Neural Computing and Applications (2024) 36:15777–15797
https://doi.org/10.1007/s00521-024-09931-4

ORIGINAL ARTICLE

Stock market index prediction using transformer neural network models and frequency decomposition

Camilo Yañez[1] · Werner Kristjanpoller[1] · Marcel C. Minutolo[2]

Received: 10 January 2024 / Accepted: 28 April 2024 / Published online: 18 May 2024

Abstract
In an increasingly complex and volatile environment, government officials, researchers, and investors alike would like to possess models that accurately forecast markets in order to make appropriate decisions. This research investigates the efficacy of Transformer-based deep neural networks in predicting financial market returns compared to traditional models, focusing on ten different market indexes. The study employs a comprehensive methodology that involves iterative dropout tests and batch size optimization to enhance model performance. By leveraging the power of deep learning, the research aims to improve prediction accuracy and capture complex patterns in stock market data. Twelve neural network architectures are compared across ten indexes to measure their performance, finding that the proposed Transformer variants produce significantly better results compared to benchmark models in all cases. The results of ablative experiments reveal the superiority of Transformer models in capturing long-term dependencies and extracting meaningful features from time series data. The findings suggest that Transformer neural networks outperform LSTM networks and other traditional models in forecasting financial market trends. This research contributes to the growing body of literature on deep learning applications in finance and provides valuable insights for government officials, researchers, and investors seeking to make informed decisions in the stock market. The implications of this study extend beyond academia, offering practical implications for enhancing prediction accuracy and optimizing investment strategies in the dynamic and volatile financial market landscape.

Keywords Transformer encoder · Convolutional neural network · Empirical mode decomposition · Complete ensemble empirical mode decomposition with adaptive noise · Stock market forecasting

1 Introduction

Market indexes are a weighted average of the most representative set of stocks in terms of market liquidity within the respective market. As such, the movement of an index suggests the market's expectations of the future of their respective economies. Further, given their composition, they reflect the general behavior of a specific market, which serves as a point of comparison for other financial assets. Hence, the index serves as a proxy for market sentiment about the past, present, and future of an economy.

Possessing the ability to accurately forecast future variations of the price of an index has the potential to result in improved benefits and reduced risks associated with investment, policy decision, and resource allocation. However, accurately predicting stock market movements is an extremely difficult task to perform since there are many things that can influence the movement of the time series. Given the difficulty of forecasting market movements, this research is inspired by the work of [1], where new hybrid algorithms capable of extracting deep features and time sequences were proposed. [1] found that their proposed models improved prediction accuracy and outperformed other models. Despite the improvement of their models, forecasting stock markets remains challenging and there is still room for greater performance. Therefore, the field

✉ Werner Kristjanpoller
werner.kristjanpoller@usm.cl

1 Departamento de Industrias, Universidad Técnica Federico Santa María, Valparaíso, Chile

2 Rockwell School of Business, Robert Morris University, Moon Township, PA, USA

should try to capture as features more scenarios and conditions from the past in order to forecast the future more precisely. To accomplish the task of developing improved forecasting models, it is first necessary to take as an assumption that markets are not random. Rather, throughout the series of an index under study there is a component capable of being modeled, and that said component repeats itself over time. Finally, features in a series which repeat over time are able to be captured by the model allowing for future projections.

Those interested in forecasting market movements have many tools at their discretion to accomplish said task. These tools capture some of the aforementioned characteristics in the time series of an index. However, these models have not yet been able to present reliable representations of reality due the nonlinearity, non-stationarity, and intrinsic volatility of stock market indexes [2]. Therefore, researchers continue to search for more precise models that result in robust predictions of index movements. In this regard, *deep learning* techniques have shown better performance in forecasting than approaches considered to be *classical*, such as Autoregressive Integrated Moving Average (ARIMA) models (e.g. [3–5]). Even hybrid models that include classical and neural network (NN) approaches have failed to achieve consistently better results compared to simple machine learning models. This is verified in the study by [6], where the hybrid Generalized AutoRegressive Conditional Heteroskedasticity Artificial Neural Network (GARCH-ANN) and Exponential GARCH-ANN (EGRACH-ANN) models developed by [7] are compared with simple ANN models.

Using frequency decomposition methods may facilitate the data analysis process by separating a volatile time series into several sequences of frequency spectra with lower volatility. Recent studies proved greater effectiveness of models that use both empirical mode decomposition (EMD) together with other models or combinations of machine learning models [8, 9], as well as the combination with Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN) [10]. Following this same line of research, [11] presented a hybrid CEEMDAN and Transformer model for short-term load forecasting, where they verified the superiority of these architectures' forecast for improving energy management. This work demonstrates a research gap in enhancing forecast accuracy by incorporating frequency decomposition into the model. Further, it underscores the limitations of current deep learning approaches, which are often insufficient when not paired with robust preprocessing techniques like EMD or CEEMDAN. In the work presented here, we apply these architectures in a similar way, but where the earlier work used the approach for energy forecasting, we apply the technique to financial time series.

It is worth noting research similar to this one, where improvements in the predictive power of stock market indexes have been studied using long short-term memory LSTM models with EMD [1, 12]. Both of these investigations obtained favorable results in the prediction of the stock market indexes studied. However, as far as we know, there are no works that seek to develop the same task of predicting financial time series with EMD taking advantage of the Transformer architecture. Therefore, the main contribution of this study is to propose a new hybrid model of a Transformer Encoder with a Convolutional Neural Network (CNN) layer based on EMD and CEEMDAN decomposition. This seeks to improve the performance of current prediction models for stock market indexes by comparing the results obtained with the performance obtained by models based on LSTM networks in the prediction of ten different market indexes.

2 Literature review

Time series are structured data in the form of a sequence separated by some interval of time. As suggested earlier, for the analysis of these series there are various tools that help us obtain information regarding its qualitative and quantitative characteristics, as well as the behavior over time. Additionally, in the field of study of financial series, mechanisms have been devised that aim to forecast the behavior of future values based on previously observed values.

Deep learning (DL) is a subset of machine learning that has shown rapid development over the past two decades with respect to time series analysis and has been widely used by various researchers in virtually every industry. Examples of this can be seen in studies by [13] and [14], where they use artificial neural networks (NN) to analyze properties of materials used in various civil engineering applications. In short, there has been a growing interest in the development of deep learning models, since they have been shown to significantly outperform their traditional Machine Learning counterparts. However, there is still a lack of work that focuses only on DL for financial time series [15].

In the development of DL models, the aim is to identify patterns and make predictions of the real world by imitating the functioning of the human brain itself. [16] describes this operation and the structure of the artificial neural networks, which are composed of layers, and these in turn by simple processing units called artificial neurons, given their similarity to the neurons of the human brain. Each artificial neuron consists of weights, bias, and a specific activation function, where its mathematical model is shown in 1.

$$Y = f\left(\sum W_m X_m + b\right) \tag{1}$$

where Y is the output of the neuron, X_m is the input vector, W_m is the weight matrix, b is the bias vector and f is the activation function (AF). Among the most popular AFs are the Sigmoid, Tanh, ReLU and ELU, although there are many others with different characteristics that may be used depending on the desired output. The connection between neurons allows the weights of each neuron to be adjusted during network training according to a loss function that evaluates the distance between the real values and the output values in order to minimize the error of the model depending on said loss function.

One of the earliest and most widely used DL models to train sequential data is the recurrent neural network (RNN) [17]. However, RNN models have a drawback known as "short-term memory", caused by the Vanishing Gradient problem. A vanishing gradient occurs in the RNN processes as more steps are added, and information retention from previous steps declines [18]. In response to this, the LSTM networks proposed by [19] have a modification of the original RNN model to address memory loss. The LSTM model implements *forget gates* and forward mechanisms can make the network retain information that is stored longer for later use and removes irrelevant information. These architectures are less frequently applied to financial time series forecasting but are inherently suitable for this domain due to the aforementioned characteristics [20].

However, the field of DL is rapidly developing more accurate models in most research areas. Transformers (detailed later) have an increasing role in deep learning, displacing even highly consistent architectures such as LSTM networks in the case of natural language processing (NLP) [21] or even in the area of medicine with image processing with convolutional neural networks [22]. LSTM networks process information sequentially, as shown by the RNN network in Fig. 1. This architecture should theoretically be able to retain important information over infinitely many sequences. However, in practice, these networks eventually suffer the vanishing gradient problem, that is, the LSTM network will at some point forget relevant information from previous sequences.

In Fig. 1 we illustrate the self-attention (SA) layer used by the Transformer and contrast its structure with that of a recurrent neural network (RNN). The self-attention layer of the transformer resolves the problem of the vanishing gradient by allowing the model to access information from any previous point in the time series. Unlike RNNs, which process sequences step by step and convey information through recurring hidden states, SA has the ability to access all previous points in the series and weight them according

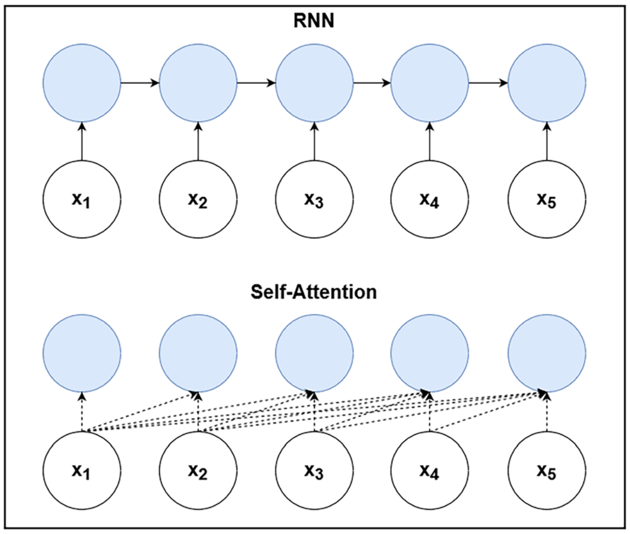


Fig. 1 RNN vs self-attention

to a learned measure of relevance, providing information about distant points. This feature is particularly useful in the study of financial time series, such as predicting prices or returns of stock market indexes, where it is crucial to integrate relevant information from distant points given the difficulty that this task represents. In addition, hybrid models of neural networks based on frequency decomposition obtain the highest precision for this task. There is still room for improvement that this study exploits by innovating in the implementation of a Transformer encoder architecture.

Like RNNs, Transformers are designed to process sequential input data. However, unlike RNNs, Transformers process the entire input all at once, rather than sequentially like RNNs. [23] introduced the Transformer in 2017 and has quickly displaced models that use RNN and LSTM for natural language processing (NLP) [24]. Google's open-source project is applying Bidirectional Encoder Representations from Transformers (BERT) to large datasets to pretrain models for applications in NLP which is resulting in faster development and novel approaches [25, 26]. In fact, [27] applied Transformers to study the impact on investor sentiment on stock volatility and [26] used them with sentiment analysis for investment strategies.

3 Methodology

For the correct understanding of the hybrid models proposed in this study, it is first necessary to understand the functioning of the individual parts of which they are composed. For this reason, this section focuses mainly on the decomposition algorithms used, such as EMD and

CEEMDAN, as well as deep learning methods such as CNN, LSTM and Transformer networks, among other components involved in these models.

3.1 Decomposition algorithms

The high volatility of financial time series data, such as the closing price of a stock or market index, makes their prediction highly challenging. Therefore, to deal with this high volatility, the series of returns are first decomposed into **7 signals** plus a **Residual** by one of two mechanisms: EMD or CEEMDAN. The purpose for carrying out two decompositions is to later be able to conclude to what extent changing the decomposition method affects and which is more efficient for these financial series.

3.1.1 Empirical mode decomposition

The Empirical Mode Decomposition (EMD) presented by [28] is an iterative procedure that decomposes a signal $f(t)$ into a set of oscillatory components called Intrinsic Mode Functions (IMF). An IMF is defined as a function that is symmetric with respect to the local zero mean and has the same number of extrema and zero-crossings, or differs at most by one. The sum of all extracted IMFs and a residue $R(t)$ reconstructs the original signal. This is extremely useful for analyzing highly complex time-frequency series.

The oscillatory algorithm can be described as follows:

Algorithm 1 Empirical mode decomposition

```
1: Initialize $k \leftarrow 0$
2: Set the initial residue $r_0(t) \leftarrow f(t)$
3: **while** the stopping criterion is not satisfied **do**
4:     Identify all local extrema of $r_k(t)$
5:     Perform cubic spline interpolation on the extrema to construct the upper $e_{max}(t)$ and lower $e_{min}(t)$ envelopes
6:     Compute the mean envelope $m(t) \leftarrow \frac{e_{min}(t)+e_{max}(t)}{2}$
7:     Extract the candidate IMF $d_{k+1}(t) \leftarrow r_k(t) - m(t)$
8:     Apply a sifting process to $d_{k+1}(t)$ until it meets the IMF criteria, then denote as $c_{k+1}(t)$
9:     Update the residue $r_{k+1}(t) \leftarrow r_k(t) - c_{k+1}(t)$, increment $k$
10: **end while**
11: The final residue is $R(t) \leftarrow r_k(t)$
12: Express the original signal as $f(t)$ as the sum of $n$ IMFs and the final residue:
```

$$f(t) = \sum_{i=1}^{n} c_i(t) + R(t) \tag{2}$$

where $c_i(t)$ represents the i^{th} IMF, and $R(t)$ denotes the residual component, capturing the trend of the signal.

3.1.2 Complete ensemble empirical mode decomposition with adaptive noise

One of the disadvantages of EMD is that it fails to perform a correct decomposition when the signal has low-frequency waves that behave similarly to the characteristics of high-frequency waves when they are mixed together [29]. This is called the "mode-mixing problem". In order to solve the mode-mixing problem, [30] proposed EEMD (Ensemble EMD), which slightly disturbs the signals to be decomposed by adding white noise. Adding white noise provides a uniformly distributed scale in the time-frequency space, generating a more robust decomposition thereby avoiding the mode-mixing problem and eliminating the need to apply subjective criteria for the selection of IMFs unlike the original EMD model. It should be noted that the added noise will be averaged over a sufficient number of trials so that nearly the only thing to survive is the original signal resulting in obtaining a pure decomposition. However, in EEMD, some useless IMFs are generated, so [31] proposed a CEEMDAN to improve the decomposition and thus achieve an exact reconstruction of the original signal with better spectral separation of the intrinsic mode functions (IMF). Finally, this method is optimized by generating fewer IMFs thanks to the fact that it manages to better separate the different components of the signal, which can

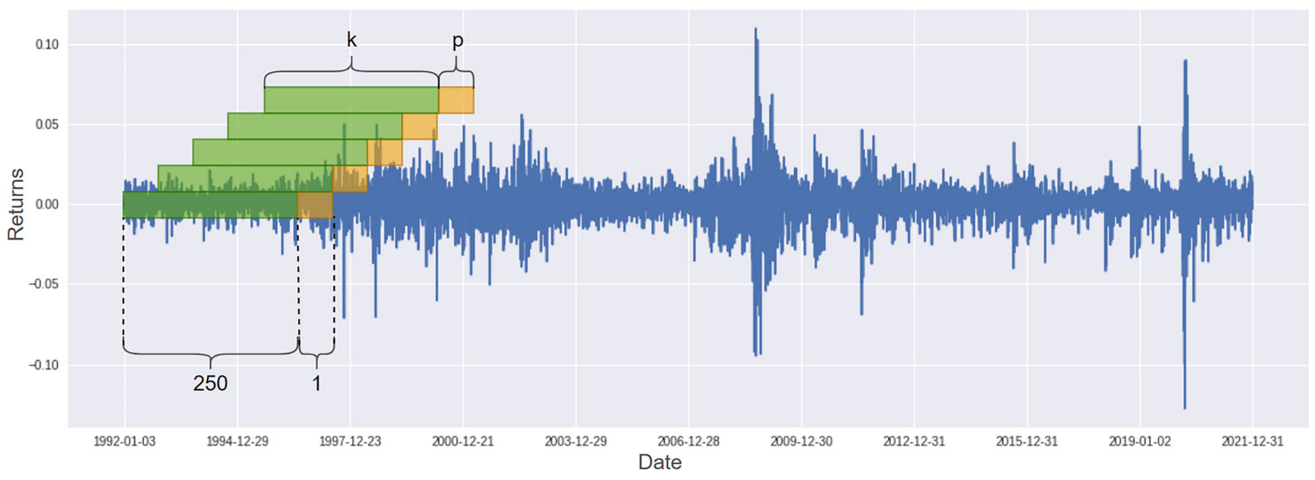


Fig. 2 One step ahead sliding window

reduce the computational cost of the methodology [32]. This last innovation is incorporated in the decomposition involving CEEMDAN in this study.

It is important to note that given the nature of CEEMDAN, to obtain identical IMFs when decomposing the same signal on different occasions, it is always necessary to add the same white noise. This is essential when comparing the performance of the models, since they have to work with the same data, and therefore, one must be careful to always study the same IMFs. Furthermore, as previously stated, the IMFs generated by both EMD and CEEMDAN encapsulate unique frequency-related information pertaining to the time series. These spectral components, while distinct in their characteristics, are collectively indispensable. Each component contributes a vital piece of the original series' structure, rendering every IMF essential for the accurate reconstruction and subsequent prediction of the series.

3.2 Sliding window

One of the most used tools by researchers to combat the high volatility of time series to improve model prediction performance is to use a sliding window [33]. This technique is useful in improving forecasts of the next day closing price of a stock based on a sliding window length and not just based on the current day's closing price. This result is due to the fact that the information contained in the current value only is not enough to capture all the information of the series. This is why when training the network with non-unique values in the form of a sequence, it is possible to deliver enough useful information to significantly improve the forecast results. This can be seen in [34], wherein it is concluded that a larger sliding window performs better than a smaller. Thus, instead of feeding the network with data with the form (n, p), where n is the current price and p is the price to be predicted, the model data is fitted with (n_{k-1}, k, p), where k is the length of the sliding window of previous prices needed to predict the next value of p.

For this work, as in the study of [1], a window of $k = 250$ is used since it is roughly equivalent to a trading year and is considered a reasonable size taking into account the amount of data available. An example of this is illustrated in Fig. 2 with the return series for the S &P 500 Index. Here, data from x_i to x_{i+249} is used to predict the x_{i+250} value. For the first window, the returns from the first position to the value of position 250 are used to predict the value in $i = 251$. Subsequently, for the second window, the returns from position 2 to 251 are used to predict $i = 252$; this cycle is repeated until the last value in the series is predicted.

3.3 Neural network models

3.3.1 Convolutional neural networks

In the use of deep learning for image recognition, convolutional neural networks, a term first coined by [35], have been shown to be one of the best-performing architectures for such a task [36]. However, these networks are also used for an endless number of tasks in various areas of study, since they are an excellent architecture to extract certain characteristics and patterns from some input, similar to those that human beings identify through vision. For images, CNN can detect vertical or horizontal lines; for stock indexes, it can detect trends, which in turn can form patterns in the price movements of the index under study.

By itself, a convolutional filter is not powerful enough to detect more complex patterns, so it is essential to complement it with another type of neural network. It is for this reason that convolutional networks are represented in the form of a funnel, since each layer reduces the initial information but extracts useful features that increase its depth. For this study, the CNN network is expected to be able to identify for each data point characteristics related to the points that are closest to each other since these should be more related to each other than more distant data points. Therefore, by adding a CNN layer to the model, we expect that it will improve its accuracy when predicting the return one step ahead. When using models that have a CNN as the first layer, this is applied as a convolution with 64 filters, since it was found to produce the best results after numerous tests.

3.3.2 Long short-term memory

As mentioned in Sect. 2, LSTM networks are presented as a modification of the original RNN model to solve the problem of "short-term memory" caused by vanishing gradient during training. These architectures have a forgetting gate f_t, which, as its name indicates, allows elements to be eliminated from memory. They also have an entry gate i_t, where new elements are added to the memory, and an output gate o_t, where the hidden state of the memory is updated. Each one of these gates is built with a *sigmoid* function that allows the total passage of information when the value is 1; a partial passage when intermediate; and, completely blocks its passage if 0.

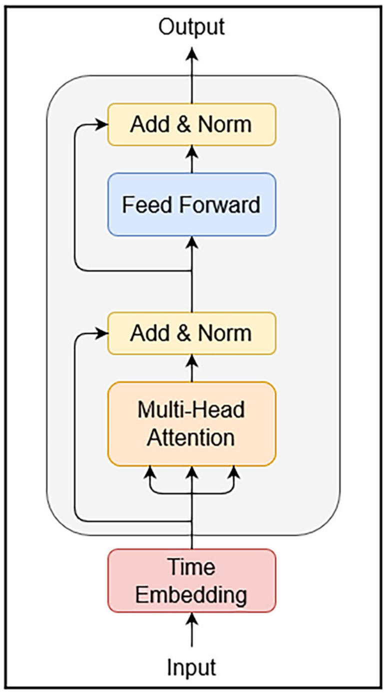


Fig. 3 Transformer encoder architecture

3.3.3 Transformers

In contrast to the RNN and LSTM, Transformers (TF) have direct connections between all their previous sequences as illustrated in Fig. 1 [23]. The increased connections allow all the information to propagate in all the sequences of the network. This has benefits such as the ability to parallel sequence and more robust long-term memory than the ones in the previously mentioned counterparts. However, this also implies that the model will be directly connected to all the information, whether it is relevant or not. Therefore, to filter what is truly important, the Transformer must use an algorithm called self-attention.

In the self-attention mechanism, vectors of *Query*, *Key*, and *Values* are created for each token. Subsequently, the current *Query* is evaluated with each *Key* of the previous tokens to determine its relevance with respect to the current token, and the resulting value is compared with the *Value* vector that represents the true representation of said token. Specifically, for a given input token embedding x, the *Query* vector is calculated as $Q = W^Q x$, the *Key* vector as $K = W^K x$, and the *Value* vector as $V = W^V x$. The matrices W^Q, W^K, and W^V are parameters of the model that are learned during the training process. Each of these matrices has a size of $d_k \times d_{model}$, where d_k is the dimensionality of the Q, K, and V vectors, and d_{model} is the dimensionality of the token embeddings. These vectors are the ones that are adjusted as the model is trained. Then, by means of the scalar product between the *Query* and *Key* vectors, a score is obtained that represents the relevance between the current token and the previous tokens in the sequence. It should be noted that high scores indicate high relevance, while low scores indicate the opposite. The attention scores for a given *Query* are divided by $\sqrt{d_k}$ to stabilize gradients during training, as large values could result in very small gradients if not scaled properly. After scaling, the scores are passed through a *softmax* function to make them all positive and sum = 1, as can be seen in 3. The final values can be interpreted as the percentage of focus that is given to each token in the sequence.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

As a result of all this, a mechanism is obtained that allows all the available information to contribute in a weighted manner according to its relevance with respect to the current token in a context vector. The tokens with high scores contribute in greater proportion than the tokens with low scores. However, no information is lost at any time, which solves the vanishing gradient problem. Additionally, for the particular case of study, where it is required to predict a single token of the sequence, it is enough to use only the

Transformer encoder and not its complete architecture, as represented in Fig. 3. This simplifies the implementation of the model and its computational cost during training.

When working with time series there is an additional challenge that must be considered compared to other Transformer implementations. Since the entire sequence is delivered to the Transformer simultaneously, values from years ago are given the same importance as the value the day before. To solve for this, we propose to use **time embedding**, which imparts the notion of time that is hidden in the series. For the implementation of time embedding, the **Time2Vec** approach described by [37] is used. With it, it is possible to obtain a vectorial representation of the periodic and non-periodic characteristics of a time series, as described in 4.

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i\tau + \varphi_i, & \text{if } i = 0. \\ \mathcal{F}(\omega_i\tau + \varphi_i), & \text{if } 1 \le i \le k. \end{cases} \tag{4}$$

where $\mathbf{t2v}(\tau)[i]$ is the ith element of $\mathbf{t2v}(\tau)$, $\mathcal{F}$ is a periodic activation function, and both ω_i and φ_i are parameters. $\omega_i\tau + \varphi_i$ represents the non-periodic characteristic of the series, while $\mathcal{F}(\omega_i\tau + \varphi_i)$ represents the periodic part, and $\mathcal{F}$ is represented as a *sine* function.

Finally, time embedding is the first part of data processing in the Transformer network and serves as input for its Encoder architecture, as can be seen in Fig. 3.

3.4 Model components

One of the components that all neural network models have is the *activation functions* (AF), which plays a very important role when learning abstract features through nonlinear transformations [38]. Some of the first and most traditional activation functions were the Logistic Sigmoid and the Tanh that produce outputs between [0, 1] and $[-1, 1]$ respectively. The Tanh function is more useful than the Sigmoid since it is centered at zero, allowing for better convergence. However, both functions have the vanishing gradient problem, which occurs when the input is a value far from 0. Therefore, when performing *"backpropagation"*, they adjust the parameters of each neuron, causing it to stop learning in the first layers of the network.

[39] have demonstrated that the Rectified Linear Unit (ReLU) activation function produces better performance than hyperbolic tangents despite its nonlinearity and non-differentiability at zero, and is currently the most successful and widely used AF [40]. This is because the ReLU AF has low computational cost and since it is not bounded for positive inputs, it generates a constant gradient with faster learning.

Finally, [41] developed the Gaussian Error Linear Unit (GELU) AF, which manages to overcome the precision of ELU and ReLU thanks to its non-monotonic nature that prevents the weights of some neurons from stagnating at a null value (dead neurons). GELU is a type of stochastic activation function that has not been widely explored due to its expensive sampling process. It is defined as:

$$\mathrm{GELU}(x) = x \times P(X \le x) \tag{5}$$

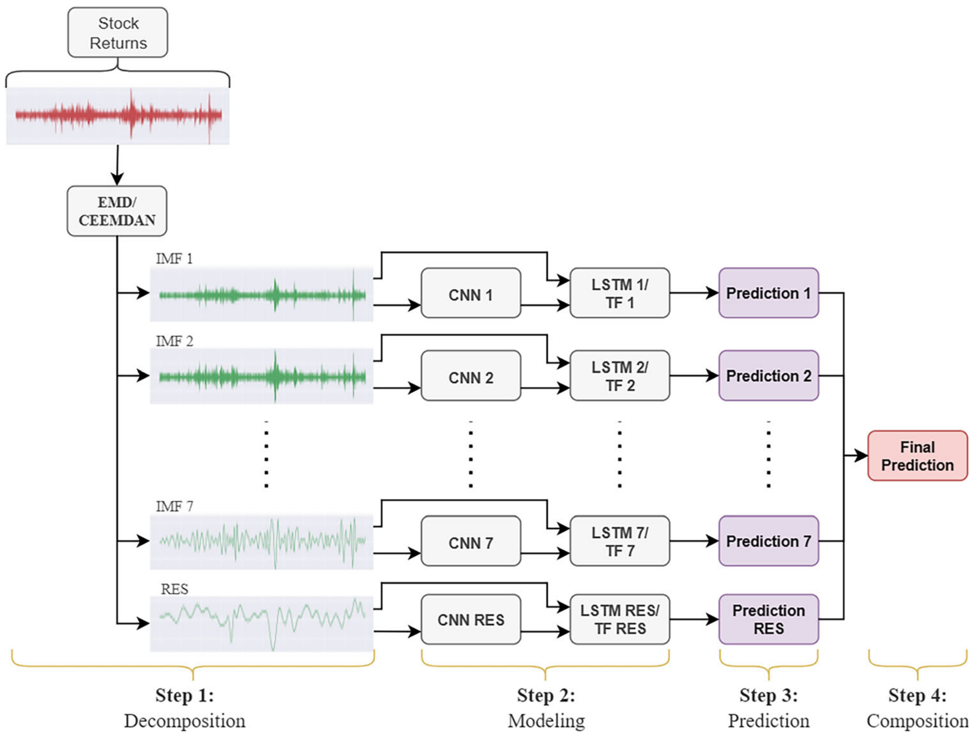
where P is the probability and cause of its high computational complexity. However, it presents a promising relationship between precision and training time [38] and the reason that we use it as the activation function for this study.

Another fundamental part of training a neural network is performing the aforementioned *"backpropagation"*. This calculates how much the error of the network would change by varying the weights of each in a hierarchical multilayer neural network. The errors of neurons in one layer are computed from the errors of the layer immediately before it. Therefore, error computations start at the final layer and flow backward which leads to the concept that errors "back propagate" through the network [42]. Once the errors for each neuron have been calculated, the final output error can be reduced by changing the input weights of each neuron.

It is also important to take into account the *loss function* and the *optimizer* working together with which the model is trained. The loss function evaluates how well the machine learning algorithm models the data supplied for fitting during training, where a high loss value indicates a poorer fit compared to a lower loss. For this, just like AFs, there are a variety of loss functions that may be used depending on the nature of the problem to be solved and the importance that is given to certain values, as there can be outliers. Among the most common loss functions for regression models (those in this study) are the mean squared error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and the Huber loss function. MSE gives greater importance to outliers, MAE gives less and therefore focuses more on the bulk of the data, and the Huber loss function is a midpoint between the two [43]. Some of these loss functions are covered in more detail in Sect. 3.7 when reviewing evaluation metrics.

As previously mentioned, the loss function works in conjunction with some optimizer. The loss function is responsible for adjusting the weights of the network with the objective that the loss function decreases its value, that is, improves the model fit with the actual values. In this case, there are also a wide variety of ways in which this task can be carried out, which range from simply monitoring the negative gradient of a function in order to locate its minimum, as is done by the Stochastic Gradient Descent (SGD), to more complex mechanisms such as the Nadam optimizer.

Fig. 4 Flowchart of models



The Nesterov-accelerated Adaptive Moment Estimation (Nadam) is an optimizer that combines the Adam algorithm with Accelerated Nesterov Gradient [44]. Adam is a gradient descent extension that adds the first and second moment of the gradient by applying moving averages to smooth out noisy objective functions and improve convergence. The Nesterov impulse modifies the calculation speed of certain parameters by using the gradient of the projected update of the parameter instead of the value of the actual current variable, this allows for slowing down the search when the optimum is located, in addition to improving the speed of convergence and the quality of the learned models.

There are mixed results regarding which optimizer may perform best. Some results suggest that the basic SGD can better generalize [45], while other results suggest that optimizers such as Adam, Nadam or Adamax produce better results [46]. Here, after multiple tests, it was determined that the best optimizer for the models under study, and therefore the one that is used, is the Nadam optimizer.

Finally, in all the models included in this research, during the training of the series, the global learning rate of the model decreases when learning becomes stagnant. This is because models often show benefits both in terms of convergence and time by doing this [47].

3.5 Benchmark models

As previously stated, CNN networks are capable of extracting useful patterns and features in time series analysis, so connecting a layer of these to an LSTM network can significantly improve accuracy in financial time series prediction. In addition, in Sect. 3.1 we discussed the potential of decomposing a series to further improve its analysis. Therefore, in this study the two decomposition techniques described are used as benchmarks, similar to what was done in [1]. However, in the previous study, the two decomposition techniques were their proposed models, while in this study they are the benchmarks. We use the same decomposition logic described in Fig. 4 of Sect. 3.6.

In addition, the simple models of the LSTM networks and series without decomposition are also included in order to measure the impact of including a CNN layer and decomposition. The comparison models are the following:

- LSTM
- CNN-LSTM
- EMD-LSTM
- EMD-CNN-LSTM
- CEEMDAN-LSTM
- CEEMDAN-CNN-LSTM

3.6 Proposed models

The models proposed in this study are those based on Transformer encoders with time embedding in replacement of the LSTM neural networks of the benchmark models. In the proposed approach, depicted in Fig. 4, the modeling process commences with an EMD or CEEMDAN frequency decomposition of the stock index time series. This foundational step deconstructs the complex series into eight simpler components: seven IMFs and one residual

Subsequent to the decomposition, Step 2 ensues, wherein each IMF and the residual are individually modeled, employing either the CNN network alone or in combination with LSTM networks or the Transformer, as appropriate.

The third step is dedicated to the forecasting of future values for each IMF and the residual, utilizing their respective trained models. These separate predictions are critical as they encapsulate diverse aspects of the market's behavior from the high-frequency noise to the long-term trend.

The final stage in our modeling process involves the consolidation of individual forecasts derived from each IMF and the residual. By summing these predictions, we synthesize a comprehensive forecast reflective of the original return series.

To summarize, the proposed models are presented below. As can be seen, the LSTM networks were replaced by a Transformer (TF) network since the contribution of this research is to check the efficiency of these algorithms when applied to financial time series.

- TF
- CNN-TF
- EMD-TF
- EMD-CNN-TF
- CEEMDAN-TF
- CEEMDAN-CNN-TF

3.7 Evaluation metrics

In Sect. 3.4, we discussed the importance of the loss functions and how they work together with the optimizer to fit the model to the data. Some loss functions such as MSE and MAE were mentioned, however, these metrics can also be used to evaluate the results obtained by the models in their predictions by quantifying the difference between the actual and predicted values, differentiating between them by the measure of the difference.

The formula to calculate the metrics used in this study are detailed below, where the mean square error (MSE) was used as a function of loss in training, while the root mean square error (RMSE) and Mean Absolute Error (MAE) were used as performance evaluation metrics.

$$MSE = \sum_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{N} \tag{6}$$

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{N}} \tag{7}$$

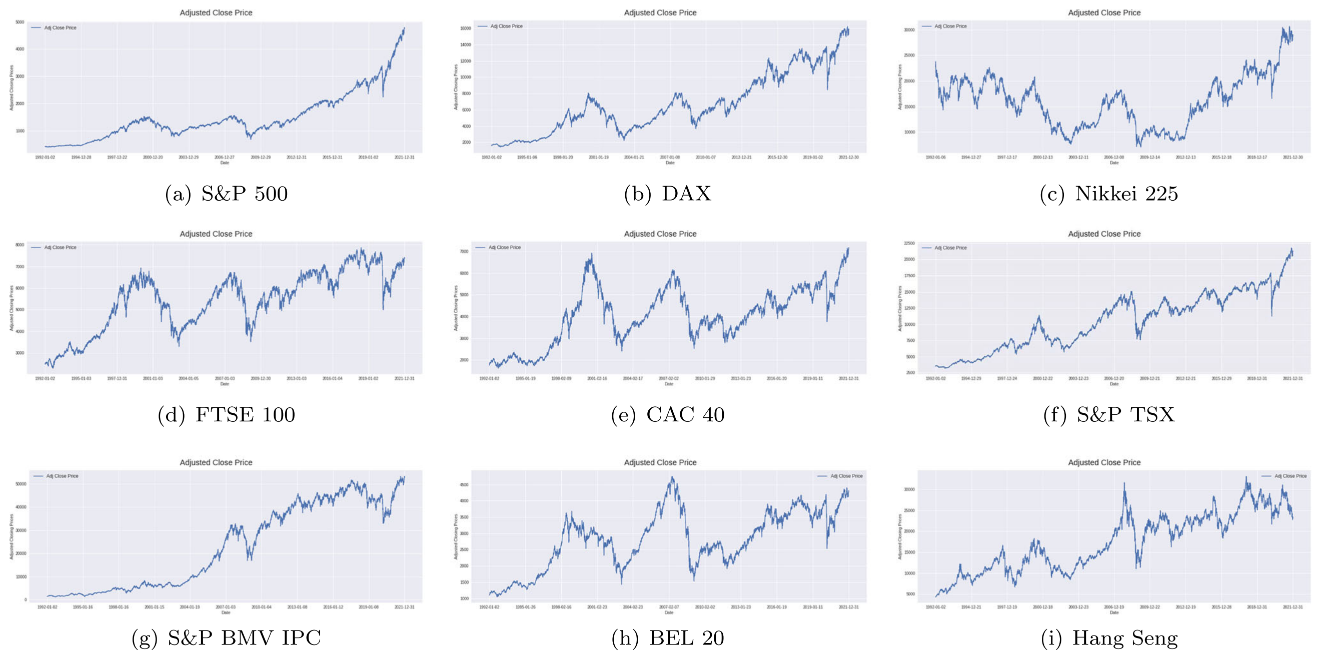
$$MAE = \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|^2}{N} \tag{8}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and N is the total number of observations.

In order to obtain more robust conclusions about the most appropriate model, it is also useful to compare the models according to their superiority in terms of their predictive capacity. To do this, the Model Confidence Set (MCS) proposed by [48] was used, where a ranking is created that allows ordering from the best to the worst model according to its predictive capacity. It should be noted that higher values delivered by the MCS are preferred unlike the error metrics, with a minimum value 0 and maximum of 1.

Table 1 Index price statistics

Index	Count	Mean	Min	Max	Standard deviation
S &P 500	7558	1493	395	4793	883
Dow Jones	7558	12966	3137	36489	7440
DAX	7592	6693	1402	16251	3674
Nikkei 225	7365	16082	7055	30670	4982
FTSE 100	7580	5445	2281	7878	1372
CAC 40	7622	4049	1611	7181	1305
S &P TSX	7543	10541	3195	21769	4367
S &P BMV IPC	7516	23018	1252	53305	17799
BEL 20	7614	2809	1046	4757	888
Hang Seng	7408	17454	4302	33154	6939

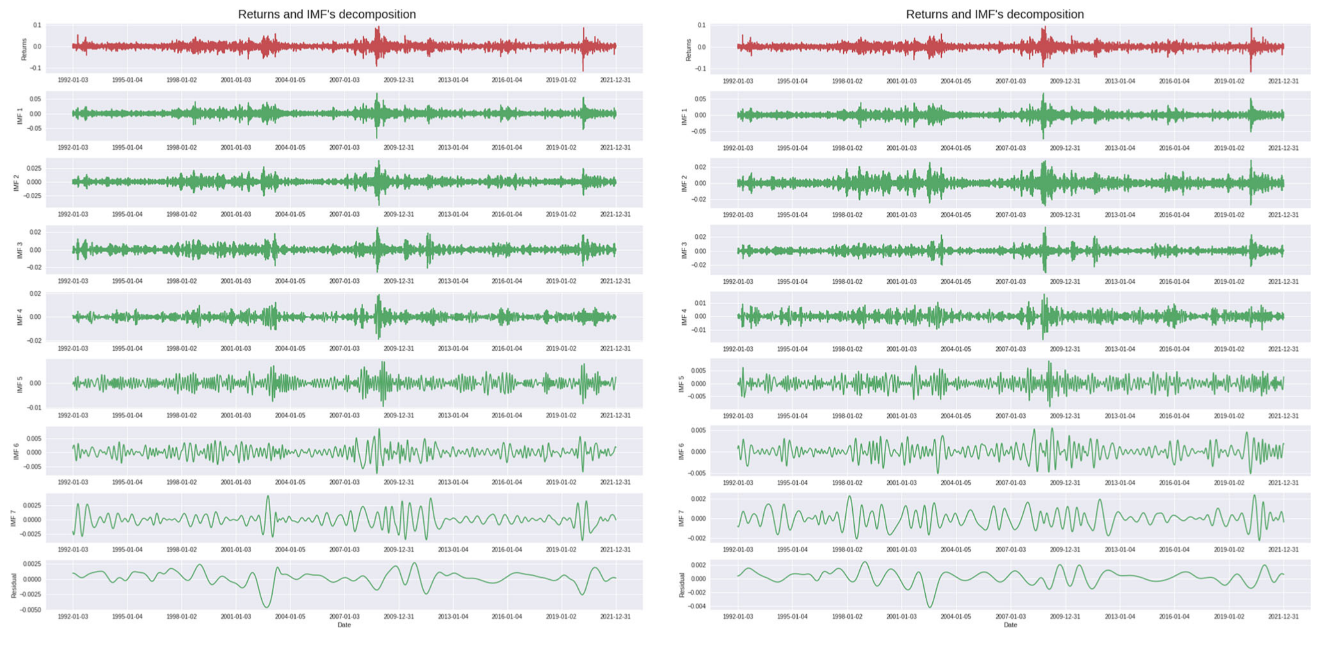


(a) S&P 500 (b) DAX (c) Nikkei 225

(d) FTSE 100 (e) CAC 40 (f) S&P TSX

(g) S&P BMV IPC (h) BEL 20 (i) Hang Seng

Fig. 5 Original index price graphics



(a) EMD (b) CEEMDAN

Fig. 6 IMFs and residual decompositions of FTSE 100 returns index

4 Analysis of experiments and results

In this section, we detail the procedures and results, as well as the analysis of the results. All models were trained with an NVIDIA Tesla T4 GPU and programmed in Python 3.7 using the Keras environment developed by [49]. Open source libraries such as Keras allow for the simplification of programming codes to create complex deep learning algorithms. Therefore, their use favors simpler implementation and the ability to replicate the models developed in this work more easily.

4.1 Data description and transformations

The dataset for this study consists of 10 stock market indexes belonging to 9 different countries to validate the

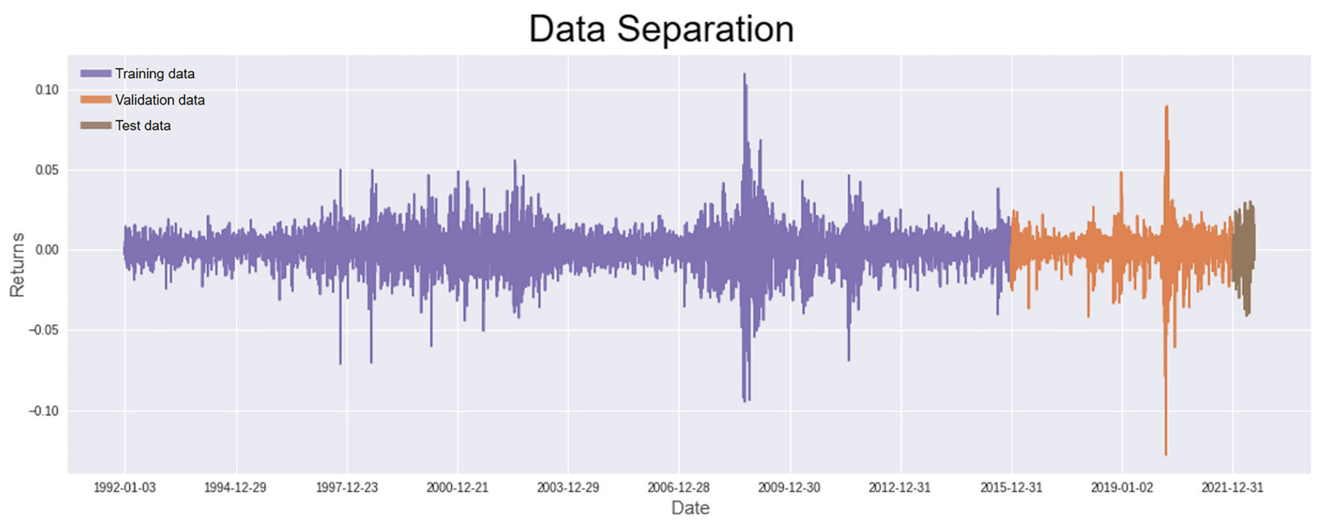


Fig. 7 Training, validation and test sets

results obtained with series that have the greatest possible diversity in terms of characteristics. Data was obtained from Yahoo Finance (https://finance.yahoo.com/). For each of the indexes, we use the opening price, adjusted closing price, highest price, lowest price and volume of the index. However, after obtaining the Pearson correlation coefficient, we found that all price variables had a coefficient greater than 0.99, while the volume variable had a correlation greater than 0.56 with the price variables. Therefore, we determined that using only the adjusted closing price variable for constructing the models would be parsimonious. Each series has historical daily prices from the first trading day of 1992 through the last trading day of 2021. On average, there are more than 7,500 trading days for each series. The descriptive statistics of the indexes are presented in Table 1 followed by graphs of nine indexes to illustrate the financial series (the Dow Jones Industrial Average was excluded given its similarity to the S &P 500) (Fig. 5).

Often, deep learning models have training problems when the input and output data vary widely over an interval [50], as is the case with the prices of the indexes evaluated. For this reason, so that all data has approximately the same scale, we use the logarithmic returns by means of 9. With this, we optimize the learning process of the network while increasing the stationarity of the study series, which produces convenient properties for analysis and improves prediction.

$$R_t = \ln(P_t) - \ln(P_{t-1}) \tag{9}$$

4.2 EMD and CEEMDAN

As mentioned in the methodology and shown graphically in Fig. 4, the first step is decomposition of the series through the EMD or CEEMDAN algorithm. The figures shown in Fig. 6 illustrate the decomposition for the FTSE 100 index, where variation in the frequencies of the IMFs and a slight mode-mixing problem in the EMD signal can be observed, as previously described as occurring in these types of decomposition. It is important to note that the applied EMD method will be complemented by time embeddings within the Transformer model architecture as mentioned in Sect. 3.3.3. These embeddings capture both periodic and non-periodic features of the time series, thus enhancing the model's temporal understanding. Moreover, the conversion of price data into logarithmic returns serves to stationarize the series, acting in itself as a form of feature engineering. This, coupled with the implementation of a sliding window approach, effectively constitutes an innovative form of feature selection tailored for financial time series analysis. Such integrations showcase the novelty of the proposed models, extending beyond conventional use of EMD for time series decomposition.

4.3 Dataset preparation

After decomposition, each IMF and residual is divided into a training and validation set. In this case, the training set is 80% of the total available data, while the remaining 20% is assigned to the validation set. In addition, returns corresponding to the first 150 additional trading days of 2022 are obtained for each index in order to visualize the performance of each model on a dataset not used during the training of the neural network (test set). Figure 7 illustrates

Table 2 Epochs for training

	Undec.	IMF 1	IMF 2	IMF 3	IMF 4	IMF 5	IMF 6	IMF 7	RES
Epochs	300	200	150	100	75	75	75	75	75

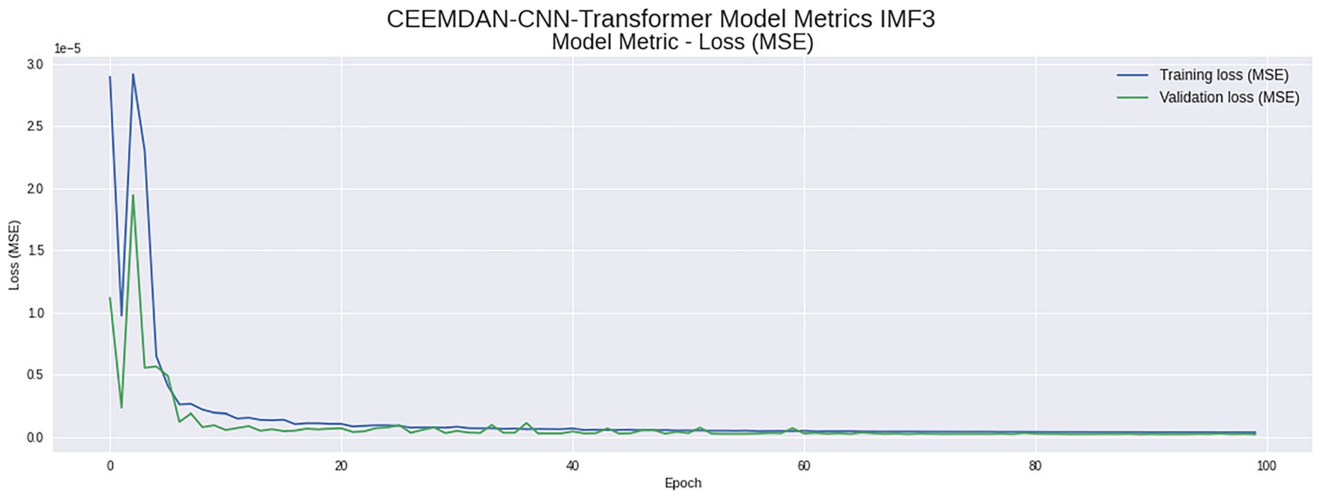


Fig. 8 Loss curve during training

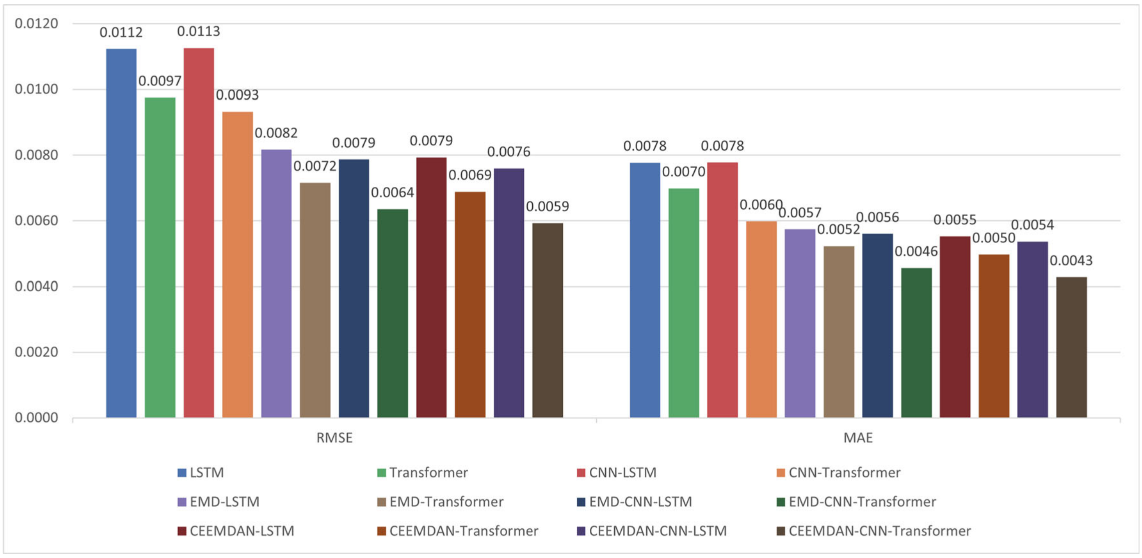


Fig. 9 Results FTSE 100

the training, validation, and test sets mentioned for the returns of the S &P 500 index.

4.4 Model training

The determination of the hyperparameters for the neural networks used in this study was the product of an exhaustive validation and experimentation process. Iterative dropout tests were carried out seeking the balance between effective regularization and retention of critical information. With respect to batch size, various configurations were explored, specifically: 4, 8, 16, 32, 64 and 128, to identify the one that would provide stable and efficient convergence of the model during training.

The configuration of the CNN networks was tuned testing combinations of kernel size, padding, and number of filters in the set {16, 32, 64, 128}. The objective was to optimize the capture of relevant patterns and features without introducing unnecessary complexity into the model. In parallel, the Transformer network architectures

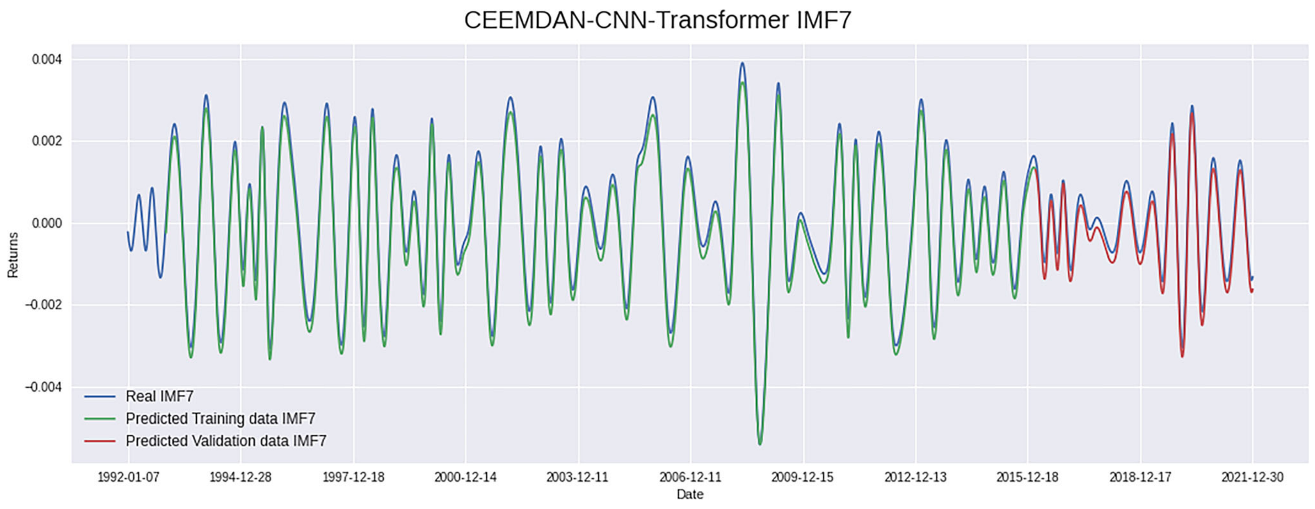


Fig. 10 Returns IMF 7 Nikkei 225

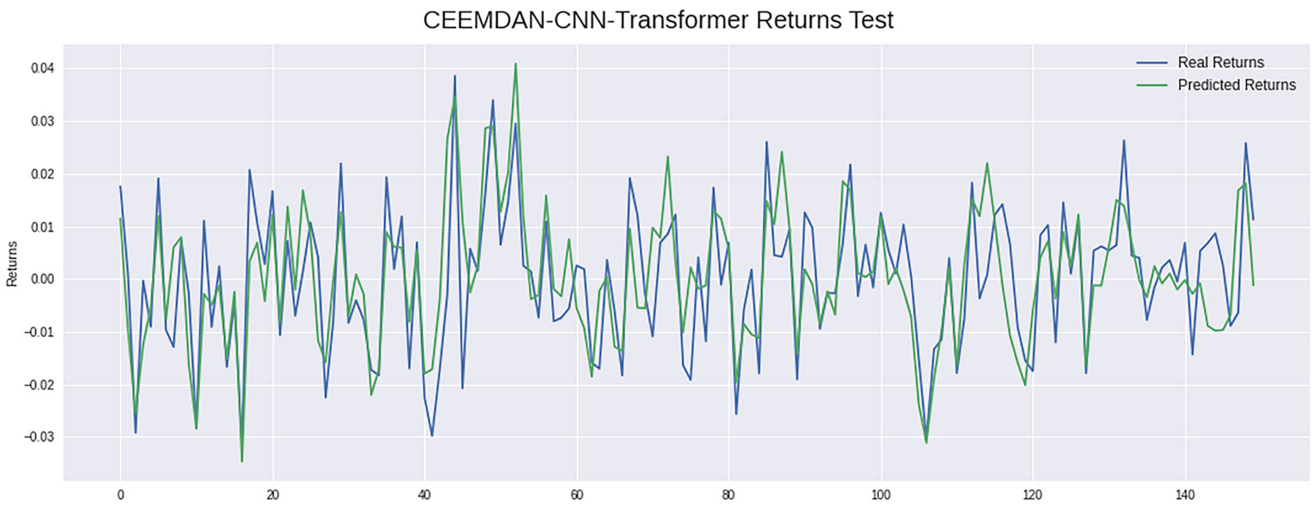


Fig. 11 Test returns Nikkei 225

were tested with different dimensions of the Feed Forward layer (32, 64, 128, 256 and 512) and multiple attention heads (1, 2, 4, 8) to evaluate their capacity of parallel processing and its impact on capturing long-term dependencies.

Similarly, LSTM networks were subjected to a exploration in terms of number of layers and hidden units, testing values of 32, 64, 128, 256 and 512 to understand how these affected the long and short-term memory of the model. After carrying out an iterative process which involved a significant number of experiments, optimal configurations were identified that ensure robust consistency in the results.

This trial and error methodology not only validated the final hyperparameters used but also reinforced the robustness of the models by demonstrating their capacity for generalization and their performance under a wide spectrum of test conditions. This ensures that the selected models, with their established hyperparameters, are optimized for the specific task of financial time series forecasting, providing a foundation for future research and practical applications.

The models share a common batch size of 32 and a “sliding window“ length of $k = 250$ days, as was appropriate for the nature of the financial time series data examined as stated in Sect. 3.2. CNN networks and LSTM were implemented with 64 filters and 64 hidden units, respectively, while the Transformers have 2 heads for multi-attention and a Feed Forward layer dimension equal to 64. Beyond these commonalities, each model type was constructed with a single layer of their respective

Table 3 EMD models results

Index	Model	EMD							
		EMD -				EMD-CNN			
		RMSE	MAE	MCS	Time (Min.)	RMSE	MAE	MCS	Time (Min.)
S &P 500	LSTM	0.00804	0.00529	0.00000	**37.43**	0.00804	0.00529	0.00000	44.82
	TF	0.00764	0.00521	0.00000	39.92	0.00687	**0.00441**	0.21040	42.78
Dow Jones	LSTM	0.00777	0.00519	0.11955	39.88	0.00840	0.00534	0.00000	43.15
	TF	0.00749	0.00503	0.00000	40.90	**0.00639**	**0.00428**	**1.00000**	41.40
DAX	LSTM	0.00990	0.00700	0.00000	40.27	0.00967	0.00683	0.00000	40.48
	TF	0.00822	0.00609	0.00000	40.82	**0.00712**	**0.00519**	**1.00000**	42.73
Nikkei 225	LSTM	0.01002	0.00732	0.00000	41.30	0.00983	0.00723	0.00000	41.45
	TF	0.00931	0.00701	0.00000	**37.33**	0.00798	0.00602	0.00790	40.60
FTSE 100	LSTM	0.00817	0.00574	0.00000	41.35	0.00787	0.00560	0.00000	43.62
	TF	0.00716	0.00523	0.00000	39.63	0.00635	0.00456	0.00000	42.13
CAC 40	LSTM	0.00980	0.00698	0.00000	42.40	0.00971	0.00692	0.00000	43.53
	TF	0.00872	0.00641	0.00000	40.27	**0.00743**	**0.00547**	**1.00000**	42.08
S &P TSX	LSTM	0.00714	0.00462	0.07790	40.72	0.00710	0.00463	0.20900	41.37
	TF	0.00681	0.00442	0.20900	**36.83**	0.00687	0.00405	0.20900	40.70
S &P IPC	LSTM	0.01029	0.00727	0.00005	41.57	0.00955	0.00681	0.00005	41.53
	TF	0.00854	0.00624	0.00005	**37.02**	0.00807	0.00594	0.00080	41.67
BEL 20	LSTM	0.00826	0.00570	0.00020	40.35	0.00817	0.00561	0.00020	42.88
	TF	0.00797	0.00558	0.00020	38.58	0.00672	0.00464	0.95515	41.27
Hang Seng	LSTM	0.01060	0.00748	0.00000	39.87	0.00984	0.00727	0.00000	40.92
	TF	0.00920	0.00691	0.00000	37.10	0.00785	0.00589	0.01020	40.62

The bold text in the Time column represents the model with the minimum processing time, and the bold text in the RMSE and MAE columns represents the model with the minimum loss function for each stock market index. The bold text in the MCS column represents the superior model for each stock market index

architecture, followed by three dense layers of 64, 8, and 1 units to finalize the structure.

The training epochs for each IMF were standardized across all models, set in a way that allowed each to reach an optimal training point without overfitting. This approach ensured that increasing the number of epochs beyond the established threshold would not yield additional benefits. The specifics of these epoch settings are detailed in Table 2.

To transparently convey the stability of the models, Fig. 8 depicts the training and validation loss curves for IMF 3 of the CAC 40 index, representing a mid-range component in terms of variability and complexity. The model swiftly reduces loss in the initial epochs, indicating rapid learning, and then stabilizes, illustrating robust learning and the absence of overfitting. This convergence pattern is crucial for establishing the model's predictive reliability.

Reinforcing this evidence, Figs. 12 and 13 demonstrate a similar convergence behavior for the most volatile decomposition (IMF 1) and the smoothest one (Residual), respectively. These figures collectively cover the full spectrum of series complexity, ensuring consistent model performance across different volatility levels. The uniformity of these curves validates the chosen hyperparameters for the complex task of financial time series forecasting.

4.5 Results

After training and using the models to predict the returns of the 10 indexes under study, the results presented in the following tables were obtained. Given the large amount of information, these are separated into three tables, the first of which contains the results of the non-decomposed models (Table 6), which, as expected, had the worst performance, where it failed to stand out in any metric, and therefore, not much of its results are detailed in the analysis. The results obtained by the EMD and CEEMDAN decomposition are shown in Tables 3 and 4, respectively.

The three tables detail the results of the **RMSE** and **MAE** metrics obtained by each model for each index. In addition to the confidence value obtained according to the **MCS**, where the closer the value is to 1 the better rather than one nearer to 0. Additionally, the total **training time** of each model (in minutes) is also included, the sum of the training times of each IMF in the decomposition models.

It should be noted that for each index, the best value obtained from RMSE, MAE, MCS and Time is highlighted in bold. EMD and CEEMDAN performed better than non-decomposed models in every aspect except training time since they require less. The reason for including training time is to be able to compare the relationship between precision and adjustment time between the studied architectures.

When first analyzing the general results of the two tables just described, it can be seen that in all cases, the performance metrics and predictive capacity (RMSE, MAE and MCS) converge to the same model, except for the case of the S &P 500, where the CEEMDAN-CNN-TF model performs better in terms of RMSE and MCS, but is in second place in terms of MAE, although the difference between the two (EMD-CNN-TF) is practically null.

There is an enormous difference between the models with and without frequency decomposition. Figure 9 shows the results obtained for the FTSE 100 index, where the models are ordered from left to right grouped according to whether the series is not decomposed, EMD or CEEMDAN, while in each grouping the networks LSTM, TF and later CNN are interspersed. The first four columns correspond to LSTM, TF, CNN-LSTM and CNN-TF, then the next four columns are in the same order but with the network divided by EMD and finally the same four divided by CEEMDAN.

In Fig. 9, we see that the Transformers always outperform the LSTM; the CEEMDAN models are better than their respective EMD counterparts; and, CEEMDAN and EMD are better than the models without decomposing.

Analyzing the results of the models in more detail, we see that in all the cases the Transformer networks outperforms the LSTM. This arises when comparing the 60 TF models with their respective LSTM counterparts, where in 100% of the cases the Transformers are better. When dividing the results, it is observed that in 52 of 60 cases (87%) adding a CNN layer improves the prediction result. Therefore, the models recommended for forecasting financial time series should include one of these architectures. Furthermore, of these 52 cases, 24 of 30 correspond to an improvement in an LSTM network, while in 28 of the

Table 4 CEEMDAN models results

Index	Model	CEEMDAN							
		CEEMDAN -				CEEMDAN-CNN			
		RMSE	MAE	MCS	Time (Min.)	RMSE	MAE	MCS	Time (Min.)
S &P 500	LSTM	0.00802	0.00531	0.00000	42.13	0.00800	0.00529	0.00000	42.78
	TF	0.00737	0.00502	0.00000	40.10	**0.00672**	0.00443	**1.00000**	40.23
Dow Jones	LSTM	0.00778	0.00520	0.00000	40.65	0.00783	0.00520	0.00000	41.12
	TF	0.00726	0.00505	0.00000	**37.67**	0.00657	0.00433	0.16850	40.23
DAX	LSTM	0.00984	0.00696	0.00000	**39.27**	0.00971	0.00684	0.00000	40.45
	TF	0.00863	0.00636	0.00000	39.65	0.00734	0.00538	0.05080	42.32
Nikkei 225	LSTM	0.01000	0.00732	0.00250	37.85	0.00975	0.00719	0.00000	40.35
	TF	0.00928	0.00693	0.00005	38.92	**0.00762**	**0.00575**	**1.00000**	40.12
FTSE 100	LSTM	0.00793	0.00552	0.00000	39.48	0.00760	0.00537	0.00000	42.40
	TF	0.00689	0.00497	0.00000	**39.23**	**0.00593**	**0.00428**	**1.00000**	41.45
CAC 40	LSTM	0.00964	0.00685	0.00000	40.85	0.00958	0.00681	0.00000	43.83
	TF	0.00939	0.00679	0.00000	**35.75**	0.00768	0.00561	0.1064	40.83
S &P TSX	LSTM	0.00709	0.00461	0.20900	39.92	0.00704	0.00460	0.20900	40.52
	TF	0.00689	0.00462	0.20900	37.85	**0.00631**	**0.00395**	**1.00000**	41.22
S &P IPC	LSTM	0.00988	0.00698	0.00005	38.85	0.00945	0.00663	0.00005	43.95
	TF	0.00911	0.00646	0.00005	37.98	**0.00696**	**0.00519**	**1.00000**	39.77
BEL 20	LSTM	0.00825	0.00567	0.15090	**36.92**	0.00812	0.00555	0.00020	40.93
	TF	0.00757	0.00534	0.00020	39.80	**0.00671**	**0.00460**	**1.00000**	40.55
Hang Seng	LSTM	0.01062	0.00752	0.00000	37.43	0.00986	0.00728	0.00000	39.55
	TF	0.00933	0.00705	0.00000	**36.90**	**0.00743**	**0.00558**	**1.00000**	40.52

The bold text in the Time column represents the model with the minimum processing time, and the bold text in the RMSE and MAE columns represents the model with the minimum loss function for each stock market index. The bold text in the MCS column represents the superior model for each stock market index

Table 5 Decompositions test results

Index	Model	EMD				CEEMDAN			
		EMD -		EMD-CNN		CEEMDAN -		CEEMDAN-CNN	
		RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
S &P 500	LSTM	0.00990	0.00766	0.01000	0.00780	0.00993	0.00774	0.01012	0.00790
	TF	0.01058	0.00812	0.00913	0.00718	0.01102	0.00838	**0.00912**	**0.00712**
Dow Jones	LSTM	0.00911	0.00695	0.00922	0.00703	0.00906	0.00701	0.00906	0.00703
	TF	0.00945	0.00747	**0.00866**	**0.00680**	0.00990	0.00754	0.00872	0.00682
DAX	LSTM	0.01427	0.00975	0.01416	0.00994	0.01417	0.00975	0.01444	0.01005
	TF	0.01563	0.01090	0.01316	**0.00912**	0.01525	0.01060	**0.01301**	0.00932
Nikkei 225	LSTM	0.01017	0.00787	0.01009	0.00779	0.00966	0.00768	0.00963	0.00762
	TF	0.01023	0.00792	**0.00936**	**0.00730**	0.00966	0.00762	0.00953	0.00747
FTSE 100	LSTM	0.00949	0.00710	0.00963	0.00719	0.00936	0.00685	0.00932	0.00678
	TF	0.01026	0.00757	0.00882	0.00678	0.00995	0.00750	**0.00847**	**0.00623**
CAC 40	LSTM	0.01118	0.00892	0.01129	0.00905	0.01118	0.00879	**0.01116**	0.00887
	TF	0.01276	0.01013	0.01145	0.00878	0.01226	0.00973	0.01131	**0.00873**
S &P TSX	LSTM	0.00713	0.00547	0.00713	0.00541	0.00714	0.00548	0.00720	0.00547
	TF	0.00733	0.00569	**0.00648**	0.00487	0.00740	0.00573	0.00659	**0.00487**
S &P IPC	LSTM	0.00795	0.00644	0.00786	0.00610	0.00779	0.00626	0.00779	0.00602
	TF	0.00847	0.00667	0.00758	**0.00570**	0.00815	0.00635	**0.00736**	0.00578
BEL 20	LSTM	0.01017	0.00715	0.01095	0.00783	0.01021	0.00715	0.01047	0.00719
	TF	0.01047	0.00753	0.01046	0.00751	0.01129	0.00807	**0.00994**	**0.00700**
Hang Seng	LSTM	**0.01490**	**0.01096**	0.01639	0.01198	0.01537	0.01147	0.01702	0.01229
	TF	0.01583	0.01131	0.01555	0.01125	0.01665	0.01223	0.01599	0.01131

The bold text in the Time column represents the model with the minimum processing time, and the bold text in the RMSE and MAE columns represents the model with the minimum loss function for each stock market index. The bold text in the MCS column represents the superior model for each stock market index

remaining 30 cases the improvement corresponds to a TF network, which could suggest that convolutional networks perform slightly better when combined with transformer networks. Additionally, it is worth mentioning that an attempt was made to make predictions using a CNN architecture by itself. However, it was only possible to obtain a constant line in the mean of the series, so it is reaffirmed that in these cases the CNN architectures contribute to the improvement of the forecasts, but they are not enough to function independently, as mentioned above.

The type of decomposition that provides the best results, as suggested by the work of [51], is the CEEDAN. In 70% of the cases (28 of 40) the CEEMDAN decomposition was better than the EMD. However, the model that benefited the most from this was the LSTM network, with a total of 16 out of 20 cases. Therefore, the result may indicate that although TF networks in general work better with CEEMDAN than with EMD (12 out of 20 cases), these are almost indifferent to the applied decomposition. Additionally, it is noted that if the two proposed models (EMD-CNN-TF and CEEMDAN-CNN-TF) are compared with all the other models under study, in all cases the proposed models outperform the others. For 30% of the cases the model with the best performance is EMD-CNN-TF, while the remaining 70% corresponds to CEEMDAN-CNN-TF. These findings demonstrate the robust efficiency of the proposed models.

While superiority is observed by the CEEMDAN decomposition in most cases, the following observation provides a more nuanced perspective on model performance across different indices. While CEEMDAN decomposition excels particularly for S &P 500, Nikkei 225, FTSE 100, S &P TSX, S &P IPC and BEL 20, EMD stands out for the Dow Jones and DAX. Also, a divergence in better decomposition is noted for the CAC 40 and Hang Seng, given their different effectiveness between LSTM and Transformer models. This suggests that, despite a general trend favoring CEEMDAN, model optimization and decomposition choice must be tailored to each specific market for maximum efficiency.

The better performance of the Transformers networks can be attributed to their high adjustability during training. As an example, Fig. 10 shows the fit of IMF 7 of the Nikkei 225 index, where we see that most of the time the TF network overlaps the real data and only in the most extreme values is there a slight difference. This is indistinct from the training or validation set. In the same way, for the rest of the IMFs, the Transformer networks also have a notable adjustment. However, it is worth mentioning that the adjustment decreases for the IMFs with the highest frequency but this does not amount to something significant in any case.

Although the differences in the total training times are not significant, it can be observed that in only 3 indexes do the LSTM network models manage to obtain the best results, while in the remaining 7 indexes the Transformer models achieve the best times, highlighting that the CEEMDAN decomposition achieves better results for 4 indexes. From these results, it can be concluded that, in general, Transformer networks not only obtain the best results regardless of the metric, but also do so with shorter training time.

It is important to note that differences across markets are evident in the predictive outcomes observed in our study, which encompassed indices from various parts of the globe. This differences in predictions can be primarily attributed to factors such as market volatility, cyclical patterns, and the distinct economic and regulatory landscapes that characterize each region. These differences are further accentuated by the unique liquidity profiles and economic events to different countries or regions, underlining the global diversity of financial markets.

Given these significant variances, it becomes imperative that each predictive model is fine-tuned specifically for the index it is intended to forecast. The distinct characteristics inherent to each market dictate that models are not interchangeable across different indices.

Besides, the evaluation of model generalization across different datasets is a crucial aspect of any machine learning study, that is why Table 5 presents the results obtained for the ten indexes using the test data. That test data incorporated an independent dataset as mentioned previously, ensuring that the models were evaluated on data not seen during the training phase. The Transformer models exhibited robust generalization capabilities across all tested indexes. When examining the RMSE and MAE, the Transformer models maintained superior performance, outperforming LSTM models in 8 of the 10 cases in terms of RMSE, and in 9 of the 10 cases with respect to MAE.

This superior performance, despite the slight drop between the training and test values, indicates a strong predictive fit and an absence of overfitting. The slight variations in the model's performance during testing are within acceptable limits, such as as mentioned in [52], reinforcing the models' ability to generalize and adapt to new data while retaining high predictive accuracy.

In the context of financial markets volatility, the Transformer models capacity to maintain high accuracy is indicative of their learning and generalization mechanisms. The model's proficiency in capturing the dynamics of financial time series, while avoiding overfitting, positions them as a promising tool for forecasting in this domain.

However, there is evidence from studies that support that modifications in the architecture of simple models, such as implementing an Attention Mechanism (AM), can improve the results obtained. Examples of this are given in [53] and [54], where the authors show that models with Attention LSTM and adaptive multi-modal bidirectional long short-term memory network (AM-Bi-LSTM) attention mechanisms, respectively, improve the performance of models based on unmodified LSTM. This suggests the need to experiment with new variants of the proposed model, and with it, further improve performance.

It is worth noting that these findings are supported by the testing across a diverse set of financial indices, which span various market conditions and geographic locations. This diversity in testing scenarios further substantiates the robustness and versatility of the Transformer models suggesting their use across financial forecasting tasks.

Finally, Fig. 11 shows the test results for the Nikkei 225 index, where the Transformer model's predictions closely tracks the actual series movements. This result, along with the summarized test outcomes in Table 5, provide clear evidence of the models effective generalization and their potential as reliable forecasting tools.

5 Conclusion

In this article, a novel model based on Transformer networks with frequency decomposition for forecasting financial time series was proposed. The main motivation was to innovate in neural network architectures because of the need to better capture the movements of stock markets. With this innovation, we sought to improve forecasting performance necessary for good decision making. After modeling 10 stock market indexes from different parts of the world, where the historical returns of each one present heterogeneous characteristics and are affected by diverse and opposing conditions in many cases, robust results were obtained.

The results obtained highlight the efficiency of Transformer models architecture for financial time series forecasts. Through this study we demonstrated the potential that Transform models have to improve the performance of investment strategies and reduce the negative effects of the high volatility inherent in this type of series. Additionally, the superiority of the proposed models over the models based on LSTM networks is demonstrated, and the importance of the decomposition of the series to substantially improve the forecasting power of the models is highlighted.

Finally, this study lays the foundations to start exploiting the possibilities offered by Transformer architectures when applying them to both financial time series and other types of series. For this reason, the results and innovations obtained from this study should encourage its implementation in time series of various areas and thus take advantage of its potential.

Model results

See Table 6.

Table 6 Undecomposed models results

Index	Model	Undecomposed							
		–				CNN			
		RMSE	MAE	MCS	Time (Min.)	RMSE	MAE	MCS	Time (Min.)
S &P 500	LSTM	0.01160	0.00764	0.00000	12.75	0.01150	0.00761	0.00000	13.85
	TF	0.01084	0.00680	0.00000	11.82	0.01010	0.00646	0.00000	13.85
Dow Jones	LSTM	0.01123	0.00741	0.00000	15.88	0.01123	0.00741	0.00000	13.85
	TF	0.01045	0.00669	0.00000	11.95	0.00959	0.00628	0.00000	13.83
DAX	LSTM	0.01420	0.00988	0.00000	11.67	0.01417	0.00987	0.00000	13.72
	TF	0.01177	0.00849	0.00000	10.50	0.01099	0.00740	0.00000	12.82
Nikkei 225	LSTM	0.01455	0.01037	0.00000	12.20	0.01426	0.01028	0.00000	12.42
	TF	0.01215	0.00889	0.00000	10.70	0.01130	0.00797	0.00000	11.72
FTSE 100	LSTM	0.01123	0.00777	0.00000	13.35	0.01126	0.00778	0.00000	12.98
	TF	0.00975	0.00699	0.00000	11.80	0.00932	0.00598	0.00000	12.30
CAC 40	LSTM	0.01376	0.00967	0.00000	13.93	0.01709	0.00992	0.00000	15.82
	TF	0.01205	0.00862	0.00000	11.73	0.01070	0.00743	0.00000	12.73
S &P TSX	LSTM	0.01050	0.00681	0.00000	12.85	0.01049	0.00681	0.00000	12.68
	TF	0.00863	0.00566	0.00000	11.83	0.00871	0.00559	0.00000	12.85
S &P IPC	LSTM	0.01412	0.00980	0.00000	14.22	0.01394	0.00975	0.00000	13.75
	TF	0.00964	0.00702	0.00005	11.17	0.00881	0.00628	0.00080	13.75
BEL 20	LSTM	0.01178	0.00806	0.00000	13.82	0.01180	0.00805	0.00000	13.75
	TF	0.01040	0.00725	0.00000	15.82	0.00982	0.00654	0.00020	12.73
Hang Seng	LSTM	0.01571	0.01074	0.00000	11.70	0.01532	0.01066	0.00000	14.57
	TF	0.01020	0.00734	0.00000	11.75	0.00997	0.00699	0.00000	13.87

CAC 40 loss curves

See Figs. 12 and 13.

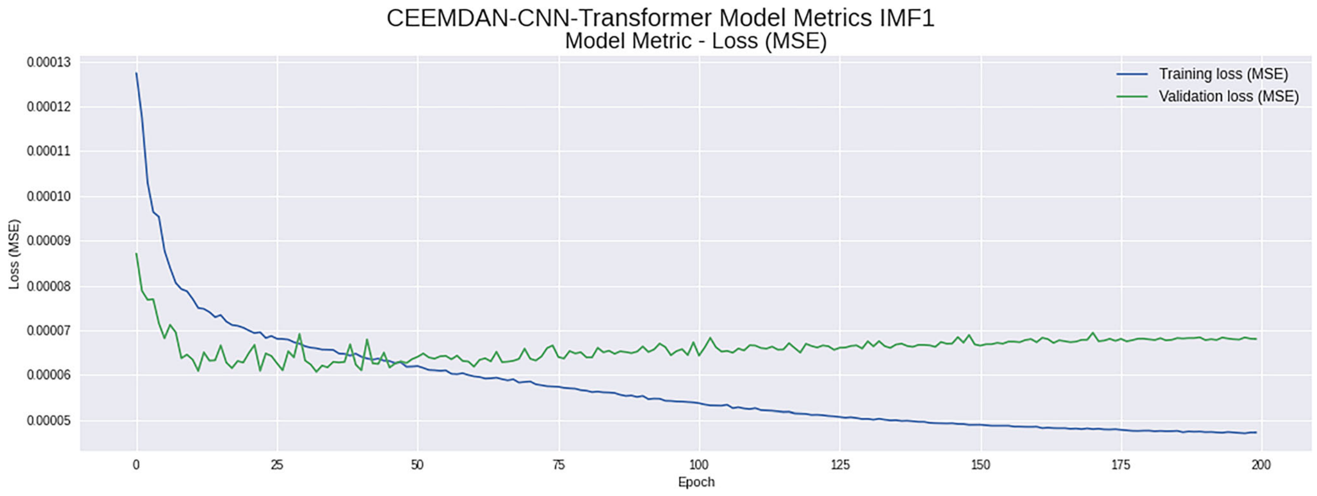


Fig. 12 IMF 1 Loss curve during training

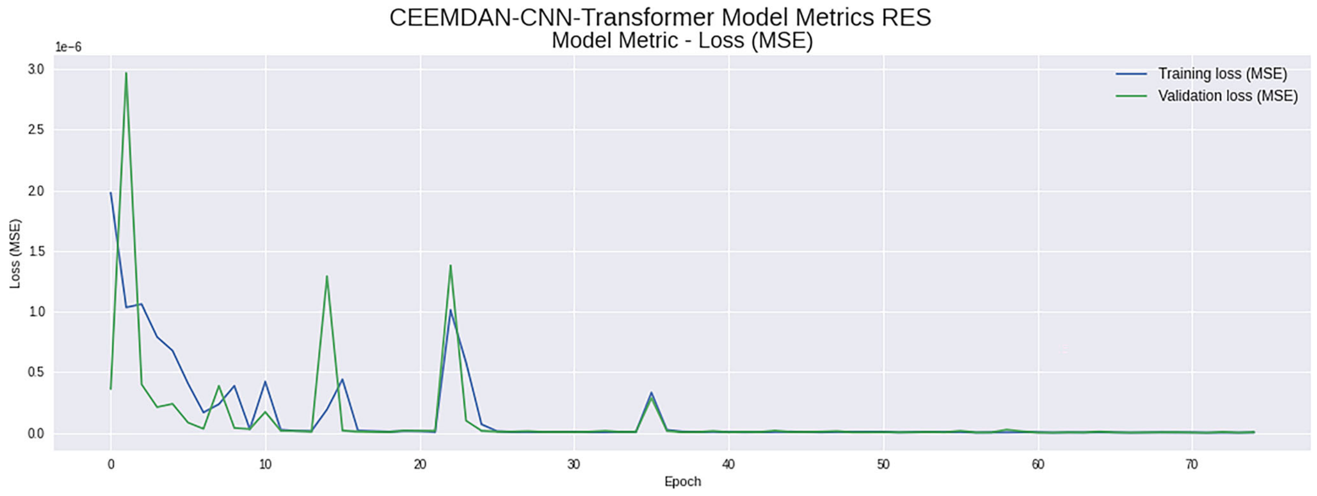


Fig. 13 Residual Loss curve during training

Data availability The data will be made available on reasonable request.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Consent to publish The authors agreed with the content and gave explicit consent to submit the manuscript.

Human and animal rights The study does not involve Human Participants. The study does not involve Animals.

References

1. Rezaei H, Faaljou H, Mansourfar G (2020) Stock price prediction using deep learning and frequency decomposition. Expert Syst Appl 169:114332. https://doi.org/10.1016/j.eswa.2020.114332
2. Huang N, Wu ML, Qu W et al (2003) Application of Hilbert–Huang transform to non-stationary financial time series analysis. Appl Stoch Model Bus Ind 19:245–268. https://doi.org/10.1002/asmb.501

3. Siami Namini S, Tavakoli N, Siami Namin A (2018) A comparison of ARIMA and LSTM in forecasting time series. In: 2018 17th IEEE international conference on machine learning and applications (ICMLA) pp 1394–1401. https://doi.org/10.1109/ICMLA.2018.00227
4. Selvin S, Ravi V, Gopalakrishnan E, et al (2017) Stock price prediction using LSTM, RNN and CNN-sliding window model. In: 2017 International conference on advances in computing, communications and informatics (ICACCI) pp 1643–1647. https://doi.org/10.1109/ICACCI.2017.8126078
5. Rhanoui M, Yousfi S, Mikram M et al (2019) Forecasting financial budget time series ARIMA random walk vs LSTM neural network. IAES Int J Artif Intell 8:317. https://doi.org/10.11591/ijai.v8.i4.pp317-327
6. Güreşen E, Kayakutlu G, Daim T (2011) Using artificial neural network models in stock market index prediction. Expert Syst Appl 38:10389–10397. https://doi.org/10.1016/j.eswa.2011.02.068
7. Roh T (2007) Forecasting the volatility of stock price index. Expert Syst Appl 33:916–922. https://doi.org/10.1016/j.eswa.2006.08.001
8. Zhang Y, Li C, Jiang Y et al (2022) Accurate prediction of water quality in urban drainage network with integrated EMD-LSTM model. J Clean Prod 354:131724. https://doi.org/10.1016/j.jclepro.2022.131724
9. Yang G, Yuan E (2022) Predicting the long-term co2 concentration in classrooms based on the BO-EMD-LSTM model. Build Environ 224:109568. https://doi.org/10.1016/j.buildenv.2022.109568
10. Lin Y, Lin Z, Liao Y et al (2022) Forecasting the realized volatility of stock price index: a hybrid model integrating CEEMDAN and LSTM. Expert Syst Appl 206:117736. https://doi.org/10.1016/j.eswa.2022.117736
11. Ran P, Dong K, Liu X et al (2023) Short-term load forecasting based on CEEMDAN and transformer. Electr Power Syst Res 214:108885. https://doi.org/10.1016/j.epsr.2022.108885
12. Cao J, Li Z, Li J (2018) Financial time series forecasting model based on CEEMDAN and LSTM. Physica A: Stat Mech Appl 519:127–139. https://doi.org/10.1016/j.physa.2018.11.061
13. Topcu I, Saridemir M (2009) Prediction of compressive strength of concrete containing fly ash using artificial neural networks and fuzzy logic. Comput Mater Sci 41:305–311. https://doi.org/10.1016/j.commatsci.2007.04.009
14. Mohtasham Moein M, Saradar A, Rahmati K et al (2022) Predictive models for concrete properties using machine learning and deep learning approaches: a review. J Build Eng 63:105444. https://doi.org/10.1016/j.jobe.2022.105444
15. Sezer O, Gudelek U, Ozbayoglu M (2020) Financial time series forecasting with deep learning: a systematic literature review: 2005–2019. Appl Soft Comput 90:106181. https://doi.org/10.1016/j.asoc.2020.106181
16. Jain AK, Mao J, Mohiuddin KM (1996) Artificial neural networks: a tutorial. Computer 29(3):31–44
17. Elman J (1990) Finding structure in time. Cogn Sci 14:179–211. https://doi.org/10.1016/0364-0213(90)90002-E
18. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 5:157–66. https://doi.org/10.1109/72.279181
19. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–80. https://doi.org/10.1162/neco.1997.9.8.1735
20. Fischer T, Krauss C (2017) Deep learning with long short-term memory networks for financial market predictions. Eur J Oper Res 270:654–669. https://doi.org/10.1016/j.ejor.2017.11.054
21. Junaid T, Sumathi D, Sasikumar AN et al (2022) A comparative analysis of transformer based models for figurative language classification. Comput Electr Eng 101:108051. https://doi.org/10.1016/j.compeleceng.2022.108051
22. Playout C, Duval R, Boucher M et al (2022) Focused attention in transformers for interpretable classification of retinal images. Med Image Anal 82:102608. https://doi.org/10.1016/j.media.2022.102608
23. Vaswani A, Shazeer N, Parmar N, et al (2017) Attention is all you need. Adv Neural Inf Process Syst 30
24. Wolf T, Debut L, Sanh V, et al (2020) Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations, pp 38–45
25. Leow EKW, Nguyen BP, Chua MCH (2021) Robo-advisor using genetic algorithm and BERT sentiments from tweets for hybrid portfolio optimisation. Expert Syst Appl 179:115060
26. de Oliveira Carosia AE, Coelho GP, da Silva AEA (2021) Investment strategies applied to the Brazilian stock market: a methodology based on sentiment analysis with deep learning. Expert Syst Appl 184:115470
27. Gao Y, Zhao C, Sun B et al (2022) Effects of investor sentiment on stock volatility: new evidences from multi-source data in china's green stock markets. Financ Innov 8(1):1–30
28. Huang N, Shen Z, Long S et al (1998) The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. Proce R Soc London Series A: Math Phys Eng Sci 454:903–995. https://doi.org/10.1098/rspa.1998.0193
29. Erdiş A, Bakir M, Jaiteh M (2021) A method for detection of mode-mixing problem. J Appl Stat 48:1–17. https://doi.org/10.1080/02664763.2021.1908969
30. Wu Z, Huang N (2009) Ensemble empirical mode decomposition: a noise-assisted data analysis method. Adv Adapt Data Anal 1:1–41. https://doi.org/10.1142/S1793536909000047
31. Torres ME, Colominas M, Schlotthauer G, et al (2011) Complete ensemble empirical mode decomposition with adaptive noise. In: ICASSP, IEEE international conference on acoustics, speech and signal processing - proceedings pp 4144–4147. https://doi.org/10.1109/ICASSP.2011.5947265
32. Colominas M, Schlotthauer G, Torres ME (2014) Improved complete ensemble EMD: a suitable tool for biomedical signal processing. Biomed Signal Process Control 14:19–29. https://doi.org/10.1016/j.bspc.2014.06.009
33. Hota H, Handa R, Shrivas A (2017) Time series data prediction using sliding window based RBF neural network. Int J Comput Intell Res 13(5):1145–1156
34. Chu CS (1995) Time series segmentation: a sliding window approach. Inf Sci 85:147–173. https://doi.org/10.1016/0020-0255(95)00021-G
35. Yann L, Bottou L, Bengio Y et al (1986) Gradientbased learning applied to document recognition. Proc IEEE 11:2278–2324
36. Guo T, Dong J, Li H, et al (2017) Simple convolutional neural network on image classification. In: 2017 IEEE 2nd international conference on big data analysis (ICBDA) pp 721–724. https://doi.org/10.1109/ICBDA.2017.8078730
37. Kazemi M, Goel R, Eghbali S, et al (2019) Time2vec: Learning a vector representation of time. arXiv preprint arXiv:1907.05321
38. Dubey S, Singh S, Chaudhuri B (2022) Activation functions in deep learning: a comprehensive survey and benchmark. Neurocomputing 503:92–108. https://doi.org/10.1016/j.neucom.2022.06.111
39. Glorot X, Bordes A, Bengio Y (2010) Deep sparse rectifier neural networks. J Mach Learn Res 15
40. Ramachandran P, Zoph B, Le QV (2017) Searching for activation functions. arXiv preprint arXiv:1710.05941
41. Hendrycks D, Gimpel K (2016) Gaussian error linear units (GELUS). arXiv preprint arXiv:1606.08415

42. Lillicrap T, Santoro A, Marris L et al (2020) Backpropagation and the brain. Nat Rev Neurosci 21:335–346. https://doi.org/10.1038/s41583-020-0277-3
43. Wang Q, Ma Y, Zhao K et al (2022) A comprehensive survey of loss functions in machine learning. Annals Data Sci 9:187–212. https://doi.org/10.1007/s40745-020-00253-5
44. Dozat T (2016) Incorporating nesterov momentum into adam
45. Hardt M, Recht B, Singer Y (2016) Train faster, generalize better: stability of stochastic gradient descent. In: International conference on machine learning pp 1225–1234
46. Vani S, Rao T (2019) An experimental approach towards the performance assessment of various optimizers on convolutional neural network. In: 2019 3rd international conference on trends in electronics and informatics (ICOEI) pp 331–336. https://doi.org/10.1109/ICOEI.2019.8862686
47. Hsueh BY, Li W, Wu IC (2019) Stochastic gradient descent with hyperbolic-tangent decay on classification. In: 2019 IEEE winter conference on applications of computer vision (WACV) pp 435–442
48. Hansen P, Nason J, Lunde A (2010) The model confidence set. Econometrica 79:453–497. https://doi.org/10.2139/ssrn.522382
49. Chollet F, et al (2015) Keras. https://github.com/fchollet/keras
50. Koprinkova-Hristova P, Petrova M (1999) Data-scaling problems in neural-network training. Eng Appl Artif Intell 12:281–296. https://doi.org/10.1016/S0952-1976(99)00008-1
51. Liu T, Luo Z, Huang J et al (2018) A comparative study of four kinds of adaptive decomposition algorithms and their applications. Sensors 18:2120. https://doi.org/10.3390/s18072120
52. Ying X (2019) An overview of overfitting and its solutions. J Phys: Conf Ser 1168:022022. https://doi.org/10.1088/1742-6596/1168/2/022022
53. Chen S, Ge L (2019) Exploring the attention mechanism in LSTM-based Hong Kong stock price movement prediction. Quant Finance 19:1–9. https://doi.org/10.1080/14697688.2019.1622287
54. Yu X, Feng Wz, Wang H et al (2020) An attention mechanism and multi-granularity-based bi-LSTM model for Chinese Q &A system. Soft Comput 24:5831–5845. https://doi.org/10.1007/s00500-019-04367-8