Forecasting SST Anomalies: Comparing MLP and LSTM Models Using EOF Time Series

Beatrice Ciancarella
beatrice.ciancarella@studio.unibo.it

Sveva Flocco
sveva.flocco@studio.unibo.it

Arianna Magagna
arianna.magagna@studio.unibo.it

Department of Physics
Science of Climate

This study investigates the efficacy of neural network models in forecasting sea surface temperature (SST) anomalies using Empirical Orthogonal Function (EOF) time series. The research compares two neural network architectures, Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM), using the ERA5 reanalysis SST data (spanning from 1940 to 2022) that had already been processed through EOF analysis to extract principal modes of variability. The preprocessing of the dataset involved normalization, partial autocorrelation function analysis, and seasonal decomposition to enhance model performance. Both neural network models were trained and evaluated on their ability to predict EOFs, thereby capturing the spatial variability in SST anomalies. The results demonstrated the LSTM's superior capability in modeling complex temporal dependencies compared to the MLP. The study concludes with a comparative analysis of the models' performance, emphasizing the practical implications of using neural networks for climate variability forecasting.

Keywords: Deep learning, Neural network, Long short-term memory network, Feedforward network, Multilayer Perceptron, Forecasting, Sea surface temperature, Empirical Orthogonal Function, Teleconnections.

1 Introduction

The purpose of this project is to investigate the efficacy of neural network models in forecasting Empirical Orthogonal Function (EOF) time series bearing information on sea surface temperature (SST) anomalies. This study focuses on comparing two neural network architectures: the Multilayer Perceptron (MLP) and the Long Short-Term Memory (LSTM) network. For studies on climate, numerical based or data-driven methods are still the major tools applied to make predictions. While there are many such data-driven approaches ranging from familiar statistical methods to the newest machine learning and artificial intelligence mechanisms[1]. Thus, the motivation behind this research stems from the need to improve predictive efficiency in climate modeling, which has significant implications for understanding and responding to climate variability and change.

Utilizing the ERA5 reanalysis dataset, which spans from 1940 to modern days, the study leveraged SST data that had undergone EOF analysis to extract the principal modes of variability. The preprocessing steps included normalization, partial autocorrelation function analysis, and seasonal decomposition, aimed at enhancing the performance of the neural network models. The goal was to train and evaluate both the MLP and LSTM models on their ability to predict EOFs, thereby capturing the spatial variability in SST anomalies.

The MLP is a type of feedforward neural network that consists of multiple layers, including input, hidden, and output layers, with no feedback loops. It is widely used for its simplicity and effectiveness in a variety of tasks. In contrast, the LSTM network is a type of recurrent neural network specifically designed to capture long-term dependencies in sequential data, making it particularly suitable for time series forecasting.

The results of this study indicated that the LSTM network outperformed the MLP in modeling complex temporal dependencies, demonstrating superior capability in forecasting EOF time series. This finding highlights the potential of LSTM networks in climate modeling applications, where capturing temporal patterns is crucial. A focus was placed on the application of both models to non-stationary series, comparing the performance of the two methods.

The research concludes with a comparative analysis of the performance of the two models, emphasizing the practical implications of using neural networks for forecasting climate variability. The insights gained from this study contribute to the broader field of climate science by providing a deeper understanding of the capabilities and limitations of different neural network architectures in predicting SST variability.

2 Neural Networks

Artificial neural networks (NN) draw inspiration from biological systems: the brain and the nervous system.[2] The basic building blocks of NNs are processing units called neurons, which are connected to each other through synaptic weights, meaning that each neuron receives weighted information from the others and is then able to produce an output by passing the weighted sum of those input signals through an activation function.

Two main categories of network architecture can be identified, depending on the type of the connections between the neurons:

i. feedforward neural networks (FFNN), if there is no feedback from the outputs of the neurons towards the inputs throughout the network;

ii. recurrent neural networks (RNN), if there exists such a feedback, so that a synaptic connection from the outputs towards the inputs (either their own or the inputs of other neurons) exists.

Usually NNs are arranged in the form of layers and they can be single-layer or multi-layer. In figure 1 a single layer FFNN fully connected is shown. Since there is no computation performed at

Version 1.18, June 25, 2025

the input layer, the latter is not considered in the layer count. Input signals are passed on to the output layer via the weights and the neurons in the output layer compute the output signals.

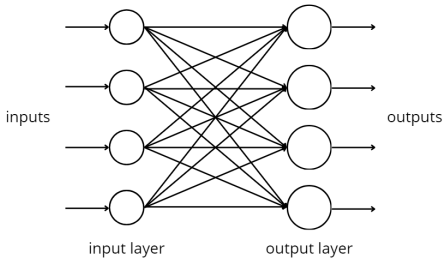

Fig. 1 Single layer FFNN.

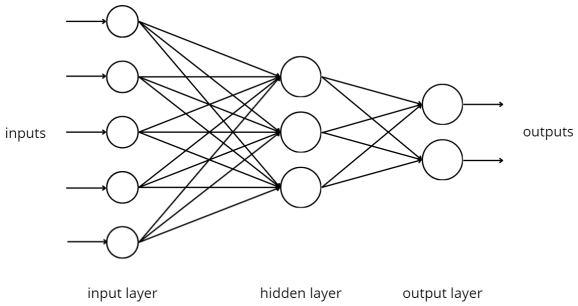

Fig. 2 Multi-layer FFNN.

In figure 2, a multilayer FFNN with one hidden layer. The function of the hidden neurons is to intervene between the external input and the network output, enabling the network to extract higher-order statistics.

In both the cases displayed, the networks are fully connected because every neuron in each layer is connected to every other neuron; if some of the synapic connections were missing, it would then be partially connected.

The most important feature of a NN is its capability to learn. In this context, learning is well defined by Haykin [3]:

Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place.

2.1 Use of NNs for time series forecasting. Time series analysis is a broadly applicable and practically significant problem across various disciplines [4]. It enables the prediction of future values in a series based on its past values, albeit with some degree of error. Literature demonstrates numerous successful applications in diverse fields including weather and climate predictions.

Neural networks have gained significant attention as a viable alternative to conventional statistical forecasting techniques [5]. Consequently, extensive research has been conducted to compare the efficacy of neural networks against traditional forecasting methods. While some studies favor traditional approaches, others show neural networks outperforming them. The inconsistency in results is attributed to varying research methodologies, such as data preprocessing techniques. To resolve the mixed findings, Hill [6] conducted a comprehensive study comparing neural network forecasts with six traditional statistical methods: neural networks consistently outperformed traditional methods across monthly and quarterly time series. Additionally, mathematical theory supports the superiority of neural networks, especially for discontinuous time series and later forecast horizons.

Neural networks have found extensive application in forecasting tasks for several reasons [7]. Firstly, they are nonparametric, meaning they don't rely on assumptions about the underlying model, which is crucial when the data generating process is unknown. Secondly, neural networks are nonlinear models, making them highly flexible and adept at modeling complex real-world phenomena with nonlinear characteristics. Thirdly, theoretical findings establish neural networks as universal functional approximators capable of accurately representing any complex function given a sufficiently large network. The ability to identify the underlying functional relationship between inputs and outputs is vital for forecasting accuracy. These characteristics collectively render neural networks valuable tools for forecasting purposes.

3 SST data

The ERA5 reanalysis[1] dataset provided by the ECMWF is used as the data source for the monthly sea surface temperatures (SST) time series, relating to the years 1940-2022.

The main modes of SST anomalies were derived with the Empirical Orthogonal Function (EOF) method, the amplitude corrisponding to the intensity of the spatial pattern of each mode was then obtained as a time series. The seasonal cycle has been removed[2] and the spatial domain is the entire globe between 35^oN and 35^oS. The data resolution is 0.25 degrees. The technique used to obtain the EOFs was a singular value decomposition (SVD) constructed with numerical libraries from Numpy.

EOFs represent orthogonal modes of variability in the SST dataset. The first EOF explains the largest amount of variance, while the second EOF explains the second-largest amount of variance, and so on.

It is important to note that these spatial patterns do not directly represent the temperature, but rather abstract models of spatial variability in the SST data. So, the forecast does not directly predict specific temperatures, but rather how the spatial patterns identified by the EOFs will change over time.

Making direct predictions in terms of EOFs can be useful for understanding spatial variability in the data, especially in complex or high-dimensional datasets, and highlight teleconnections. Specifically, climatic teleconnections are defined as long-distance connections between weather conditions in different parts of the world: a common and spectacular example of such is El Niño Southern Oscillation (ENSO). Besides this, several other more regional oscillations have been discovered that are perhaps less spectacular than the global ENSO phenomenon but are found to be of considerable significance for describing regional climate anomalies (e.g., Bjerknes, 1964) [9]. These regional oscillations may also be regarded as free internal oscillations in the atmosphere-ocean system. Some attractive methods for determining regional oscillations and associated teleconnection patterns in the atmosphere and ocean are to use EOF analysis as done in this project and to compute correlation maps as illustrated in the conclusions of this report in Figure 31.

Although useful also by itself, for most practical applications, it is more common to use EOFs as part of a broader approach to forecasting specific quantities, such as temperature or atmospheric

[1] Reanalysis data combine a numerical weather prediction (NWP) model with sophisticated data assimilation to retrieve an optimized estimate on the atmospheric state [8].

[2] While neural networks may be capable of automatically modelling seasonality, there is some evidence that deseasonalizing data prior to modelling with a neural network may be advantageous [5].

pressure. This involves converting predictions made in terms of EOFs into predictions in the original quantities using statistical relationships or models.

3.1 Data preprocessing. In this project, we performed several preprocessing steps. The preprocessing is crucial for preparing the data for input into our Long Short-Term Memory (LSTM) neural network and feedforward neural network models. Here, we detail the steps taken.

3.1.1 Normalization. The time series data were normalized to ensure that the input values are on a consistent scale, this may help to increase the performance and speed of convergence of the model. The normalization was performed on the entire dataset collectively, rather than individually for each series, and it scaled the data to the range $[-1, 1]$.

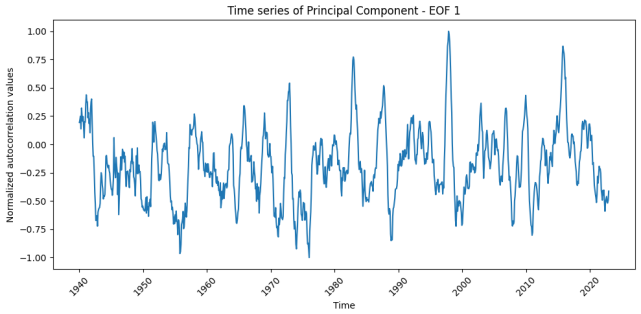

Fig. 3 Normalized time series for EOF1.

3.1.2 Partial Autocorrelation Function (PACF). The PACF was computed and plotted to identify the extent of the correlation between the series and its lagged values. This is important for understanding the intrinsic dependencies in the data and it helps in determining the appropriate number of lagged observations to use as input features in the models.

3.1.3 Seasonal Decomposition. Each series was decomposed into three components: seasonal, trend, and residual. This decomposition helps in understanding the underlying patterns in the data, which is beneficial for model training and evaluation. The trend is the result of a simple running mean with a window of the same length of the period, using linear least-squares extrapolation for both ends. The seasonality computed on the detrended series is inferred by the parameter `period` (12). Then the seasonality and the trend were subtracted from the original series to look at anomalies. Despite the fact that our computed anomalies depict only the first mode of only SST values and for all longitudes, the spikes in the series (specifically around 1982-1983, 1986-1990, 1997-2000, 2010-2011, 2015-2016) seem related to Multivariate ENSO Index (MEI) time series[3], shown in Figure 5.

[3]The bi-monthly MEI.v2 is the time series of the leading combined EOF of five different variables (sea level pressure (SLP), SST, zonal and meridional components of the surface wind, and outgoing longwave radiation (OLR)) over the tropical Pacific basin (30°S-30°N and 100°E-70°W)[10].

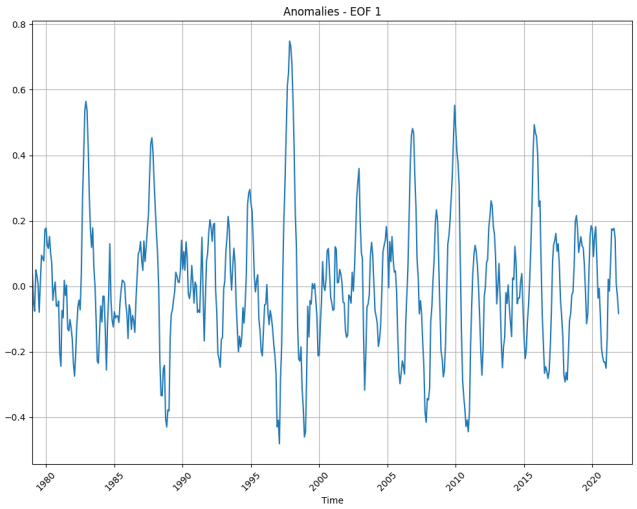

Fig. 4 Seasonal Decomposition for EOF1, 1980-2022 time window.

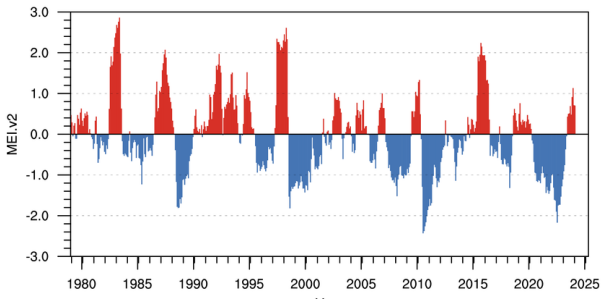

Fig. 5 MEI Version 2 (MEI.v2)[10].

3.1.4 Stationarity. A linear trend was fitted to each series, and the series were detrended by subtracting this trend as displayed in Figure 6 for EOF1 and in Figure 7 for EOF2. The detrended series were then used for stationarity testing. This step is essential because many time series models assume stationarity. The LSTM NN is built to manage both stationary and non stationary time series, whilst the FFNN does not naturally manage non stationarity as well as LSTM NN does. The Augmented Dickey-Fuller (ADF) test was conducted on the trended and detrended series to verify stationarity. The results indicated that the first series is stationary at a 5% significance level (both trended and detrended), while the second series is stationary at a 5% significance level only when detrended. We will keep in mind this specific when looking at the result of both models.

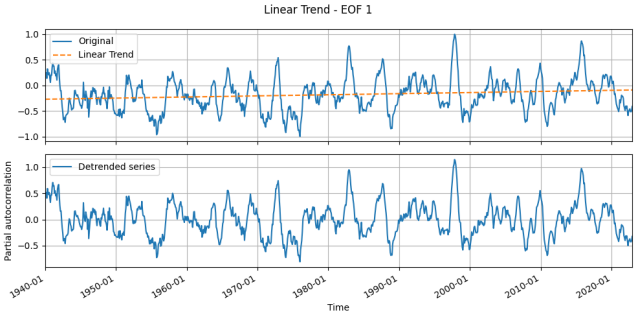

Fig. 6 Linear trend for EOF1.

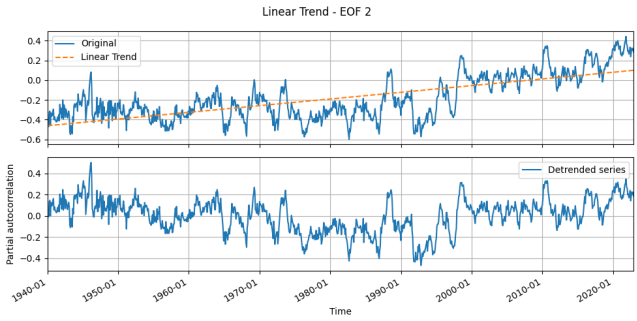

Fig. 7 Linear trend for EOF2.

4 Feed-forward Neural Network

As anticipated in the previous paragraphs, FFNNs are frameworks having only direct connections feeding from neurons of one layer to neurons of the next layer [11].

4.1 MLP method. The multilayer perceptron (MLP) is one type of supervised FFNN and it is a powerful modeling tool that applies a supervised training procedure using examples of data with known outputs [12].

To understand its functioning, an introduction to the one neuron perceptron and a single layer perceptron is due.

4.1.1 One neuron perceptron. A one neuron perceptron is the simplest neural network, which has only one output to which all inputs are connected. Given $i = 0, 1, ..., n$, where n is the number of inputs, let $\{w_i\}$ be the weight of the i-th neuron, so that the output y is obtained through three steps:

i weighting: each input feature is multiplied by its weight $\{x_i w_i\}$

ii sum: the weighted input are summed in a linear combination $(x_0 w_0 + x_1 w_1 + ... + x_n w_n)$

iii transfer: an activation function f is applied to the sum producing the output

$$y = f\left(\sum_{i=0}^{n} w_i x_i\right) = f(W \cdot X)$$

where W and X respectively denote the weight vector and the input vector.

The activation function can be of different nature: commonly used functions are Heaviside, linear and sigmoid.

The training step involves optimizing weights to minimize a cost function, typically the squared error between known and estimated responses. Techniques like gradient descent are used to find the optimal weight vector, allowing the network to generalize to new data.

4.1.2 Single layer perceptron. A single layer perceptron (SLP) is created by connecting multiple perceptrons in parallel, suitable for linearly separable multiclass problems.

4.1.3 Multilayer perceptron. For nonlinearly separable problems, a multi-layer perceptron (MLP) architecture is used, adding layers between the input and output layers to form hidden layers. In an MLP, information flows unidirectionally from the input to the output layer, passing through hidden layers, with each connection having its own weight. Hidden layers typically use a sigmoid activation function, while the output layer can be sigmoid or linear.

4.1.4 Backpropagation algorithm. Training a MLP involves determining the optimal weights to accurately model the desired relationship [13]. This process aims to minimize the network's error, represented on an error surface. Gradient descent, particularly the backpropagation algorithm, is commonly used to find the global minimum of this error surface.

The backpropagation algorithm commences by initializing weights with small random values, mirroring the process of selecting a random point on the error surface. Following this initialization, the algorithm progresses through a set sequence of operations. Initially, it initializes the network weights. Then, it presents the first input vector from the training data to the network and proceeds to propagate this input through the network to obtain an output. Subsequently, the algorithm calculates the error by comparing the actual output to the target output. This computed error is then propagated back through the network. Upon completion of this backpropagation step, the algorithm adjusts the weights to minimize the overall error. This iterative process continues, with subsequent input vectors being presented to the network and the error being reduced further through weight adjustments, until the error reaches a satisfactory threshold.

Effective training typically requires many iterations. The error surface may have multiple minima, and the goal is to avoid local minima. The backpropagation algorithm uses a learning rate to determine the step size during gradient descent and a momentum term to help escape local minima. The learning rate must be balanced to avoid erratic changes or prolonged training, while the momentum term helps the network move past local minima by adding a portion of the previous weight change to the current adjustment. Training stops when the network performs optimally on independent test data, which might not coincide with the minimal network error.

4.2 Model implementation. The complete code for the implementation of the model is provided on GitHub[4]. In this paragraph, the key steps and components involved in building and training the MLP model for time series forecasting are presented.

4.2.1 Data preparation. As described in section 3.1, the data was normalized to ensure that the values lie within a similar range, which is crucial for the effective training of neural networks. The dataset was split into training, validation, and test sets. Specifically, 80% of the data was used for training, 10% for validation, and 10% for testing. A sliding window approach was employed to create sequences of fixed length (12 time steps) from the time series data. Each sequence is used as an input to the model, and the following value in the series is the target output.

4.2.2 Model Architecture. An MLP model was defined with an input layer, one hidden layer, and an output layer. The input layer takes sequences of length 12, the hidden layer contains 64 neurons with ReLU activation[5], and the output layer has a single neuron to predict the next value in the sequence. The model was implemented using PyTorch, a popular deep learning framework.

4.2.3 Training. The training process involved using the Mean Squared Error (MSE) as the loss function and the Adam optimizer for updating the model weights.

During the training, backpropagation algorithm was employed, as prescribed in section 4.1.4. To better clarify, its implementation consisted of the following steps. During the forward pass, the input data is passed through the network layer by layer until an output is produced. This process is handled by the forward function in our model. After obtaining the model's output, the error is calculated

[4] https://github.com/ariannamagagna/EOF-time-series/tree/main

[5] ReLU stands for Rectified Linear Unit: it performs a threshold operation to each input element where values less than zero are set to zero: $\mathrm{ReLU}(x) = (x)^+ = max(0, x)$. For further information, refer to PyTorch website [14].

by comparing the predicted output with the target value. In our case, we use the MSE as the loss function. During the backward pass, the error is propagated backward through the network. This occurs when we call `loss.backward()`. PyTorch automatically computes the gradients of the errors with respect to the weights of the network using automatic differentiation. Finally, the network weights are updated using an optimization algorithm, such as the Adam optimizer in our case. This occurs when we call `optimizer.step()`, which updates the weights based on the calculated gradients.

The model was initially set to train for 100 epochs, but early stopping was implemented to monitor the validation loss and prevent overfitting. To track progress, the validation losses were printed every 10 epochs.

4.2.4 Evaluation. After training, the model's performance was evaluated on the test set. Key metrics calculated include the Mean Squared Error (MSE), Mean Absolute Error (MAE), correlation coefficient, and the R-squared (R^2) value. These metrics provide insights into the accuracy and effectiveness of the model in predicting the time series data.

4.2.5 Forecasting. A forecasting function was implemented to extend the time series by predicting future values. This function uses the last observed sequence to iteratively predict the next value and update the sequence, allowing for multi-step forecasts. For demonstration, the model was used to forecast the next 9 months. The predictions were appended to the original series and plotted to visualize the extended series.

4.3 MLP model results and performance on EOF1. The model was run on the EOF1 time series and early stopping occurred at epoch 37. In figure 8 is presented a plot that illustrates the training and validation losses over the epochs. The training loss represents the error on the training dataset, indicating how well the model is learning the data it was trained on. The validation loss shows the error on a separate validation dataset, providing an indication of how well the model is generalizing to unseen data. The early stopping mechanism halted training at epoch 37 when the validation loss ceased to improve, thereby preventing overfitting and ensuring better model performance on new data.

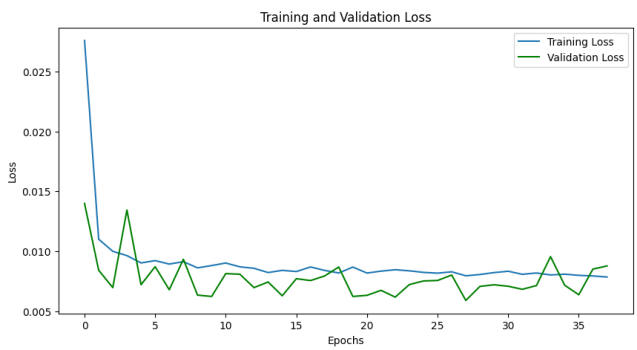

Fig. 8 MLP training and validation losses.

The following plot (Figure 9) compares the original data (in blue) with the predicted data generated by the model (in orange). The original data represents the actual values from the EOF1 time series, while the predicted data shows the values forecasted by our model. The close alignment between the two lines indicates that the model has successfully captured the underlying patterns in the time series, demonstrating its predictive accuracy.

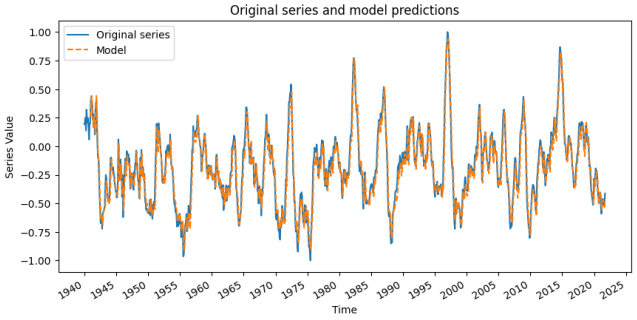

Fig. 9 MLP predictions.

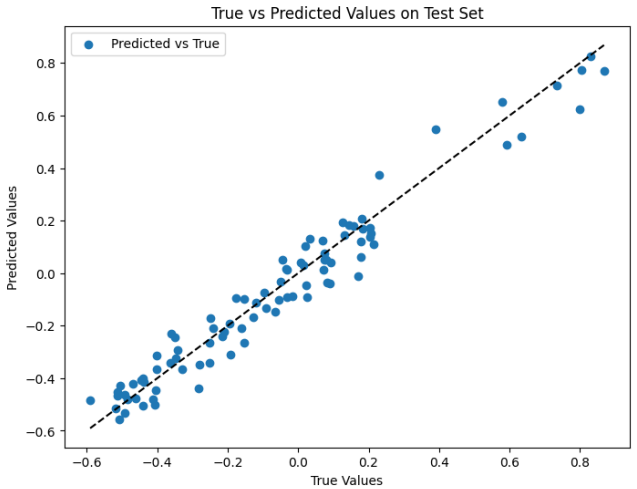

Fig. 10 MLP scatterplot true vs predicted values.

A scatter plot having true values (actual data) on one axis and predicted values (output of the model) on the other axis is also generated. Each dot represents a pair of true and predicted values for a specific data point. By comparing the position of the dots relative to the diagonal line, we can assess how well the model's predictions align with the actual data. A tight clustering of dots around the diagonal line indicates accurate predictions, while deviations from the line suggest discrepancies between the predicted and true values.

To gain a more precise understanding of the model's performance, it was assessed using metrics including mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), correlation, and R-squared (R^2). The computed metrics on the test set were as follows:

MSE	0.006
RMSE	0.074
MAE	0.373
Correlation	0.977
R^2	0.954

Table 1 MLP model evaluation on EOF1 time series.

The MSE and RMSE are both low, suggesting that the model's predictions are relatively accurate and the errors are not heavily influenced by outliers. The MAE of 0.373 indicates that, on average, the model's predictions deviate from the true values by approximately 0.373 units. The high correlation coefficient of 0.977 and the R-squared value of 0.954 indicate a strong linear relationship between the predicted and true

values, suggesting that the model explains approximately 95.4% of the variance in the data. Overall, these metrics suggest that the model performs well in capturing the patterns and trends present in the EOF1 time series, with relatively low errors and a high degree of correlation between predicted and true values.

Finally, a 9-month forecast was executed, and its outcomes are depicted in figure 11. Additionally, a closer examination of the forecasted data is provided in the subsequent figure 12.

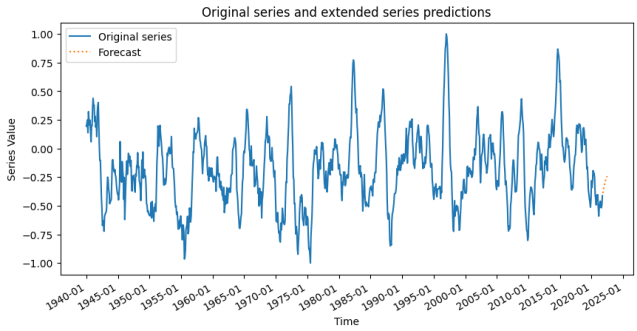

Fig. 11 9 months forecast with MLP.

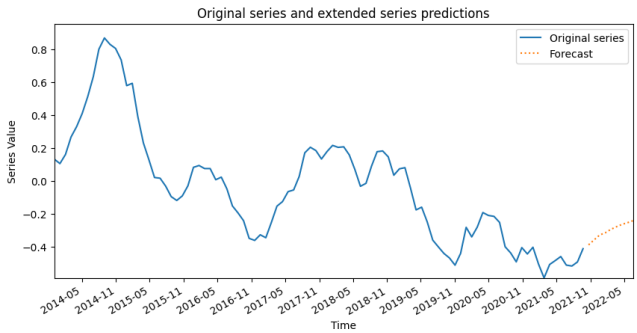

Fig. 12 9 months forecast zoom with MLP.

The forecast for the next 9 months[6] indicates a moderate upward trend in the principal component associated with the primary EOF mode of SST anomalies. This trend suggests a potential stabilization and slight intensification of the corresponding spatial pattern, possibly indicating a recovery from the dip observed in late 2021. Such changes could lead to more stable or slightly warmer sea surface temperatures in the coming months. Given the moderate nature of the forecast, it is crucial to continuously monitor and validate the model's predictions with updated data to ensure accurate forecasting and to understand the broader climatic implications.

4.3.1 Conclusions. The evaluation of the MLP model's performance on the EOF1 time series revealed promising results. The early stopping mechanism at epoch 37 helped prevent overfitting and ensured better generalization to unseen data, as demonstrated by the close alignment between the training and validation losses. The model accurately captured the underlying patterns in the time series, as evidenced by the comparison between the original and predicted data, as well as the scatter plot analysis.
Moreover, the computed evaluation metrics on the test set, including test loss, MSE, RMSE, MAE, correlation, and R-squared, indicated that the model performed well in predicting the target variable. Notably, the model achieved a low test loss and exhibited a strong linear relationship between predicted and true values, suggesting high predictive accuracy and explaining approximately 95.4% of the variance in the data.
Furthermore, the 9-month forecast generated by the MLP model provided valuable insights into future trends, as depicted in the forecast figures. The model's ability to accurately forecast future data points underscores its utility in practical applications.
Overall, these findings highlight the effectiveness of the MLP model in modeling and predicting time series, emphasizing its potential for various forecasting tasks in related domains.

[6]The plot in figures 11 and 12 shows a small jump just because the extended series is plotted from the first forecasted value which is located a month later the last value of the original series.

5 LSTM Neural Network

LSTM is a class of recurrent neural network (RNN), which was developed for learning long-term dependencies. Each neuron in LSTM is a memory cell, which includes three gates, as shown in figure 13: input gate, forget gate, and output gate to control the flow of information between different time steps.

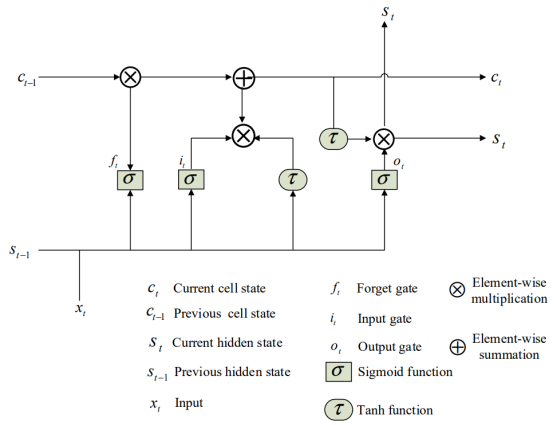

Fig. 13 The structure of a long short-term memory cell.

Unlike conventional NNs, the LSTM cells generate two separate values by a series of activations and operations. One value is the cell state (c_t) that carries information and stores memory in the long term, and the other is the output of the hidden layer (s_t). When the number of inputs increases, the gradients to the first several inputs vanish and become equal to zero. The LSTM can solve this problem by using the internal gates that can add, edit, or remove information in the cell.[15] LSTM method has proven to be effective in dealing with time series data with long-term dependencies.

5.1 LSTM. At each time step, i, the hidden state of the LSTM, h_{i-1}, depends on the hidden state from the previous time step, h_{i-1}, as well as the input in the current time step, x_i. We compactly represent this relationship using the following equation:

$$h_t = LSTM(x_t, h_{t-1})$$

LSTM maintains the long-term history of a time series with a cell state c_t. The information that flows into and out of the cell state are regulated with several gates. First, the input x_t at current time t and the previous hidden state h_{t-1} are used to change the cell memory as follows:

$$\bar{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c)$$

The amount of change to the current cell state is controlled by an input gate i_t while the amount for the cell to maintain its previous state information is regulated by a forget gate f_t:

$$c_t = i_t \circ_{t-1} + f_t \circ \bar{c}_t,$$

where the input and forget gates also depend on x_t and h_{t-1}.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

The cell state provides the information needed to compute the output h_t, whose value is regulated by an output gate.

$$h_t = \sigma \circ tanh(c_t)$$

$$o_t = \sigma(W_0 x_t + U_0 h_{t-1} + b_0)$$

In the above equation $\sigma(\cdot)$ denotes a sigmoid activation function, $tanh(\cdot)$ denotes a hyperbolic tangent function, and $\circ$ denotes the Hadamard product operation.[16]

5.2 Model implementation. In this section, the implementation of the LSTM model for time series forecasting will be illustrated, including a description of the fundamental steps and essential components for constructing and training the model. The complete code is provided on GitHub.

5.2.1 Data preparation. Normalized data was recalled, as illustrated in section 3.1. Using normalized data is essential when using neural networks to ensure uniformity of values. The dataset was split into training, validation, and test sets, with 80% allocated to training, 10% to validation, and 10% to testing. A moving window method was used to generate fixed length sequences (12 time steps) from the time series data. Each set is further divided into sequences of fixed length. Each sequence serves as the input to the model, while the next value in the series serves as the target output. The sequences are converted into tensors,which represent the format required by PyTorch.
Subsequently, DataLoader are created for each set to facilitate data loading in batches during model training and evaluation, with shuffling enabled only for the training and validation sets. For the test set ('test_loader'), 'shuffle' is set to false because typically test data should not be shuffled to ensure evaluation occurs in an orderly and reproducible manner.

5.2.2 Model Architecture. The LSTM model for time series forecasting was defined using the PyTorch library. The LSTM model is defined through a class named *LSTMModel* inheriting functionality from the nn.Module class. This class features an *init method* serving as a constructor and accepting parameters to define the dimensions of the model's layers, including input_*size*, hidden_size, and output_size. The hidden layer has a dimension of 64, enabling the model to capture complex information from the input data. The model's output is also 1, corresponding to the prediction of the next value in the time series. The LSTM layer is defined within the *init method* using *nn.LSTM*, with *input_size* as input and hidden_size as output. A linear layer (*self.linear*) transforms the output of the LSTM layer into the dimension specified by *output_size*. The forward method defines the prediction logic of the model, passing the data through the LSTM layer and the linear layer, returning only the last prediction. The model is initialized with the specified layer dimensions, along with the definition of the loss function (*Mean Squared Error Loss*) and the optimizer (*Adam Optimizer*) with a learning rate of 0.001.

5.2.3 Training. The *train_model* function was utilized to train the model. It took as input the model itself, the training and validation data, the optimizer, the loss function, and the number of epochs. During each epoch, the model underwent training and evaluation on batches of data. For each batch, the model was set to training mode, the gradient was zeroed, data passed through the model to obtain predictions, loss was computed, and gradient back-propagation was performed. Following training on all training batches, the model was evaluated on batches of validation data. At the end of each epoch, the average validation loss was computed and printed if the epoch was a multiple of 10, aiding in tracking progress. Training proceeded for the specified number of epochs, which was 100.

5.2.4 Evaluation. The performance of the model was subsequently evaluated using the *assess_model* function which takes as input the trained model, the test data data loader and a loss function. In this function, the model was set to evaluation mode and model predictions were calculated for each sequence of test data. The predictions were compared with the effective labels to calculate the loss and then the evaluation metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and the R-squared value (R^2) were calculated. model evaluation was then performed on the test data using the *assess_model* function.

5.2.5 Forecasting. First, forecasting were made on the entire series using the hidden state initialized in each step using a function called *predict_whole_series*. This function takes as input the trained LSTM model, the time series, and the length of the sequence used to generate predictions. Before making predictions, the model was put into evaluation mode and an empty list was created to store the predictions. We then iterated over all points in the time series minus the sequence length to extract sequences from the series. These sequences were converted into PyTorch tensors and passed through the LSTM model. The resulting predictions were added to the prediction list which was returned for later use. Subsequently, the forecast was made for a number of future steps, spanning 9 months. The model was set to evaluation mode using *model.eval()* to disable dropout and other regularization techniques. A copy of the original series was then prepared to avoid direct modifications, and the last time window was taken as the starting point for predictions. Predictions were generated within a with *torch.no_grad()* block to disable gradient computation. The hidden state of the LSTM model was initialized, and the model then made a prediction that was added to the extended series. The moving window was updated with the latest elements of the extended series for the next prediction step.

5.3 EOF model results and performance on EOF1. The model was run on the EOF1 time series and early stopping occurred at epoch 14. In figure 14 is presented a plot that illustrates the training and validation losses over the epochs.

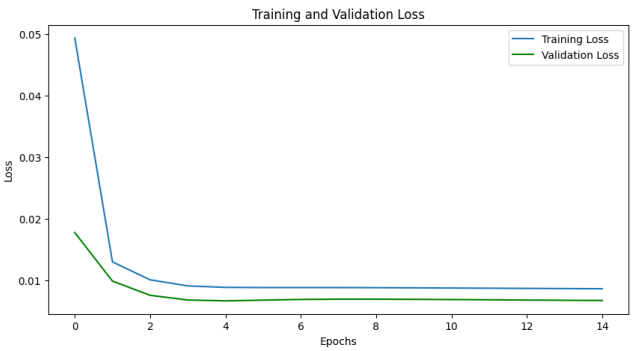

Fig. 14 LSTM training and validation losses.

The comparison between the original data (in blue) and the model-generated predicted data (in orange) is depicted in Figure 15. The close correspondence between these two lines underscores the model's ability to accurately capture the underlying patterns within the time series, thus affirming its predictive accuracy.

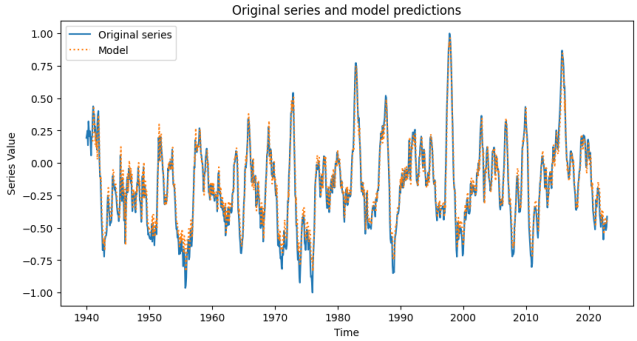

Fig. 15 LSTM predictions.

A scatter plot depicting true values (actual data) on one axis and predicted values (model output) on the other axis has been generated and is displayed in Figure 16.

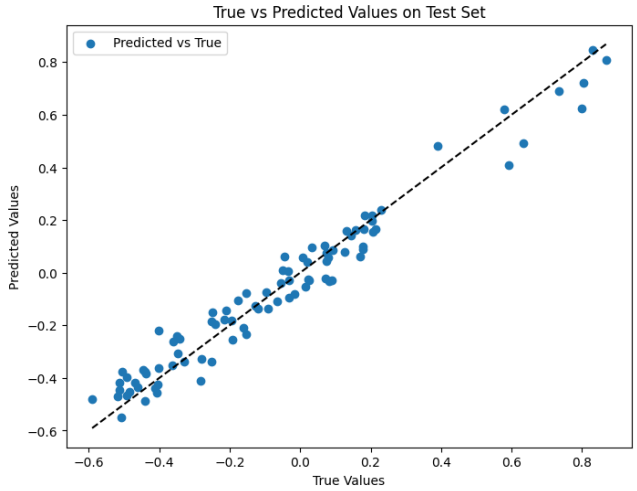

Fig. 16 LSTM scatter plot true vs predicted values.

For a more detailed assessment of the model's performance, we utilized the same evaluation metrics as those employed for the MLP model. These metrics include mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), correlation, and R-squared (R^2). The computed metrics on the test set are as follows:

MSE	0.005
RMSE	0.071
MAE	0.058
Correlation	0.981
R^2	0.958

Table 2 LSTM model evaluation on EOF1 time series

The evaluation metrics for the LSTM model on the EOF1 time series provide a comprehensive assessment of its performance. The test loss, MSE, RMSE, and MAE metrics collectively indicate that the model's predictions closely align with the actual values, with low errors and accurate estimation of the variance of the data. A high correlation coefficient and R-squared value further affirm the model's ability to capture the underlying patterns in the data, explaining approximately 95.8% of the variance. Overall, these metrics underscore the LSTM model's effectiveness in accurately modeling and predicting the EOF1 time series, highlighting its strong performance and reliability for forecasting tasks.

Finally, a 9-month forecast was executed, and its outcomes are depicted in figure 17. Additionally, a closer examination of the forecasted data is provided in the subsequent figure 18.

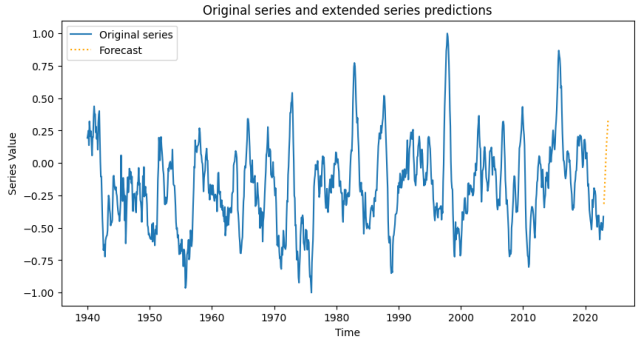

Fig. 17 9 months forecast with LSTM.

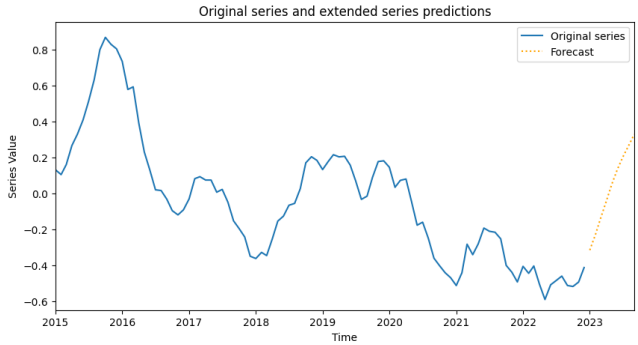

Fig. 18 9 months forecast zoom with LSTM.

The forecast for the next 9 months shows a significant upward trend in the principal component associated with the primary EOF mode of SST anomalies. This trend suggests a strong intensification of the corresponding spatial pattern, potentially indicative of a major climatic event such as an El Niño phase. Such a shift could have wide-ranging impacts on global weather patterns, including altered precipitation and temperature distributions. However, it is crucial to interpret this forecast with caution, considering potential model uncertainties. Ongoing monitoring and model validation are recommended to ensure accurate predictions and to understand the implications of these changes fully.

5.3.1 Conclusions. The LSTM model performed effectively on the EOF1 time series, with early stopping initiated at epoch 14 to prevent overfitting. Visualizations of training and validation losses, original versus predicted data, and a scatter plot of true versus predicted values confirm the model's ability to capture underlying patterns and exhibit predictive accuracy. Assessment metrics, including test loss, MSE, RMSE, MAE, correlation, and R-squared, indicate strong performance, with low errors and high correlation between predicted and true values. The model's efficacy has been employed to make a 9-months forecast.[7] Overall, these results affirm the LSTM model's reliability for accurately modeling and predicting time series.

[7] Both MLP and LSTM models, despite performing well during training and validation, fail in long-term forecasts (beyond 20-25 months), stabilizing at a constant value (0.0). This may be due to memory scalability issues, insufficient model complexity, and prediction drift. Limited data availability (under 1000 points) exacerbates these problems. Including more variables and creating synthetic datasets could address these limitations.

6 Comparative Results

Both models performance on the first EOF was evaluated using several metrics. The LSTM model performed slightly better in terms of MSE, RMSE, and MAE. The R^2 values for both models were close, showing both explained the variance in the data effectively.

	MLP	LSTM
MSE	0.006	0.005
RMSE	0.074	0.071
MAE	0.373	0.058
Correlation	0.977	0.981
R^2	0.954	0.958

Table 3 Models evaluation on EOF1 time series.

Based on these results, the LSTM model appears to have a slight edge in accuracy, while the MLP model is also a strong performer with a high correlation to the actual data. The choice of model may depend on specific use cases and computational efficiency considerations.

6.1 Application on a non-stationary time series. Both models were also applied to the second EOF of our data[8]. As anticipated, this time series did not pass the stationarity check, so the results were expected to be less impressive compared to the first EOF.

6.1.1 MLP. The MLP was run on the original EOF2 time series and early stopping occurred at epoch 16. In the following plot (Figure 19) the training and validation losses can be seen. It is evident how the losses are much higher than they were for the first EOF.

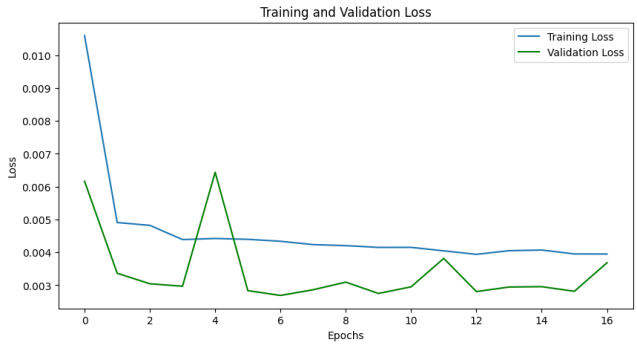

Fig. 19 MLP training and validation losses.

The comparison between the original data (blue) and the model-generated predicted values (orange) is shown in Figure 20.

[8]The entire code, which can be found on GitHub, is structured to facilitate easy generalization to any of the available EOFs using a for loop.

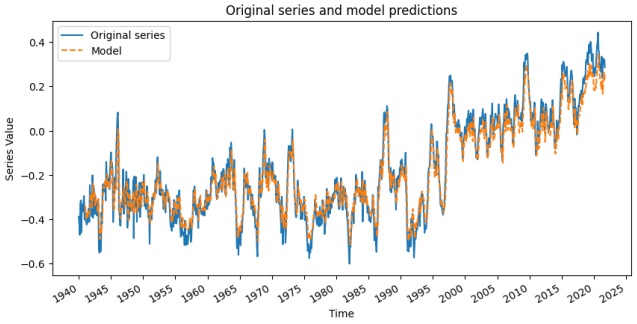

Fig. 20 MLP predictions on EOF2.

In figure 21 the scatter plot for predicted vs true values.

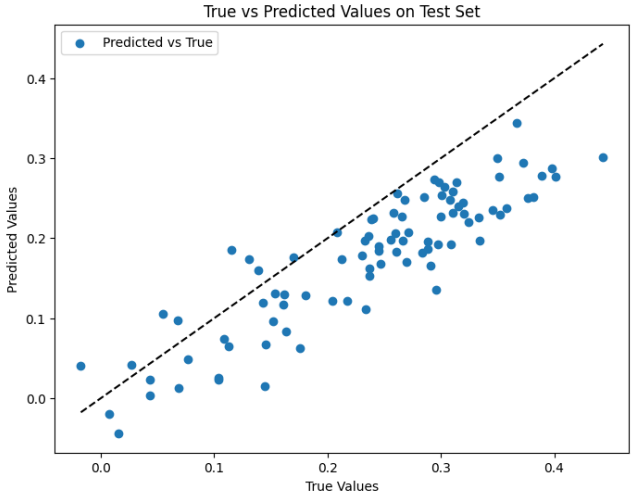

Fig. 21 MLP scatter plot for EOF2.

In this picture we can see how the model tends to underestimate the values which is consistent with a misinterpretation of the positive trend.

6.1.2 LSTM. The LSTM model was run on the original EOF2 time series and early stopping occurred at epoch 37. To obtain the best results the parameters of the model were changed, the hidden layer dimension was decreased to 20 and since it overshooted the peaks we enlarge the moving window sequence lenght up to 18. In the following plot (Figure 22) the training and validation losses can be seen.

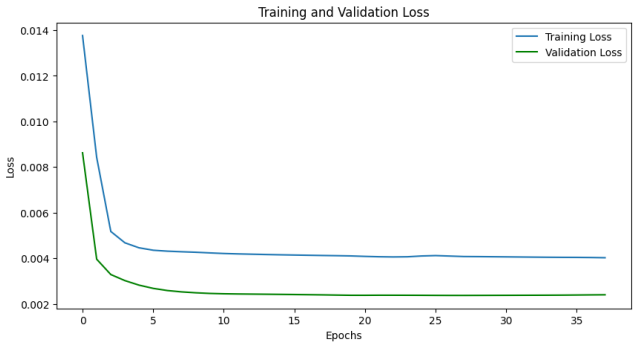

Fig. 22 LSTM training and validation losses.

The comparison between the original data (blue) and the model-generated predicted values (orange) is shown in Figure 23.

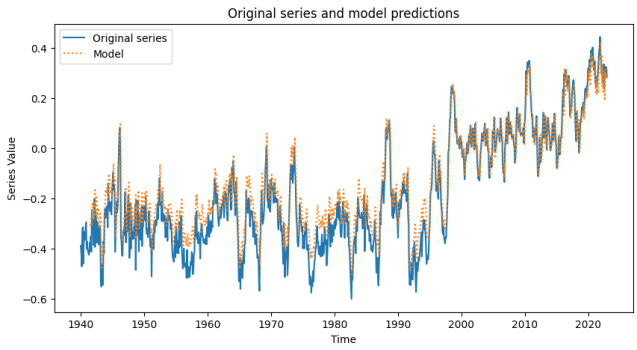

Fig. 23 LSTM predictions on EOF2.

In figure 24 the scatter plot for predicted vs true values.

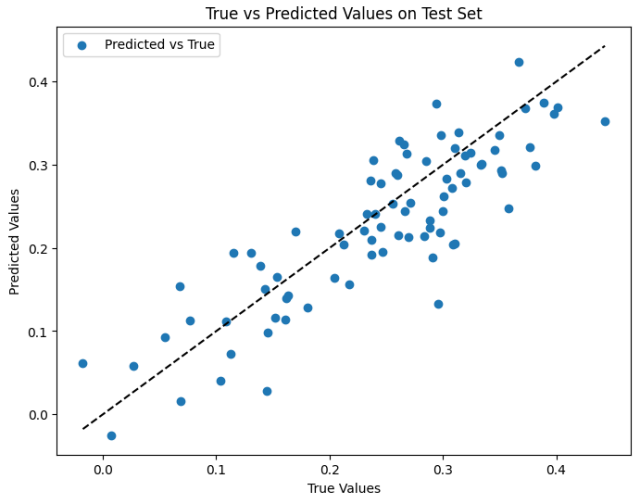

Fig. 24 LSTM scatter plot for EOF2.

In comparison to the scatter plot for the MPL model, the predicted values are now more symmetric with respect to the diagonal line.

6.1.3 Models evaluation on the non-stationary series. For a more quantitative evaluation, here is a table reporting the comparison between the two models.

	MLP	LSTM
MSE	0.006	0.003
RMSE	0.076	0.054
MAE	0.119	0.045
Correlation	0.893	0.859
R^2	0.455	0.698

Table 4 Models evaluations on original EOF2 time series.

The performance comparison between the MLP and LSTM models on a non-stationary time series reveals that the LSTM generally achieves lower overall error metrics, indicating smaller average prediction errors, and explains a larger portion of the variance of the data. Moreover, the LSTM demonstrates more consistent accuracy with fewer large errors, as reflected in its significantly better MAE. Consistently with our expectations, the LSTM NN is better equipped in dealing with non-stationary series. However, it is essential to exercise caution in selecting the model's parameters and tune them to the peculiarities of the series we are dealing with. While the MLP's predictions have a strong linear relationship with the true values, the LSTM's consistent accuracy might make it preferable in scenarios where minimizing individual large errors is critical. This highlights a trade-off between overall fit and consistency, influencing model selection based on specific application requirements.

6.2 Application on de-trended series. To improve the models' outputs, the next step was to remove the linear trend from the data to achieve stationarity. The models were then applied to this detrended series[9], and the trend was subsequently added back to the results. This approach led to improved performance for both models. The following plots display these predictions along with the evaluations.

6.2.1 MLP. The MLP was run on the detrended EOF2 time series and early stopping occurred at epoch 13. In the following plot (Figure 25) the training and validation losses can be seen. It is evident how the losses are much higher than they were for the first EOF.

Fig. 25 MLP training and validation losses for detrended EOF2.

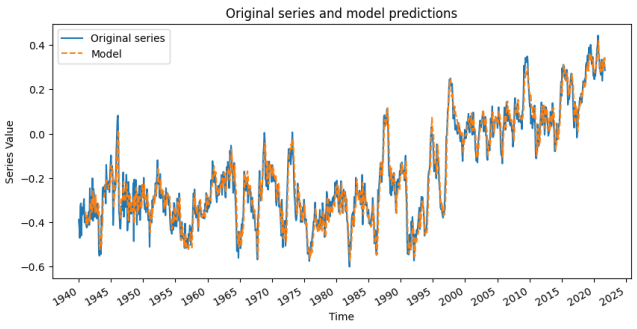

Fig. 26 MLP predictions on detrended EOF2.

The comparison between the original data (blue) and the model-generated predicted values (to which the trend was added)(orange) is shown in Figure 26. It is immediate to notice how the prediction has improved with respect to the one showed in Figure 20, especially towards the end of the time series, where the non stationarity is more evident.

In figure 27 the scatter plot for predicted vs true values.

[9] In this case, LSTM model parameters were set again to the values used for EOF1 predictions.

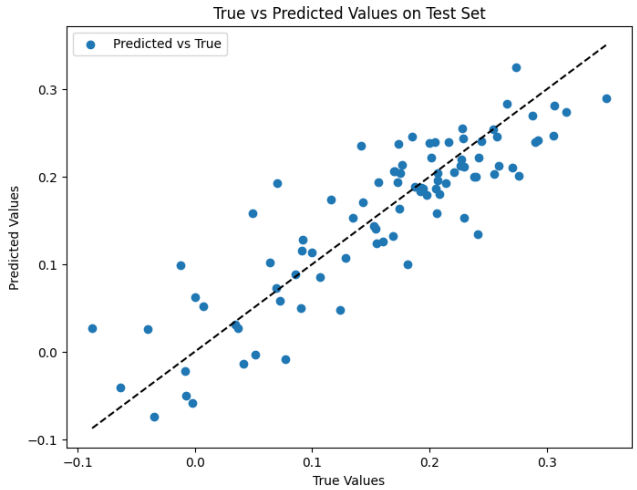

Fig. 27 MLP scatter plot for detrended EOF2.

In this picture we can see how the model improved its estimate of the data: the removal of the trend improved the predicted values, so that there is no systematic underestimation, as was seen in Figure 21.

6.2.2 LSTM. The LSTM model was run on the detrended EOF2 time series and early stopping occurred at epoch 16. In the following plot (Figure 28) the training and validation losses can be seen. It is evident how the losses are much higher than they were for the first EOF.

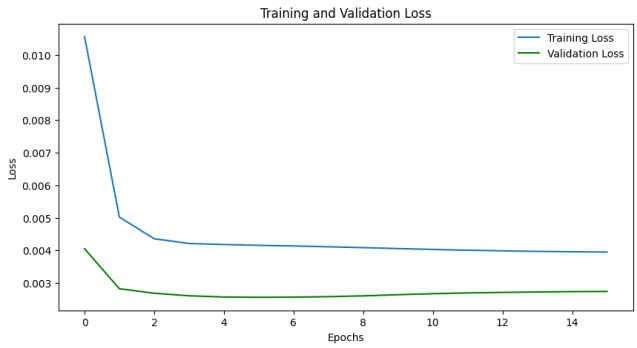

Fig. 28 LSTM training and validation losses for detrended EOF2.

The comparison between the original data (blue) and the model-generated predicted values (to which the trend was added)(orange) is shown in Figure 29. We can notice how the peaks are more contained, especially towards the last part of the time series, with respect to the predictions made in Figure 23.

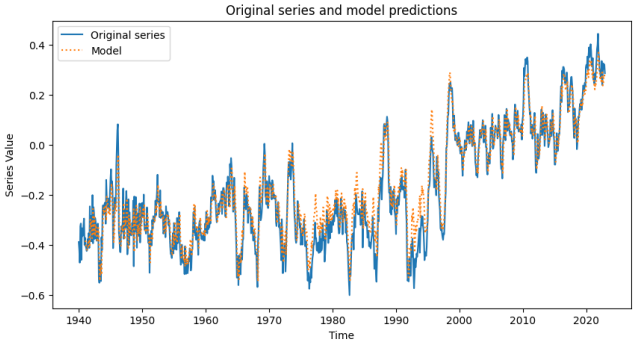

Fig. 29 LSTM predictions on detrended EOF2.

In figure 30 the scatter plot for predicted vs true values.

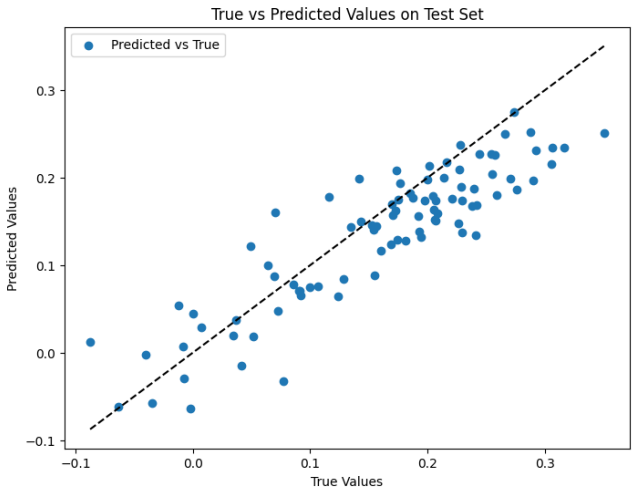

Fig. 30 LSTM scatter plot for detrended EOF2.

Opposite to what happened for the MLP model, the removal of the trend had the effect of creating an underestimate of the values towards the end of the time series; on the other hand, it improved the dispersion, which was higher in Figure 24.

6.2.3 Models evaluation on the de-trended non-stationary series. For a more quantitative evaluation, here is a table reporting the comparison between the two models.

	MLP	LSTM
MSE	0.002	0.002
RMSE	0.047	0.050
MAE	0.104	0.041
Correlation	0.876	0.895
R^2	0.763	0.732

Table 5 Models evaluations on detrended EOF2 time series.

After removing the trend to make the data stationary, applying the models, and then adding the trend back, the performance metrics for the MLP and LSTM models improved, significantly in the MLP case. Both models achieved lower MSE and RMSE, indicating smaller average prediction errors. The MLP maintains a lower overall error, but the gap between the MLP and LSTM has narrowed. The LSTM's MAE remains better, suggesting that it continues to produce more consistently accurate predictions with fewer large errors. Both models now explain a substantial portion of the variance in the data, as indicated by their improved

R^2 values. Overall, making the data stationary before modeling and then reintroducing the trend enhanced the models' predictive performance and their ability to explain the data variance.

7 Conclusions

This study evaluated the performance of neural network models, specifically the Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM) networks, in forecasting sea surface temperature (SST) anomalies using Empirical Orthogonal Function (EOF) time series. To enhance model performance, several processing steps were performed on the data. The processed data was subsequently employed to train and evaluate both neural network models in predicting EOFs, thus enabling them to capture the spatial variability in SST anomalies.
The results demonstrated that the LSTM network outperformed the MLP in modeling complex temporal dependencies. This was true for both stationary and non-stationary time series, with the performance gap widening in favor of the LSTM in the non-stationary case, making it the better choice.

The LSTM model's superior performance in capturing temporal dependencies and its robustness in handling both stationary and non-stationary series make it a valuable tool for climate variability forecasting. Accurate SST anomaly predictions are crucial for understanding and responding to climatic events, such as El Niño, which have significant global impacts.

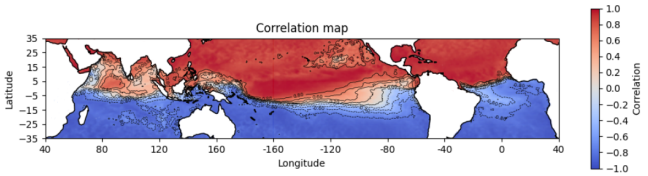

Fig. 31 Forecasted correlation map for EOF1.

Ultimately, one application of this project could involve calculating the correlation between SSTs at each point on the geographical grid and the temporal coefficients of the principal EOF modes. Figure 31 illustrates an example for the forecast calculated from EOF1. Specifically, it was computed using LSTM forecast series and SSTs values from ERA5 reanalysis dataset in the same temporal window 01/2023-09/2023 and geographical region of the forecast. The sctipt for the SST teleconnection forecast can be found in the LSTM script. The correlation highlighted large-scale climatic phenomena, such as the spatial patterns characteristic of ENSO. The first few modes represent the dominant spatial patterns of SST variability, and mapping these allow us to determine how much each point on the grid is correlated with the temporal pattern of the EOF, aiding in visualizing teleconnections. These connections enable us to understand how changes in one region can influence the climate in another non-adjacent distant region. Teleconnections are important because they enhance our understanding and prediction of large-scale climatic phenomena, improve seasonal and long-term weather forecasts, help analyze trends, and better plan and adapt for global climatic impacts like heatwaves, droughts, or heavy rainfall.

References

[1] Sarkar, P. P., Janardhan, P., and Roy, P., 2020, "Prediction of sea surface temperatures using deep learning neural networks," SN Applied Sciences, **2**(8), p. 1458.

[2] Sazlı, M. H., 2006, "A brief review of feed-forward neural networks," Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering, **50**(01).

[3] Haykin, S. and Network, N., 2004, "A comprehensive foundation," Neural networks, **2**(2004), p. 41.

[4] Tealab, A., 2018, "Time series forecasting using artificial neural networks methodologies: A systematic review," Future Computing and Informatics Journal, **3**(2), pp. 334–340.

[5] Nelson, M., Hill, T., Remus, W., and O'Connor, M., 1999, "Time series forecasting using neural networks: Should the data be deseasonalized first?" Journal of forecasting, **18**(5), pp. 359–367.

[6] Hill, T., O'Connor, M., and Remus, W., 1996, "Neural network models for time series forecasts," Management science, **42**(7), pp. 1082–1092.

[7] Zhang, G. P., 2001, "An investigation of neural networks for linear time-series forecasting," Computers & Operations Research, **28**(12), pp. 1183–1202.

[8] Gong, B., Langguth, M., Ji, Y., Mozaffari, A., Stadtler, S., Mache, K., and Schultz, M. G., 2022, "Temperature forecasting by deep learning methods," Geoscientific model development, **15**(23), pp. 8931–8956.

[9] Jose P. Peixoto, A. H. O., 1992, *Physics of climate*, American Institute of Physics.

[10] Physical Sciences Laboratory - NOAA, "Multivariate ENSO Index Version 2 (MEI.v2)," Consulted on the 28th of May 2024, https://psl.noaa.gov/enso/mei/

[11] Khaldi, R., Chiheb, R., and El Afia, A., 2018, "Feedforward and recurrent neural networks for time series forecasting: comparative study," Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications, pp. 1–6.

[12] Taud, H. and Mas, J.-F., 2018, "Multilayer perceptron (MLP)," Geomatic approaches for modeling land change scenarios, pp. 451–455.

[13] Deshpande, R. R., 2012, "On the rainfall time series prediction using multilayer perceptron artificial neural network," International Journal of emerging technology and advanced engineering, **2**(1), pp. 2250–459.

[14] PyTorch, 2024, "ReLU," [Accessed: 2024-05-23], https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html

[15] Tran, T. T. K., Bateni, S. M., Ki, S. J., and Vosoughifar, H., 2021, "A Review of Neural Networks for Air Temperature Forecasting," Water, **13**(9), p. 1294.

[16] Liu, X., Wilson, T., Tan, P.-N., and Luo, L., 2019, "Hierarchical LSTM framework for long-term sea surface temperature forecasting," 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 41–50.

[17] Brezak, D., Bacek, T., Majetic, D., Kasac, J., and Novakovic, B., 2012, "A comparison of feed-forward and recurrent neural networks in time series forecasting," 2012 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), pp. 1–6.

[18] Tang, Z. and Fishwick, P. A., 1993, "Feedforward neural nets as models for time series forecasting," ORSA journal on computing, **5**(4), pp. 374–385.

[19] Hill, T., Marquez, L., O'Connor, M., and Remus, W., 1994, "Artificial neural network models for forecasting and decision making," International journal of forecasting, **10**(1), pp. 5–15.

List of Figures

List of Tables