forecasting

MDPI

Article

Decompose and Conquer: Time Series Forecasting with Multiseasonal Trend Decomposition Using Loess

Amirhossein Sohrabbeig [1], Omid Ardakanian [2] and Petr Musilek [1,*]

[1] Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada; sohrabbe@ualberta.ca
[2] Computing Science, University of Alberta, Edmonton, AB T6G 1H9, Canada; ardakanian@ualberta.ca
* Correspondence: pmusilek@ualberta.ca

Abstract: Over the past few years, there has been growing attention to the Long-Term Time Series Forecasting task and solving its inherent challenges like the non-stationarity of the underlying distribution. Notably, most successful models in this area use decomposition during preprocessing. Yet, much of the recent research has focused on intricate forecasting techniques, often overlooking the critical role of decomposition, which we believe can significantly enhance the performance. Another overlooked aspect is the presence of multiseasonal components in many time series datasets. This study introduced a novel forecasting model that prioritizes multiseasonal trend decomposition, followed by a simple, yet effective forecasting approach. We submit that the right decomposition is paramount. The experimental results from both real-world and synthetic data underscore the efficacy of the proposed model, Decompose&Conquer, for all benchmarks with a great margin, around a 30–50% improvement in the error.

Keywords: time series forecasting; decomposition; multiseasonal

Citation: Sohrabbeig, A.; Ardakanian, O.; Musilek, P. Decompose and Conquer: Time Series Forecasting with Multiseasonal Trend Decomposition Using Loess. *Forecasting* **2023**, *5*, 684–696. https://doi.org/10.3390/forecast5040037

Academic Editor: Sonia Leva

Received: 1 October 2023
Revised: 14 November 2023
Accepted: 10 December 2023
Published: 12 December 2023

1. Introduction

Time series forecasting plays a crucial role in a wide variety of real-world applications, ranging from weather prediction to power system planning and operation to financial risk assessment [1–3]. While the forecast horizon varies in real-world applications, many of them require a long-term forecast. This is challenging due to long-term dependencies, error propagation, and complex dynamics, which are difficult to model [4]. Despite these challenges, the increasing demand for precise and reliable long-term forecasts has drawn attention to this area [4–6].

Real-world time series data frequently display complex traits such as non-stationarity and non-normality, complicating the task of Long-Term Time Series Forecasting (LTSF) [7]. Non-stationarity refers to the evolving nature of the data distribution over time. More precisely, it can be characterized as a violation of the Strict-Sense Stationarity condition, defined by the following equation:

$$F_X(x_{t_1+\tau}, \cdots, x_{t_n+\tau}) = F_X(x_{t_1}, \cdots, x_{t_n}) \text{ for all } \tau,\ t_1, \cdots, t_n \in \mathbb{R} \text{ and } n \in \mathbb{N}, \quad (1)$$

where F_x is the Cumulative Distribution Function (CDF).

Seasonality, deterministic and stochastic trends, heteroscedasticity, and structural breaks are common types of non-stationarity usually observed in time series data. Furthermore, the data distribution can deviate significantly from a normal distribution, manifesting features such as fat tails or exhibiting conditions such as outliers or intermittency within the series [7].

The LTSF problem can be addressed using two approaches [8], illustrated in Figure 1:

1. **Iterative Multi-Step (IMS)** : In this approach, forecasts are made incrementally for each time step within the forecast horizon. Specifically, a single forecast is generated,

after which, the look-back window is shifted forward by one time step to incorporate the newly created prediction. This process is then repeated iteratively.

2. **Direct Multi-Step (DMS):** Contrary to the IMS method, this method generates forecast values for the entire horizon in a single computational pass, thereby producing all the required forecasts simultaneously.

Each approach excels under specific conditions. For instance, the IMS method is particularly advantageous when dealing with shorter forecast horizons, where single-step accuracy is paramount. However, this technique is prone to error accumulation and typically requires a longer inference time compared to its counterpart. As a result, the DMS method is often the go-to choice for LTSF, offering a more-efficient and often more-reliable solution for extended forecast horizons [6]. Forecasting can also be approached through univariate or multivariate methods. In the univariate approach, each time series is modeled and predicted independently, neglecting its interactions with others. On the contrary, the multivariate method accounts for the relationships among different varieties.

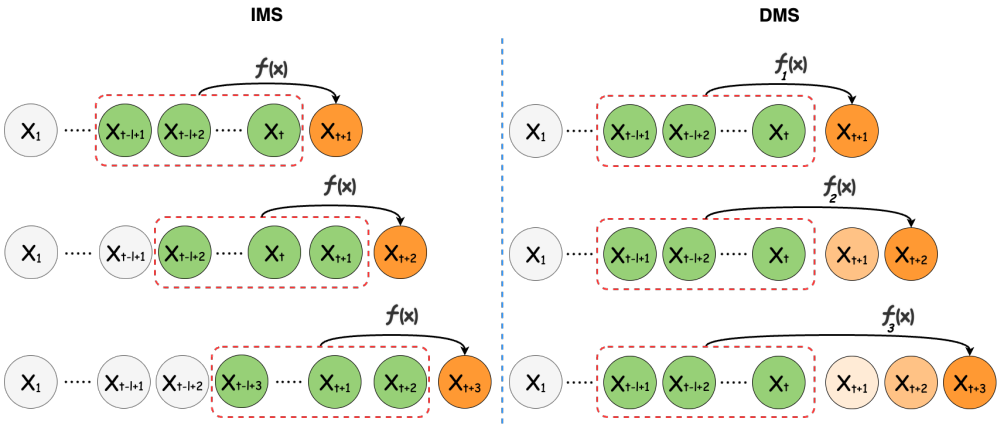

Figure 1. A visual comparison of the IMS and DMS methods: The IMS method (**left**) predicts future values step by step, moving the retrospective window forward one step each time. To determine the complete prediction horizon, this process is repeated. Only one function, $f(x)$, is learned. Conversely, the DMS method (**right**) generates all prediction values at once. A distinct function is required for each prediction value, such as $f_1(x)$, $f_2(x)$, etc.

In today's world, with the vast amounts of data available, there is a growing trend of using Machine Learning and Deep Learning for time series predictions. These advanced models outperform traditional statistical methods in both efficacy and accuracy. Many recent studies advocating deep neural network approaches for LTSF propose increasingly intricate networks, often more elaborate than previous ones, to address the challenges involved. However, these studies often overlook simple, but highly effective techniques, such as decomposing a time series into its constituents as a preprocessing step, as their focus is mainly on the forecasting model.

We advocate for a univariate DMS method to address the intrinsic challenges associated with the LTSF task. This approach employs a multiseasonal decomposition technique, known as the MSTL [9], to break down the time series data into its foundational elements, namely the trend, various seasonal components, and residuals. Each component is then analyzed and predicted separately. Experiments with real-world and synthetic data demonstrated that the proposed method, Decompose & Conquer, outperformed state-of-the-art methods by a substantial margin. We attributed this improvement to the better choice of the decomposition method and to the handling of the extracted components separately. This approach and its name were inspired by the renowned divide-and-conquer algorithm design paradigm to overcome complexity.

The contributions of this paper can be summarized as follows:

- We propose a novel forecasting approach that breaks down time series data into their fundamental components and addresses each component separately.
- We assessed the model's efficiency with real-world time series datasets from various fields, demonstrating the enhanced performance of the proposed method. We further show that the improvement over the state-of-the-art was statistically significant.
- We designed and implemented a synthetic-data-generation process to further evaluate the effectiveness of the proposed model in the presence of different seasonal components.

2. Related Work

Data-driven approaches to time series forecasting offer several benefits, including flexibility, scalability, and robustness to noise, making them valuable tools for understanding and predicting complex systems. Existing data-driven time series forecasting methods can be broadly categorized into classical, Machine Learning, and Deep Learning models.

2.1. Classical Time Series Models

Classical time series forecasting models, such as the Autoregressive Integrated Moving Average (ARIMA) [10,11] and Exponential Smoothing [12,13], have long served as foundational pillars in the field of predictive analytics. These models excel at capturing linear relationships and identifying underlying patterns within the data, such as trends and seasonality. ARIMA, for instance, combines autoregressive, integrated, and moving average components to model a wide range of time series structures. Exponential Smoothing methods, such as Holt–Winters, focus on updating forecast estimates by considering the most-recent observations with exponentially decreasing weights for past data. These classical models lack the complexity to tackle some of the intricacies present in modern datasets, such as the non-stationarity of the underlying distribution and the non-linearity of temporal and spatial relationships.

2.2. Machine Learning Models

Machine-Learning-based time series forecasting models such as Random Forests [14], Support Vector Machines [15], and various types of neural networks have risen to prominence for their ability to handle complex, non-linear relationships within data. Unlike traditional statistical models, which are often constrained by assumptions such as linearity and stationarity, Machine Learning models offer a more-flexible and -adaptive framework to model time series data. However, they lack the interpretability that classical time series models provide.

2.3. Deep Neural Networks

With the rise of deep neural networks, Recurrent Neural Networks (RNNs) have emerged as specialized tools for handling sequential data. Within the RNN family, both LSTM [16] and GRU [17] utilize gated mechanisms to manage the flow of information, thus addressing challenges such as vanishing or exploding gradients. Attention-based RNNs [18] employ temporal attention mechanisms to capture long-term relationships within the data. Nevertheless, recurrent models face limitations in parallelization and struggle with capturing long-term dependencies. Temporal Convolutional Networks (TCNs) [19] present another efficient option for sequence-related tasks, but their capability is restricted by the receptive field of their kernels, making it difficult to grasp long-term dependencies.

2.4. Transformer-Based Architectures

The success of Transformer-based models [20] in various AI tasks, such as natural language processing and computer vision, has led to increased interest in applying these techniques to time series forecasting. This success is largely attributed to the strength of the multi-head self-attention mechanism. The standard Transformer model, however, has certain shortcomings when applied to the LTSF problem, notably the quadratic time/memory

complexity inherent in the original self-attention design and error accumulation from its autoregressive decoder. Informer [21] seeks to mitigate these challenges by introducing an improved Transformer architecture with reduced complexity and adopting the DMS forecasting approach. Autoformer [22] enhances data predictability by implementing a seasonal trend decomposition prior to each neural block, employing a moving average kernel on the input data to separate the trend–cyclical component. Building on Autoformer's decomposition method, FEDformer [5] introduces a frequency-enhanced architecture to capture time series features better. These Transformer-based models were used as baselines in this paper.

A recent study suggested that Transformer-based architectures may not be ideal for LTSF tasks [6]. This is largely attributed to the permutation-invariant characteristic of the self-attention mechanism. Even with positional encoding, this mechanism does not completely preserve temporal information, which is crucial for attaining high accuracy in the LTSF task.

2.5. Time Series Decomposition

Time series decomposition concerns breaking time series data into components such as the trend, seasonality, and remainder. The decomposition methods provide clarity and structure to complex time series data, making it easier to model, interpret, and predict this kind of data.

In decomposition, an additive or multiplicative structure can be assumed for the forming components [23]. Specifically, we can write:

$$y_t = S_t + T_t + R_t, \tag{2}$$

for the additive method, where y_t is the value of the time series at t and S_t, T_t, and R_t are its corresponding seasonality, trend, and remainder components at that time, respectively. Multiplicative methods can also be formulated as follows:

$$y_t = S_t * T_t * R_t. \tag{3}$$

If the size of seasonal changes or deviations around the trend–cycle remain consistent regardless of the time series level, then the additive decomposition is suitable. On the other hand, if these variations correspond proportionally to the time series level, a multiplicative decomposition is better. Multiplicative decompositions are frequently used in economics [23].

The classical way of time series decomposition consists of three main steps [24]. First, the trend component is calculated using the moving average technique and removed from the data by subtraction or division for the additive or multiplicative cases. The seasonal component is then calculated simply by averaging the detrended data and then removed in a similar fashion. What is left is the remainder component. The studies [6,22] followed a similar approach by calculating the trend component, detrending the data, and treating the remainder as the second component. Although the use of this decomposition method has enormously improved previous results, showing the essence of decomposition for time series forecasting, there are still more-complex and -accurate decomposition methods to explore, such as X-13-ARIMA-SEATS [25], X-12-ARIMA [26], and Seasonal Trend decomposition using Loess (STL) [27], to name a few.

While the aforementioned traditional methods are popular in many practical scenarios due to their reliability and effectiveness, they are often only suitable for time series with a singular seasonal pattern. Recently, however, approaches that accommodate multiseasonal trends have emerged, e.g., Seasonal Trend decomposition by Regression (STR) [28], FastRobustSTL [29], and forecasting-based models [30].

One successful member of this family is Multiple Seasonal Trend decomposition using Loess (MSTL) [9]. The MSTL is a versatile and robust method for decomposing a time series into its constituent components, especially when the data exhibit multiseasonal

patterns. Building upon the classical Seasonal Trend decomposition procedure based on Loess (STL), the MSTL extends its capabilities to handle complex time series with more than one seasonal cycle. The method applies a sequence of STL decompositions, each tailored to a specific seasonal frequency, allowing for a more-subtle extraction of seasonal effects of different lengths.

3. Problem Formulation

Let $X = (x_1, x_2, ..., x_T) \in \mathbb{R}^{T \times n}$ be a time series of T observations, each having n dimensions. The goal of time series forecasting is that, given a look-back window of length l, ending at time t: $X_{t,l} = (x_{t-l+1}, ..., x_{t-1}, x_t)$, predict the next h steps of the time series as $f_\theta(X_{t,h}) = (x_{t+1}, ..., x_{t+h})$, where θ denotes the parameter of the forecasting model. We refer to a pair of look-back and forecast windows as a sample.

4. Methodology

In the first step, we employed the MSTL [9] method to decompose time series data. The MSTL is an entirely self-operating additive algorithm for decomposing time series that exhibit several seasonal patterns. It is essentially an enhanced version of the traditional STL [27] decomposition, wherein the STL technique is used iteratively to determine the various seasonal elements present within a time series. The MSTL modifies Equation (2) to encompass several seasonal components within a time series as follows:

$$y_t = S_t^1 + S_t^2 + ... + S_t^n + T_t + R_t, \tag{4}$$

where n is the number of seasonal components. Figure 2 is an example of decomposing a time series into its components.

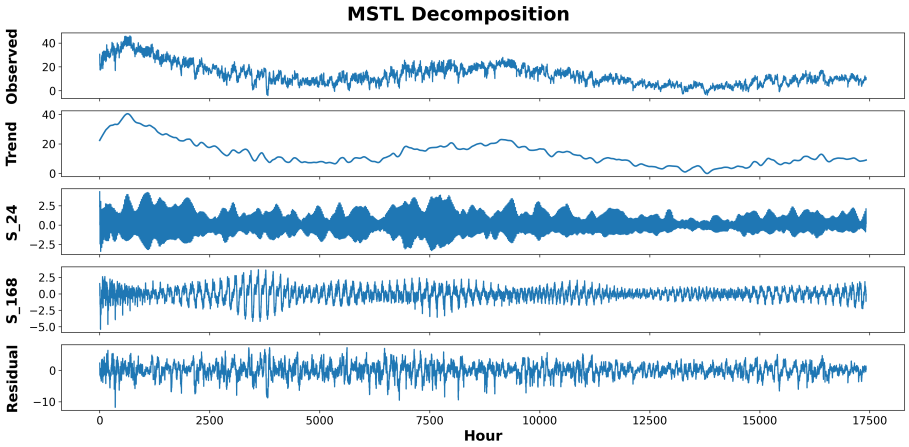

Figure 2. MSTL decomposition of one year of ETTh1 data with 24 and 168 h seasonality periods.

This method excels at deconstructing time series that exhibit multiseasonal trends. The decomposition results in various components that, when added up, recreate the original data. Subsequently, each component undergoes individual training and evaluation in a dedicated module.

To perform the decomposition, first, the whole dataset is split into training, validation, and test sets to prevent data leakage, and then, each set is decomposed independently using the MSTL decomposition method. The MSTL algorithm requires the length of seasonality periods as the input. To determine this, we can draw upon expert insights, i.e., visually inspecting the data by plotting or conducting a frequency domain analysis using the Fast Fourier Transform (FFT). The frequency domain analysis is performed by first detrending the dataset and transforming it to the frequency domain using the FFT, then picking up the dominant frequencies by looking at the power spectrum of the signal. For real datasets, we adopted the former approach, and for synthetic data, we followed the latter.

A solitary linear layer is sufficiently robust to model and forecast time series data provided it has been appropriately decomposed. Thus, we allocated a single linear layer for each component in this study. Upon receiving an input sequence, every linear layer independently generates the complete output sequence in a DMS fashion. These outputs are then aggregated to formulate the final forecast. The overall architecture of the proposed model is depicted in Figure 3.

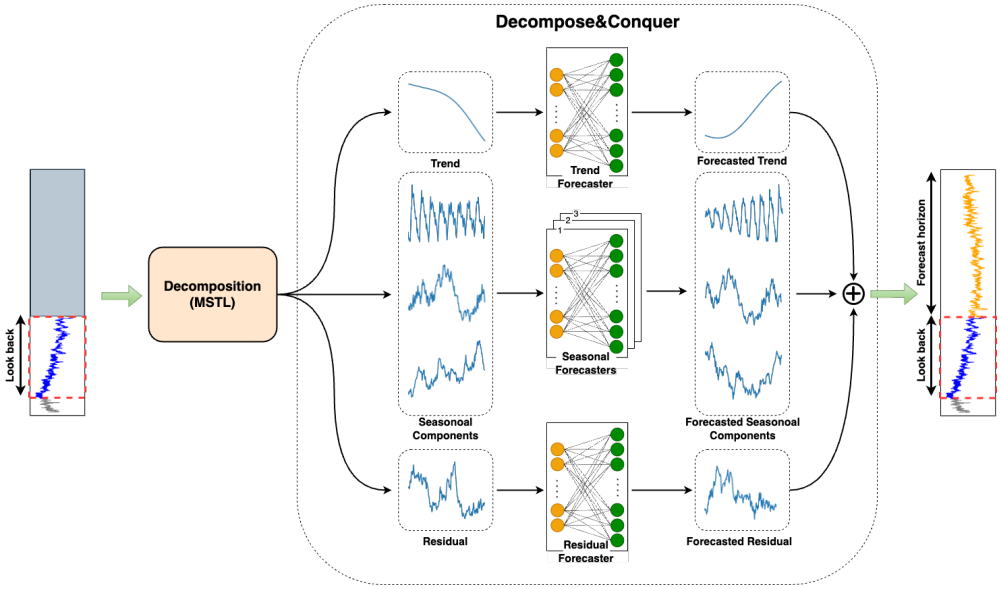

Figure 3. Overall architecture of the model.

4.1. Synthetic Data Generation

To further validate the model's performance, we generated some synthetic data by rendering random trend, seasonality, and noise components and adding them together to make a non-stationary time series. The generative process is expressed in the following equation:

$$y_t = T_t + S_t^{(1)} + S_t^{(2)} + ... + S_t^{(n)} + N_t, \tag{5}$$

where T_t, $S_t^{(i)}$, and N_t are the values of the trend, the i-th seasonal component, and the noise at time t, respectively.

To generate the trend component, we used a linear function of time with a slope of m_t and an intercept of b. The slope, m_t, is an extension of the Gaussian random walk process, in which, at each time, we may take a Gaussian step with a probability of p or stay in the same state with a probability of $1 - p$:

$$\begin{aligned} T_t &= m_t \times t + b, \\ m_t &= m_{t-1} + x_t \times z_t, \\ x_t &\sim \text{Bernoulli}(p), \\ z_t &\sim \text{Normal}(0, \sigma), \\ m_0 &\sim \text{Normal}(\mu, \sigma), \end{aligned} \tag{6}$$

Notice that m_t is a Gaussian random variable itself because it is the sum of independent Gaussian random variables. The parameter p controls the frequency of potential changes in the trend component.

To generate each seasonal component, first, we generated one signal period using a Gaussian random walk process:

$$\begin{aligned} s_t &= s_{t-1} + x_t \\ x_t &\sim \text{Normal}(0, \sigma) \\ s_0 &\sim \text{Normal}(0, \sigma) \end{aligned} \tag{7}$$

We then repeated the generated pattern (S) for the requested length (l). Each repetition is multiplied by the signal's amplitude, coming from another Gaussian random process. The i-th repetition of the period is as follows.

$$\begin{aligned} a_i &= a_{i-1} \times x_i, \\ x_i &\sim \text{Normal}(0, \sigma), \\ S_i &= S \times (1 + a_i). \end{aligned} \tag{8}$$

Lastly, the noise component is generated using a white noise process. An example of a time series generated by the described process is depicted in Figure 4.

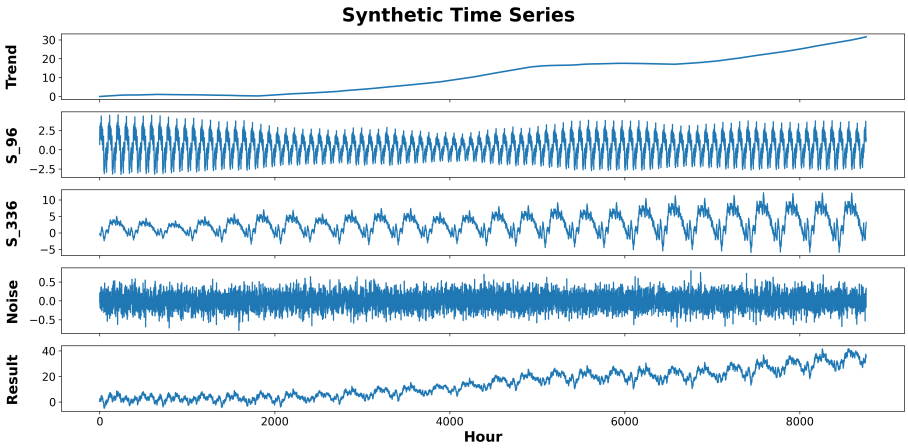

Figure 4. Synthetic data generated for one year with 96- and 336-hour seasonality periods.

4.2. Implementation Details

This study used the L2 loss paired with the ADAM [31] optimization method. The learning rate was initialized at 1e-4, although it was subject to modification based on the ReduceLROnPlateau method. The batch size was configured as 32, and an early stoping criterion was established to stop the training after the evaluation measure (e.g., validation loss) did not improve for three consecutive checks. Each experiment was conducted five times, and the final results were obtained using the average metric value. The training/validation/testing split in all experiments was 2/3, 1/6, and 1/6. The Decompose & Conquer model was implemented using PyTorch [32], wrapped with PyTorch Lightning [33], and trained on a GeForce RTX 3090 24 GB GPU (by Nvidia).

5. Results

5.1. Datasets

The datasets used in this are described as follows. **Electricity** (https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014 (accessed on 9 December 2023)): Records the hourly electricity usage of 321 customers over a span of three years. **ETT** [21] datasets: Include the target value "oil temperature" along with six power load attributes. ETTm1 and ETTm2 data were noted every 15 min, while ETTh1 and ETTh2 data were logged hourly from 2016 to 2018. **Exchange** [34]: Features daily exchange rate data across eight countries from 1990 to 2010. **ILI** (https://pems.dot.ca.gov/ (accessed on 9 December

2023)): Documents the weekly ratio of patients with influenza-like symptoms to the overall patient count. This information was provided by the U.S. Centers for Disease Control and Prevention between 2002 and 2020. **Weather** (https://www.bgc-jena.mpg.de/wetter/ (accessed on 9 December 2023)): Represents a meteorological series with 21 weather metrics collected every ten minutes in 2020 from the Weather Station at the Max Planck Biogeochemistry Institute. **Traffic** (http://pems.dot.ca.gov (accessed on 9 December 2023)): Contains hourly data related to road occupancy rates from various sensors on freeways in the San Francisco Bay area. Provided by the California Department of Transportation, this dataset has 862 attributes and spans from 2016 to 2018. The main characteristics of all the datasets are summarized in Table 1.

Table 1. Statistical details of the real-world datasets used in this study.

Dataset	Electricity	ETTh1 & ETTh2	ETTm1 & ETTm2	Exchange	ILI	Weather	Traffic
Features	321	7	7	8	7	21	862
Length	26,304	17,420	69,680	7,588	966	52,696	17,544
Granularity	1 h	1 h	5 min	1 d	1 wk	10 min	1 h

5.2. Evaluation Metrics

The forecasting errors were evaluated using two common metrics: the mean-squared error:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \tag{9}$$

and the mean absolute error:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|. \tag{10}$$

5.3. Baselines

For comparison, we included three effective Transformer models: Informer [21], Autoformer [22], and FEDformer [5]. Two other baselines that have recently achieved state-of-the-art results are the NLinear and DLinear models [6]. We also used Closest Repeat (Repeat Last) and Average Repeat (Repeat Avg.) as two straightforward baselines replicating the last and average in the look-back period, respectively. In this study, we did not compare the obtained results with classical models, such as ARIMA, SARIMA, etc., as their parameter selection procedure takes too long, which makes the training process extremely slow. Instead, we based our conclusions on the experimental results of these models from other studies [21,22], which demonstrated their poor performance on the LSTF task, and opted not to include them in this study directly.

5.4. Experimental Results on Real Data

Table 2 shows the results obtained using the proposed model and the baselines for all the real datasets included in this study.

While a model's performance is best compared using results from the entire dataset and a single instance is not conclusive proof of superiority, visualizing a few results can provide insights into the differences. To do this, we plot the forecasts made by the proposed model, Decompose & Conquer, with the second-best results achieved from other models using four arbitrarily selected datasets in Figure 5.

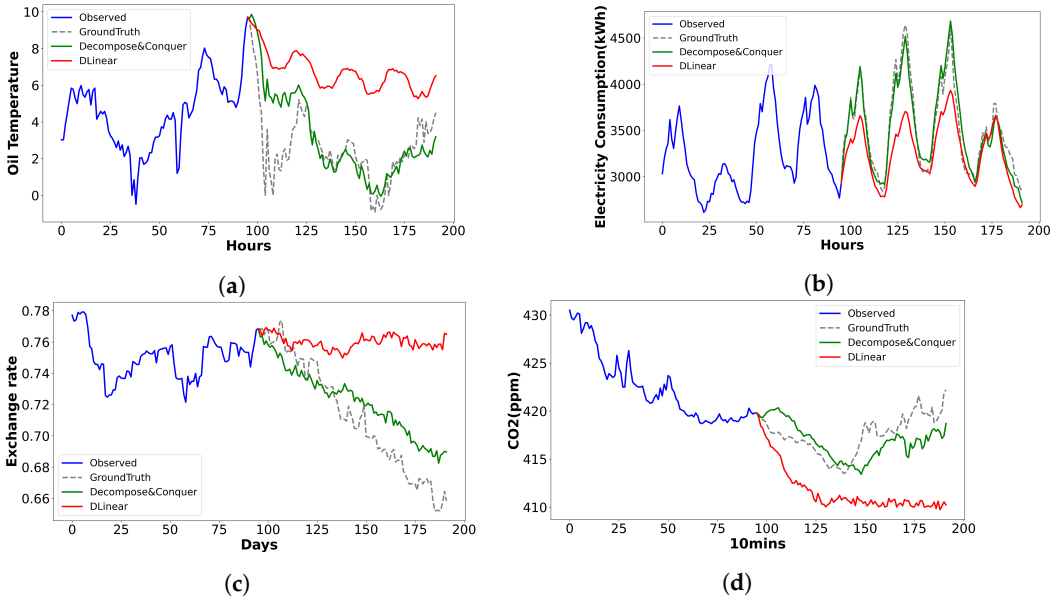

Figure 5. All the forecasts were made for a prediction length of 96 time steps. The comparison was made between Decompose & Conquer and the second-best-performing model, which was the DLinear model, in all situations considered. (**a**) **ETTh1**: Oil temperature forecast of Decompose & Conquer vs. DLinear. (**b**) **Electricity**: Electricity consumption forecast of Decompose & Conquer vs. DLinear. (**c**) **Exchange-rate**: Exchange rate forecast of Decompose & Conquer vs. DLinear. (**d**) **Weather**: CO_2 forecast of method vs. DLinear.

Table 2. Experimental results on real data (column "Transformers" reports the best result from FEDformer [5], Autoformer [22], and Informer [21]). The results were averaged over five runs to eliminate the effect of randomness. The IMP column displays the error reduction of the best model compared to the second-best one. The best results are highlighted in **bold** and the second-best results are underlined.

Dataset	Length	Decompose & Conquer		DLinear		NLinear		Transformers		Repeat Last		Repeat Avg.		IMP	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.190**	**0.292**	0.443	0.453	0.452	0.458	0.494	0.515	1.679	0.878	0.941	0.709	57.1%	35.5%
	192	**0.294**	**0.369**	0.500	0.494	0.513	0.498	0.553	0.552	1.757	0.913	0.989	0.732	41.2%	25.3%
	336	**0.383**	**0.435**	0.550	0.532	0.563	0.531	0.613	0.582	1.806	0.934	1.033	0.750	30.4%	18.2%
	720	**0.556**	**0.549**	0.675	0.611	0.719	0.625	0.752	0.666	1.958	0.992	1.158	0.812	17.6%	10.1%
ETTh2	96	**0.067**	**0.183**	0.164	0.286	0.159	0.280	0.185	0.309	0.259	0.362	0.199	0.320	57.86%	34.6%
	192	**0.104**	**0.223**	0.189	0.304	0.189	0.306	0.205	0.325	0.306	0.393	0.221	0.336	45.0%	26.6%
	336	**0.148**	**0.268**	0.202	0.316	0.207	0.322	0.221	0.344	0.328	0.407	0.232	0.344	26.7%	15.2%
	720	**0.227**	**0.341**	0.315	0.402	0.263	0.365	0.248	0.360	0.376	0.436	0.268	0.367	8.46%	5.3%
ETTm1	96	**0.205**	**0.296**	0.379	0.412	0.392	0.418	0.415	0.452	1.534	0.808	0.862	0.660	45.9%	28.2%
	192	**0.215**	**0.305**	0.432	0.440	0.454	0.452	0.483	0.494	1.589	0.837	0.900	0.678	50.2%	30.7%
	336	**0.232**	**0.321**	0.502	0.477	0.535	0.493	0.537	0.527	1.654	0.867	0.947	0.702	53.8%	32.7%
	720	**0.343**	**0.400**	0.570	0.524	0.615	0.543	0.655	0.597	1.740	0.906	1.011	0.735	39.8%	23.7%
ETTm2	96	**0.061**	**0.172**	0.111	0.230	0.108	0.226	0.114	0.236	0.192	0.308	0.138	0.267	43.5%	24.0%
	192	**0.065**	**0.178**	0.146	0.265	0.133	0.253	0.143	0.266	0.220	0.332	0.158	0.283	51.1%	29.6%
	336	**0.072**	**0.188**	0.176	0.291	0.162	0.280	0.162	0.281	0.251	0.355	0.184	0.304	55.6%	32.9%
	720	**0.121**	**0.242**	0.210	0.323	0.209	0.317	0.216	0.326	0.302	0.390	0.228	0.336	42.1%	23.7%
Electricity	96	**0.073**	**0.170**	0.189	0.273	0.190	0.268	0.195	0.307	1.621	0.954	0.862	0.768	61.4%	36.6%
	192	**0.096**	**0.195**	0.188	0.275	0.189	0.270	0.203	0.314	1.627	0.959	0.864	0.767	48.9%	27.8%
	336	**0.123**	**0.224**	0.200	0.290	0.203	0.284	0.216	0.327	1.644	0.968	0.872	0.769	38.5%	21.1%
	720	**0.170**	**0.271**	0.234	0.323	0.244	0.318	0.252	0.355	1.666	0.980	0.896	0.774	27.4%	14.7%

Table 2. *Cont.*

Dataset	Length	Decompose & Conquer		DLinear		NLinear		Transformers		Repeat Last		Repeat Avg.		IMP	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Exchange	96	**0.043**	**0.152**	0.092	0.218	0.091	0.214	0.133	0.272	0.083	0.203	0.135	0.275	48.2%	25.1%
	192	**0.081**	**0.210**	0.204	0.338	0.181	0.308	0.249	0.380	0.173	0.300	0.235	0.364	53.2%	30.0%
	336	**0.138**	**0.270**	0.303	0.422	0.349	0.431	0.484	0.505	0.334	0.421	0.402	0.476	54.5%	35.8%
	720	**0.465**	**0.507**	0.825	0.692	1.161	0.832	1.279	0.905	0.985	0.769	1.079	0.822	43.6%	26.7%
Weather	96	**0.060**	**0.109**	0.152	0.229	0.157	0.202	0.217	0.310	0.241	0.245	0.201	0.262	60.5%	46.0%
	192	**0.100**	**0.164**	0.195	0.274	0.199	0.244	0.297	0.376	0.290	0.282	0.250	0.297	48.7%	32.7%
	336	**0.165**	**0.233**	0.251	0.317	0.257	0.285	0.310	0.360	0.358	0.328	0.301	0.327	34.3%	18.2%
	720	**0.276**	**0.331**	0.331	0.373	0.336	0.334	0.387	0.400	0.444	0.381	0.366	0.365	16.6%	0.9%
Traffic	96	**0.257**	**0.243**	0.695	0.420	0.678	0.402	0.608	0.373	2.781	1.085	1.448	0.813	57.7%	34.9%
	192	**0.313**	**0.265**	0.641	0.391	0.633	0.380	0.647	0.398	2.824	1.097	1.454	0.816	50.6%	30.2%
	336	**0.364**	**0.287**	0.646	0.393	0.640	0.382	0.683	0.420	2.871	1.107	1.471	0.820	43.1%	24.9%
	720	**0.430**	**0.321**	0.682	0.415	0.674	0.401	0.677	0.417	2.885	1.108	1.482	0.821	36.2%	20.0%
Illness	24	**0.622**	**0.540**	2.589	1.044	2.689	1.096	3.402	1.324	5.547	1.499	5.076	1.758	76.0%	48.3%
	36	**0.889**	**0.650**	2.840	1.106	2.572	1.069	2.813	1.132	7.312	1.833	4.788	1.674	65.4%	39.2%
	48	**1.023**	**0.700**	3.134	1.172	2.654	1.093	2.868	1.166	7.806	1.943	4.491	1.601	61.5%	36.0%
	60	**1.299**	**0.819**	3.286	1.193	2.767	1.111	3.175	1.248	6.917	1.788	4.485	1.575	53.1%	26.3%

5.5. Experimental Results on Synthetic Data

Figure 6 illustrates the variations in the MSE as new seasonal components are introduced through the outlined data-generation process. This chart indicates that the proposed model not only delivered superior performance, but remained robust when additional seasonal components were added.

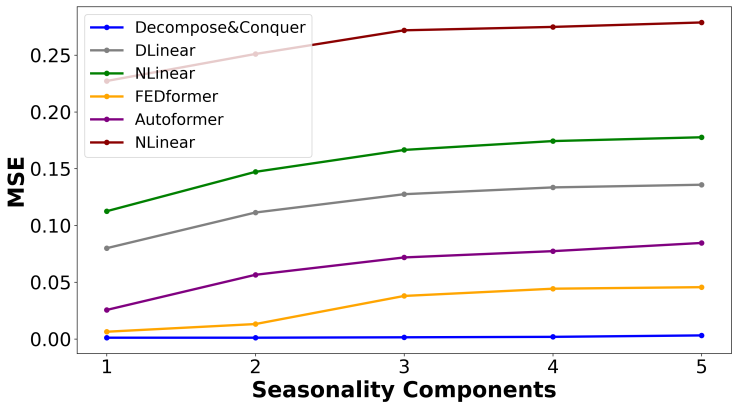

Figure 6. Changes in the MSE by adding seasonality components.

5.6. Statistical Tests

A straightforward method for deciding between two predictions is to opt for the one with the lower error or highest performance according to the evaluation metrics outlined in Section 5.2. However, it is important to recognize if the improvement with respect to the evaluation metrics is meaningful or simply a result of the data points selected in the sample. For this evaluation, we used the Diebold–Mariano test [35], a statistical test designed to understand whether the difference in performance between two forecasting models is statistically significant. It does this by comparing the prediction errors of the two models over a certain period. The test checks the null hypothesis that the two models have the same performance on average, against the alternative that they do not. If the test statistic exceeds a critical value, we reject the null hypothesis, indicating that the difference in the forecast accuracy is statistically significant.

The Diebold–Mariano test is formulated under the null hypothesis $H_0 : E[d_t] = 0$ against the alternative hypothesis $H_a : E[d_t] \neq 0$, where d_t is the loss differential at time

t, defined as $d_t = L(y_t, \hat{y}_t^{(1)}) - L(y_t, \hat{y}_t^{(2)})$, with L being the squared error loss, y_t the true value, and $\hat{y}_t^{(1)}$, $\hat{y}_t^{(2)}$ the forecasts from models 1 and 2, respectively. To calculate the test statistic, the average loss differential $\overline{d} = \frac{1}{T}\sum_{t=1}^{T} d_t$ and its standard error, $SE(\overline{d})$, must be calculated. The Diebold–Mariano test statistic is then given by $DM = \frac{\overline{d}}{SE(\overline{d})}$.

Table 3 shows the DM values and corresponding p-values of the null hypothesis obtained from the Diebold–Mariano test, which compares the forecast accuracy of the Decompose & Conquer model and several established forecasting models, namely DLinear, NLinear, FEDformer, Autoformer, and Informer. It does this by comparing the prediction errors of two models over the period of 96 time steps. The low p-values for the baselines suggest that the difference in the forecast accuracy of the Decompose & Conquer model and that of the baselines is statistically significant. The results highlighted the predominance of the Decompose & Conquer model, especially when compared to the Autoformer and Informer models, where the difference in performance was most pronounced. In this set of tests, the significance level (α) was set to 0.05, and as the underlying distribution for the test statistics can be approximated to the standard normal, the critical values for the DM value were ± 1.96 (the null hypothesis is rejected when $|\text{DM}| > 1.96$.)

Table 3. DM statistics and p-values from a Diebold–Mariano test comparing the results of the Decompose & Conquer model with various baselines.

DLinear		**NLinear**		**FEDformer**		**Autoformer**		**Informer**	
DM	***p*-Value**	**DM**	***p*-Value**	**DM**	***p*-Value**	**DM**	***p*-Value**	**DM**	***p*-Value**
-7.78	4.9×10^{-15}	-5.89	2.1×10^{-9}	-8.72	2.2×10^{-18}	-17.93	8.0×10^{-69}	-21.5	6.2×10^{-96}

6. Conclusions and Future Work

In this article, we demonstrated the effectiveness of a suitable decomposition technique (MSTL) for the time series forecasting task in the presence of single or multiseasonal components. Using a reliable decomposition method, one can achieve surprisingly promising results, even with an uncomplicated network architecture as simple as a linear layer. This was confirmed by the results of the experiments conducted using real-world and synthetic data. The Decompose & Conquer model outperformed all of the latest state-of-the-art models across the benchmark datasets, registering an average enhancement of approximately 43% over the next-best outcomes for the MSE and 24% for the MAE. Additionally, the difference between the accuracy of the proposed model and the baselines was found to be statistically significant.

It is important to highlight that the proposed model demonstrated a distinct advantage in forecasting complex time series data over extended periods, especially when dealing with multiseasonal components. In the context of short-term forecasting, the efficacy of the new model was found to be comparable to that of conventional statistical models.

In this study, the experiments were carried out in the univariate setting. We explored multivariate time series forecasting tasks, but contrary to what may be expected, the use of exogenous variables did not improve the results. This problem can be attributed to the complex dynamics and relationships between variables, which cannot be fully extracted using this network and require more-complicated architectures. Thus, one limitation of the current approach is that it does not harness potential spatial dependencies between different variables, which could provide additional predictive power.

Future work should explore the development of an enhanced model that can capture and leverage these spatial relationships, which could lead to more-precise forecasting across multivariate time series data. Moreover, the robustness of the proposed model to the data quality issues was not investigated in the current work and is deferred to future work. This is a significant consideration, as data quality can substantially impact the performance of predictive models. Issues such as missing values, outliers, and noise in the data can skew the results and lead to inaccurate forecasts. Additionally, integrating exogenous variables

introduces the challenge of dealing with varying scales and distributions, further complicating the model's ability to learn the underlying patterns. Addressing these concerns will require the implementation of preprocessing and adversarial training techniques to ensure that the model is robust and can maintain high performance despite data imperfections. Future research will also need to assess the model's sensitivity to different data quality issues, potentially incorporating anomaly detection and correction mechanisms to enhance the model's resilience and reliability in practical applications.

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

Funding: This research was supported by Huawei Canada.

Data Availability Statement: All datasets used in this study are stored in a Google drive account and are accessible using this link: https://drive.google.com/drive/folders/1FWHcFVzwh_u4oMEXzQtDjSR6Uhg1hfO3?usp=sharing (accessed on 9 December 2023).

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

References

1. Lim, B.; Zohren, S. Time-series forecasting with deep learning: A survey. *Philos. Trans. Royal Soc. A* **2021**, *379*, 20200209. [CrossRef] [PubMed]
2. Mahalakshmi, G.; Sridevi, S.; Rajaram, S. A survey on forecasting of time series data. In Proceedings of the 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16), Kovilpatti, India, 7–9 January 2016; pp. 1–8.
3. Deb, C.; Zhang, F.; Yang, J.; Lee, S.E.; Shah, K.W. A review on time series forecasting techniques for building energy consumption. *Renew. Sustain. Energy Rev.* **2017**, *74*, 902–924. [CrossRef]
4. Zhou, T.; Ma, Z.; Wen, Q.; Sun, L.; Yao, T.; Yin, W.; Jin, R. Film: Frequency improved legendre memory model for long-term time series forecasting. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 12677–12690.
5. Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; Jin, R. Fedformer: Frequency enhanced decomposed Transformer for long-term series forecasting. In Proceedings of the International Conference on Machine Learning, PMLR, Baltimore, MD, USA, 25–27 July 2022; pp. 27268–27286.
6. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are Transformers effective for time series forecasting? In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 11121–11128. [CrossRef]
7. Hewamalage, H.; Ackermann, K.; Bergmeir, C. Forecast evaluation for data scientists: Common pitfalls and best practices. *Data Min. Knowl. Discov.* **2023**, *37*, 788–832. [CrossRef] [PubMed]
8. Taieb, S.B.; Hyndman, R.J. *Recursive and Direct Multi-step Forecasting: The Best of Both Worlds*; Department of Econometrics and Business Statistics, Monash University: Clayton VIC, Australia, 2012; Volume 19.
9. Bandara, K.; Hyndman, R.; Bergmeir, C. MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns. *Int. J. Oper. Res.* **2022**, *1*, 4842. [CrossRef]
10. Box, G.E.; Jenkins, G.M. Some recent advances in forecasting and control. *J. R. Stat. Society. Ser. C (Appl. Stat.)* **1968**, *17*, 91–109. [CrossRef]
11. Box, G.E.; Pierce, D.A. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *J. Am. Stat. Assoc.* **1970**, *65*, 1509–1526. [CrossRef]
12. Holt, C.C. Forecasting seasonals and trends by exponentially weighted moving averages. *Int. J. Forecast.* **2004**, *20*, 5–10. [CrossRef]
13. Winters, P.R. Forecasting sales by exponentially weighted moving averages. *Manag. Sci.* **1960**, *6*, 324–342. [CrossRef]
14. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
15. Boser, B.E.; Guyon, I.M.; Vapnik, V.N. A training algorithm for optimal margin classifiers. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, PA, USA, 27–29 July 1992; pp. 144–152.
16. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
17. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In Proceedings of the NIPS 2014 Workshop on Deep Learning, Montreal, QC, Canada, 13 December 2014.
18. Qin, Y.; Song, D.; Cheng, H.; Cheng, W.; Jiang, G.; Cottrell, G.W. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, VIC, Australia, 19–25 August 2017; pp. 2627–2633.
19. Bai, S.; Kolter, J.Z.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv* **2018**, arXiv:1803.01271.

20. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
21. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient Transformer for long sequence time-series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 11106–11115.
22. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition Transformers with auto-correlation for long-term series forecasting. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22419–22430.
23. Hyndman, R.J.; Athanasopoulos, G. Components of Time Series Data. 2021. Available online: https://otexts.com/fpp3/components.html (accessed on 27 August 2023).
24. Hyndman, R.J.; Athanasopoulos, G. Classical Decomposition. 2021. Available online: https://otexts.com/fpp3/classical-decomposition.html (accessed on 27 August 2023).
25. Bell, W.R.; Hillmer, S.C. Issues involved with the seasonal adjustment of economic time series. *J. Bus. Econ. Stat.* **1984**, *2*, 291–320.
26. Findley, D.F.; Monsell, B.C.; Bell, W.R.; Otto, M.C.; Chen, B.C. New capabilities and methods of the X-12-ARIMA seasonal-adjustment program. *J. Bus. Econ. Stat.* **1998**, *16*, 127–152.
27. Cleveland, R.B.; Cleveland, W.S.; McRae, J.E.; Terpenning, I. STL: A seasonal-trend decomposition. *J. Off. Stat* **1990**, *6*, 3–73.
28. Dokumentov, A.; Hyndman, R.J. *STR: A Seasonal-Trend Decomposition Procedure Based on Regression*; Working Paper 13/15; Monash University: Melbourne, VIC, Australia 2015; Volume 13, pp. 1–32.
29. Wen, Q.; Zhang, Z.; Li, Y.; Sun, L. Fast RobustSTL: Efficient and robust seasonal-trend decomposition for time series with complex patterns. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 6–10 June 2020; pp. 2203–2213.
30. Bandara, K.; Bergmeir, C.; Hewamalage, H. LSTM-MSNet: Leveraging forecasts on sets of related time series with multiple seasonal patterns. *IEEE Trans. Neural Networks Learn. Syst.* **2020**, *32*, 1586–1599. [CrossRef]
31. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
32. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1–10.
33. PyTorch Lightning. 2023. Available online: https://lightning.ai/ (accessed on 19 September 2023).
34. Lai, G.; Chang, W.C.; Yang, Y.; Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In Proceedings of the The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 95–104.
35. Diebold, F.X.; Mariano, R.S. Comparing predictive accuracy. *J. Bus. Econ. Stat.* **2002**, *20*, 134–144. [CrossRef]