AI

MDPI

Article

xLSTMTime: Long-Term Time Series Forecasting with xLSTM

Musleh Alharthi * and Ausif Mahmood

Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, CT 06604, USA; mahmood@bridgeport.edu
* Correspondence: muslehal@my.bridgeport.edu

Abstract: In recent years, transformer-based models have gained prominence in multivariate long-term time series forecasting (LTSF), demonstrating significant advancements despite facing challenges such as high computational demands, difficulty in capturing temporal dynamics, and managing long-term dependencies. The emergence of LTSF-Linear, with its straightforward linear architecture, has notably outperformed transformer-based counterparts, prompting a reevaluation of the transformer's utility in time series forecasting. In response, this paper presents an adaptation of a recent architecture, termed extended LSTM (xLSTM), for LTSF. xLSTM incorporates exponential gating and a revised memory structure with higher capacity that has good potential for LTSF. Our adopted architecture for LTSF, termed xLSTMTime, surpasses current approaches. We compare xLSTMTime's performance against various state-of-the-art models across multiple real-world datasets, demonstrating superior forecasting capabilities. Our findings suggest that refined recurrent architectures can offer competitive alternatives to transformer-based models in LTSF tasks, potentially redefining the landscape of time series forecasting.

Keywords: xLSTM; transformer; linear network; time series forecasting; state-space model

Citation: Alharthi, M.; Mahmood, A. xLSTMTime: Long-Term Time Series Forecasting with xLSTM. *AI* **2024**, *5*, 1482–1495. https://doi.org/10.3390/ai5030071

Academic Editor: Mehdi Neshat

Received: 13 July 2024
Revised: 12 August 2024
Accepted: 20 August 2024
Published: 23 August 2024

1. Introduction

Time series forecasting with Artificial Intelligence has been a prominent research area for many years. Historical data on electricity, traffic, finance, and weather are frequently used to train models for various applications. Some of the earlier techniques in time series forecasting relied on statistics and mathematical models like SARIMA [1–3], and TBATs [4]. These used moving average and seasonal cycles to capture patterns for future prediction. With the advent of machine learning, new approaches using linear regression [5] were developed. Here, a grouping-based quadratic mean loss function is incorporated to improve linear regression performance in time series prediction. Another approach in machine learning is based on an ensemble of decision trees termed XGBoost [6]. This uses Gradient Boosted Decision Trees (GBDTs), where each new tree focuses on correcting the prediction errors of the preceding trees.

Deep learning introduced some newer approaches. Some of the earlier techniques used Recurrent Neural Networks (RNNs) [7] with varying architectures based on Elman RNN, LSTM (Long Short-Term Memory), and GRU (Gated Recurrent Units). These designs capture sequential dependencies and long-term patterns in the data [8]. The recurrent approaches were followed by the use of Convolutional Neural Networks (CNNs) in time series e.g., [9–11]. In recent years, transformer-based architectures have become the most popular approach for Natural Language Processing (NLP). Their success in NLP has given rise to the possibility of using them in other domains such as image processing, speech recognition, as well as time series forecasting. Some of the popular transformer-based approaches to time series include [12–18]. Of these, Informer [12] introduces a ProbSparse self-attention mechanism with distillation techniques for efficient key extraction. Autoformer [13] incorporates decomposition and auto-correlation concepts from classic time

series analysis. FEDformer [14] leverages a Fourier-enhanced structure for linear complexity. One of the recent transformer-based architectures, termed PatchTST [16], breaks down a time series into smaller segments to be used as input tokens for the model. Another recent design, iTransformer [18], independently inverts the embedding of each time series variate. The time points of individual series are embedded into variate tokens, which are utilized by the attention mechanism to capture multivariate correlations. Further, the feed-forward network is applied for each variate token to learn nonlinear representations. While the above-mentioned designs have shown better results than traditional machine learning and statistical approaches, transformers face challenges in time series forecasting due to their difficulty in modeling nonlinear temporal dynamics, order sensitivity, and high computational complexity for long sequences. Noise susceptibility and handling long-term dependencies further complicate their use in fields involving volatile data such as financial forecasting. Different transformer-based designs such as Autoformer, Informer, and FEDformer aim to mitigate the above issues, but often at the cost of some information loss and interpretability. Thus, alternative approaches not relying on the attention mechanism of transformer need to be explored for further advancements in this field.

As a result, some recent time series research tried to explore approaches other than transformer-based designs. These include LTSF-Linear [19], ELM [20], and Timesnet [21]. LTSF-Linear is extremely simple and uses a single linear layer. It outperforms many transformer-based models such as Informer, Autoformer, and FEDformer [12–14] on the popular time series forecasting benchmarks. TimesNet [20] uses modular TimesBlocks and an inception block to transform 1D time series into 2D, effectively handling variations within and across periods for multi-periodic analysis. ELM further improves LTSF-Linear by incorporating dual pipelines with batch normalization and reversible instance normalization. With the recent popularity of state-space approaches [22], some research in time series has explored these ideas and has achieved promising results, e.g., SpaceTime [23] captures autoregressive processes and includes a "closed-loop" variation for extended forecasting.

The success of LTSF-Linear [19] and ELM [20], with straightforward linear architectures, in outperforming more complex transformer-based models has prompted a reevaluation of approaches to time series forecasting. This unexpected outcome challenges the assumption that increasingly sophisticated architectures necessarily lead to better prediction performance. In light of these findings, we propose enhancements to a recently proposed improved LSTM-based architecture, termed xLSTM. We adapt and improve xLSTM for time series forecasting and call our architecture xLSTMTime. This model incorporates exponential gating and a revised memory structure, designed to improve performance and scalability in time series forecasting tasks.

Our contributions include the incorporation of series decomposition, the extraction of the trend and seasonal components in the xLSTM-based architecture, and a post-reversible instance normalization step that has demonstrated good success in improving prior time series forecasting models. We are also the first to propose xLSTM for time series applications and provide an integrated model for it. We compare our xLSTMTime against various state-of-the-art time series prediction models across multiple real-world datasets and demonstrate its superior performance, highlighting the potential of refined recurrent architectures in this domain.

2. Related Work

While LSTM was one of the first popular deep learning approaches with applications to NLP, it was over-shadowed by the success of transformers. Recently, this architecture has been revisited and greatly improved. The revised LSTM is termed xLSTM—Extended Long Short-Term Memory [24]. xLSTM improves the classical LSTM architecture by focusing on the shortcomings of traditional LSTMs. One of the drawbacks of the LSTM design is that it has difficulty in effectively updating previously stored information when more relevant data are encountered in a long sequence. As the training process proceeds, the state updates can cause LSTM to lose important relevant information. LSTM also has limited memory

capacity as the memory cells compress the history information into a scalar value. Further, the architecture of LSTM is inherently sequential as the next state depends on the previous state, thus preventing efficient training of LSTMs on large datasets.

The forgetful aspect of LSTM for important information is improved in xLSTM via exponential gating and modified memory structures. Two variations of the xLSTM architecture have been proposed, termed sLSTM and sLSTM. sLSTM provides improved scalar updates for the state with exponential gating functions for the input and forget gates, which provide better state management than LSTM. sLSTM has multiple memory cells, which enhances its capability in storing complex patterns. This aspect is especially useful in time series prediction.

The second variant of xLSTM, termed mLSTM, supports matrix memory, which can store much larger state information. Very similar to the popular transformer architecture, mLSTM uses the query key values to efficiently store and retrieve relevant information. Eliminating the sequential memory update in mLSTM and the use of matrices make this architecture extremely scalable, efficient, and suitable for large-scale problems. Overall, xLSTM provides better normalization, a revised memory structure featuring scalar and matrix variants, and the integration of residual block backbones for better stability. Before describing our adaptation of xLSTM for time series forecasting, we detail the two architecture variations of xLSTM, i.e., sLSTM and mLSTM, in the following subsections.

2.1. sLSTM

The stabilized Long Short-Term Memory (sLSTM) [24] model is an advanced variant of the traditional LSTM architecture that incorporates exponential gating, memory mixing, and stabilization mechanisms. These enhancements improve the model's ability to make effective storage decisions, handle rare token prediction in NLP, capture complex dependencies, and maintain robustness during training and inference. The equations describing sLSTM are as described in [24]. We present these here for the sake of completeness of our work before describing the adaptation of these to the time series forecasting domain.

The architecture of sLSTM is depicted in Figure 1.

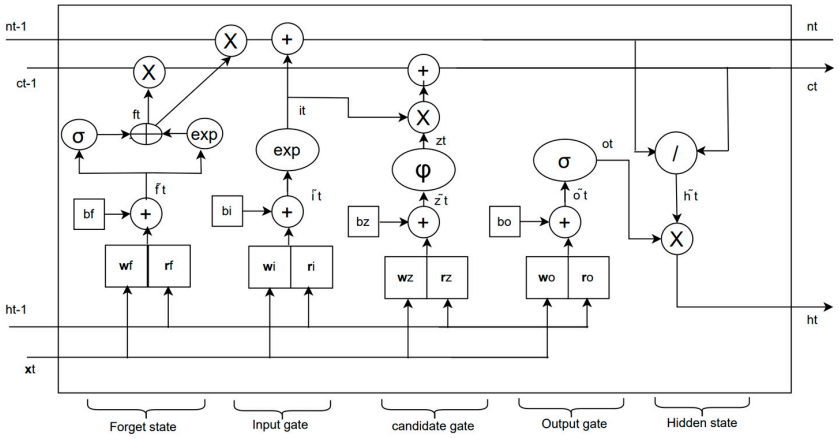


Figure 1. Architecture of sLSTM.

For sLSTM, the recurrent relationship between the input and the state is described as follows:

$$c_t = f_t \, c_{t-1} + i_t z_t \tag{1}$$

where c_t is the cell state at time step t and retains long-term memory of the network, f_t is the forget gate, i_t is the input gate, and z_t controls the amount of input and the previous hidden state h_{t-1} to be added to the cell state, as described below.

$$z_t = \varphi\left(\widetilde{z}_t\right), \qquad \widetilde{\mathcal{Z}}_t = \mathcal{W}_z^{\mathsf{T}} x_t + r_{\mathcal{Z}} h_{t-1} + b_z \tag{2}$$

In the above equations, x_t is the input vector, φ is an activation function, $\mathcal{W}_z^\intercal$ is the weight matrix, r_z is the recurrent weight matrix, and b_z represents bias.

The model also uses a normalization state:

$$n_t = f_t \, n_{t-1} + i_t \tag{3}$$

where n_t is the normalized state at time step t. It helps in normalizing the cell state updates. Hidden state h_t is used for recurrent connections:

$$h_t = o_t \widetilde{h}_t, \quad \widetilde{h}_t = c_t / n_t \tag{4}$$

where o_t is the output gate. The input gate i_t controls the extent to which new information is added to the cell state:

$$i_t = \exp(\widetilde{i}_t), \qquad \widetilde{i}_t = \mathcal{W}_z^\intercal \, x_t + r_i h_{t-1} + b_i \tag{5}$$

Similarly the forget gate f_t controls the extent to which the previous cell state c_{t-1} is retained.

$$f_t = \sigma(\widetilde{f}_t) \; OR \; exp(\widetilde{f}_t) \qquad \widetilde{f}_t = \mathcal{W}_f^\intercal \, x_t + r_f h_{t-1} + b_f \tag{6}$$

The output gate o_t controls the flow of information from the cell state to the hidden state:

$$o_t = \sigma(\widetilde{o}_t), \qquad \widetilde{o}_t = \mathcal{W}_o^\intercal \, \mathcal{X}_t + r_o h_{t-1} + b_o \tag{7}$$

where $\mathcal{W}_o^\intercal$ is the weight matrix that is applied to the current input x_t, r_o is the recurrent weight matrix for the output gate that is applied to the previous hidden state h_{t-1}, and b_o is the bias term for the output gate.

To provide numerical stability for exponential gates, the forget and input gates are combined into another state m_t:

$$m_t = \max(\log(f_t) + m_{t-1}, \log(i_t)) \tag{8}$$

$$i'_t = \exp(\log(i_t) - m_t) = \exp(\widetilde{i}_t - m_t) \tag{9}$$

where i'_t is a stabilized input gate, which is a rescaled version of the original input gate. Similarly, the forget gate is stabilized via f'_t, which is a rescaled version of the original forget gate:

$$f'_t = \exp(\log(f_t) + m_{t-1} - m_t) \tag{10}$$

To summarize, compared to the original LSTM, sLSTM adds exponential gating, as indicated by Equations (5) and (6); uses normalization via Equation (3); and finally, the stabilization isachieved via Equations (8)–(10). These provide considerable improvements to the canonical LSTM.

2.2. mLSTM

The Matrix Long Short-Term Memory (mLSTM) model [24] introduces a matrix memory cell along with a covariance update mechanism for key–value pair storage, which significantly increases the model's memory capacity. The gating mechanisms work in tandem with the covariance update rule to manage memory updates efficiently. By removing hidden-to-hidden connections, mLSTM operations can be executed in parallel, which speeds up both training and inference processes. These improvements make mLSTM highly efficient for storing and retrieving information, making it ideal for sequence modeling tasks that require substantial memory capacities, such as language modeling, speech recognition, and time series forecasting. mLSTM represents a notable advancement in Recurrent Neural

Networks, addressing the challenges of complex sequence modeling effectively. Figure 2 shows the architecture of mLSTM.

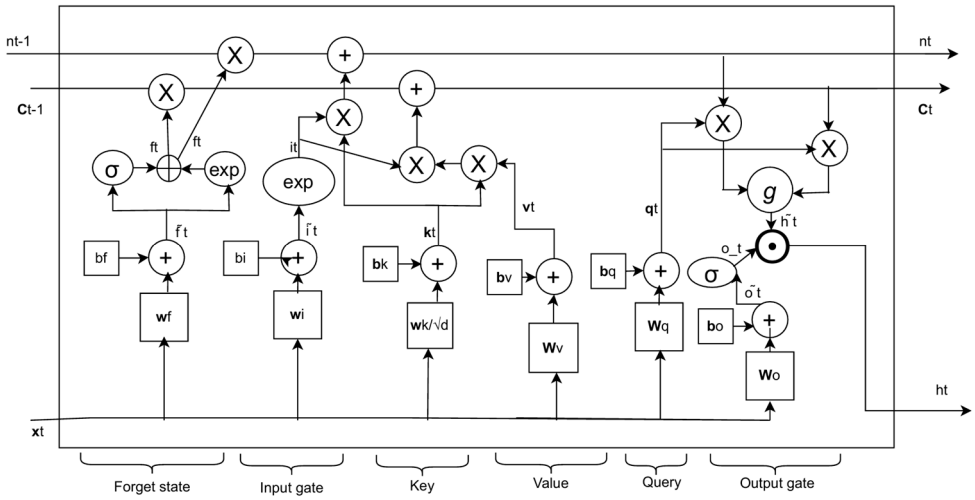


Figure 2. Architecture of mLSTM.

Equations (11)–(19) describe the operations of mLSTM [24].

$$C_t = f_t\, C_{t-1} + i_t\, v_t k_t \tag{11}$$

C_t is the matrix memory that stores information in a more complex structure than the scalar cell state in a traditional LSTM. Normalization is carried out similar to sLSTM:

$$n_t = f_t\, n_{t-1} + i_t\, k_t \tag{12}$$

$$h_t = o_t \odot \tilde{h}_t\ ,\ \tilde{h}_t = g(C_t, q_t, n_t) = C_t q_t / \max\{n_t^{\intercal}\, q_t, 1\} \tag{13}$$

Similar to the transformer architecture, query q_t, key k_t, and value v_t are created as follows:

$$q_t = W_q\, x_t\ + b_q \tag{14}$$

$$k_t = \frac{1}{\sqrt{d}} W_k\, x_t\ + b_k \tag{15}$$

$$v_t = W_v\, x_t\ + b_v \tag{16}$$

$$i_t = \exp(\tilde{i}_t), \quad \tilde{i}_t = \mathrm{w}_i \mathrm{x}_t + \mathrm{b}_i \tag{17}$$

where i_t is the input gate that controls the incorporation of new information into the memory. The forget gate is slightly different compared to sLSTM, as shown below. It determines how much of the previous memory C_{t-1} is to be retained.

$$f_t = \sigma\,(\tilde{f}_t)\ OR\ exp(\tilde{f}_t), \quad \tilde{f}_t = \mathrm{w}_f \mathrm{x}_t + \mathrm{b}_f \tag{18}$$

The output gate is also slightly different in mLSTM, as shown below.

$$o_t = \sigma\,(\tilde{o}_t), \quad \tilde{o}_t = \mathrm{w}_o \mathrm{x}_t + \mathrm{b}_o \tag{19}$$

The output gate controls how much of the retrieved memory is passed to the hidden state. In the next section, we describe how we adapt sLSTM and mLSTM to the time series domain.

2.3. XLSTM for Time Series Applications

For time series applications, both sLSTM and mLSTM provide a viable architecture due to their better handling of the long-term context in terms of important past patterns. We further enhance the capabilities of xLSTM using the proven techniques of sequence decomposition [13] and reversible instance normalization (RevIn [25]). The sequence decomposition extracts the trend and seasonal patterns from the time series data so that xLSTM can do an effective job of storing and retrieving this information. Since the statistical properties such as mean and variance often change over time in time series (known as the distribution shift problem), the RevIN (Reversible Instance Normalization has been shown to be effective at better time series prediction [25]. RevIN is a normalization-and-denormalization method with learnable affine transformation. It is symmetrically structured to remove and restore the statistical information of a time series instance. These two enhancements of sequence decomposition and RevIn when added to the improved xLSTM architecture have the potential for superior time series forecasting models. We detail the design of our xLSTM-based architecture next and then empirically demonstrate that our xLSTM-based design performs competitively with the state-of-the-art transformers [26] and state-space models [22] for time series forecasting.

3. Proposed Method

Our proposed xLSTMTime-based model combines several key components to effectively handle time series forecasting tasks. Figure 3 provides an overview of the model architecture.

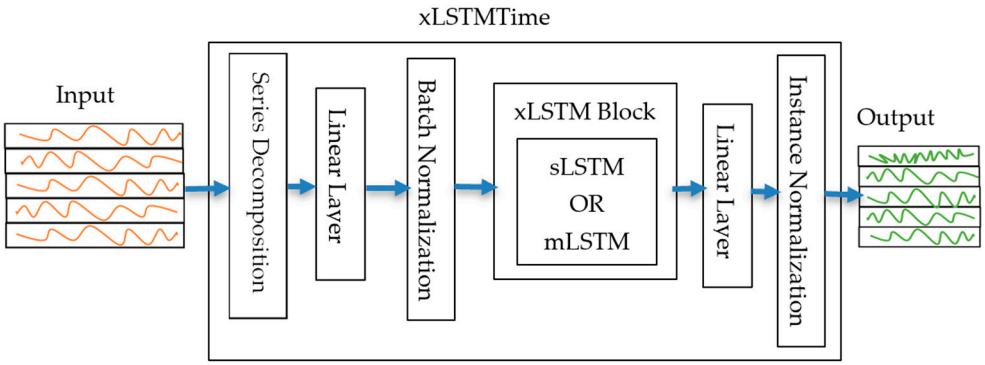


Figure 3. xLSTMTime—processing pipeline for xLSTM-based model for time series forecasting.

The input to the model is a time series comprising multiple sequences. The series decomposition block splits the input time series data into two components for each series to capture trend and seasonal information. We implement the approach as proposed in [13] and described as follows. For the input sequence with a context length of L and m number of features, i.e., $\boldsymbol{x} \in \mathbb{R}^{L \times m}$, we apply learnable moving averages on each feature via 1D convolutions. Then, the trend and seasonal components are extracted as follows:

$$\boldsymbol{x}_{trend} = AveragePool(Padding(\boldsymbol{x}))\boldsymbol{x}_{seasonal} = \boldsymbol{x} - \boldsymbol{x}_{trend} \tag{20}$$

After decomposition, the data pass through a linear transformation layer to be transformed into the dimensionality needed for the xLSTM modules. We further perform a batch normalization [27] to provide stability in learning before feeding the data to the xLSTM modules. Batch normalization is a transformative technique in deep learning that stabilizes the distribution of network inputs by normalizing the activations of each layer. It allows for higher learning rates, accelerates training, and reduces the need for strict initialization and some forms of regularization like Dropout. By addressing internal covariate shift, batch normalization improves network stability and performance across various tasks. It

introduces minimal overhead with two additional trainable parameters per layer, enabling deeper networks to train faster and more effectively [27].

The xLSTM block contains both the sLSTM and mLSTM components. The sLSTM component uses scalar memory and exponential gating to manage long-term dependencies and control the appropriate memory for the historical information. The mLSTM component uses matrix memory and a covariance update rule to enhance storage capacity and relevant information retrieval capabilities. Depending upon the attributes of the dataset, we choose either the sLSTM or mLSTM component. For smaller datasets such as ETTm1, ETTm2, ETTh1, ETTh2, ILI, and weather, we use sLSTM, whereas for larger datasets such as Electricity, Traffic, and PeMS, mLSTM is chosen due to its higher memory capacity in better learning for time series patterns. The output from the xLSTM block goes through another linear layer. This layer further transforms the data, preparing it for the final output via instance normalization. Instance normalization operates on each channel of time series independently. It normalizes the data within each channel of each component series to have a mean of 0 and a variance of 1. The formula for instance normalization for a given feature map is as follows:

$$IN(x) = \frac{x - \mu(x)}{\sigma(x)} \tag{21}$$

where x represents the input feature map, $\mu(x)$ is the mean of the feature map, and $\sigma(x)$ is the standard deviation of the feature map [27]. We use reversible instance normalization (RevIn), which was originally proposed in [27], to operate on each channel of each sample independently. It applies a learnable transformation to normalize the data during training, such that it can be reversed to its original scale during prediction.

The pseudo-code for xLSTMTime is described below in Figure 4.

1. Apply series decomposition to split each series to extract trend and seasonal information.
2. Transform the series data using linear projections to xLSTM dimensions.
3. Apply batch normalization.
4. Process data through a sequence of sLSTM or mLSTM blocks depending upon the complexity and number of feature in series data.
   sLSTM block implements the Equations (1)–(10).
   mLSTM block implements Equations (11)–(19).
5. Apply linear transformation to transform the data from sLSTM/mLSTM units to time series format.
6. Apply reversible instance normalization on each channel for improved series forecasting.

Figure 4. Pseudo-code for our xLSTMTime algorithm.

4. Results

We test our proposed xLSTM-based architecture on 12 widely used datasets from real-world applications. These datasets include the Electricity Transformer Temperature (ETT) series, which are divided into ETTh1 and ETTh2 (hourly intervals), and ETTm1 and ETTm2 (5 min intervals). Additionally, we analyze datasets related to traffic (hourly), electricity (hourly), weather (10 min intervals), influenza-like illness (ILI) (weekly), and exchange rate (daily). Another dataset, PeMS (PEMS03, PEMS04, PEMS07, and PEMS08) traffic, was sourced from the California Transportation Agencies (CalTrans) Performance Measurement System (PeMS). Table 1 presents the attributes of different datasets that we use.

Table 1. Characteristics of the different datasets used.

Datasets	Features	Time Steps	Granularity
Weather	21	52,696	10 min
Traffic	862	17,544	1 h
ILI	7	966	1 week
ETTh1/ETTh2	7	17,420	1 h
ETTm1/ETTm2	7	69,680	5 min
PEMS03	358	26,209	5 min
PEMS04	307	16,992	5 min
PEMS07	883	28,224	5 min
PEMS08	170	17,856	5 min
Electricity	321	26,304	1 h

Each model follows a consistent experimental setup, with prediction lengths T of {96, 192, 336, 720} for all datasets except the ILI dataset. For the ILI dataset, we use prediction lengths of {24, 36, 48, 60}. The look-back window L is 512 for all datasets except ILI dataset, for which we use L of 96 [16]. We use Mean Absolute Error (MAE) during the training. For evaluation, the metrics used are MSE (Mean Squared Error) and MAE (Mean Absolute Error). Table 2 presents the results for the different benchmarks, comparing our results to the recent works in the time series field.

Table 2. Comparison of our xLSTMTime model with other models on the time series datasets.

Models		xLSTMTime		PatchTST		DLinear		FEDformer		Autoformer		Informer		Pyraformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.144	0.187	0.149	0.198	0.176	0.237	0.238	0.314	0.249	0.329	0.354	0.405	0.896	0.556
	192	0.192	0.236	0.194	0.241	0.220	0.282	0.275	0.329	0.325	0.370	0.419	0.434	0.622	0.624
	336	0.237	0.272	0.245	0.282	0.265	0.319	0.339	0.377	0.351	0.391	0.583	0.543	0.739	0.753
	720	0.313	0.326	0.314	0.334	0.323	0.362	0.389	0.409	0.415	0.426	0.916	0.705	1.004	0.934
Traffic	96	0.358	0.242	0.360	0.249	0.410	0.282	0.576	0.359	0.597	0.371	0.733	0.410	2.085	0.468
	192	0.378	0.253	0.379	0.256	0.423	0.287	0.610	0.380	0.607	0.382	0.777	0.435	0.867	0.467
	336	0.392	0.261	0.392	0.264	0.436	0.296	0.608	0.375	0.623	0.387	0.776	0.434	0.869	0.469
	720	0.434	0.287	0.432	0.286	0.466	0.315	0.621	0.375	0.639	0.395	0.827	0.466	0.881	0.473
Electricity	96	0.128	0.221	0.129	0.222	0.14	0.237	0.186	0.302	0.196	0.313	0.304	0.393	0.386	0.449
	192	0.150	0.243	0.147	0.240	0.153	0.249	0.197	0.311	0.211	0.324	0.327	0.417	0.386	0.443
	336	0.166	0.259	0.163	0.259	0.169	0.267	0.213	0.328	0.214	0.327	0.333	0.422	0.378	0.443
	720	0.185	0.276	0.197	0.290	0.203	0.301	0.233	0.344	0.236	0.342	0.351	0.427	0.376	0.445
Illness	24	1.514	0.694	1.319	0.754	2.215	1.081	2.624	1.095	2.906	1.182	4.657	1.449	1.420	2.012
	36	1.519	0.722	1.579	0.870	1.963	0.963	2.516	1.021	2.585	1.038	4.650	1.463	7.394	2.031
	48	1.500	0.725	1.553	0.815	2.130	1.024	2.505	1.041	3.024	1.145	5.004	1.542	7.551	2.057
	60	1.418	0.715	1.470	0.788	2.368	1.096	2.742	1.122	2.761	1.114	5.071	1.543	7.662	2.100
ETTh1	96	0.368	0.395	0.370	0.400	0.375	0.399	0.376	0.415	0.435	0.446	0.941	0.769	0.664	0.612
	192	0.401	0.416	0.413	0.429	0.405	0.416	0.423	0.446	0.456	0.457	1.007	0.786	0.790	0.681
	336	0.422	0.437	0.422	0.440	0.439	0.443	0.444	0.462	0.486	0.487	1.038	0.784	0.891	0.738
	720	0.441	0.465	0.447	0.468	0.472	0.490	0.469	0.492	0.515	0.517	1.144	0.857	0.963	0.782

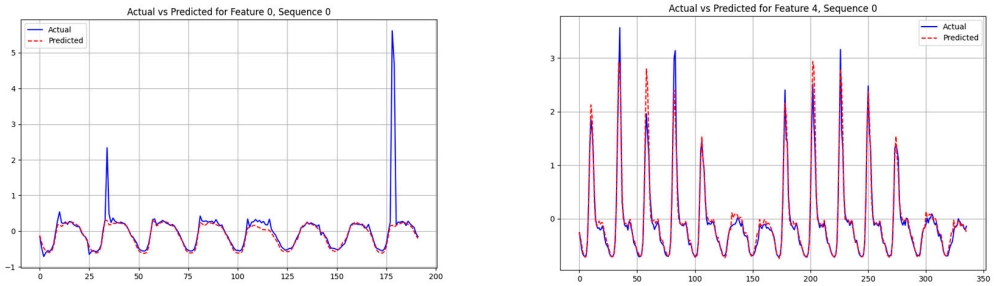
Table 2. *Cont.*

Models		xLSTMTime		PatchTST		DLinear		FEDformer		Autoformer		Informer		Pyraformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh2	96	0.273	0.333	0.274	0.337	0.289	0.353	0.332	0.374	0.332	0.368	1.549	0.952	0.645	0.597
	192	0.340	0.378	0.341	0.382	0.383	0.418	0.407	0.446	0.426	0.434	3.792	1.542	0.788	0.683
	336	0.373	0.403	0.329	0.384	0.448	0.465	0.400	0.447	0.477	0.479	4.215	1.642	0.907	0.747
	720	0.398	0.430	0.379	0.422	0.605	0.551	0.412	0.469	0.453	0.490	3.656	1.619	0.963	0.783
ETTm1	96	0.286	0.335	0.293	0.346	0.299	0.343	0.326	0.390	0.510	0.492	0.626	0.560	0.543	0.510
	192	0.329	0.361	0.333	0.370	0.335	0.365	0.365	0.415	0.514	0.495	0.725	0.619	0.557	0.537
	336	0.358	0.379	0.369	0.392	0.369	0.386	0.392	0.425	0.510	0.492	1.005	0.741	0.754	0.655
	720	0.416	0.411	0.416	0.420	0.425	0.421	0.446	0.458	0.527	0.493	1.133	0.845	0.908	0.724
ETTm2	96	0.164	0.250	0.166	0.256	0.167	0.260	0.180	0.271	0.205	0.293	0.355	0.462	0.435	0.507
	192	0.218	0.288	0.223	0.296	0.224	0.303	0.252	0.318	0.278	0.336	0.595	0.586	0.730	0.673
	336	0.271	0.322	0.274	0.329	0.281	0.342	0.324	0.364	0.343	0.379	1.270	0.871	1.201	0.845
	720	0.361	0.380	0.362	0.385	0.397	0.421	0.410	0.420	0.414	0.419	3.001	1.267	3.625	1.451

Table 2: Multivariate long-term forecasting outcomes with prediction intervals T = {24, 36, 48, 60} for the ILI dataset and T = {96, 192, 336, 720} for other datasets. The best results are highlighted in red, and the next best results are in blue. Lower numbers are better.

As can be seen from Table 2, for a vast majority of the benchmarks, we outperform existing approaches. Only in the case of electricity and ETTh2, in a few of the prediction lengths, are our results the second best.

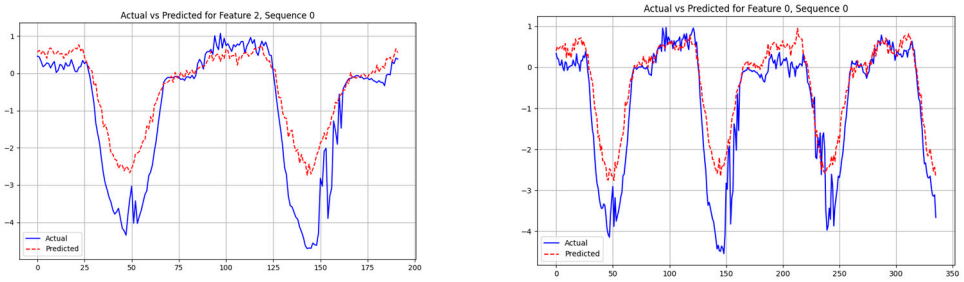
Figures 5 and 6 show the graphs for actual versus predicted time series values for a few of the datasets. The prediction lengths in Figures 5a and 6a are 192, while in Figures 5b and 6b, they are 336. Both figures use a context length of 512. As can be seen, our model learns the periodicity and variations in the data very nicely for the most part.



(**a**) T = 192 (**b**) T = 336

Figure 5. Predicted vs. actual forecasting using our model with L = 512 and T = {192, 336} for the traffic dataset.

Table 3 presents the comparison results for the PeMS datasets. Here, our model produces either the best or the second best results when compared with the recent state-of-the-art models. Figure 6 shows the actual versus predicted graphs for some of the PeMS datasets.



(**a**) T = 192 (**b**) T = 336

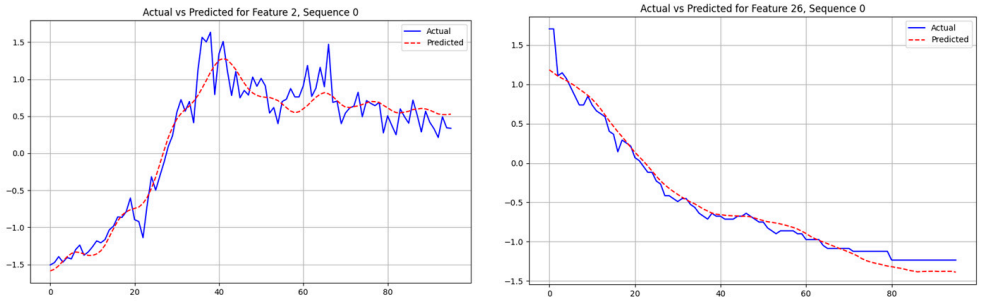
Figure 6. Predicted vs. actual forecasting using our model with L = 512 and T = {192, 336} for the ETTm1 dataset.

Table 3. Comparison of our xLSTMTime model with other models on PEMS datasets. Multivariate forecasting outcomes with prediction intervals T = {12, 24, 48, 96} for all datasets, look-back window L = 96. The best results are highlighted in red, and the next best results are in blue. The lower number is better.

Models		**Our xlstmTime**		**iTransformer**		**RLinear**		**PatchTST**		**Crossformer**		**DLinear**		**SCINet**	
Metric		**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
PEMS03	12	0.065	0.166	0.071	0.174	0.126	0.236	0.099	0.216	0.090	0.203	0.122	0.243	0.066	0.172
	24	0.087	0.194	0.093	0.201	0.246	0.334	0.142	0.259	0.121	0.240	0.201	0.317	0.085	0.198
	48	0.125	0.232	0.125	0.236	0.551	0.529	0.211	0.319	0.202	0.317	0.333	0.425	0.127	0.238
	96	0.192	0.291	0.071	0.174	0.126	0.236	0.099	0.216	0.090	0.203	0.457	0.515	0.178	0.287
PEMS04	12	0.074	0.175	0.078	0.183	0.138	0.252	0.105	0.224	0.098	0.218	0.148	0.272	0.073	0.177
	24	0.090	0.195	0.095	0.205	0.258	0.348	0.153	0.275	0.131	0.256	0.224	0.340	0.084	0.193
	48	0.123	0.230	0.120	0.233	0.572	0.544	0.229	0.339	0.205	0.326	0.355	0.437	0.099	0.211
	96	0.174	0.280	0.150	0.262	1.137	0.820	0.291	0.389	0.402	0.457	0.452	0.504	0.114	0.227
PEMS07	12	0.059	0.151	0.067	0.165	0.118	0.235	0.095	0.207	0.094	0.200	0.115	0.242	0.068	0.171
	24	0.077	0.170	0.088	0.190	0.242	0.341	0.150	0.262	0.139	0.247	0.210	0.329	0.119	0.225
	48	0.105	0.204	0.110	0.215	0.562	0.541	0.253	0.340	0.311	0.369	0.398	0.458	0.149	0.237
	96	0.148	0.247	0.139	0.245	1.096	0.795	0.346	0.404	0.396	0.442	0.594	0.553	0.141	0.234
PEMS08	12	0.072	0.169	0.079	0.182	0.133	0.247	0.168	0.232	0.165	0.214	0.154	0.276	0.087	0.184
	24	0.101	0.199	0.115	0.219	0.249	0.343	0.224	0.281	0.215	0.260	0.248	0.353	0.122	0.221
	48	0.149	0.238	0.186	0.235	0.569	0.544	0.321	0.354	0.315	0.355	0.440	0.470	0.189	0.270
	96	0.224	0.289	0.221	0.267	1.166	0.814	0.408	0.417	0.377	0.397	0.674	0.565	0.236	0.300

Figure 7 shows the graphs for actual versus predicted time series values for a few of the PEMS datasets. The prediction length in Figure 7 is 96. As can be seen, our model learns the trend and the variations in the data for the PEMS dataset nicely for the most part.

For learning rate optimization, we use dynamic learning rate adjustment, where the program monitors the training process, and suggests an effective learning rate based on the observed loss. This results in a higher learning rate at the beginning of the training process and gradually decreases as the loss improves. This leads to a faster and more stable training process. Figure 8 depicts the learning rate change during the training of the ETTm1 dataset.



(**a**) T = 96 (**b**) T = 96

Figure 7. Predicted vs. actual forecasting using our model with L = 96 and T = {96} for the PEMS03 and PEMS07 datasets.

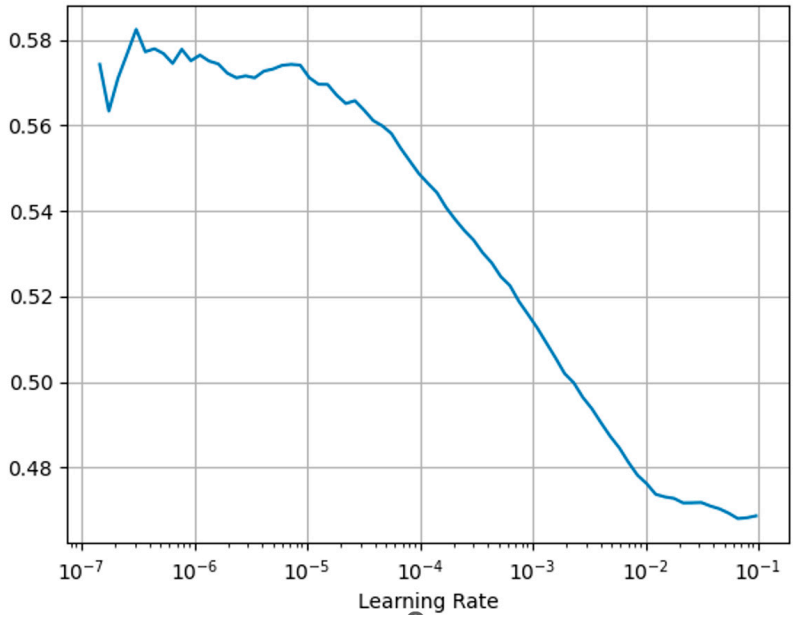


Figure 8. Learning rate schedule followed during the training of the ETTm1 dataset.

The time complexity of xLSTM has been shown to be linear. We measure the exact parameter count and the execution time for training the different models in Tables 4 and 5. While the model sizes are fairly similar for xLSTMTime and transformer-based models (e.g., PathTST and iTransformer), the execution time for xLSTMTime is lower with respect to transformer-based models. This is because the transformer-based models have quadratic complexity with respect to the sequence length due to the pair-wise attention being computed.

Table 4. Model sizes for different architectures for time series forecasting for the Ettm1 dataset (sizes are in millions).

Model	xLSTM (sLSTM Blocks)	xLSTM (mSLTM Blocks)	PatchTST	DLinear	iTransformer
Model Size	1.616 m	1.584 m	0.920 m	0.0984 m	0.9480 m

Table 5. Execution time (training) for different architectures on different datasets.

Model	DataSet	Training Time
xLSTMTime	ETTm1	116.16 s
	Weather	254.32 s
	ETTh1	32.09 s
PatchTST	ETTm1	613.95 s
	Weather	1767.5 s
	ETTh1	142.31 s
DLinear	ETTm1	73.31 s
	Weather	137.69 s
	ETTh1	25.66 s
iTransformer	ETTm1	251.04 s
	Weather	443.47 s
	ETTh1	74.23 s

5. Discussion

One of the recent most effective models for time series forecasting is Dlinear. When we compare our approach to the Dlinear model, we obtain substantial improvements across various datasets, as indicated by the results in Table 2. The most significant enhancements are seen in the weather dataset, with improvements of 18.18% for T = 96 and 12.73% for T = 192. Notable improvements are also observed in the illness dataset (22.62% for T = 36) and ETTh2 dataset (11.23% for T = 192). These results indicate that our xLSTMTime model consistently outperforms DLinear, especially in complex datasets for varying prediction lengths.

Another notable recent model for time series forecasting is PatchTST. The comparison between our xLSTMTime model and PatchTST reveals a nuanced performance landscape. xLSTMTime demonstrates modest but consistent improvements over PatchTST in several scenarios, particularly in the weather dataset, with enhancements ranging from 1.03% to 3.36%. The most notable improvements were observed in weather forecasting at T = 96 and T = 336, as well as in the ETTh1 dataset for T = 720 (1.34% improvement). In the electricity dataset, xLSTMTime shows slight improvements at longer prediction lengths (T = 336 and T = 720). However, xLSTMTime also shows some limitations. In the illness dataset, for shorter prediction lengths, it underperforms PatchTST by 14.78% for T = 24, although it outperforms for T = 60 by 3.54%. Mixed results were also observed in the ETTh2 dataset, with underperformance for T = 336 but better performance at other prediction lengths. Interestingly, for longer prediction horizons (T = 720), the performance of xLSTMTime closely matches or slightly outperforms PatchTST across multiple datasets, with differences often less than 1%. This could be attributed to the better long-term memory capabilities of the xLSTM approach.

Overall, the comparative analysis suggests that while xLSTMTime is highly competitive with PatchTST, a state-of-the-art model for time series forecasting, its advantages are specific to certain datasets and prediction lengths. Moreover, its consistent outperformance of DLinear across multiple scenarios underscores its robustness. The overall performance profile of xLSTMTime, showing significant improvements in most cases over DLinear and PatchTST, establishes its potential in the field of time series forecasting. Our model demonstrates particular strengths at longer prediction horizons in part due to the long context capabilities of xLSTM coupled with the extraction of seasonal and trend information in our implementation.

In comparing the xLSTMTime model with iTransformer, RLinear, PatchTST, Crossformer, DLinear, and SCINet on the PeMS datasets (Table 3), we also achieve superior performance. For instance, in the PEMS03 dataset, for a 12-step prediction, xLSTMTime

achieves approximately 9% better MSE and 5% better MAE. This trend continues across other prediction intervals and datasets, highlighting xLSTMTime's effectiveness in multivariate forecasting. Notably, xLSTMTime often achieves the best or second best results in almost all cases, underscoring its effectiveness in various forecasting scenarios.

The traditional LSTM architecture had difficulty in capturing long-term dependencies. While xLSTM overcomes these to some extent, it may have its limitations in long sequence handling as compared to transformer-based architectures which do not rely on any past state to predict the future token. On the standard datasets that researchers use to compare time series forecasting models, we have not encountered this limitation in xLSTM, but it remains to be seen when much larger time sequences are used to see if xLSTM can still be as good as we have seen in the work.

While we have tested xLSTMTime for different time series forecasting domains such as electricity, weather, traffic, illness, transportation, etc., it is applicable to effective forecasting in other domains e.g., financial and economics. Each domain's series data consists of number of features (or channels). To adapt a given domain's data to our xLSTM model, the data loaders that feed the training and test data to our model will need to be adjusted according to the number of features and the time steps in the context length. We have implemented the code in a flexible way that this can be easily specified.

6. Conclusions

In this paper, we adapt the recently enhanced recurrent architecture of xLSTM which has demonstrated competitive results in the NLP domain for time series forecasting. Since xLSTM with its improved stabilization, exponential gating, and higher memory capacity offers a potentially better deep learning architecture, by properly adapting it to the time series domain via series decomposition, and batch and instance normalization, we develop the xLSTMTime architecture for LTSF. Our xLSTMTime model demonstrates excellent performance against state-of-the-art transformer-based models as well as other recently proposed time series models. Through extensive experiments on diverse datasets, xLSTMTime showed superior accuracy in terms of MSE and MAE, making it a viable alternative to more complex models. We highlight the potential of xLSTM architectures in the time series forecasting arena, paving the way for more efficient and interpretable forecasting solutions, and further exploration using recurrent models.

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

Funding: The authors received no financial support for this research.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All materials related to our study, including the trained models, detailed results reports, source code, and datasets, are publicly accessible via our dedicated GitHub repository: https://github.com/muslehal/xLSTMTime (accessed on 10 August 2024). Dataset link: https://drive.google.com/drive/folders/1ZOYpTUa82_jCcxIdTmyr0LXQfvaM9vIy (accessed on 10 August 2024).

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

References

1. Box, G.E.; Jenkins, G.M.; Reinsel, G.C.; Ljung, G.M. *Time Series Analysis: Forecasting and Control*; John Wiley & Sons: Hoboken, NJ, USA, 2015.
2. Dubey, A.K.; Kumar, A.; García-Díaz, V.; Sharma, A.K.; Kanhaiya, K. Study and analysis of SARIMA and LSTM in forecasting time series data. *Sustain. Energy Technol. Assess.* **2021**, *47*, 101474.

3. Zhang, G.P. An investigation of neural networks for linear time-series forecasting. *Comput. Oper. Res.* **2001**, *28*, 1183–1202. [CrossRef]
4. De Livera, A.M.; Hyndman, R.J.; Snyder, R.D. Forecasting time series with complex seasonal patterns using exponential smoothing. *J. Am. Stat. Assoc.* **2011**, *106*, 1513–1527. [CrossRef]
5. Ristanoski, G.; Liu, W.; Bailey, J. Time series forecasting using distribution enhanced linear regression. In *Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, 14–17 April 2013*; Proceedings, Part I 17; Springer: Berlin/Heidelberg, Germany, 2013; pp. 484–495.
6. Chen, T.; Carlos, G. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
7. Hewamalage, H.; Bergmeir, C.; Bandara, K. Recurrent neural networks for time series forecasting: Current status and future directions. *Int. J. Forecast.* **2021**, *37*, 388–427. [CrossRef]
8. Petneházi, G. Recurrent neural networks for time series forecasting. *arXiv* **2019**, arXiv:1901.00069.
9. Zhao, B.; Lu, H.; Chen, S.; Liu, J.; Wu, D. Convolutional neural networks for time series classification. *J. Syst. Eng. Electron.* **2017**, *28*, 162–169. [CrossRef]
10. Borovykh, A.; Bohte, S.; Oosterlee, C.W. Conditional time series forecasting with convolutional neural networks. *arXiv* **2017**, arXiv:1703.04691.
11. Irena, K.; Wu, D.; Wang, Z. Convolutional neural networks for energy time series forecasting. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
12. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 11106–11115. [CrossRef]
13. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*; Neural Information Processing Systems Foundation, Inc. (NeurIPS): San Diego, CA, USA, 2021; pp. 22419–22430.
14. 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 39th International Conference on Machine Learning PMLR 2022, Baltimore, MD, USA, 17–23 July 2022; pp. 27268–27286.
15. Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.X.; Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*; Neural Information Processing Systems Foundation, Inc. (NeurIPS): San Diego, CA, USA, 2019.
16. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A Time Series is worth 64 words: Long-term forecasting with Transformers. *arXiv* **2022**, arXiv:2211.14730.
17. Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A.X.; Dustdar, S. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In Proceedings of the International Conference on Learning Representations 2022, Online, 25–29 April 2022.
18. Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; Long, M. itransformer: Inverted transformers are effective for time series forecasting. *arXiv* **2023**, arXiv:2310.06625.
19. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are transformers effective for time series forecasting? *Proc. AAAI Conf. Artif. Intell.* **2023**, *37*, 11121–11128. [CrossRef]
20. Alharthi, M.; Mahmood, A. Enhanced Linear and Vision Transformer-Based Architectures for Time Series Forecasting. *Big Data Cogn. Comput.* **2024**, *8*, 48. [CrossRef]
21. Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv* **2022**, arXiv:2210.02186.
22. Gu, A.; Goel, K.; Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv* **2021**, arXiv:2111.00396.
23. Zhang, M.; Saab, K.K.; Poli, M.; Dao, T.; Goel, K.; Ré, C. Effectively modeling time series with simple discrete state spaces. *arXiv* **2023**, arXiv:2303.09489.
24. Beck, M.; Pöppel, K.; Spanring, M.; Auer, A.; Prudnikova, O.; Kopp, M.; Klambauer, G.; Brandstetter, J.; Hochreiter, S. xLSTM: Extended Long Short-Term Memory. *arXiv* **2024**, arXiv:2405.04517.
25. Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; Choo, J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In Proceedings of the 10th International Conference on Learning Representations, Virtual, 25–29 April 2022.
26. Vaswani, A. Attention is all you need. *arXiv* **2017**, arXiv:1706.03762.
27. Ioffe, S. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2015**, arXiv:1502.03167.