtering, the red line is the averaged ensemble prediction. Left to right: linear-interpolated, 11 interpolation points, Lyapunov exponents and Hurst exponent filter, error: 0.097567 ± 0.03563; fractal-interpolated, 13 interpolation points, Shannon's entropy and Fisher's information filter, error: 0.08099 ± 0.00961; fractal-interpolated, 13 interpolation points, Shannon's entropy and SVD entropy filter, error: 0.08099 ± 0.00961.

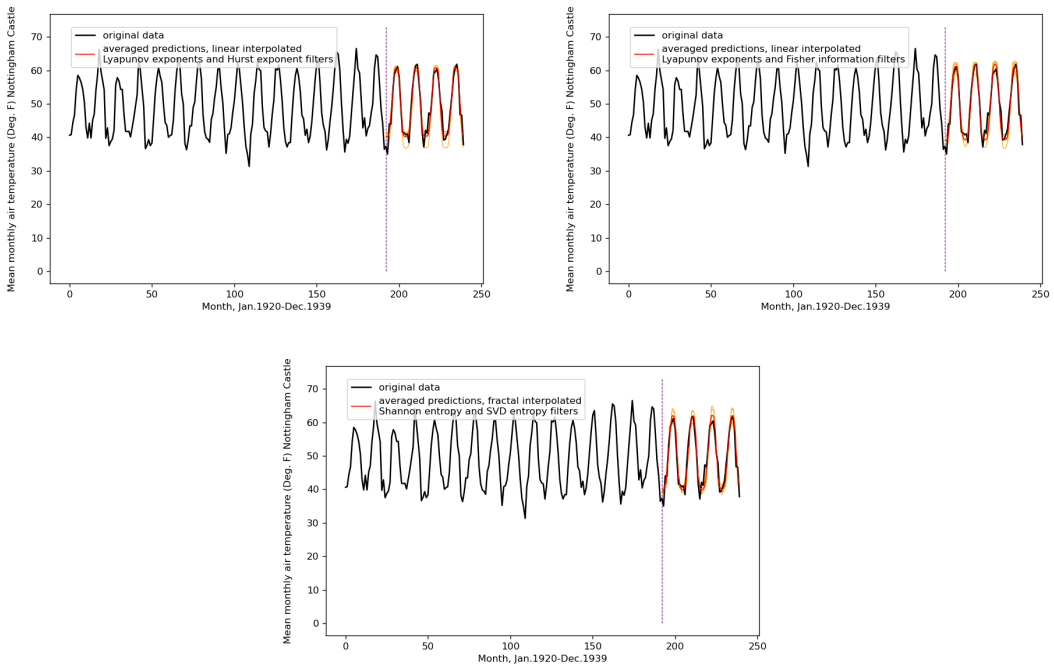


Figure A11. Best results monthly mean temperatures in Nottingham castle dataset. The orange lines show the remaining ensemble predictions after filtering, the red line is the averaged ensemble prediction. Left to right: linear-interpolated, 3 interpolation points, Lyapunov exponents and Hurst exponent filter, error: 0.05625 ± 0.00632; linear-interpolated, 7 interpolation points, Lyapunov exponent and Fisher's information filter, error: 0.05635 ± 0.00481; fractal-interpolated, 1 interpolation point, Shannon's entropy and SVD entropy filter, error:0.05724 ± 0.00495.

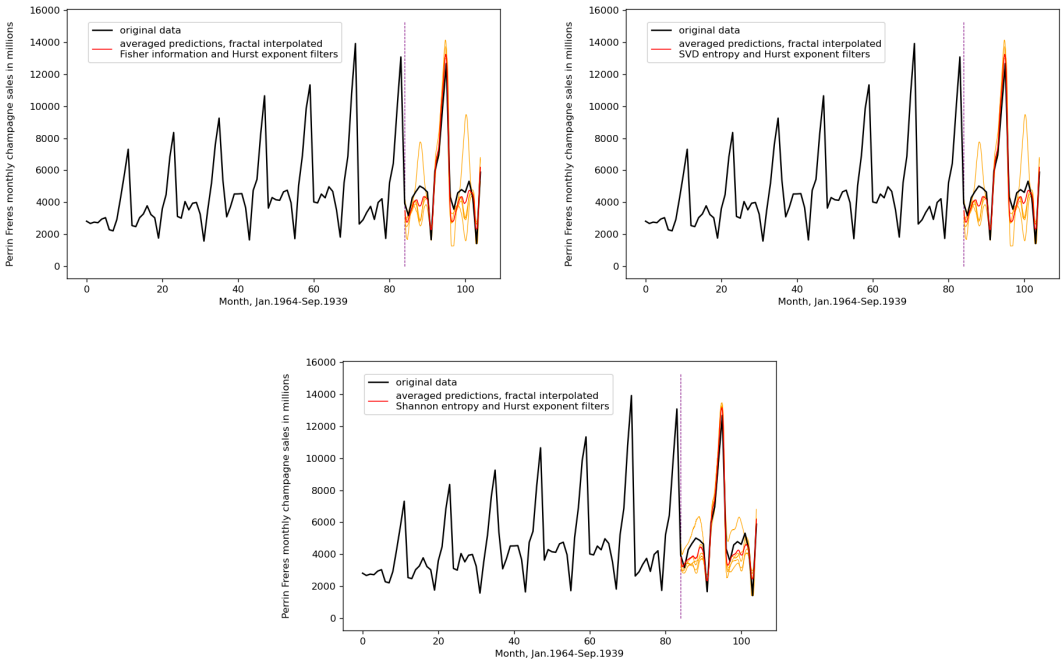


Figure A12. Best results for the Perrin Freres monthly champagne sales dataset. The orange lines show the remaining ensemble predictions after filtering, the red line is the averaged ensemble prediction. Left to right: fractal-interpolated, 13 interpolation points, Fisher's information and Hurst exponent filter, error: 0.04968 ± 0.02155; fractal-interpolated, 13 interpolation points, SVD entropy and Hurst exponent filter, error: 0.04968 ± 0.02155; fractal-interpolated, 11 interpolation point, Shannon's entropy and Hurst exponent filter, error: 0.05001 ± 0.01416.

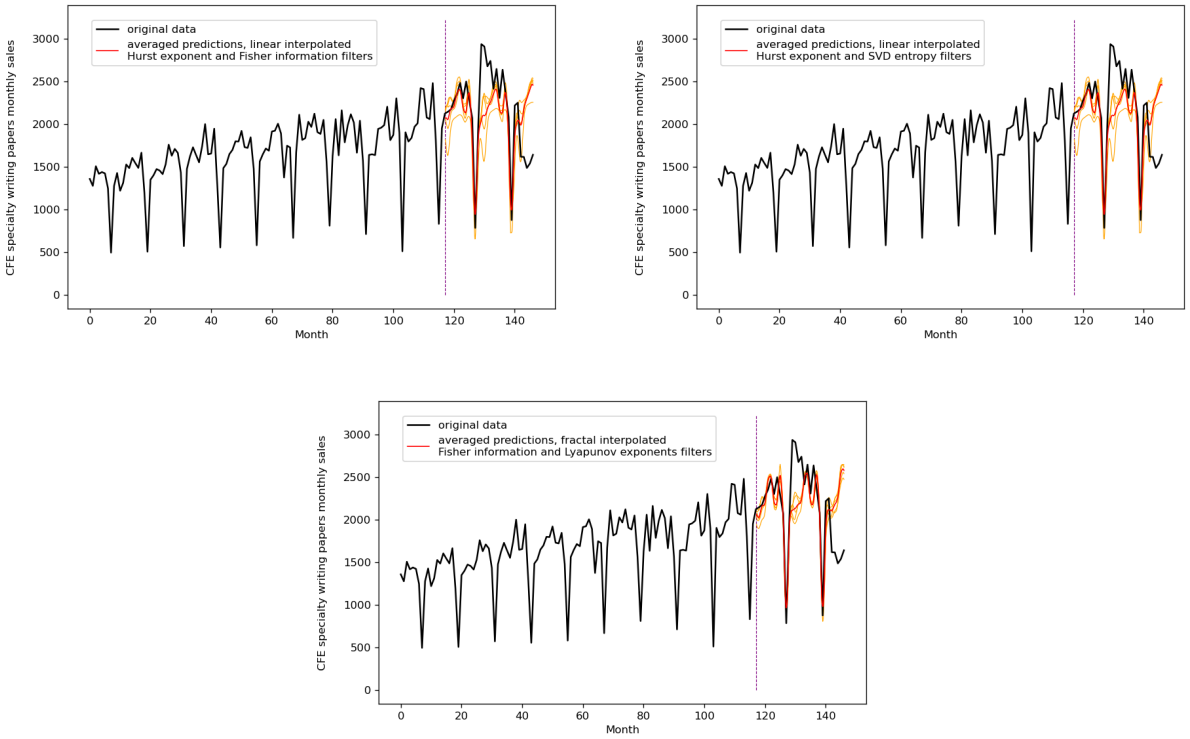


Figure A13. Best results CFE specialty monthly writing paper sales dataset. The orange lines show the remaining ensemble predictions after filtering, the red line is the averaged ensemble prediction. Left to right: linear-interpolated, 7 interpolation points, Hurst exponent and Fisher's information filter, error: 0.17651 ± 0.01096; fractal-interpolated, 7 interpolation points, Hurst exponent and SVD entropy filter, error: 0.17651 ± 0.01096; fractal-interpolated, 5 interpolation points, Fisher's information and Lyapunov exponents filter, error: 0.17685 ± 0.00601.

Appendix D. Baseline Predictions

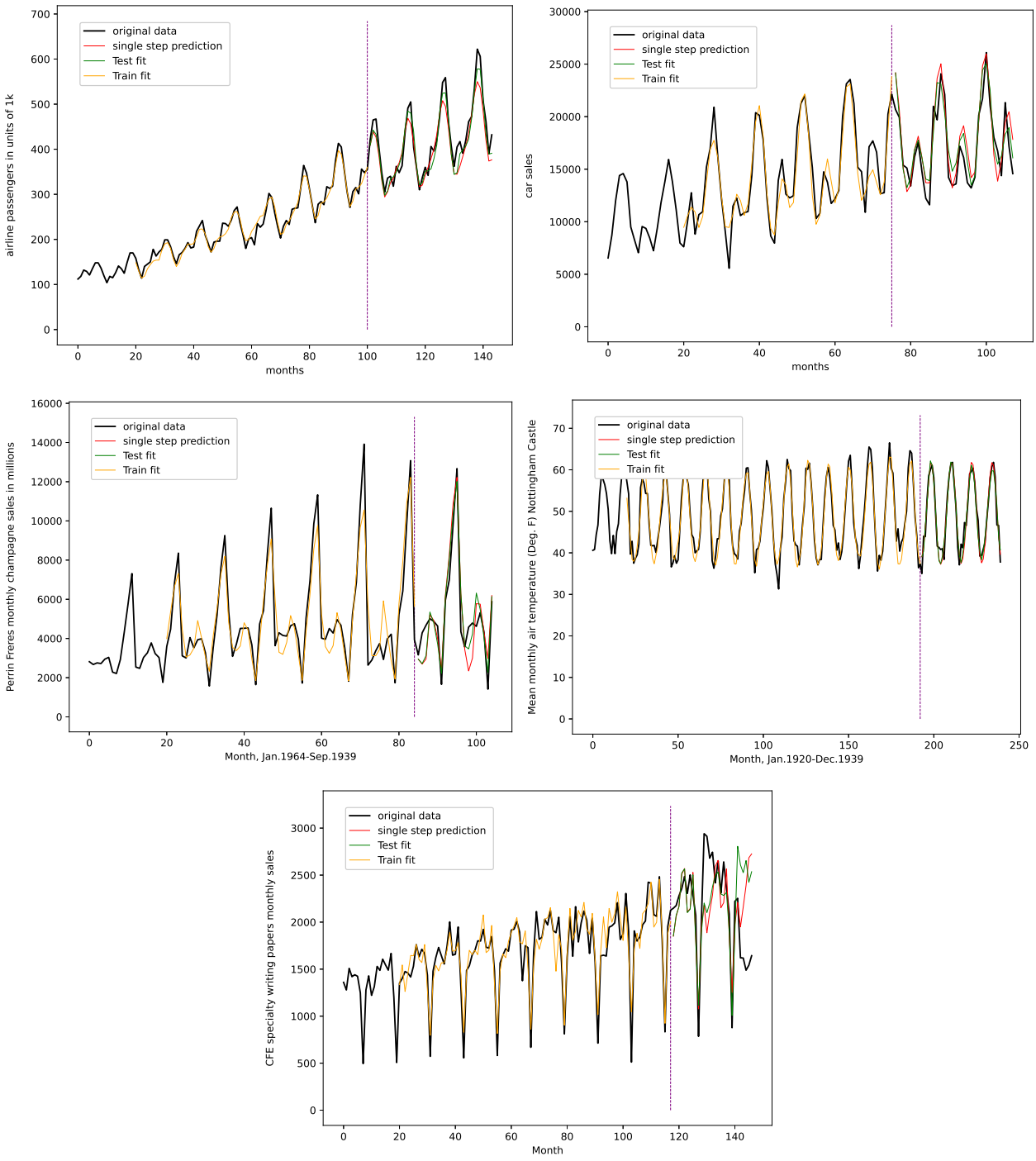


Figure A14. Baseline predictions for each dataset, LSTM.

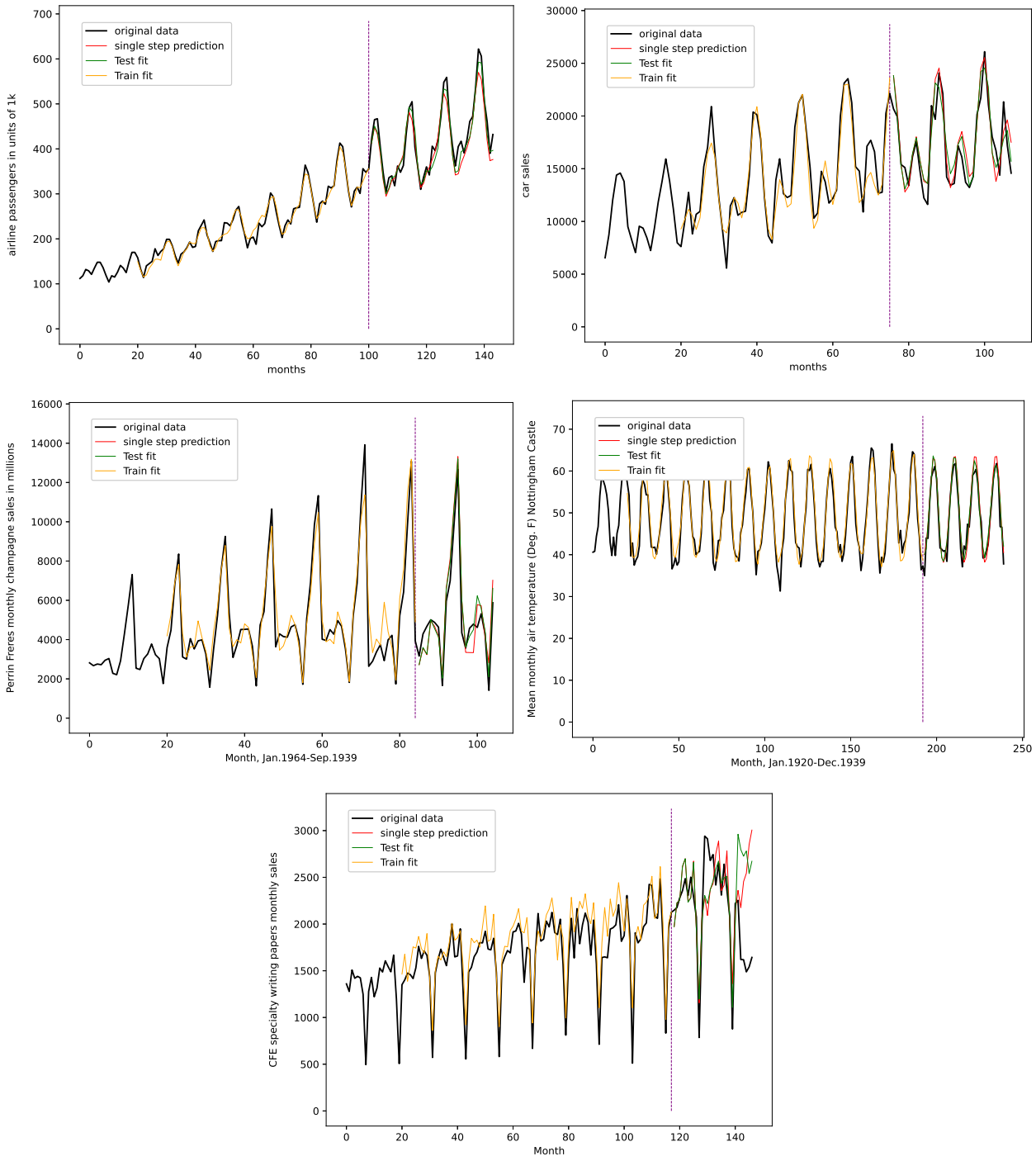


Figure A15. Baseline predictions for each dataset, GRU.

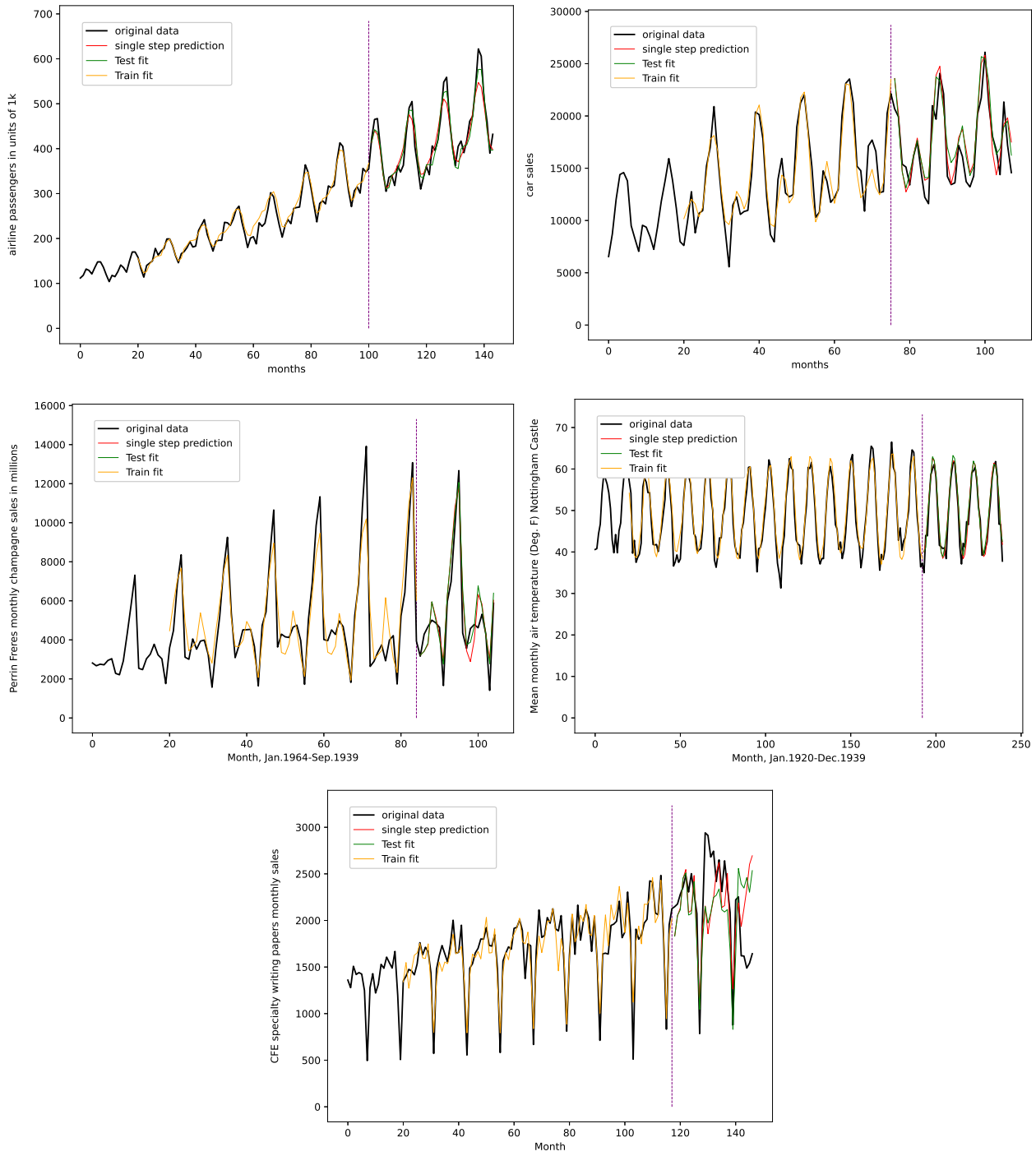


Figure A16. Baseline predictions for each dataset, RNN.

Appendix E. Unfiltered Ensemble Prediction Errors

Table A5. Unfiltered ensemble prediction errors for all datasets, linear-interpolated.

# of Interpolation Points	Monthly International Airline Passengers	Monthly Car Sales in Quebec	Monthly Mean Air Temperature in Nottingham Castle	Perrin Freres Monthly Champagne Sales	CFE Specialty Monthly Writing Paper Sales
0	0.16771 ± 0.01537	0.20779 ± 0.03961	0.28503 ± 0.04816	0.20641 ± 0.05741	0.38041 ± 0.05203
1	0.16076 ± 0.01917	0.20239 ± 0.04121	0.31272 ± 0.04877	0.19946 ± 0.06391	0.38070 ± 0.05338
3	0.16487 ± 0.01758	0.18964 ± 0.04448	0.30624 ± 0.05029	0.18770 ± 0.07113	0.38474 ± 0.05309
5	0.15347 ± 0.01988	0.18814 ± 0.04654	0.30236 ± 0.05090	0.18858 ± 0.07011	0.37807 ± 0.05443
7	0.15710 ± 0.02002	018252 ± 0.04657	0.30020 ± 0.05112	0.18439 ± 0.07311	0.37935 ± 0.05479
9	0.14610 ± 0.02088	0.17944 ± 0.04784	0.29381 ± 0.05185	0.17920 ± 0.07327	0.39043 ± 0.05280
11	0.15410 ± 0.02082	0.18092 ± 0.04787	0.30588 ± 0.05123	0.18487 ± 0.07284	0.36708 ± 0.05583
13	0.15361 ± 0.02014	0.17582 ± 0.04781	0.30105 ± 0.05151	0.18408 ± 0.07380	0.39382 ± 0.05228
15	0.15359 ± 0.02091	0.17476 ± 0.04773	0.31103 ± 0.05033	0.17973 ± 0.07522	0.39385 ± 0.05276
17	0.16245 ± 0.02004	0.17571 ± 0.04754	0.30171 ± 0.05125	0.18219 ± 0.07404	0.37625 ± 0.05515

Table A6. Unfiltered ensemble prediction errors for all datasets, linear-interpolated.

# of Interpolation Points	Monthly International Airline Passengers	Monthly Car Sales in Quebec	Monthly Mean Air Temperature in Nottingham Castle	Perrin Freres Monthly Champagne Sales	CFE Specialty Monthly Writing Paper Sales
0	0.16771 ± 0.01537	0.20779 ± 0.03961	0.28503 ± 0.04816	0.20641 ± 0.05741	0.38041 ± 0.05203
1	0.16294 ± 0.01917	0.20283 ± 0.04332	0.26516 ± 0.05052	0.21457 ± 0.06303	0.36967 ± 0.05309
3	0.15199 ± 0.01758	0.19681 ± 0.04584	0.29448 ± 0.05088	0.19397 ± 0.0691	0.37922 ± 0.05435
5	0.15088 ± 0.01988	0.17882 ± 0.04761	0.27367 ± 0.05107	0.19438 ± 0.07132	0.35778 ± 0.05520
7	0.14553 ± 0.02002	0.17105 ± 0.04771	0.28405 ± 0.05130	0.18642 ± 0.07327	0.37685 ± 0.05533
9	0.15033 ± 0.02088	0.18831 ± 0.04813	0.28135 ± 0.05186	0.20273 ± 0.07183	0.35956 ± 0.05501
11	0.15664 ± 0.02082	0.18738 ± 0.04832	0.28566 ± 0.05130	0.18151 ± 0.07370	0.38573 ± 0.05557
13	0.15459 ± 0.02014	0.17700 ± 0.04855	0.30069 ± 0.05168	0.19560 ± 0.07281	0.38573 ± 0.05504
15	0.15090 ± 0.02091	0.18368 ± 0.04825	0.30349 ± 0.05114	0.19760±0.07235	0.38506 ± 0.05515
17	0.15428 ± 0.02004	0.18468 ± 0.04903	0.28451 ± 0.05208	0.18838 ± 0.07347	0.36044 ± 0.05591

Appendix F. Phase Space Embeddings

For all time series data, we estimated a phase space embedding. A phase space embedding consists of two parts.

First, estimating the time lag/delay, i.e., the lag between two consecutive components in the embedding vector. For this research, we used the method based on the average mutual information from [41] to estimate the time delay.

Second, estimating the embedding dimension, here, we used the algorithm of false-nearest-neighbors, [53].

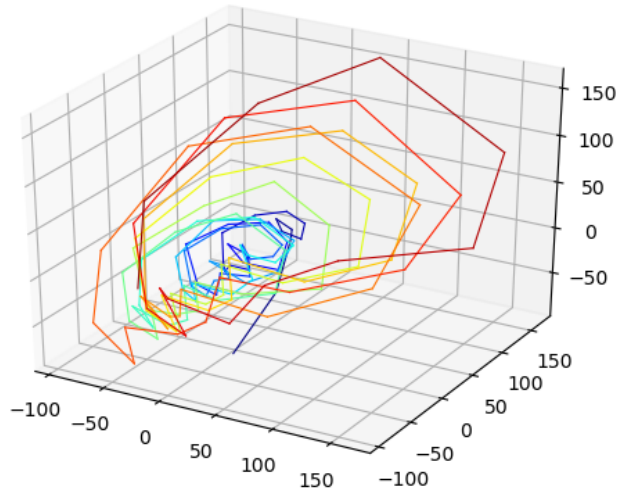
All time series data were detrended by subtracting a linear fit before applying the algorithms to estimate the phase space embedding. The employed algorithms yielded the following results:

- **Monthly international airline passengers:**
  Time delay, $\tau = 1$
  Embedding dimension, $d_E = 3$;
- **Monthly car sales in Quebec:**
  Time delay, $\tau = 1$
  Embedding dimension, $d_E = 6$;
- **Monthly mean temperature in Nottingham Castle:**
  Time delay, $\tau = 1$
  Embedding dimension, $d_E = 3$;
- **Perrin Freres monthly champagne sales:**
  Time delay, $\tau = 1$
  Embedding dimension, $d_E = 7$;
- **CFE specialty monthly writing paper sales:**
  Time delay, $\tau = 2$
  Embedding dimension, $d_E = 1$.

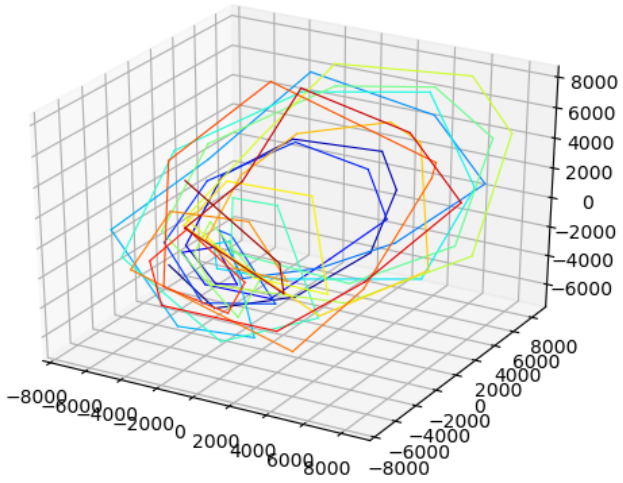
As these algorithms can only estimate a phase space embedding, we plotted all time series data in three-dimensional phase space by using each previously determined time delay τ and building three-dimensional coordinate vectors from the univariate signal. Further, the fact that two of the estimated embedding dimensions are three gave rise to assuming that a three-dimensional phase space embedding might be reasonable for all employed time series data. Thus, given a signal $[x_1, x_2, \ldots, x_n]$, we obtain the phase space vectors as:

$$\vec{y}(i) = [x_i, x_{i+\tau}, x_{i+2\tau}] \quad . \tag{A1}$$

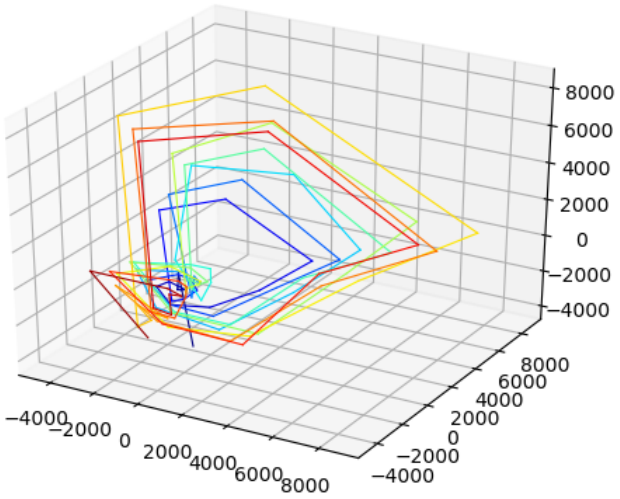
This yields the plots from Figure A17. Thus, given that the phase space reconstructions look reasonable in 3 dimensions, i.e., one can see some rough symmetry/antisymmetry/fractality, and that the false-nearest-neighbor algorithm can only give estimates on a suitable embedding dimension, we chose the embedding dimension to be $d_E = 3$ for Fisher's information and SVD entropy.



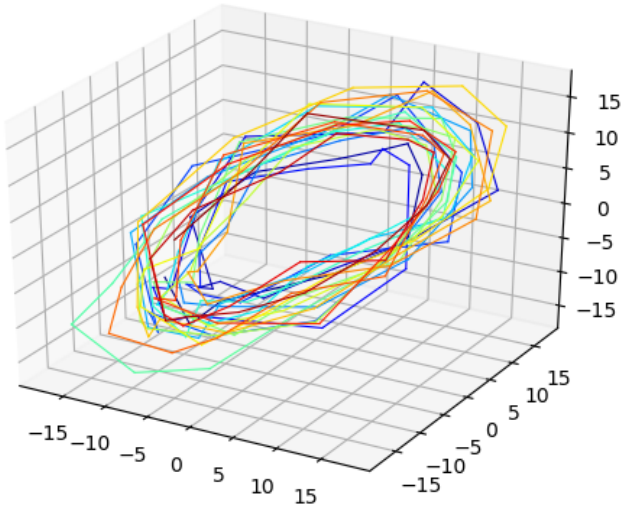
Monthly international airline passengers



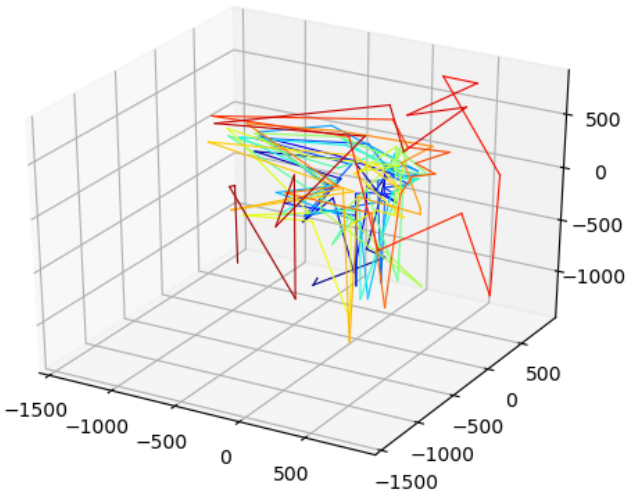
Monthly car sales in Quebec



Monthly mean air temperature in Nottingham Castle



Perrin Freres monthly champagne sale



CFE specialty monthly writing paper sales

Figure A17. Phase space reconstruction in $d_E = 3$ dimensions for all time series data.

References

1. Rasheed, K.; Qayyum, A.; Qadir, J.; Sivathamboo, S.; Kwan, P.; Kuhlmann, L.; O'Brien, T.; Razi, A. Machine Learning for Predicting Epileptic Seizures Using EEG Signals: A Review. *IEEE Rev. Biomed. Eng.* **2021**, *14*, 139–155. [CrossRef]
2. Wang, M.C.; Uhlenbeck, G.E. On the Theory of the Brownian Motion II. *Rev. Mod. Phys.* **1945**, *17*, 323–342. [CrossRef]
3. Karaca, Y.; Zhang, Y.D.; Muhammad, K. A Novel Framework of Rescaled Range Fractal Analysis and Entropy-Based Indicators: Forecasting Modelling for Stock Market Indices. *Expert Syst. Appl.* **2019**. [CrossRef]
4. Manousopoulos, P.; Drakopoulos, V.; Theoharis, T. Curve Fitting by Fractal Interpolation. In *Transactions on Computational Science I*; Gavrilova, M.L., Tan, C.J.K., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 85–103. [CrossRef]
5. Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
6. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1724–1734. [CrossRef]
7. Zhang, J.; Man, K.F. Time series prediction using RNN in multi-dimension embedding phase space. In Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218), San Diego, CA, USA, 14 October 1998; Volume 2, pp. 1868–1873.
8. Sapankevych, N.I.; Sankar, R. Time Series Prediction Using Support Vector Machines: A Survey. *IEEE Comput. Intell. Mag.* **2009**, *4*, 24–38. [CrossRef]
9. Trafalis, T.B.; Ince, H. Support vector machine for regression and applications to financial forecasting. In Proceedings of thee IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, Como, Italy, 27 July 2000; Volume 6, pp. 348–353. [CrossRef]
10. Cao, L.J.; Tay, F.E.H. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Trans. Neural Netw.* **2003**, *14*, 1506–1518. [CrossRef]
11. Chang, M.; Chen, B.; Lin, C. *EUNITE Network Competition: Electricity Load Forecasting*; Technical Report; National Taiwan University: Taipei, Taiwan, 2001.
12. Voyant, C.; Notton, G.; Kalogirou, S.; Nivet, M.L.; Paoli, C.; Motte, F.; Fouilloy, A. Machine learning methods for solar radiation forecasting: A review. *Renew. Energy* **2017**, *105*, 569–582. [CrossRef]
13. Qi, M.; Zhang, G.P. Trend Time—Series Modeling and Forecasting With Neural Networks. *IEEE Trans. Neural Netw.* **2008**, *19*, 808–816. [CrossRef] [PubMed]
14. Zhang, J.S.; Xiao, X.C. Predicting Chaotic Time Series Using Recurrent Neural Network. *Chin. Phys. Lett.* **2000**, *17*, 88–90. [CrossRef]
15. Connor, J.T.; Martin, R.D.; Atlas, L.E. Recurrent neural networks and robust time series prediction. *IEEE Trans. Neural Netw.* **1994**, *5*, 240–254. [CrossRef] [PubMed]
16. Sherstinsky, A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Phys. D Nonlinear Phenom.* **2020**, *404*, 132306. [CrossRef]
17. Zhang, G.; Berardi, V. Time series forecasting with neural network ensembles: An application for exchange rate prediction. *J. Oper. Res. Soc.* **2001**, *52*, 652–664. [CrossRef]
18. Chen, J.; Zeng, G.Q.; Zhou, W.; Du, W.; Lu, K.D. Wind speed forecasting using nonlinear-learning ensemble of deep learning time series prediction and extremal optimization. *Energy Convers. Manag.* **2018**, *165*, 681–695. [CrossRef]
19. Zhang, G.P. A neural network ensemble method with jittered training data for time series forecasting. *Inf. Sci.* **2007**, *177*, 5329–5346. [CrossRef]
20. Hurst, H.; Black, R.; Sinaika, Y. *Long-Term Storage in Reservoirs: An Experimental Study*; Constable: London, UK, 1965.
21. Cincotta, P.M.; Helmi, A.; Méndez, M.; Núñez, J.A.; Vucetich, H. Astronomical time-series analysis—II. A search for periodicity using the Shannon entropy. *Mon. Not. R. Astron. Soc.* **1999**, *302*, 582–586. [CrossRef]
22. Vörös, Z.; Jankovičová, D. Neural network prediction of geomagnetic activity: A method using local Hölder exponents. *Nonlinear Process. Geophys.* **2002**, *9*, 425–433. [CrossRef]
23. Castillo, O.; Melin, P. Hybrid Intelligent Systems for Time Series Prediction Using Neural Networks, Fuzzy Logic, and Fractal Theory. *IEEE Trans. Neural Netw.* **2002**, *13*, 1395–1408. [CrossRef]
24. Raubitzek, S.; Neubauer, T. A fractal interpolation approach to improve neural network predictions for difficult time series data. *Expert Syst. Appl.* **2021**, *169*, 114474. [CrossRef]
25. Hyndman, R.; Yang, Y. Time Series Data Library v0.1.0.. 2018. Available online: https://pkg.yangzhuoranyang.com/tsdl/ (accessed on 28 October 2021).
26. Mazel, D.; Hayes, M. Using iterated function systems to model discrete sequences. *IEEE Trans. Signal Process.* **1992**, *40*, 1724–1734. [CrossRef]
27. Barnsley, M.F.; Demko, S.; Powell, M.J.D. Iterated function systems and the global construction of fractals. *Proc. R. Soc. London. A Math. Phys. Sci.* **1985**, *399*, 243–275. [CrossRef]
28. Rasekhi, S.; Shahrazi, M. Modified R/S and DFA Analyses of Foreign ExchangeMarket Efficiency under Two Exchange Rate Regimes: A Case Study of Iran. *Iran. J. Econ. Res.* **2014**, *18*, 1–26.

29. Feder, J. *Fractals*; Physics of Solids and Liquids; Springer: New York, NY, USA, 1988.
30. Julián, M.; Alcaraz, R.; Rieta, J. Study on the Optimal Use of Generalized Hurst Exponents for Noninvasive Estimation of Atrial Fibrillation Organization. In Proceedings of the Computing in Cardiology 2013, Zaragoza, Spain, 22–25 September 2013.
31. Di Matteo, T. Multi-scaling in finance. *Quant. Financ.* **2007**, *7*, 21–36. [CrossRef]
32. Feller, W. *An Introduction to Probability Theory and Its Applications*; Wiley-Blackwell: Hoboken, NJ, USA, 1971.
33. Selvaratnam, S.; Kirley, M. Predicting Stock Market Time Series Using Evolutionary Artificial Neural Networks with Hurst Exponent Input Windows. *Lect. Notes Comput. Sci.* **2006**, *4304*. [CrossRef]
34. Schölzel, C. *Nonlinear Measures for Dynamical Systems*; version 0.5.2; Zenodo: 2019. [CrossRef]
35. Eckmann, J.P.; Kamphorst, S.; Ciliberto, S. Liapunov exponents from time series. *Phys. Rev. A* **1987**, *34*, 4971–4979. [CrossRef] [PubMed]
36. Henriques, T.; Ribeiro, M.; Teixeira, A.; Castro, L.; Antunes, L.; Costa Santos, C. Nonlinear Methods Most Applied to Heart-Rate Time Series: A Review. *Entropy* **2020**, *22*, 309. [CrossRef]
37. Zengrong, L.; Liqun, C.; Ling, Y. On properties of hyperchaos: Case study. *Acta Mech. Sin.* **1999**, *15*, 366–370. [CrossRef]
38. Mayer, A.L.; Pawlowski, C.W.; Cabezas, H. Fisher Information and dynamic regime changes in ecological systems. *Ecol. Model.* **2006**, *195*, 72–82. [CrossRef]
39. Klema, V.; Laub, A. The singular value decomposition: Its computation and some applications. *IEEE Trans. Autom. Control* **1980**, *25*, 164–176. [CrossRef]
40. Makowski, D.; Pham, T.; Lau, Z.J.; Brammer, J.C.; Lespinasse, F.; Pham, H.; Schölzel, C.; Chen, S.A. NeuroKit2: A Python Toolbox for Neurophysiological Signal Processing. *Behav. Res. Methods* **2020**, 1–8. [CrossRef]
41. Fraser, A.M.; Swinney, H.L. Independent coordinates for strange attractors from mutual information. *Phys. Rev. A* **1986**, *33*, 1134–1140. [CrossRef]
42. Krakovská, A.; Mezeiová, K.; Budáčová, H. Use of False Nearest Neighbours for Selecting Variables and Embedding Parameters for State Space Reconstruction. *J. Complex Syst.* **2015**, *2015*, 932750. [CrossRef]
43. Stark, J.; Broomhead, D.S.; Davies, M.E.; Huke, J. Takens embedding theorems for forced and stochastic systems. *Nonlinear Anal. Theory Methods Appl.* **1997**, *30*, 5303–5314. [CrossRef]
44. Caraiani, P. The predictive power of singular value decomposition entropy for stock market dynamics. *Phys. Stat. Mech. Its Appl.* **2014**, *393*, 571–578. [CrossRef]
45. Gu, R.; Shao, Y. How long the singular value decomposed entropy predicts the stock market? —Evidence from the Dow Jones Industrial Average Index. *Phys. A Stat. Mech. Its Appl.* **2016**, *453*. [CrossRef]
46. Roberts, S.J.; Penny, W.; Rezek, I. Temporal and spatial complexity measures for electroencephalogram based brain-computer interfacing. *Med Biol. Eng. Comput.* **1999**, *37*, 93–98. [CrossRef] [PubMed]
47. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
48. Zhou, R.; Cai, R.; Tong, G. Applications of Entropy in Finance: A Review. *Entropy* **2013**, *15*, 4909–4931. [CrossRef]
49. Kim, S.; Ku, S.; Chang, W.; Song, J.W. Predicting the Direction of US Stock Prices Using Effective Transfer Entropy and Machine Learning Techniques. *IEEE Access* **2020**, *8*, 111660–111682. [CrossRef]
50. Kim, S.S. Time-delay recurrent neural network for temporal correlations and prediction. *Neurocomputing* **1998**, *20*, 253–263. [CrossRef]
51. A. Waibel. Modular Construction of Time-Delay Neural Networks for Speech Recognition. *Neural Comput.* **1989**, *1*, 39–46. [CrossRef]
52. Sun, W.; Wang, Y. Short-term wind speed forecasting based on fast ensemble empirical mode decomposition, phase space reconstruction, sample entropy and improved back-propagation neural network. *Energy Convers. Manag.* **2018**, *157*, 1–12. [CrossRef]
53. Rhodes, C.; Morari, M. The false nearest neighbors algorithm: An overview. *Comput. Chem. Eng.* **1997**, *21*, S1149–S1154. [CrossRef]