mathematics

MDPI

Review

Deep Time Series Forecasting Models: A Comprehensive Survey

Xinhe Liu and Wenmin Wang *

School of Computer Science and Engineering, Macau University of Science and Technology, Macao 999078, China
* Correspondence: wmwang@must.edu.mo

Abstract: Deep learning, a crucial technique for achieving artificial intelligence (AI), has been successfully applied in many fields. The gradual application of the latest architectures of deep learning in the field of time series forecasting (TSF), such as Transformers, has shown excellent performance and results compared to traditional statistical methods. These applications are widely present in academia and in our daily lives, covering many areas including forecasting electricity consumption in power systems, meteorological rainfall, traffic flow, quantitative trading, risk control in finance, sales operations and price predictions for commercial companies, and pandemic prediction in the medical field. Deep learning-based TSF tasks stand out as one of the most valuable AI scenarios for research, playing an important role in explaining complex real-world phenomena. However, deep learning models still face challenges: they need to deal with the challenge of large-scale data in the information age, achieve longer forecasting ranges, reduce excessively high computational complexity, etc. Therefore, novel methods and more effective solutions are essential. In this paper, we review the latest developments in deep learning for TSF. We begin by introducing the recent development trends in the field of TSF and then propose a new taxonomy from the perspective of deep neural network models, comprehensively covering articles published over the past five years. We also organize commonly used experimental evaluation metrics and datasets. Finally, we point out current issues with the existing solutions and suggest promising future directions in the field of deep learning combined with TSF. This paper is the most comprehensive review related to TSF in recent years and will provide a detailed index for researchers in this field and those who are just starting out.

Keywords: time series forecasting; deep learning; Transformer; convolutional neural network (CNN); recurrent neural network (RNN); multi-layer perceptron (MLP); state space model (SSM); large language model (LLM)

MSC: 68T07; 62M10

Citation: Liu, X.; Wang, W. Deep Time Series Forecasting Models: A Comprehensive Survey. *Mathematics* **2024**, *12*, 1504. https://doi.org/10.3390/math12101504

Academic Editors: Jonathan Blackledge and Radu Tudor Ionescu

Received: 22 March 2024
Revised: 28 April 2024
Accepted: 8 May 2024
Published: 11 May 2024

1. Introduction

Time series forecasting (TSF) is a field of study that involves all aspects of society, and the concept has been around for decades in statistics and economics. Early TSF methods mostly evolved gradually through the contributions of many statisticians, economists, and mathematicians spanning multiple periods, a relatively long process, and primarily found applications in economics. For example, conducting research and forecasting trends in stock prices aid in the development of investment strategies and risk management. Similarly, predicting fluctuations in commodity prices facilitates inventory adjustments and supply chain management. Additionally, forecasting macroeconomic indicators such as gross domestic product (GDP) and the unemployment rate enables timely economic decision-making at the national level. With the advent of the information age, the rate of data generation has surged significantly, prompting the need for TSF across various domains, such as electricity [1], transportation [2], meteorology [3], and disease prevention and control [4].

Statistical models played a crucial role in the early stages of TSF, where viewing TSF as a regression problem was a common approach. With the refinement of the theory and gradual evolution of the application of the autoregressive (AR) model and the moving average (MA) model, these models were eventually amalgamated into the autoregressive moving average (ARMA) model. The ARMA model not only considers the long-term dependencies in the time series but also addresses the influence of short-term white noise; thus, it can better capture structural patterns at different time scales using time series data. Further, the ARIMA model, which incorporates a differencing component based on ARMA, can handle the trend part of the time series data more effectively and makes the coefficients in the model more interpretable. Other models also exist including the error-trend-seasonality (ETS) model, which takes seasonal patterns into account, the non-parametric Gaussian process (GP) model, etc. However, these models are better at handling small datasets that are less noisy or highly trending. Additionally, these early statistical models are mostly designed for univariate time series forecasting and often require data preprocessing.

However, with the development of this field and burgeoning volume of data, the influence of external variables (covariates) on the prediction results has been increasingly considered by researchers; for instance, the vector autoregression (VAR) model [5] considers the setting of multiple variables.

In recent years, with the development of machine learning, especially deep learning, deep learning models have gained prominence due to their ability to flexibly capture complex relationships and better capture features and long-term dependencies in data. RNN variants (LSTM, GRU), CNNs, and Transformers have also been introduced to handle multivariate time series data after improving upon the original. However, whether in univariate or multivariate, single-step or multi-step TSF scenarios, deep learning models have shown the ability to positively improve the accuracy of the results.

In the fields of natural language processing (NLP) and computer vision (CV), some models or components exhibiting superior performance have also made remarkable contributions to improving prediction accuracy after being introduced into the field of TSF and have become an important part of the new generation of TSF models [6,7].

In this paper, we aim to fill this gap by summarizing the development of deep models for TSF in recent years. We also present a milestone chart depicted in Figure 1. We first introduce several models commonly used in TSF over recent years and their evolving trends, and we then propose a novel taxonomy from the perspectives of neural network architectures. In addition, we explore the improvement directions of various models, such as RNN, Transformer, and MLP models. We identify key models in each category to analyze and elaborate on their advantages and innovations, aiming to offer insights for enhancing the performance of TSF models. Additionally, we outline the common problems of TSF in detail, summarize the existing solutions and further categorize the solutions according to the approach they involved. Finally, we also point out potential research directions, as well as the gaps and challenges that still need to be addressed, to provide researchers with fresh perspectives on the future trajectory of the TSF domain.

Existing surveys mainly focus on the entire field of time series, including classification and anomaly detection. Not many surveys focus on TSF tasks, and those that do only explore a limited number of early works or provide an overview of specific neural networks of models, failing to cover the overall development trend in this field. Our work provides a comprehensive overview for researchers who are seeking to enter the important field of TSF, an important area of AI development. The main contributions of this paper are as follows:

- **Development trend and new taxonomy:** We analyze the development trend of TSF models over the past five years and propose a new taxonomy of the network architecture of TSF models. We categorize existing TSF models into the following four categories: RNN based, Transformer based, CNN based, and MLP based.
- **Fine-grained problem investigation:** We summarize and categorize the common issues in TSF as well as the existing solutions to these problems to facilitate targeted

model selection for researchers in different fields when integrating or transferring TSF models to specific areas.

- **Rich resources:** Our paper includes almost all the research in the field of TSF over the past five years, involving a wide range of state-of-the-art models and datasets. This survey can serve as a practical guide to get started, understand, apply, and explore various deep learning methods in the field of TSF.
- **Future directions:** We explore the limitations of existing TSF models and propose several potentially effective future innovations as well as research directions in terms of seven aspects.

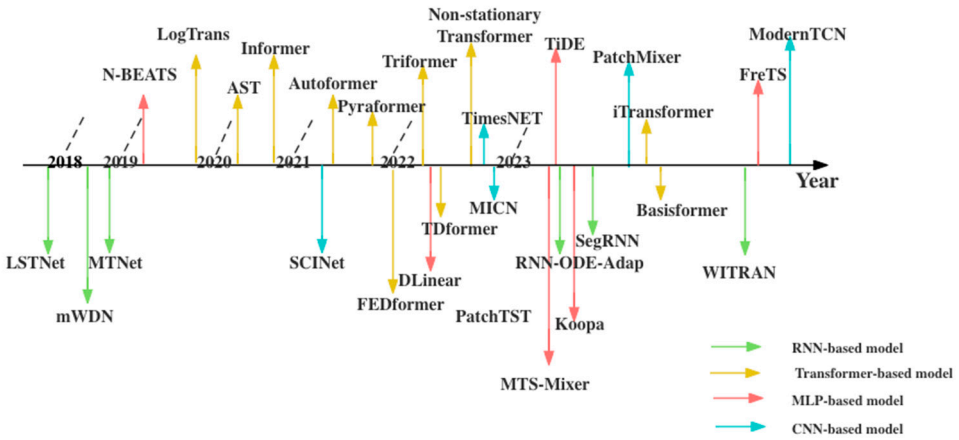

Figure 1. Milestone timeline for deep learning-based time series forecasting.

The rest of our survey is organized to guide the readers as follows. Section 2 outlines the preliminaries of TSF and enumerates commonly used abbreviations and concepts related to TSF. Section 3 clarifies the categorization and taxonomy of TSF models, offering an overview of various models. Section 4 presents commonly used experimental evaluation metrics and datasets. Section 5 discusses current challenges in deep learning-based TSF tasks. Section 6 explores future directions for research. In Section 7, we summarize this survey.

2. Preliminaries of Time Series Forecasting

2.1. Overview of Time Series Forecasting

The essence of TSF simply refers to using historical observations of variables to predict the future values of target variable. Further, the target of TSF is to build a predictor P, where the time series is denoted by $Y = (y_1, y_2, ..., y_t)$ and y_t is the value in time t. Based on a given forecast horizon h, predicted values are generated. This generalized definition takes the following form:

$$Y = (y_1, y_2, ..., y_t) \overset{P}{\Rightarrow} \hat{Y} = (\hat{y}_{t+1}, \hat{y}_{t+2}, ..., \hat{y}_{t+h}) \tag{1}$$

As described above, this is only a generalized formal definition. Depending on the type of TSF, different definitions exist, which will be provided in this section.

Early statisticians often viewed TSF tasks as regression problems, and ARIMA, which combines autoregressive, differential, and moving average elements, can be utilized for models with trends and seasonality; it can also be applied in situations where non-stationary data need to be predicted. However, ARIMA focuses only on linear relationships and fails to capture non-linear relationships. More specifically, the autoregressive part of the model focuses on the linear relationship between the current observed value and historical value, while the moving average (MA) indicates the linear relationship between the current value and the white noise term. This is a very significant limitation of the ARIMA family, namely, the linearity assumption for the observed system is commonly deemed invalid.

In 2017, the research team at Facebook launched a piece of commercial software named Prophet (https://www.prophet-web.com/) [8] for large-scale TSF, which was essentially based on statistical methods. Prophet offers a modular regression model with interpretable parameters, allowing for modeling based on domain knowledge related to time series. According to Prophet, the data are decomposed into four basic factors of variation: trend, periodic changes (e.g., weekly, and yearly seasonality), holiday effects, and extra events. The primary distinction between extra events and holiday effects is that holiday effects focus on dealing with periodic occurrences such as traditional holidays, whereas extra events are more flexible and are not unconstrained by fixed periodic or seasonal patterns, and these special events may be irregular and one-time, which is currently a research hotspot. Compared with ARIMA, the early-stage statistical model applying decomposition architecture, the advantage lies in its better accuracy when predicting change points and trends as well as its applicability to large-scale data. However, due to applying additive model and fitting components, forecasting an exact value is challenging due to the influence of noise and other implicit factors. It is worth noting that none of these models can be used for long sequence time series forecasting (hereinafter called LSTF, Informer proposed) scenario.

However, RNNs face serious problems such as vanishing gradients and exploding gradients when dealing with long sequences. In 1997, Jürgen Schmidhuber proposed long short-term memory (LSTM) [9]. By introducing a gating mechanism (including forget gates, input gates and output gates) to control information flow, LSTM effectively captures long-term dependencies between different time steps in sequence. Furthermore, the gated recurrent unit (GRU) [10] simplifies the network architecture by including only two gates, an update gate and a reset gate, which reduces the number of parameters, prevents overfitting, and makes computation more efficient. DeepAR [1] is a probabilistic forecasting tool proposed by Amazon based on an autoregressive recurrent network architecture, and its predicted output is not a definite value but rather a probability distribution of the predicted value. Compared to LSTM, DeepAR considers the stochastic nature of many processes. The probability distribution as the output is closer to the essence and can effectively reflect the uncertainty of forecasting, offering a comprehensive basis for decision-making with higher accuracy. It seems that RNN variants can potentially resolve long sequence forecasting. However, their effectiveness remains unsatisfactory when confronted with real-world applications involving massive, high-frequency time series data and longer prediction demand. Therefore, integrating long sequence and addressing these data characteristics is one of the key challenges in deploying sequence data analysis.

Before the advent of Transformers [11], LSTF was a relatively neglected issue in the field of TSF. On the one hand, the models mentioned above only provide relatively short-term prediction; on the other hand, the coarse-grained prediction for a long-term trend in the future is insufficient for real-world application. This has hindered development and progress in fields where the support of LSTF is urgently needed to reduce resource consumption in real-life applications, such as the commissioning of large electrical Transformers. Initially successful in the NLP, Transformers have been gradually applied in TSF tasks since both NLP and time series involve sequence data. The successful application of models in NLP and CV has led to a surge in research activity in the field of TSF over the past couple of years. Initially, Wu et al. [4] directly applied Transformers for TSF. Subsequently, researchers adopted a combination of CNN or LSTM with Transformers [12] to address the issue of position information dependent on position embedding. This ensured the perfect application of the sequence modeling capability and ultra-long-period information extraction capability of the attention mechanism.

In the classical encoder–decoder structure [13], between Input and output of the model, there is a recurrent neural network modeling process one by one. When the model is presented with a longer prediction sequence, the cumulative number of neurons, i.e., the traveling paths of network signals, between the input and output will also increase, raising the cumulative error. However, the self-attention mechanism in the Transformer,

by virtue of the fact that the output of each position only depends on the weights of the other positions in the sequence and the weighted sum of the corresponding feature values, is able to avoid having a recurrent structure. Moreover, the maximum length of network signals' traveling paths can be reduced to $\mathcal{O}(1)$. However, given the overall length of the sequence, solving the quadratic computational complexity still needs to be addressed. Informer [14] utilized key and few critical queries to perform sparse attention, thereby reducing computation. Since then, innovation in Transformer-based approaches for solving LSTF problems has entered a phase of rapid growth, facilitated by the integration with traditional statistical TSF methods and inspiration from leading advances in domains like CV.

Due to recurrent architecture's inherent advantages in modeling sequence, it is often considered the default starting point for sequence modeling tasks. However, some convolutional architectures can achieve good accuracy in machine translation [15], audio synthesis [16], and language modeling [17], so the scope of applicability for convolutional sequence modeling has expanded. The CNN is a type of deep feedforward neural network with convolution and pooling operations at its core. Although the CNN was originally designed to handle image recognition in CV [18], it is still effective for TSF. (1) The advantage of using a CNN in TSF tasks is its convolution kernels, and its local receptive fields mean that each convolution kernel only views a small part of the input data. (2) Sliding kernels can capture local patterns and features, with varying scales of patterns and features captured by adjusting the sizes and strides of the convolution kernels. Meanwhile, the pooling operation can save key data, reduce the amount of redundant information, and ultimately yield forecasting results. (3) In addition, kernel weights are shared across the input, which means that a CNN can recognize the same pattern at any position in time series, increasing its capacity for generalization. (4) By stacking CNNs, complex patterns in time series can be effectively learned.

The characteristics above enable CNNs to perform well in some specific time series tasks, especially those involving multi-scale patterns. CNN-based TSF models have limitations in handling non-stationary data and long-term dependencies, possibly failing to fully capture the contextual information of the data. In comparison with other network architectures, such as RNNs, CNNs do not exhibit superior prediction accuracy, particularly in scenarios involving TSF with longer intervals. Nonetheless, they are frequently integrated into other advanced algorithmic models as robust modules for prediction tasks. In 2016, DeepMind proposed an audio generative model called WaveNet [16] based on PixelCNN [19], which regards audio data as 1D waveform data. By combining dilated causal convolution with residual and skip connection, it shows very promising results in audio modeling and speech recognition. Transferring to the TSF realm, time series can also be reviewed as one-dimensional vectors and then be fed into WaveNet to obtain the predicted value for the next time step. Inspired by WaveNet, Anastasia Borovykh et al. [20] used ReLU instead of the gated activation function with novel parametrized skip connections to simplify and optimize the model structure for multivariate TSF. In 2016, Lea et al. proposed a temporal convolutional network (TCN) [21] for video-based action segmentation, which was later extended to the TSF field. Moreover, in 2018, Bai et al. introduced causal convolutions based on TCN while combining residual connection and dilation convolution to avoid future information leakage and realize the introduction of longer historical information with the same complexity to effectively improve the prediction performance. Further, in 2022, Liu et al. proposed SCINet [22] based on downsample–convolve–interact architecture, with a unique binary tree structure where both short- and long-term dependencies can be extracted, and its performance once surpassed Transformer-based models that obtained SOTA in the LSTF task. Following this, influenced by decomposition [23] and other novel ideas successfully applied in TSF, CNN-based models and Transformer-based models have demonstrated leading performance in recent years.

The MLP method is constrained by its point-wise mapping approach, rendering it ineffective in addressing the global dependencies within time series data, thereby signifi-

cantly impeding its performance, but it has a simpler structure. However, over the past year, many works have moved away from RNNs, CNNs and Transformers, instead opting for MLP-based models in TSF tasks, especially for LSTF, resulting in significant improvements [24–27]. In other words, the architecture of TSF models is developing towards a new direction: **simplicity**. Zeng et al. questioned the effectiveness of using Transformers in TSF, especially the necessity of the attention mechanism, and proposed DLinear [26], a model that combines the decomposition scheme with the linear layer. Raw time series data are decomposed into trend and seasonal components by a moving average. Each component is processed by one-layer linear layers and then added to obtain the result. This simple model outperformed SOTA's Transformer model at the time in both univariate and multivariate TSF tasks. Further, influenced by Mixer [28] in CV, which realizes the interaction between the channel dimension and token dimension through MLP, MTS-Mixer [29] applies the structure of Mixer to the TSF scenario, and studies the low-rank property of multivariate time series, that is, only a small part of data can represent the nearly complete original matrix and be used to design a model with a factorized temporal and channel mixing strategy. Since then, institutions like Google have continued to explore the potential capacity of MLP, optimizing results in TSF.

Additionally, a kind of model known as Time-index model, which is defined in detail by woo et al. [30]. This model uses time-related features rather than the historical observations mentioned before, such as datetime features, as inputs to predict the value on a specific timestep. In 2017, Luke B. Godfrey and Michael S. Gashler proposed ND [31] with a regression-based extrapolation method. In simple terms, a complex curve is obtained based on historical values by implementing regression, whose horizontal axis and vertical axis represent timestamps and sequence values, respectively. In this way, the future timestamp is used as an input to obtain the predicted values. Many recent works have been heavily influenced by its idea. This method obviously takes global information into account rather than just making predictions based on a small time window in the past.

2.2. Some Definitions in Time Series Forecasting

In this paper, we elaborate on some ambiguous or undefined issues in previous research.

Firstly, in this paper, TSF is primarily separated into two categories based on variate: univariate TSF and multivariate TSF.

For univariate TSF, given historical time series Y with look-back window size length T, $Y = (y_1, y_2, ..., y_T)$, the goal of our task is to predict the forecasting horizon with length L, $\hat{Y} = (\hat{y}_{T+1}, \hat{y}_{T+2}, ..., \hat{y}_{T+L})$. We can obtain $T \times 1 \rightarrow L \times 1$, denoting input T and prediction L with one variable. Most of the early TSF models were designed for univariate TSF due to the difficulty in acquiring and recording data as well as the scarcity of computational resources in the past, meaning that the focus of research was usually on a single variate. However, as science and technology advances, more and more applications are required to consider the complexity of the data more thoroughly, so multivariate TSF has gradually emerged.

For multivariate TSF, given historical data χ containing M variables, $\chi = \left\{ X_1^t, X_2^t, ..., X_M^t \right\}_{t=1}^{T}$, wherein the look-back window size is length T and X_i^t means the value of the i_{th} variate at the t_{th} timestep, the goal of our task is to predict the corresponding sequence $\mathcal{Y} = \left\{ Y_1^t, Y_2^t, ..., Y_N^t \right\}_{t=T+1}^{T+L}$ at future L-time steps. Most of the models surveyed in this article, especially those developed in the past two years, are aimed at tackling LSTF problems. Therefore, the length of L is longer than in previous works, and we can obtain $T \times M \rightarrow L \times N$, denoting input T with feature dimensions or numbers of variables $M (M > 1)$ and prediction L with N dimensions. It should be noted here that the number of variables will also be considered as multiple channels.

The essential distinction between univariate TSF and multivariate TSF lies in whether to consider the relationship between multiple variables. In real-world scenarios, multivariate TSF models can often be directly applied to univariate TSF. Therefore, the categorization in this paper is only used for describing the models' application in original papers. Most

researchers believe that the process of modeling multivariate TSF modeling is focused on exploring the dependency relationship between multiple variables, so early studies tend to create a complex multivariate relationship extraction model, leading to improved accuracy in prediction results. Coincidentally, with the evolution of the model structure, more and more researchers have discovered that simpler Channel-Independent (later called CI) modeling approaches [6,26], specifically modeling each variable individually in multivariate TSF, yield better results. We will get into the specifics in the subsequent section. However, it is clear that this simplification of complex problems has played an important role in various research areas.

Above is the division of TSF with respect to the variate dimension of the input. Similarly, for the output, we have one-step and multi-step TSF depending on the time horizon of forecasted output values. For one-step TSF, it simply forecasts the next time step, i.e., it forecasts only one value in the future, so it is used for short-term forecasting, such as the stock price on the next trading day. Multi-step TSF predicts values of multiple time steps in the future, which is the primary focus for LSTF, because we need longer predicted values.

Furthermore, multi-step TSF can be categorized according to the way model outputs and the predicted sequences are generated. The following are the main strategies, usually selected according to the requirements of tasks and the characteristics of the data:

1. **Iterated multi-step forecasting (IMS):** This strategy means that the model iteratively generates forecasts for multiple future time steps; each time step's forecast result is dependent on the forecast of the previous step. In other words, the model uses the forecast of the first future time step as an input to predict the value of the next time step, and the whole process is carried out iteratively. However, this kind of iterative forecasting results in significant cumulative errors in the LSTF problem; we are particularly concerned with the prediction accuracy being greatly affected. The commonly used RNN-based TSF method falls within the category of IMS forecasting because of its recursive nature.
2. **Direct multi-step forecasting (DMS):** This strategy means that the model directly generates forecasts for multiple time steps, unlike IMS, where each forecast depends on the prediction of the previous time step. Informer [14] has designed a generative-style decoder that generates the forecasts in one forward procedure and is no longer affected by dynamic decoding. Meanwhile, it is no longer affected by the inability to make fast predictions when tasks require long outputs. MQ-RNN [32], Autoformer [33], FEDfomer [34], DLinear [26], Pyraformer [35], etc., all implement the DMS strategy. Zeng et al. [26] notes that the essential difference between these two strategies lies in whether the decoders are implemented in an autoregressive way: IMS if yes, and DMS otherwise.

In the early stage, deep learning-based TSF models considered the properties of sequence data such as long- or short-term dependencies, meaning that the current observed value is influenced by past distant time points. However, the unique characteristics of time series data are not adequately taken into account:

- **Trend:** The overall long-term direction of change shown by time series data, with three specific trends: stationary, upward, and downward.
- **Seasonality:** The periodicity of the time series data shown over a specific period, which is usually associated with a specific time interval (weekly, monthly, quarterly, etc.). For example, sales of cold drinks increase every summer.
- **Cyclical Patterns:** Although seasonality is a special form of cyclical pattern and can be considered a subset of cyclicality, it more specifically refers to cyclical patterns that occur within a fixed period. There is still a difference, theoretically. Cyclical patterns represent alternating peaks and troughs in a time series, characterized by fluctuations that tend to be irregular but predictable over the long term. These patterns often arise in reality from various business or economic activities, scientific and technological

developments, etc. In deep learning models, we typically decide whether to consider it based on the characteristics of the data.

- **Noise:** Stochastic fluctuations in the time series that may come from measurement errors or external factors.

The above are the components obtained using decomposition methods the deep learning models usually used to assist them in obtaining more accurate predictions. For example, N-BEATS [36] uses fully connected layers to perform decomposition for time series data; the interpretability of results is aided by the analysis of the trend and seasonal part. A typical characteristic of the trend is that it is typically a monotonic function, or at least a slowly varying one. N-BEATS utilizes polynomial regression and sine–cosine functions to fit the trend and seasonal part, respectively. Autoformer [33] innovatively incorporates a commonly used statistical preprocessing method into an embedded module, called series-decomposition block, which is based on the moving average (MA), in order to highlight the trend term and smooth the seasonal term. Recent SOTA models have widely embraced this strategy. We will organize detailed developments in the following sections for reference.

3. Taxonomy of Deep Time Series Forecasting Models

To systematically summarize the existing deep TSF models, we propose a taxonomy from the perspectives of the neural network architecture, and the discussed models cover RNN-based, CNN-based, Transformer-based, and MLP-based models, as outlined in Figure 2. We present the innovation points and trends of each category along the timeline. The four architectures represent the main research directions in the field of deep learning-based TSF, covering the evolution from traditional RNNs to more modern Transformer models, and the recent development of MLP models achieving good performance with simple architectures. These methods are based on different neural network architectures, each with its unique characteristics and advantages, as detailed in Section 2. Categorizing them helps highlight the architectural differences, making it easier for readers to understand how different types of architecture models have improved in forecasting performance over time, particularly in addressing the increasingly focused LSTF issue, and where innovations and improvements can be made. Moreover, this taxonomy can clearly demonstrate how innovative ideas within different neural network architectures mutually influence and promote each other. In this way, it could be helpful for researchers who are new to this field to straighten out the context and choose specific models according to their own research direction in academic research. The content of articles listed under each classification serves to foster the development of the TSF realm, an approach less commonly encountered in previous surveys but providing a comprehensive understanding of the TSF field's evolution through this approach. Clear summaries of these models are presented in tables after each category. It is worth noting that most of the models are designed for the LSTF problem, which is consistent with the research focus changing as real-world data develop.

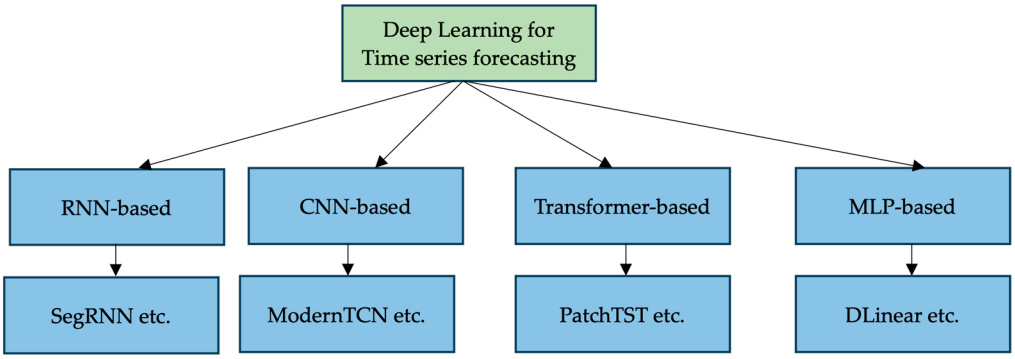

Figure 2. The taxonomy of deep TSF models from neural network architecture. A representative model is shown for each category.

3.1. RNN-Based Models

Recurrent neural network (RNN)-based models are the pioneer of deep learning in the TSF realm. Because of their recursive structure, RNN-based models are suitable for working with time series data or tasks that involve obtaining time dependencies. However, there is still a limit to handling long-term dependencies or capturing intricate patterns, particularly when working with LSTF, where the issue of vanishing gradients or inflating gradients is likely to occur.

LSTNet [37], proposed by Lai et al. for multivariate TSF, introduced a convolutional layer to capture short-term features of time series data with LSTM or GRU for long-term feature extraction. Considering that RNNs cannot capture very long-term connections, a recurrent skip component is designed to record ultra-long-term features of time series data. Using the periodic pattern of real data, a periodic p hyperparameter is introduced to solve this problem. Finally, considering that the neural network is not flexible enough to the change in the input scale (changes in data size and dimension), which will lead to fluctuations in performance, an AR model is introduced to add linear components, making the non-linear deep learning model more robust to time series that violate the scale change, and the output can respond to the scale change to the input. The DA-RNN model [38] used a dual-stage attention mechanism to adaptively assign higher weights to highly correlated feature variables in the input stage and find the encoder hidden states with the strongest temporal correlation over the entire time steps in the decoder stage. This kind of design ensures that DA-RNN can not only adaptively select the most relevant input features but can also capture the long-term temporal dependence of time series. MQ-RNN [32], which also adopts Seq2Seq architecture, considers that a single value provided by point forecasting methods cannot reflect the uncertainty of the model for the predicted value, but probabilistic ones are suitable for this situation and provide abundant information for many decision-making scenarios. A novel forking-sequences approach is applied in the training process (reflected in the connection of a decoder after every time point in the encoder part), and a DMS strategy is adopted; an encoder–decoder structure is also optimized with the RNN being replaced with a MLP for prediction in the decoder.

Further, mWDN [39] introduced wavelet decomposition as part of the network framework with trainable parameters rather than a preprocessing method before modeling. Subseries obtained by mDWN are sent to corresponding LSTMs and then ensemble the results. MTNet [40] introduced a memory network and redesigned the attention weights. As with LSTNet, the traditional autoregressive linear model is paralleled with the non-linear neural network to address its insensitive output scale. Hybrid ES-LSTM [41] combines the exponential smoothing (ES) method with LSTM and trains the preprocessed parameters together with a neural network. At the same time, the ES method can effectively capture active components such as seasonality, and the neural network can iteratively eliminate seasonal factors. By hierarchically leveraging both local and global components, sequence information is extracted and integrated simultaneously. This approach has inspired numerous subsequent models. Moreover, Fan et al. [42] focuses on multi-modal fusion, regarding different historical periods as different modalities and fusing them with multimodal attention weights for better prediction.

C2FAR [43] is a univariate probabilistic TSF model based on DeepAR [1] that introduces the binning approach that already exists in CV and NLP. It first classifies the coarse bin in which the predicted value is located and then uses that as a condition to sort out which finer bin the prediction value is located in. It is worth mentioning that C2FAR focus a practical concern regarding the diverse nature of real-world time series, which often exhibit a combination of discrete, continuous, and semi-continuous characteristics. C2FAR deals with prediction problems through the discretization of sequence values and is suitable for prediction that does not require high precision. Moreover, Neural ODE is frequently used to describe the change in hidden state over time, and the derivative of the hidden state is used to model this change. Based on that, RNN-ODE-Adap [44] included the data itself in the derivative of the hidden state to model together and selected the time steps adaptively

according to the local variations in the time series, capturing the potential trend with fewer steps, and achieved higher prediction accuracy with lower time complexity.

With the emergence of Transformer-based models, the Transformer's superior performance has rendered RNN-based models no longer the preferred choice for LSTF problem, and its performance has even been surpassed by MLP-based models, such as DLinear [26] and TiDE [24]. In the LSTF problem, with the expansion of the prediction horizon, the cumulative error of RNN-based model increases rapidly, and the inference time also increases rapidly. Although many previous efforts have made great improvements, they are dwarfed by the reality of longer data inputs: longer forecast horizons. However, SegRNN [45] emerged at a time when Transformer-based models and MLP-based models were alternating in prominence. This was influenced by the patching technique used to preprocess data, which has been utilized in both of these kinds of models. SegRNN assumes that the main reason for RNNs' failure in LSTF problems is their high recurrent iteration counts. It introduces a segment technique into the RNN and a parallel multi-step forecasting (PMF) strategy while replacing pointwise iterations with segment-wise iterations to reduce the number of iterations. These methods greatly improve the prediction accuracy of RNN-based models. Following this, WITRAN [46] continued to emphasize capturing long- and short-term repeating patterns. However, inspired by TimesNET's reconstruction of 1D time series in 2D space to simulate both intra- and inter-period changes, the author further rearranged the sequence, and multiple adaptive cycles of the input were learned by setting up hyperparameters instead of fast Fourier transformations (FFT). Gate select cells were designed in both horizontal and vertical directions to merge and select information. The computational efficiency is further improved through the parallel processing of two directions of information transmission through a recurrent acceleration network. Theoretically, both WITRAN and SegRNN solve the vanishing gradient and exploding gradient of RNNs by **reducing the length of the transmission path**. This approach aligns with the main direction of restarting RNN-based models to solve LSTF problems.

While RNN-based methods have experienced periods without good performance in either predictive accuracy or forecasting capability for LSTF issues, they are lower in complexity compared to Transformer-based models. Moreover, the structure of RNN-based models is more suitable for processing time series data and allows long- and short-term patterns to be captured simultaneously. By studying strategies that have proven successful in other network models, such as patching techniques, RNN-based models may shine again. We summarize the RNN-based models in Table 1.

Table 1. Summary of RNN-based TSF models involved in taxonomy.

Models	Journal/Conference	Category	Year
LSTNet [37]	SIGIR	RNN-based	2018
DA-RNN [38]	IJCAI	RNN-based	2017
MQ-RNN [32]	Arxiv	RNN-based	2017
mWDN [39]	KDD	RNN-based	2018
MTNet [40]	AAAI	RNN-based	2019
Hybrid-ES-LSTM [41]	IJoF	RNN-based	2020
LSTM-based ED [42]	KDD	RNN-based	2019
C2FAR [43]	NeurIPS	RNN-based	2022
RNN-ODE-Adap [44]	Arxiv	RNN-based	2023
SegRNN [45]	Arxiv	RNN-based	2023
WITRAN [46]	NeurIPS	RNN-based	2023

3.2. CNN-Based Models

Convolutional neural network (CNN) is crucial for the success of deep learning in the CV realm. In the field of TSF, CNNs can leverage their high efficiency in local feature extraction. Meanwhile, temporal convolutional networks (TCNs), after effectively introducing causal convolution and dilated convolution, are highly suitable for processing sequence data and improve the performance of handling long-term sequences in some

scenarios. They are designed to address the problems of RNNs such as vanishing gradients and high computational complexity encountered in handling long sequences and utilize the advantages of modeling long-term dependencies faster and the ability to compute in parallel. Even though TCNs have increased performance, other neural network types may still perform better on long sequences.

DSANet [47] focuses on time series with dynamic periodic or non-periodic patterns. It independently inputs each univariate time series into two parallel convolution structures, each operating at a different scale, to model global and local complex patterns, respectively. Meanwhile, each branch combines an individual self-attention module to learn the dependencies between different sequences. DSANet also employs an autoregressive component to address the issue of neural network output scale insensitivity to input scale, and, finally, considers the mixture of linear and non-linear components as a result. MLCNN [48] places the construal-level theory of psychology as the core of the entire model design. From the human perspective, MLCNN uses abstract features to predict distant future values. Some specific features are used to predict near-future values, thereby improving the prediction performance by fusing prediction information from different future moments. To achieve this objective, an MLCNN was used to build a multi-tasking architecture wherein one main task and four auxiliary tasks were used to learn about the target future moment and its near- and distant-future values. The raw time series constructed five intermediate feature maps with different levels through a 10-layer one-dimension CNN. After that, it was used as the input for the subsequent LSTM-based fusion-encoder–main-decoder part to obtain the result of the non-linear part, and then the linear part's results obtained using AR were merged to obtain the final prediction.

SCINet [22] believed that the unique characteristic of time series compared to other sequence data (e.g., text sequences) lies in the fact that after downsampling them into subsequences, they still contain relevant information and connections of time series. It is proposed that causal convolution in TCN is not only unnecessary but also self-restrictive in its ability to extract temporal information. Therefore, SCINet is designed as a recursive downsample–convolve–interact architecture. It also adopts a multi-layer binary tree structure, iteratively capturing information at different time resolutions. As the depth of the binary tree increases, finer-grained information is extracted. This hierarchical structure allows for the extraction of both short- and long-term dependencies in the sequence. Further, more complex sequences can be processed by stacking multiple SCINets. MICN [49] uses a moving average similar to Autoformer [33] and FEDformer [34] to decompose the original sequence into trend terms and seasonal terms and predicts them separately, and it then integrates them to obtain the result. The innovation is that a multi-scale branch structure was designed for the seasonal part and down-sampled convolution was used to extract local features of time series. Then, instead of masked self-attention, an isometric convolution module is used to model the correlation between all local features and obtain global features. Isometric convolution involves padding the original sequence, which has a length of S, with S-1 zeros at the beginning, followed by convolving the sequence with a kernel of size S. Integrating both CNN and Transformer modeling perspectives enhances the performance of CNN models in LTSF. From the perspective of multi-periodicity, TimesNet [50] breaks through the limitation that the original one-dimensional time series structure can only represent changes between adjacent time points. Further, complex temporal variations are divided into multiple intra-period and inter-period variations. By using the fast Fourier transform (FFT) algorithm to convert 1D sequences into 2D tensors, the processing of two different types of variations is extended to 2D space, which enables temporal patterns to be captured using various neural network backbones in CV such as Inception. It is important to note that TimesNet serves as a versatile neural network framework for time series analysis, capable of handling various tasks, including TSF, classification, and anomaly detection. Instead of depending on advancing the neural network, TLNet [51] utilized a transform-based network architecture for interpretability and better performance in LSTF tasks. It was realized by transforming the input's features into a domain defined by

large receptive fields and then building representations within that domain with potential space for interpretability. In the new domain, it may be easier to learn the global features and structure.

MPPN [52] focuses on how to automatically mine multi-resolution and multi-periodicity patterns in time series and proposes an entropy-based method to evaluate the predictability of time series to prevent the negative effects brought about by the introduction of unforeseeable noise in training. In terms of resolution, convolutions with different kernel sizes are employed to extract features from original sequences to form representations at different resolutions. For the periodic pattern, Fourier transform is used to map the time series to the frequency domain, and frequencies with the top-k amplitudes are selected as the corresponding pre-defined periods. Further, dilated convolution is adopted to achieve multi-periodic and multi-resolution pattern mining. Finally, the resolution and periodic information are concatenated for the encoding result. FDNet [23], with a simple structure consisting of basic linear projection layers and CNN, uses a focal input sequence decomposition method to divide the input sequence into multiple continuous subsequences based on the temporal distance. The length of subsequences decreases as the temporal distance from the predicted element shortens, with the numbers determined by a hyperparameter. This kind of design effectively solves the problem of long sequence time series input (LSTI). PatchMixer [7] is a model that integrates patching data-organizing strategy and CI strategy with CNN architecture, which have shown significant effectiveness in TSF. It still employs trend–seasonal decomposition and applies linear and non-linear methods to two branches, respectively, eventually summing their prediction results up. Two convolutional branches are creatively employed to model intra- and inter-patch data, facilitating the extraction of local and global information and further enhancing the performance in LSTF tasks.

ModernTCN [53] drew inspiration from the superior performance of CNNs in the field of CV compared to Transformer-based models, and it further explored the potential of improving the performance of CNNs in the TSF realm. Specifically, it increases the kernel size to expand the effective reception field (ERF) and draws inspiration from the architectural advantages of Transformers. In addition, ModernTCN adopts the patching strategy to split each variable into patches for embedding to prevent variable mixing embedding methods from disrupting the modeling of relationships between subsequent variables. By cleverly decoupling the temporal part, channel part (referring to depth-wise convolution), and variable part through three types of convolutions, the prediction performance in LSTF problem was improved. Moreover, UniRepLKNet [54] essentially serves as a feature extraction network, further exploring the potential of large-kernel CNNs. It still focuses on the ERF as its design center and decouples various elements when designing large-kernel CNNs: ensuring a large receptive field with a few large kernels, efficiently capturing features in local regions with flexible small kernels, enhancing the abstract hierarchy of spatial patterns, and improving the model depth and representational capacity with efficient structures like SE Blocks. In handling time series data, it draws inspiration from the embedding layer in CorrFormer, transforming data into tensors in latent space and then simply reshaping them into a single channel embedding map. This approach demonstrates promising results in prediction tasks, potentially even replacing Transformer model architectures.

We have systematically reviewed the latest developments of CNN-based networks in the field of TSF over time, as summarized in Table 2. We can observe that the advantage of using CNN-based models for TSF is in feature extraction, especially when facing multivariate TSF tasks. Although the CNN has experienced a period of silence, the effectiveness of causal convolution, the crucial structure of the TCN and an important advancement in the TSF realm, has been questioned and challenged. In addition, the large-kernel CNN has shown promising results in various tasks in CV, such as semantic segmentation and object detection. As a result, researchers have taken notice of this technology and have modified convolution for use in TSF tasks. Numerous versions based on this concept have

been proposed, particularly in the last few months, effectively solving the LSTF issue. This architecture will undoubtedly present a better future.

Table 2. Summary of CNN-based TSF models involved in taxonomy.

Models	Journal/Conference	Category	Year
DSANet [47]	CIKM	CNN based	2019
MLCNN [48]	AAAI	CNN based	2020
SCINet [22]	NeurIPS	CNN based	2022
MICN [49]	ICLR	CNN based	2023
TimesNet [50]	ICLR	CNN based	2023
MPPN [52]	Arxiv	CNN based	2023
FDNet [23]	KBS	CNN based	2023
PatchMixer [7]	Arxiv	CNN based	2023
ModernTCN [53]	ICLR	CNN based	2024
UniRepLKNet [54]	Arxiv	CNN based	2023

3.3. Transformer-Based Models

Transformer [11] was initially proposed by Vaswani et al. in their paper "Attention is All You Need". Its original model was an encoder–decoder structure that showed state-of-the-art performance across a wide range of NLP tasks. Models such as GPT use this structure. By treating each position in the sequence as a vector and utilizing the multi-head self-attention mechanism and feedforward neural network's outstanding ability in capturing long-term dependencies, it has great potential for development in modeling time series. Significant breakthroughs that have revolutionized the TSF field during the last three years have built upon the Transformer's foundation by exploring the nature of the time series itself (different from textual sequences), such as periodicity.

Initially, many applications considered directly applying Transformer, such as Wu et al., who directly applied Transformer to the influenza-like disease forecasting task. LogTrans [55] emphasizes the specificity of time series data, which cannot be used only with a point-to-point attention mechanism similar to that in NLP realm but focuses on the contextual information around data points. Meanwhile, sparse attention is used to improve the efficiency of operation. The LogSparse self-attention mechanism proposed opts to select only elements from the range 1, 2, 4, 8, etc., adhering to the time step of the exponential growth interval. At the same time, a local attention mechanism is applied to enhance the influence of nearby elements to the current point. In addition, LogTrans integrates convolution and Transformer in its entirety; 1D convolution was used to extract the surrounding information of each node in the input sequence, and then multi-head attention mechanism was applied to learn the relationship between nodes and establish connections between parts with similar shapes. Similarly, Informer [14] also optimized Transformer from the perspective of efficiency to solve the problem of LSTF. Following the discovery that the attention scores show long tail distribution, the emphasis shifts to modeling significant relationships, specifically by forming sparse query–key pairs between critical queries and keys to reduce computational overhead. Simultaneously, the distilling operation is added between each attention block, and the input sequence length of each layer is shortened so that the model can accept longer input sequences. Furthermore, a generative decoder is designed to prevent the propagation of cumulative errors in the inference stage. TFT [12] comprehensively considers the necessary features for a high-quality TSF model, such as statistical properties like static information, adaptability to complex time series with increased noise, the application of DMS strategy, and the acquisition of uncertainty interval of predicted value. The entire model utilizes a combination of Transformer and LSTM. From the bottom up, the variable selection network (VSN) weights variables for feature screening to reduce unnecessary noise input; LSTM replaces the position embedding to capture long- and short-term information simultaneously; further, the gating mechanism is applied to identify the importance of different features and advance the network at the

same time; the temporal self-attention layer learns the long-term dependence of time series data and provides additional knowledge about the importance of features to facilitate explanation; finally, quantile regression is used to forecast the interval. Another interval prediction method, AST [56], embeds Transformer into GAN, and the α-entmax function is used to compute the sparse attention weights. This sparse Transformer functions as the generator, with the quantile loss measuring the disparity between the predicted output and ground truth serving as the generator's loss function. The discriminator is attached to the Transformer's decoder and classifies inputs to identify predicted values or ground truth, utilizing the cross-entropy loss function to calculate the adversarial loss. These two loss functions are leveraged for updating the sparse Transformer and the discriminator, respectively. The adversarial training process shapes the output distribution of the generator through backpropagation, thereby mitigating error accumulation. Furthermore, Autoformer [33] innovates based on the modular architecture from the Informer, with embedded decomposition modules, a commonly used preprocessing method in traditional statistics, to decompose the time series into the trend term and seasonal term. To break the bottleneck of information loss caused by sparse pointwise attention mechanism in previous models, a series-wise mechanism is proposed, which uses subsequences that are correlated with the current time series representation as the current time point's decision-making enhanced the capture of feature information with less computational complexity. Motivated by Autoformer [33], FEDformer [34] further exploits the characteristic wherein preserving a small number of frequency components in the frequency domain can be used to almost entirely reconstruct time domain signals without significant information loss. Raw time series are converted from the time domain to the frequency domain through Fourier transform and the attention part is performed in the frequency domain to improve the computational efficiency while better capturing the global view of time series. Aliformer [57] systematically performs a targeted exploration on the influence of known future knowledge on prediction in real life, especially in the field of sales. To address the issue, Aliformer utilizes a bidirectional self-attention mechanism that allows future information to be leaked. A knowledge-guide branch was designed to modify the attention map to minimize the effect of noise due to replaced learnable tokens from introduced future statistics. In addition, the emphasis on future knowledge is emphasized by adding span masking in the middle of the sequence.

Pyraformer [35] proposed a C-ary tree attention-based Transformer model. The mechanism relies on the path of pyramidal graph for information transfer. From the bottom to top is the fine-grained time series of the original input to coarse-grained ones obtained through convolution. The tree structure offers benefits by enabling direct interaction between any two nodes, effectively balancing the computational complexity and shortening the maximum signal traveling path. Preformer [58] follows the modular structure, focusing on multi-scale construction by choosing segments with different lengths. It leverages the characteristics of time series' continuous variations. Segment-wise attention was proposed to cope with its strong local characteristics by using matrix dot product to calculate the attention score instead of vectors. A new computing paradigm is designed to shift the query and value from the perspective of periodicity to make precise predictions. For time series decomposition, ETSformer [59] was inspired by the traditional Holt–Winters method and utilized exponential smoothing to extract the trend while incorporating a damping coefficient for stable trend generation. In other words, ETSformer improves upon Autoformer's simple detrending operation, which only relies on moving averages of sequences. The decomposed components offer interpretable insights into the prediction results. Furthermore, ETSformer employs residual learning to construct a deep architecture for modeling complex dependencies. Triformer [60] designed a model with patch attention combined with a triangle hierarchical structure. From bottom to top, the layer input shrinks exponentially to replace the stacked attention layer that requires an additional pooling layer to maintain consistent dimensions to ensure accuracy while achieving linear complexity. Different from the patch as an input unit, the complexity is here reduced only by treating the

pseudo timestamp as a query within the patch without considering the semantic features behind it. Meanwhile, according to the fact that the time series with different variables often have different temporal dynamics, the variable-specific modeling method is adopted, drawing on the idea of matrix factorization, to only learn the most prominent characteristics of each variable that are different from other variables and achieve lightweight generation of variable-specific parameters. TDformer [61] analyzed the applicability of time domain attention and two frequency domain attentions and a TSF model based on Transformer combined with MLP. Decomposition remains to be used as a preprocessing method. The trend items after decomposition are directly modeled by MLP while seasonal items are modeled using Fourier attention. The above two parts are concatenated as the results. Non-stationary Transformer [62] aims at tackling the over-stationarization problem caused by the traditional normalization methods, showing the attention matrix convergence phenomenon in the modeling of different time series after normalizing in Transformer-based models. By restoring statistical characteristics through normalization operations on inputs and de-normalization operations on outputs, Non-stationary Transformer designed a De-stationary Attention, a new form of attention from theory, where the attention matrix of the stationarized sequence is used to approximate the attention matrix of the original non-stationary sequence. Scaleformer [63] developed a multi-scale model-agnostic framework to sample time series at different sampling frequencies by average pooling. The framework employs a hierarchical forecasting strategy, progressing from coarse to fine-grained levels. The forecasting results at lower scales serve as inputs for the decoder at higher scales, with each scale equipped with its own forecasting module. Cross-scale normalization is incorporated to mitigate errors caused by distributions at different scales and in the data itself. Furthermore, adaptive loss, rather than Mean Squared Error (MSE) loss, is utilized for model training to alleviate error accumulation due to iterative processes. Quatformer [64] focuses on modeling complex cyclical patterns and mitigating the quadratic complexity in LSTF tasks. To solve these two challenges, learning-to-rotate attention (LRA) using quaternion was proposed, a mathematical tool used to represent rotation and direction; at the same time, a learnable period was introduced and phase information was used to describe complex periodic patterns. Meanwhile, linear complexity is achieved by stripping the global information and storing the global memory with an additional fixed-length latent series.

Persistence Initialization [65] designed a model-agnostic framework and its overall structure is similar to nesting the ReZero normalization outside the Transformer-based model, in fact, consisting of a residual skip connection and a learnable scalar gating mechanism. Simultaneously, Rotary encoding is used in the Transformer-based model, and replace the normalization in different positions, such as Pre- or Post-Layer Normalization, with ReZero, which can achieve a better effect. W-Transformer [66] is also a model-agnostic framework used for univariate TSF. The adopted strategy involves decomposing first using the MODWT algorithm, forecasting components separately after decomposition, and then performing aggregation before inverse MODWT. It leverages the advantage of the MODWT's shift-invariance property thoroughly.

Crossformer [67] focuses on modeling the relationship between multiple variables, aiming to simultaneously capture the dependences of time dimension and cross-variables simultaneously. Inspired by Vision Transformer (ViT) [68] in CV on image segmentation tasks, the input sequence is embedded into 2D vector array by patching (different from Triformer) with designed Dimension-Segment-Wise (DSW) embedding method. In order to keep both time and variable dimension information, a two-stage attention (TSA) layer is used to capture these two dependencies, respectively. In the variable dimension, an efficient routing attention mechanism is proposed. Patch merging can be used to obtain hierarchical representations at different scales. After linear projection to each layer, sum them up to obtain results. PatchTST [6] also adopts patching technique. Different from Crossformer, PatchTST adopts CI strategies. Each channel contains only one dimension of multivariate time series, which is processed separately and then input into Transformer

backbone, which is equivalent to processing univariate series. All series share the same embedding and Transformer weights, and then concatenate the predicted results along the dimension direction. Improved performance on LSTF task is achieved by this design. From this perspective, we can observe that simpler models may outperform complex ones. Continuing the trend of simplifying models, the innovation of TVT [69] lies in replacing the previous Time Point Tokenization (TPT) strategy with Time Variable Tokenization strategy (TVT), namely, it treats each time variable in the multivariate time series as a token instead of simultaneous data points which effectively lower over-smoothing degree and enhances the correlation between different variables. Moreover, TVT abandons the sinusoidal and cosinusoidal positional embedding strategy, questioned the effectiveness of the Transformer decoder and replacing it with a simple linear layer. Further, Conformer [70] integrates the previously used Fourier transform, multi-frequency sequence sampling and recursive trend-seasonal decomposition strategy to improve the performance in LSTF tasks with complex periodic patterns. In addition, an RNN based module is designed to model the global information and combines it with the local information extracted by sliding-window attention to compensate for the loss of fitting ability caused by linear complexity. Finally, a Normalizing Flow module uses the latent states generated in the above module to directly generate the distribution of future sequences to obtain more stable prediction results. CARD (Channeled Aligned Dual Transformer) [71] concentrates on Channel-Dependent (CD) design with patching technique to process input sequence, while capturing the temporal correlation and the dynamic relationship of different variables over time. Rethinking that widely used MSE loss function has the same weight when dealing with errors of different time steps, which fails to fully reflect the stronger correlation between near-future observations and historical ones. A signal decay-based loss function with superior performance is developed, weighted according to the importance of predictions within a finite horizon. JTFT [72] proposed a joint time-frequency domain Transformer. Inspired by the frequency sparsity FEDformer leveraged to extract temporal dependencies, a customize discrete cosine transform (CDCT) with strong energy compaction property was developed to calculate custom frequency-domain components, and then combined with time-domain patches to generate a joint time-frequency domain representation (JTFR) as input for subsequent models. To mitigate the loss of predictability caused by non-stationarity and extract the latest local relationships. Further, JTFT utilizes a two-stage approach, namely, using Transformer encoder and low-rank attention (LRA) layers inspired by the router mechanism in Crossformer to extract time-frequency and cross-variable dependencies, respectively. Client [73] continues the strategy in this stage of dividing time series modeling and multivariate relationship modeling into two modules and uses linear model and Transformer to carry out them, respectively. Taylorformer [74], an autoregressive probabilistic model, only focuses on better capturing target data distribution instead of efficiency, so only autoregressive modeling of the target variable is implemented. By integrating concepts from Taylor series and Gaussian processes, it enables Taylorformer to approximate continuous processes consistently to improved performance of the model. GCformer [75] combines the advantages of convolutional and self-attention mechanism, and adopts a two-branch design overall to extract local and global information. For long input sequences, a global convolutional branch is utilized which operates on all elements of the input data simultaneously. Additionally, three parameterization methods are introduced, including weights-decaying sub-kernels, kernel generation in the frequency domain, and perspectives from state space models, enabling slower sublinear parameter growth than linear while capturing long-term dependencies. Furthermore, integrating the local attention-based module enhances prediction accuracy. SageFormer [76] designed a general framework that can be applied to various Transform-based models, aiming to effectively introduce the relationship between various variables, bringing information gain while avoiding redundant information from interfering with the model training process. The input data are still processed with patching technique and global information of each variable sequence is extracted by adding global tokens to each sequence, and then multi-

variable relationship is extracted by graph learning. DifFormer [77] is a multi-task model for time series analysis, which adopts a two-stream structure to extract information from the time domain and the frequency domain, respectively. Meanwhile, breaking through the traditional statistical approach of viewing neural differencing as a preprocessing technique, DifFormer uses it as a built-in module to capture temporal variances progressively and flexibly, thus displaying the desired properties in various temporal modes. DSformer [78] incorporates a parallel structure throughout the whole model. Designed double sampling (DS) block performs down sampling operation through larger sampling intervals to avoid the influence of local noise and obtain more global information. At the same time, the original data are segmented by the piecewise sampling method in parallel to obtain local information. The following temporal variable attention (TVA) block also performs attention operations from the time and variable dimensions in parallel. Finally, a TVA block is used to fuse the information of the above two branches appending MLP-based generative decoder to obtain the results. SBT [79] is also a multi-task model, in terms of TSF, only for multivariate TSF with single-step forecasting. It applied sparse and binary-weighted Transformer with attention masks to reduce the computational complexity which only focus on the current time step's attention, allowing the propagation of the entire input sample to multiple attention layers, helping to keep relevant historical information for downstream layers and thus improving the model's performance and accuracy with lightweight implementation. PETformer [80] explored the novel structure of Transformers and the effective modeling of multivariable relationships. After processing the input data by patching techniques, integrate the historical and future segments as inputs to the Transformer in a placeholder enhanced manner, where each placeholder represents a part of the future data to be predicted. The distinction between encoder and decoder is eliminated, making it possible for the predicted component to access historical sequence information more naturally. In order to incorporate the relationship between variables into the model, the Inter-Channel Interaction module is incorporated at the same time.

TACTiS [81] combines the Transformer and the probabilistic copula model for complex real-world time series with irregular sampling and missing values. It primarily uses a copula-based decoder to mimic the properties of non-parametric copulas, which are flexible and can adapt to data with varying features and structures. Different from the TACTiS, TactiS-2 [82] designed the dual encoder and the decoder to generate the distributional parameters for the marginal CDFs and the copula, respectively, showing the training curriculum in a two-stage approach. These approach makes the number of distributed parameters varies linearly with the number of variables, while avoiding the problem of training dynamic difference and suboptimal prediction. PrACTiS [83] improves the encoder of TACTiS and expands it by integrating perceiver model as an encoder to enhance the expression of covariates' dependencies. At the same time, midpoint inference and local attention mechanisms are combined to solve the high computational complexity problems related to self-attention mechanisms.

iTransformer [84] presents an innovative inverted perspective to rethink the appropriate responsibilities of Transformer in modeling time series data. Specifically, the token embedding is constructed by variate dimension instead of the original token embedding in time dimension. And attention mechanism is utilized to learn the relationships between variable sequences, while an inverted feedforward network is employed to extract complex temporal features. BasisFormer [85] broke through the previous simple basis learning methods (such as generating only corresponding basis coefficients) and obtained the basis through adaptive self-supervised learning with bidirectional cross-attention to calculate the similarity coefficient to select and consolidate the basis in the future view to achieve accurate future predictions. MTST [86] further learns temporal patterns with different frequencies by controlling the size of the patch. Each layer of the stacked network uses a multi-branch structure to process patches of different sizes, respectively, with independent Transformer. Finally, the results of each branch are fused as the input of the next layer to learn the representation of time series at different resolutions.

From the recent volume of work, it is evident that Transformer-based models undergo rapid iterations and updates, which are compiled in detail in Table 3 along with the relevant papers. It has made significant explorations in the field of TSF, implying that mainstream trends can be easily extracted. We explore these trends from the following perspectives:

Table 3. Summary of Transformer-based TSF models involved in our taxonomy.

Models	Journal/Conference	Category	Year
InfluenzaTransformer [4]	Arxiv	Transformer based	2020
LogTrans [55]	NeurIPS	Transformer based	2019
TFT [12]	IJoF	Transformer based	2021
AST [56]	NeurIPS	Transformer based	2020
Informer [14]	AAAI	Transformer based	2021
Autoformer [34]	NeurIPS	Transformer based	2021
Aliformer [57]	Arxiv	Transformer based	2021
Pyraformer [35]	ICLR	Transformer based	2022
Preformer [58]	ICASSP	Transformer based	2023
FEDformer [34]	ICML	Transformer based	2022
ETSformer [59]	ICML	Transformer based	2022
TACTiS [81]	ICML	Transformer based	2022
Triformer [60]	IJCAI	Transformer based	2022
TDformer [61]	NeurIPS Workshop	Transformer based	2022
Non-stationary Transformer [62]	NeurIPS	Transformer based	2022
Scaleformer [63]	ICLR	Transformer based	2023
Quatformer [64]	KDD	Transformer based	2022
Persistence Initialization [65]	APPL INTELL	Transformer based	2022
W-Transformers [66]	ICMLA	Transformer based	2022
Crossformer [67]	ICLR	Transformer based	2023
PatchTST [6]	ICLR	Transformer based	2023
TVT [69]	Arxiv	Transformer based	2022
Conformer [70]	ICDE	Transformer based	2023
CARD [71]	ICLR	Transformer based	2024
JTFT [72]	Arxiv	Transformer based	2023
Client [73]	Arxiv	Transformer based	2023
Taylorformer [74]	ICML Workshop	Transformer based	2023
GCformer [75]	CIKM	Transformer based	2023
SageFormer [76]	IEEE Internet Things J.	Transformer based	2024
DifFormer [77]	TPAMI	Transformer based	2023
DSformer [78]	CIKM	Transformer based	2023
SBT [79]	KDD	Transformer based	2023
PETformer [80]	Arxiv	Transformer based	2023
TACTiS-2 [82]	Arxiv	Transformer based	2023
PrACTiS [83]	Arxiv	Transformer based	2023
iTransformer [84]	ICLR	Transformer based	2024
BasisFormer [85]	NeurIPS	Transformer based	2023
MTST [86]	ICAIS	Transformer based	2024

First, **design for attention**. **In the initial phase**, Transformer-based models focused on improving efficiency by building sparse attention, Longformer designed three sparse attention mechanisms to reduce the time complexity of Transformers to linear growth with exponential growth of sequence length, and later Logtrans [55], AST [56] and Informer [14] adopted this design concept. It greatly improved the efficiency of TSF, but the attention operation at this stage was still based on point-wise and did not properly take into account the characteristics of local continuous variations in time series. **In the second stage**, researchers believed that sparse point-wise attention would sacrifice information utilization, so models in this stage often adopted series/segment/patch-wise attention to reduce the number of elements participating in attention calculation, including Autoformer [33] and Preformer [58]. Lower complexity was realized, and the calculation formula of attention was redesigned according to the periodicity of time series. On the basis of segment-wise

attention, the model **in the third stage** fully redesigned the hierarchical structure of stacked attention layers, such as presenting a triangle structure or tree structure. With this approach, each node only needs to perform attention calculation with adjacent nodes rather than on all positions of sequence. This approach further reduces the number of operations, thereby decreasing the complexity, such as Triformer [60] and Pyraformer [35]. The design of the attention mechanism **in the fourth stage** is based on decoupling of the dependencies among variables and temporal dependencies in multivariate time series. Several models adopt the strategy of modeling temporal dependencies and cross-variable dependencies, respectively, so temporal attention and cross-variable attention are designed synchronously, such as in DSformer [78], or only one of them is modeled by the attention mechanism, while another one is modeled by other models such as Client [73] and PETformer [80], etc.

Second, **input format of time series data**. It is widely recognized that NLP is the initial development area of Transformers, which primarily centered around textual data. Usually, a segment of text is selected as the model input, with each token in the text being mapped to a vector. The vector combined with position embedding, type embedding, etc. is input into the model. When applying Transformer to time series, the earliest approach is to directly input, specifically, each time step is represented as a vector, and then combined with position embedding, time-related embedding and Holiday embedding related to human activities are combined as the input of Transformer-based model. The work in this stage includes InfluenzaTransformer [4] and Informer [14], etc. **In the second stage**, the correlation of the time series itself and the nature of local continuous variations are completely considered, by combining contextual information rather than only studying individual points as part of the input. From above articles, it is evident that by appending LSTM or convolutional layer behind the input layer to assist in establishing the context information of each point in the time series, the advantages of these two kinds of networks in capturing long- and short-term dependencies are cleverly used to improve the prediction performance, such as Logtrans and TFT. **In the third stage**, it is apparent that all the models discussed in survey published over the past two years are influenced by ViT, which has achieved excellent performance in field of CV, and adopt patching technique to process input data and conduct modeling. It results from summarizing the similarities between image data and time series data, researchers in the field of TSF believe that a single pixel in an image is equivalent to each time point in a time series, and research at single point is meaningless, only aggregation can contribute to training. The specific method involves patching input windows, regarding the subsequences within each window as a whole, mapping them through MLP layers, and ultimately obtaining a vector as subsequent input. This approach obviously improves model efficiency by shortening the length of the input sequence, such as PatchTST [6], SageFormer [76], and PETformer [80]. However, an increasing amount of research is starting to concentrate on the flexible way of patch generation, considering the multi-scale patching approach instead of applying fixed window for patching previously. The mainstream approach is to design a stacked network architecture, where different layers or different branches at the same layer handle patches with different sizes to realize different scales of modeling, such as MTST [86]. The combination of this multi-scale patching technique and Transformer is more flexible and has become one of the most important directions of current innovations.

Third, **modular design of Network Architecture**. Starting from Informer [14], Autoformer [33], FEDformer [34], the mainstream models continue the modular design. Building upon this foundation, the gradual combination of methods originally used for preprocessing are served as in-bult modules, leading to further advancements in TSF, especially in the task of forecasting data with periodicity.

Fourth, **the modeling of relationships between multiple variables in multivariate TSF**. In the initial TSF methods, the conventional approach was similar to that of univariate TSF, such as mapping multi-dimensional values into multi-dimensional vectors [14,33–35]. Later on, models like PatchTST found that better results can be achieved by using CI, i.e., a strategy that completely disregards the relationship between the variables, and

models each variable individually. It may result from the increased noise and redundancy when introducing multiple variables. According to recent works, some researchers insist that capturing the dependencies between multiple variables provide useful information for prediction, but it is necessary to decouple the time dimension modeling from the multiple-variables relationship modeling to prevent problems such as overfitting or complex computation. For example, PETformer introduced an Inter-Channel Interaction module to keep the relationship between the channels.

However, it is essential to recognize that despite the gradual improvement in predictive performance, certain models still have limitations. For instance, some models struggle to capture temporal patterns in poorly structured time series data [35,70]. Some may overly rely on identifying periodic features in time series data, making them less effective for training on datasets with weak periodicity [33]. Additionally, some of them overemphasize extracting future information, thereby restricting the scope of tasks to those entailing easily obtainable or predictable future information [57]. Therefore, when applying them to tasks in specific domains, we need to further analyze the models themselves.

3.4. MLP-Based Models

With Transformer-based models propelling the TSF realm into a period of rapid advancement, Zeng et al. questioned the effectiveness of Transformers in LSTF tasks and proposed DLinear [26], in which the author argued that the ordering of the time series itself is very important. Although the position embedding retains some ordering information, the self-attention mechanism has permutation invariant and inevitably loses some temporal information. Therefore, DLinear simply performs the time series decomposition first, fits the trend and seasonal terms obtained after decomposition with two linear layers, respectively, and finally merges them, which outperforms all the Transformer-based models at that time in terms of the data with obvious trend, which brings about a competition for optimal performance between MLP-based and Transformer-based models among a period.

N-BEATS [36] is a relatively early work using MLP architecture to carry out TSF and built a method for univariate TSF which is as simple and effective as possible and has certain interpretability. It serves as the foundation for many subsequent models. N-BEATS adopts "stack-block" structure overall, and time series decomposition is achieved through multi-layer fully connected networks. According to the designed Doubly Residual Stacking, each layer fitted part of the time series information, that is, the residual fitting of the previous layers, and iteratively refined the prediction process. The interpretability of N-BEATS is achieved through the concept of basis, when projecting onto the basis, learn the coefficients of each time series in a flexible way. By weighting and combining the basis based on the coefficients, it is easier to observe which basis is more important to current output. N-BEATSx [87] further extends the N-BEATS to enable it to input and process covariates while maintaining interpretability of the covariates' influence on the prediction. Furthermore, NHITS [88] improves N-BEATS for the perspective of long-horizon forecasting (different from LSTF, which just refers to a relatively long forecasting horizon), in order to alleviate increasing error as the forecast horizon increases. In brief, the concept of input-downsampling and output-upsampling is incorporated, and time series is divided into multiple granular sequences through downsampling, which reduces the number of parameters and the complexity to achieve higher efficiency. For periodic sequences, DEPTS [89] applied Fourier transform to extract periods based on N-BEATS, where MLP is used to extract periodicity dependencies to solve the challenges of complex dependencies and multiple periodicities in periodic time series. Works at this stage benefited from the Residual learning introduced by N-BEATS firstly, which showed the potential to build a deep learning architecture with better expressive and generalization capabilities, at the same time, explored the periodicity of time series adequately. These ideas serve as references for follow-up works.

The second stage of development has been influenced by some advanced Transformer-based models, which focused research on the LSTF problem. The emergence of DLinear [26] prompts researchers to think about the necessity of complex models. At the same time, the

Mixer structure in the field of CV has gradually been applied to the TSF realm promoting the trend of structural simplification developing. For example, FreDO [90] proposed a frequency-domain-based LSTF model. Through experimental comparison, it is found that it is easier to capture multiple periodicities in the frequency domain, and periodicity-based mining is conducive to long-term forecasting. Same as TimesNet [50], LightTS [27] also converts the 1D time series into 2D in structure and capture short-term local patterns and long-term dependencies, respectively, through interval sampling and continuous sampling. Finally, MLP is used to extract the features, which reduces the complexity and running time while ensuring the accuracy. MTS-Mixers [29] further applies the structure of Mixer to TSF by designing a factorized temporal and channel mixing structure. Time series is divided into multiple subsequences, with each subsequence undergoing temporal information learning independently. Subsequently, these subsequences are concatenated in their original order and redundancy in channel dimension is addressed through matrix decomposition. TSMixer [91] studied the problem that multivariate TSF models sometimes cannot outperform univariate TSF models. In order to use cross-variate information assist in improving forecasts, by introducing cross-variate feed-forward layers, extend the capabilities of linear models and model the time dimension and the feature dimension, respectively. TiDE [24] designed an encoder–decoder model based on MLP with CI strategies. Static covariates and dynamic covariates information are utilized. Both the encoder and decoder of the model are stacked with multiple MLP based residual blocks. Through the global residual connection with only one linear layer, it is guaranteed that TiDE can theoretically achieve similar performance to DLinear. Koopa [92] innovatively combined Koopman theory with TSF and designed the model from the perspective of dynamic system. The model itself was realized by a very simple MLP, which could achieve faster forecasting speed, and more accurate results compared with other models. TSMixer [25] designed by IBM adopts patching technique for input data and considers intra-patch, inter-patch, and cross-variable information interaction. By using CI backbone with cross-channel forecast reconciliation heads, it achieves better performance than other mixing methods. FITS [93] shows the similar idea to DLinear. The difference is that FITS operates in the frequency domain. After converting the time series data into the frequency domain by using Discrete Fourier Transform (DFT), and then passing them through complex linear transformation, finally inverse Fourier transform was used to return back to the time domain to obtain the result. TFDNet [94] adopts a branching structure overall, capturing long-term latent patterns and temporal periodicity from the time domain and the frequency domain, respectively, while combining Channel-Dependent (CD) strategy and trend-seasonal decomposition method. FreTS [95] still adopts the application of MLP in the frequency domain, by separately considering the complex numbers' real mappings and imaginary mappings before stacking the output to obtain output results. This kind of method performed on both channel dimension and temporal dimension in the frequency domain.

So far, we have summarized the achievements from four network architecture in recent years, and showed in Table 4. The development of TSF architecture from RNN/CNN-based models to Transformer-based models, and then to the popularity of MLP-based models, the whole structure presents a simplified process. However, this kind of simplification of structure is not unique to the field of TSF. We can observe that after N-BEATS [36] proposed to perform TSF tasks in a fully connected way, the models were expanded just around how to make longer forecasting horizons or increasing functionality over almost two years. No advanced models came out, until MLP-Mixer achieved a performance comparable to CNN and ViT [68] on ImageNet datasets in the field of CV with a pure MLP architecture, and then achieved competitive results on NLP tasks with a small number of parameters. The research of MLP-based models has been further deepened. We observed that recent models carefully incorporate innovations in other architectures, especially Transformer-based ones, such as processing data in the frequency domain, and reconstructing time series data in 2D, etc. All developments are accompanied with the innovations in other deep learning fields.

Table 4. Summary of MLP-based TSF models involved in Taxonomy.

Models	Journal/Conference	Category	Year
N-BEATS [36]	ICLR	MLP based	2020
N-BEATSx [87]	IJoF	MLP based	2022
NHITS [88]	AAAI	MLP based	2023
DEPTS [89]	ICLR	MLP based	2022
DLinear [24]	AAAI	MLP based	2023
FreDo [90]	Arxiv	MLP based	2022
LightTS [27]	Arxiv	MLP based	2022
MTS-Mixers [29]	Arxiv	MLP based	2023
TSMixer [91]	Arxiv	MLP based	2023
TiDE [25]	Arxiv	MLP based	2023
Koopa [92]	NeurIPS	MLP based	2023
TSMixer [26]	KDD	MLP based	2023
FITS [93]	ICLR	MLP based	2024
TFDNet [94]	Arxiv	MLP based	2023
FreTS [95]	NeurIPS	MLP based	2023

4. Experimental Evaluation Metrics and Datasets

This chapter is divided into two parts. First, we will introduce the evaluation metrics used to evaluate the performance of deep learning models, especially point-forecasting models in TSF tasks in detail. Furthermore, we collect the commonly used public datasets in recent years, focusing on their data categories, number of variables, sources, data characteristics, etc., which is convenient for researchers to choose according to the preferences of models and research field during their research process, and is helpful for gaining a thorough understanding of the research background.

4.1. Evaluation Metrics

- **Mean Squared Error (MSE)** MAE is one of the most commonly used evaluation metrics to measure the degree of difference between the predicted value and the actual value of the model, especially when it is desirable to penalize larger prediction errors more severely. For each sample, the squared difference between the predicted and true value is calculated, and then is summed up across all samples to obtain the MSE. The formula is as follows:

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

- **Mean Absolute Error (MAE)** Like MAE, most models designed for the LSTF task on the same open-source datasets consider it together with MSE. MAE provides a simple and easy-to-understand way to measure the overall performance of models. MAE calculates the absolute error between the predicted value and the actual value for each sample and divides the sum by the number of samples to obtain MAE. The formula is expressed as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

- **Root Mean Squared Error (RMSE)** Measures the average magnitude of the error between the predicted and actual values. RMSE can be chosen when it is desirable to combine the advantages of MSE in the evaluation and express the evaluation metrics in the same unit as the target variable. A lower value of RMSE indicates a closer proximity between the predicted values and the actual values, signifying superior model performance. The formula is as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4}$$

- **Mean Absolute Percentage Error (MAPE)** It is calculated as the average of the percentage error between the predicted and actual values. The value of MAPE is between 0 and positive infinity, the closer the value of MAPE is to 0, the more accurate the prediction is. MAPE can be chosen when we need to visually assess the accuracy of the model's predictions because it expresses the error as a percentage, which also means that MAPE can be used when comparing the relative performance of different datasets. The formula is as follows:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \tag{5}$$

- **Symmetric Mean Absolute Percentage Error (SMAPE)** It considers the symmetry between predicted and actual values, that is, the proportionality between predicted and actual values. When there is some kind of fluctuation in the dataset, these fluctuations may be periodic or symmetrical, and this pattern often emerges as a very important feature of data, so we need to take it into account. The formula is as follows:

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{||\hat{y}_i - y_i||}{(|\hat{y}_i| + |y_i|)/2} \times 100\% \tag{6}$$

- **Mean Absolute Scaled Error (MASE)** It measures the forecasting performance of a TSF model relative to a simple benchmark model. MASE compares the model's MAE with the MAE obtained through a simple method to examine the model's performance relative to the benchmark model. The smaller the value of MASE, the better the model's performance relative to the benchmark model and is suitable for evaluating the forecasting accuracy between different datasets. MASE has different calculation methods for non-seasonal time series and seasonal time series, and the formulas are specified as follows:

$$MASE = \frac{MAE}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|} \text{ or } MASE = \frac{MAE}{\frac{1}{n-m}\sum_{i=m+1}^{n}|y_i - y_{i-m}|} \tag{7}$$

- **Overall Weighted Average (OWA)** In the article of N-BEATS [36], it is mentioned as a metric used in the M4 dataset to rank entries. OWA provides an easy method to evaluate multiple metrics in a comprehensive way, which facilitates decision-making. Here, we generalize the formula by not explicitly including multiple evaluation metrics but rather describe it in a more generalized way:

$$OWA = \sum_{i=1}^{n} w_i \cdot x_i \tag{8}$$

- **Normalized Root Mean Squared Error (NRMSE)** Formulaically, it is the ratio between the RMSE and the mean of the actual observations. By standardizing RMSE, NRMSE removes the effect of data size and can be used to compare models across different scales:

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=2}^{n}(y_i - \hat{y}_i)^2}}{\overline{y}} \tag{9}$$

- **Normalized Deviation (ND)** By calculating the mean of the absolute differences between observed and predicted values across all samples, divided by the observed values. A smaller value of ND indicates a smaller prediction error, reflecting better model performance:

$$ND = \frac{1}{n}\sum_{i=2}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{10}$$

The pros and cons of the applicability of the above methods are compared in Table 5, and Table 6 displays the meaning of the involved notations.

Table 5. Comparison of several evaluation metrics.

Metric	Scale-Free	Temporal Structure
MAE	✘	✘
MSE	✘	✘
RMSE	✘	✘
MAPE	√	✘
SMAPE	√	✘
MASE	√	√
NRMSE	√	✘
ND	✘	✘

Table 6. Notions in evaluation metrics.

Notions	Meaning
n	number of samples
y_i	observed value for the ith sample
$\hat{y}_i$	predicted value for the ith sample
m	the seasonal period
w_i	the weight of ith evaluation metric
x_i	the value of ith evaluation metric
$\overline{y}$	the average of the observed value

While $R^2\left(1-\frac{MSE}{Var(y)}\right)$ is a commonly used statistical metric, it assumes a linear relationship between the dependent and independent variables, which may not hold true in deep learning models where the relationship between variables is often non-linear. Consequently, using R^2 may not fully capture the complexity and non-linear relationships of the model. Additionally, R^2 does not penalize overfitting and tends to reward an increase in model complexity. In contrast, metrics such as MSE and MAE are more intuitive in TSF scenarios. Therefore, in the context of deep learning-based time series forecasting, R^2 may not effectively provide its value compared to regression problems.

4.2. Commonly Used Public Datasets

We collect commonly used public datasets in recent years, as shown in Table 7.

Table 7. Commonly used public datasets.

Dataset	Source/Paper	Variate Number	Feature Type	Description
ETTh	Informer [14]	7	real	Electricity Transformer Temperature for 1 h level
ETTm	Informer [14]	7	real	Electricity Transformer Temperature for 15-min level
ECL	UCI [1]	321	real	Hourly electricity consumption of 321 clients from 2012 to 2014
Weather	BGC Jena [2]	21	real	Meteorological indicators collected every 10 min
ILI	CDC [3]	8	real	Weekly ratio of influenza-like illness to total patients reported
EXCHANGE	LSTNet [37]	9	real	Daily exchange rate of 8 countries from 1990 to 2016
Traffic	Cal-trans [4]	862	real	Hourly road occupancy of San Francisco freeways
Solar-Energy	LSTNet [37]	137	real	Solar power production sampled every 10 min
PEMS03	SCINet [22]	358	real	Traffic dataset starting from 1 May 2012 collected every 5 min
PEMS04	SCINet [22]	307	real	Traffic dataset starting from 1 July 2017 collected every 5 min
PEMS07	SCINet [22]	883	real	Traffic dataset starting from 1 May 2017 collected every 5 min
PEMS08	SCINet [22]	170	real	Traffic dataset starting from 1 March 2012 collected every 5 min

[1] https://archive.ics.uci.edu/mL/datasets/ElectricityLoadDiagrams20112014 (accessed on 1 January 2023); [2] https://www.bgc-jena.mpg.de/wetter/ (accessed on 1 January 2023); [3] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html (accessed on 1 January 2023); [4] https://pems.dot.ca.gov/ (accessed on 1 January 2023).

5. Challenges and Possible Solutions

5.1. Distribution Shift

Whether differences arise between the distributions of training and test data of a predictive model or among the distributions of different input sequences can lead to degradation in the performance of the predictive model. Therefore, the problem of distribution shift is urgently needed to be solved, and the essential idea behind this problem's solution is to allow the model to fit the stable part extracted from the data. A brief organization of some current works is shown in Table 8, which divides it into two categories: Model-Agnostic ones and in-model ones based on the relationship between the method and the model.

Some existing works, such as RevIN [96], reduce the difference of data distribution and improves model performance by removing non-stationary information such as mean and variance from input sequences. In the output stage, this information is used to restore the sequence, so as to prevent some conventional normalization methods from eliminating non-stationary information which is important for forecasting the future values. Since occasionally, non-stationary information may provide some important clues about the dynamics of the data. To some extent, it benefits prediction. This method can be applied as a plug-in to numerous models such as Informer [14] and TDformer [61]. Compared with the effect of the time series model itself, it can be found that the use of this framework can improve the prediction effect. Utilizing it enhances performance when compared to the effects of these TSF models themselves.

Another category of work, such as AdaRNN [97], uses maximum entropy (ME) to divide the sequence into segments, and then learns the common knowledge of segments from different distributions of the sequence, which uses domain adaptation as its primary method and allows the model to generalize across different distributions. Although the problem of distribution shift is not explicitly pointed out in Non-stationary Transformer, the designed simple parameter-free Series Stationarization is good enough for stationarize time series, thanks to its redesigned attention formula that can effectively alleviate over-stationarization problem.

In addition, Han et al. [98] mentioned in their article that CI strategy can alleviate distribution drift (note: there is no clear distinction between drift and shift in some articles) problem. Therefore, researchers may need to consider the impact of more factors and strategies when mitigating this type of problem.

Table 8. Summary and classification of solutions for distribution shift in TSF.

	Type	**Method**
Distribution shift	Model-Agnostic	DAIN [99], RevIN [96], Dish-TS [100]
	In-Model	AdaRNN [97], Non-stationary Transformer [62] etc.

5.2. Irregular Time Series Forecasting

Irregular time series is a common type in real-world data, usually referring to uneven intervals between samples or observations, frequently observed in financial and medical data. However, there are some issues with modeling irregular time series because most of the models we employed rely on the assumption of uniform sampling to model the sequence. Typical techniques include imputation and resampling to make these data with equal intervals to facilitate prediction by conventional forecasting models. However, these operations still have negative influences on downstream tasks, such as over-fitting and introducing noises. Some recent works are in [101–103].

5.3. Time Domain or Frequency Domain

Through the above summary, we have observed that in recent years, it is an important innovation point to operate time series data in the frequency domain through conversion algorithms, such as Fourier transform [34,59,90], in order to improve the performance of

TSF. In particular, there are a lot of achievements in solving the LSTF problem, mostly due to the fact that periodic information is more easily obtained in the frequency domain, and the sparsity of the frequency domain. TDformer [61] analyzes the application and selection of the time domain and two frequency domain attentions in detail. At the same time, some models began to pay attention to time and frequency dimension of time series data [72] simultaneously, allowing the model to leverage the benefits in both domains, such as the ability to record the dynamic evolution of data in the time domain and identify periodic patterns in the frequency domain, in order to enhance comprehension of time series data and effectively extract long-term and local dependencies. However, we still need to consider the potential issue of information redundancy that arises from this combination.

6. Future Directions

The proposed taxonomy provides a framework for researchers to understand the current research, while also discussing some challenges faced and corresponding solutions. However, some areas of TSF learning seem to be understudied, or are only at an early stage, and we outline several promising directions for future research.

6.1. Large Language Model for Time Series Forecasting

Large language models (hereinafter called LLMs) have shown great advancements in fields of CV and NLP, and their strong ability of generalization enables them to perform well even when dealing with limited datasets. The large number of parameters and complex network structure make it better to capture the dependencies in time series. Over recent months, there has been a surge of LLMs developed for TSF field. We have summarized them, as shown in Table 9.

Table 9. Summary and classification of LLMs for TSF are cited.

	Type	Method
LLMs (Large Language Models)	Transferring LLMs in NLP to TSF	LLMTIME [104], Time-LLM [105], FPT [106], LLM4TS [107], TEST [108], TEMPO [109], LSTPrompt [110], AutoTimes [111]
	Training LLMs for TSF	TimeGPT-1 [112], Lag-Llama [113], MOIRAI [114]

The earliest type of work was to transfer LLMs used in the NLP field to TSF. This kind of method's primary goal is to address how to load time series data into LLMs for handling. There has been a lot of research in CV field, such as processing from data-level or adding cross-modal adaptor to models. For example, LLMTIME [104] designed a special tokenizer to address the problem that existing LLMs, such as GPT-3 [115], cannot be used to encode time series directly. And LLMs are successfully applied to the zero-shot TSF tasks which eliminates the requirement for collecting historical data on the target datasets, and after pre-training with other text corpus datasets, the pre-trained model is directly used for forecasting. Although the forecasting effect of LLMTIME is inferior to the SOTA's TSF models and cannot handle large context windows well. However, the most recent LLMs listed below are progressively addressing this drawback, such as AutoTimes [111]. It will be a very meaningful research direction for dealing very long sequence forecasting. Moreover, in real-world datasets, particularly in financial datasets, the data are relatively messy, noisy and the availability of a substantial amount of data is limited. So, it is easy to encounter the situations of overfitting or parameter stability with some traditional deep learning methods such as DLinear [26]. Therefore, the application of this kind of model is very practical significance.

After deeply exploring the characteristics of time series data, Time-LLM adopts CI and patching strategies to make the model more suitable for time series tasks, and realizes cross-modal alignment between time series and context by reprogramming methods. At the

same time, two extra modules designed by Time-LLM have a small number of parameters, and only require a small number of time series samples for fine-tuning, which means that Time-LLM has a strong ability of few-shot forecasting. Moreover, by introducing prior knowledge by prompts, Time-LLM [105] has a wider range of predictive scenarios, as does LSTPrompt [110]. FPT [106] only trains positional embedding, input embedding, output linear Layer, normalization layers with freezing self-attention and FFN to allow inputs in different domains. Meanwhile, the method of obtaining token after patching is more suitable for time series tasks. LLM4TS [107] also uses patching technique and CI strategy to tokenize time series data, and further integrate temporal information properly. Other models in this category include TEST [108] and TEMPO [109].

Another type is to construct LLMs directly in the field of TSF. For instance, Lag-Llama [113] endeavors to build a foundational model for TSF with neural scaling laws. TimeGPT-1 [112] constructed a GPT-based model, applying a large amount of time series data in various domains. In addition, MOIRAI [114] launched LOTSA, the largest collection of open time series datasets, and then trained on the datasets. However, the first type is relatively easier to implement, less complicated, and has better application prospects.

6.2. Graph Neural Network for Modeling Cross-Variable Dependences

The methods discussed in this paper for modeling cross-variable rarely involve the utilization of Graph Neural Network (GNN). In fact, GNNs have been continuously studied in the field of TSF for a very long period and the dependencies between variates can be efficiently learnt by treating the variates as nodes, the relationships between them as edges and modeling them in the structure of a network or a graph. Particularly impressive outcomes have been obtained in spatial TSF tasks, such as traffic flow forecasting. Although it may require more computational cost to use GNNs to build multivariate relationships in normal TSF task, but researchers can choose different GNNs according to specific application scenarios. It is a very worthwhile area for future research.

6.3. Developping New Loss Functions

The commonly used loss function for TSF is MSE [6,14,45], which aims to minimize the difference between the predicted value and the actual value to optimize the model parameters. It is a point-to-point approach. However, it is widely acknowledged that time delay is a typical TSF problem, and many models have taken this problem into consideration in design, such as Autoformer [33]. Similarly, in the loss function, we can better optimize the model by adding penalty terms or assigning different weights to the loss. Although there are already relevant loss functions, these loss functions are more likely to apply in specific fields, such as finance and weather forecasting. Therefore, a general adaptive loss function which considers the time delay between the output sequence itself and actual values will hold promising prospects for the future.

6.4. Generative Models

Most of the models used in the TSF domain are usually discriminative models, in contrast to generative methods which are less used in solving TSF problems, such as Diffusion, GAN, and Normalizing Flows. It is mainly because TSF is typically categorized as a discriminative issue, which is easier to implement and straightforward. Meanwhile, discriminative models are more concerned with achieving the mapping relationship between inputs and outputs, that is, forecasting future outputs based on known inputs. However, generative models need to model the joint distribution of inputs and outputs. Therefore, certain probabilistic TSF models often adopt generative-style ideas. In real-world datasets, there are often some small samples with missing values or irregular-sampling datasets. In this case, applying generative models is more appropriate and somewhat eliminate the requirement for preprocessing, and it represents a very promising direction for research. A simple summary of recent generative models for TSF and categorized cases is provided in Table 10.

Table 10. A simple summary of generative models in TSF.

	Type	Method
Generative model	GAN	ForGAN [116], AST [56]
	Diffusion	TimeGrad [117], CSDI [118], TSDiff [119], TDSTF [120]
	Normalizing Flows	MAF [121], MANF [122]
	VAE	D^3VAE [123]
	SSM	DSSM [124]

6.5. MAMBA

MAMBA [125] is a new architecture that has been proposed recently which claims to perform better than Transformer. It is widely known that Transformer is the main component of LLMs. Therefore, an architecture better than Transformer will undoubtedly lead to the future development of the TSF realm. Moreover, MAMBA can remember the previous information while parallelizing training like Transformer, as well as achieve faster inference and its time and computation complexity are linearly related to the sequence length. MAMBA simplifies the deep sequence model architecture by utilizing a selective SSM architecture that can selectively decide whether to focus on or ignore inputs, and Transformer's MLP block and then merges them into a single block.

State Space Model (SSM) mentioned before provides another perspective to understand time series. Almost all the models we learned before, can be represented by SSM through the construction of state function and observation function. Before MAMBA appeared, there had been a lot of work combining SSM with deep learning models, such as DSSM [124]. Combining their benefits allows the model to learn similar patterns from a huge number of sequences and features, as well as to make models have certain interpretability, which offers a promising direction for future exploration.

6.6. Domain-Specific Time Series Forecasting

The deployment of high-performance deep learning-based TSF models is widely recommended in specific domains in practice. Updating the algorithms with domain-specific requirements has great potential for development.

6.7. Exploring the Integration of Traditional Statistical Methods with Deep Learning Methods

In the past few years, we have explored the application of models or techniques from traditional statistics in time series analysis, such as trend-seasonal decomposition and SSM, and combined them with deep learning models to achieve huge success. So, continuing to explore along this direction, while utilizing the characteristics of the time series data itself, to further enhance the interpretability and improve the forecasting performance simultaneously.

7. Conclusions

In this survey, we conducted the most comprehensive overview so far of a promising field in AI: TSF. First, we outlined the historical development of TSF and its early method evolution in academia. Then, we systematically organized the reviewed methods in a new taxonomy of architecture designs, categorizing them into four groups: RNN-based, CNN-based, Transformer-based, and MLP-based methods. In each category, we offered thorough descriptions of the important models with their innovation points and trend analyses for recent years, which were not available in previous surveys. The included papers were also organized in tabular format at the end of each category. We observed a trend towards simplification in model structures and modeling approaches in the TSF domain. In addition, commonly used datasets and evaluation metrics for TSF in recent years were summarized. Finally, to promote the development of the TSF field, this paper offered seven potential development directions, including research into LLMs and novel architectures, anticipating their ability to enhance prediction performance and address the increasing demand for prediction diversity, robustness, and applicability driven by the

growing availability of computational resources. We believe that this survey will inspire further innovations in the TSF field and provide a valuable guide for researchers and newcomers seeking to enter the domain of deep learning-based TSF.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

References

1. Salinas, D.; Flunkert, V.; Gasthaus, J.; Januschowski, T. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *Int. J. Forecast.* **2020**, *36*, 1181–1191. [CrossRef]
2. Zheng, J.; Huang, M. Traffic Flow Forecast Through Time Series Analysis Based on Deep Learning. *IEEE Access* **2020**, *8*, 82562–82570. [CrossRef]
3. Barrera-Animas, A.Y.; Oyedele, L.O.; Bilal, M.; Akinosho, T.D.; Delgado, J.M.D.; Akanbi, L.A. Rainfall Prediction: A Comparative Analysis of Modern Machine Learning Algorithms for Time-Series Forecasting. *Mach. Learn. Appl.* **2022**, *7*, 100204. [CrossRef]
4. Wu, N.; Green, B.; Ben, X.; O'Banion, S. Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. *arXiv* **2020**, arXiv:2001.08317.
5. Sims, C.A. Macroeconomics and Reality. *Econometrica* **1980**, *48*, 1–48. [CrossRef]
6. 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.
7. Gong, Z.; Tang, Y.; Liang, J. PatchMixer: A Patch-Mixing Architecture for Long-Term Time Series Forecasting. *arXiv* **2023**, arXiv:2310.00655.
8. Taylor, S.J.; Letham, B. Forecasting at Scale. *Am. Stat.* **2018**, *72*, 37–45. [CrossRef]
9. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
10. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.
11. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates, Inc.: Glasgow, UK, 2017; Volume 30.
12. Lim, B.; Arık, S.Ö.; Loeff, N.; Pfister, T. Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting. *Int. J. Forecast.* **2021**, *37*, 1748–1764. [CrossRef]
13. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Moschitti, A., Pang, B., Daelemans, W., Eds.; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1724–1734.
14. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *AAAI* **2021**, *35*, 11106–11115. [CrossRef]
15. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional Sequence to Sequence Learning. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 1243–1252.
16. van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; Kavukcuoglu, K. WaveNet: A Generative Model for Raw Audio. *arXiv* **2016**, arXiv:1609.03499.
17. Dauphin, Y.N.; Fan, A.; Auli, M.; Grangier, D. Language Modeling with Gated Convolutional Networks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 933–941.
18. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–8 December 2012; Curran Associates, Inc.: Glasgow, UK, 2012; Volume 25.
19. van den Oord, A.; Kalchbrenner, N.; Espeholt, L.; Kavukcuoglu, K.; Vinyals, O.; Graves, A. Conditional Image Generation with PixelCNN Decoders. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Curran Associates, Inc.: Glasgow, UK, 2016; Volume 29.
20. Borovykh, A.; Bohte, S.; Oosterlee, C.W. Conditional Time Series Forecasting with Convolutional Neural Networks. *arXiv* **2017**, arXiv:1703.04691.
21. Lea, C.; Vidal, R.; Reiter, A.; Hager, G.D. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. In Proceedings of the Computer Vision—ECCV 2016 Workshops, Amsterdam, The Netherlands, 8–10 October 2016; Hua, G., Jégou, H., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 47–54.
22. Liu, M.; Zeng, A.; Chen, M.; Xu, Z.; Lai, Q.; Ma, L.; Xu, Q. SCINet: Time Series Modeling and Forecasting with Sample Convolution and Interaction. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 5816–5828.

23. Shen, L.; Wei, Y.; Wang, Y.; Qiu, H. FDNet: Focal Decomposed Network for Efficient, Robust and Practical Time Series Forecasting. *Knowl. Based Syst.* **2023**, *275*, 110666. [CrossRef]
24. Das, A.; Kong, W.; Leach, A.; Mathur, S.; Sen, R.; Yu, R. Long-Term Forecasting with TiDE: Time-Series Dense Encoder. *arXiv* **2023**, arXiv:2304.08424.
25. Ekambaram, V.; Jati, A.; Nguyen, N.; Sinthong, P.; Kalagnanam, J. TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Long Beach, CA, USA, 6 August 2023; pp. 459–469.
26. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are Transformers Effective for Time Series Forecasting? *AAAI* **2023**, *37*, 11121–11128. [CrossRef]
27. Zhang, T.; Zhang, Y.; Cao, W.; Bian, J.; Yi, X.; Zheng, S.; Li, J. Less Is More: Fast Multivariate Time Series Forecasting with Light Sampling-Oriented MLP Structures. *arXiv* **2022**, arXiv:2207.01186.
28. Tolstikhin, I.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. MLP-Mixer: An All-MLP Architecture for Vision. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 24261–24272.
29. Li, Z.; Rao, Z.; Pan, L.; Xu, Z. MTS-Mixers: Multivariate Time Series Forecasting via Factorized Temporal and Channel Mixing. *arXiv* **2023**, arXiv:2302.04501.
30. Woo, G.; Liu, C.; Sahoo, D.; Kumar, A.; Hoi, S. Learning Deep Time-Index Models for Time Series Forecasting. In Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023; pp. 37217–37237.
31. Godfrey, L.B.; Gashler, M.S. Neural Decomposition of Time-Series Data for Effective Generalization. *IEEE Trans. Neural. Netw. Learn. Syst.* **2017**, *29*, 2973–2985. [CrossRef] [PubMed]
32. Wen, R.; Torkkola, K.; Narayanaswamy, B.; Madeka, D. A Multi-Horizon Quantile Recurrent Forecaster. *arXiv* **2017**, arXiv:1711.11053.
33. 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.
34. Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; Jin, R. FEDformer: Frequency Enhanced Decomposed Transformer for Long-Term Series Forecasting 2022. In Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022.
35. Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A.X.; Dustdar, S. Pyraformer: Low-complexity pyramidal at- tention for long-range time series modeling and forecasting. In Proceedings of the International Conference on Learning Representations, Virtual Event, 25–29 April 2022.
36. Oreshkin, B.N.; Carpov, D.; Chapados, N.; Bengio, Y. N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting. *arXiv* **2019**, arXiv:1905.10437.
37. Lai, G.; Chang, W.-C.; Yang, Y.; Liu, H. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; ACM: New York, NY, USA, 2018; pp. 95–104.
38. Qin, Y.; Song, D.; Chen, H.; Cheng, W.; Jiang, G.; Cottrell, G.W. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; International Joint Conferences on Artificial Intelligence Organization: Melbourne, Australia, 2017; pp. 2627–2633.
39. Wang, J.; Wang, Z.; Li, J.; Wu, J. Multilevel Wavelet Decomposition Network for Interpretable Time Series Analysis 2018. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 2437–2446.
40. Chang, Y.-Y.; Sun, F.-Y.; Wu, Y.-H.; Lin, S.-D. A Memory-Network Based Solution for Multivariate Time-Series Forecasting. *arXiv* **2018**, arXiv:1809.02105.
41. Smyl, S. A Hybrid Method of Exponential Smoothing and Recurrent Neural Networks for Time Series Forecasting. *Int. J. Forecast.* **2020**, *36*, 75–85. [CrossRef]
42. Fan, C.; Zhang, Y.; Pan, Y.; Li, X.; Zhang, C.; Yuan, R.; Wu, D.; Wang, W.; Pei, J.; Huang, H. Multi-Horizon Time Series Forecasting with Temporal Attention Learning. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 2527–2535.
43. Bergsma, S.; Zeyl, T.; Anaraki, J.R.; Guo, L. C2FAR: Coarse-to-Fine Autoregressive Networks for Precise Probabilistic Forecasting. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 21900–21915.
44. Tan, Y.; Xie, L.; Cheng, X. Neural Differential Recurrent Neural Network with Adaptive Time Steps. *arXiv* **2023**, arXiv:2306.01674.
45. Lin, S.; Lin, W.; Wu, W.; Zhao, F.; Mo, R.; Zhang, H. SegRNN: Segment Recurrent Neural Network for Long-Term Time Series Forecasting. *arXiv* **2023**, arXiv:2308.11200.
46. Jia, Y.; Lin, Y.; Hao, X.; Lin, Y.; Guo, S.; Wan, H. WITRAN: Water-Wave Information Transmission and Recurrent Acceleration Network for Long-Range Time Series Forecasting. *Adv. Neural Inf. Process. Syst.* **2024**, *36*.
47. Huang, S.; Wang, D.; Wu, X.; Tang, A. DSANet: Dual Self-Attention Network for Multivariate Time Series Forecasting. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 2129–2132.

48. Cheng, J.; Huang, K.; Zheng, Z. Towards Better Forecasting by Fusing Near and Distant Future Visions 2019. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.
49. Wang, H.; Peng, J.; Huang, F.; Wang, J.; Chen, J.; Xiao, Y. Micn: Multi-scale local and global context modeling for long-term series forecasting. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
50. Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; Long, M. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In Proceedings of the The Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
51. Wang, W.; Liu, Y.; Sun, H. TLNets: Transformation Learning Networks for Long-Range Time-Series Prediction. *arXiv* **2023**, arXiv:2305.15770.
52. Wang, X.; Wang, Z.; Yang, K.; Feng, J.; Song, Z.; Deng, C.; Zhu, L. MPPN: Multi-Resolution Periodic Pattern Network For Long-Term Time Series Forecasting. *arXiv* **2023**, arXiv:2306.06895.
53. Donghao, L.; Xue, W. ModernTCN: A Modern Pure Convolution Structure for General Time Series Analysis. Available online: https://openreview.net/forum?id=vpJMJerXHU (accessed on 13 October 2023).
54. Ding, X.; Zhang, Y.; Ge, Y.; Zhao, S.; Song, L.; Yue, X.; Shan, Y. UniRepLKNet: A Universal Perception Large-Kernel ConvNet for Audio, Video, Point Cloud, Time-Series and Image Recognition. *arXiv* **2023**, arXiv:2311.15599.
55. 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 Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Curran Associates, Inc.: Glasgow, UK, 2019; Volume 32.
56. Wu, S.; Xiao, X.; Ding, Q.; Zhao, P.; Wei, Y.; Huang, J. Adversarial Sparse Transformer for Time Series Forecasting. In Proceedings of the Advances in Neural Information Processing Systems, Virtual, 6–12 December 2020; Curran Associates, Inc.: Glasgow, UK, 2020; Volume 33, pp. 17105–17115.
57. Qi, X.; Hou, K.; Liu, T.; Yu, Z.; Hu, S.; Ou, W. From Known to Unknown: Knowledge-Guided Transformer for Time-Series Sales Forecasting in Alibaba. *arXiv* **2021**, arXiv:2109.08381.
58. Du, D.; Su, B.; Wei, Z. Preformer: Predictive Transformer with Multi-Scale Segment-Wise Correlations for Long-Term Time Series Forecasting 2022. In Proceedings of the ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023.
59. Woo, G.; Liu, C.; Sahoo, D.; Kumar, A.; Hoi, S. ETSformer: Exponential Smoothing Transformers for Time-Series Forecasting. *arXiv* **2022**, arXiv:2202.01381.
60. Cirstea, R.-G.; Guo, C.; Yang, B.; Kieu, T.; Dong, X.; Pan, S. Triformer: Triangular, Variable-Specific Attentions for Long Sequence Multivariate Time Series Forecasting. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, Vienna, Austria, 23–29 July 2022; International Joint Conferences on Artificial Intelligence Organization: Vienna, Austria, 2022; pp. 1994–2001.
61. Zhang, X.; Jin, X.; Gopalswamy, K.; Gupta, G.; Park, Y.; Shi, X.; Wang, H.; Maddix, D.C.; Wang, Y. First De-Trend Then Attend: Rethinking Attention for Time-Series Forecasting. *arXiv* **2022**, arXiv:2212.08151.
62. Liu, Y.; Wu, H.; Wang, J.; Long, M. Non-Stationary Transformers: Exploring the Stationarity in Time Series Forecasting. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 9881–9893.
63. Shabani, A.; Abdi, A.; Meng, L.; Sylvain, T. Scaleformer: Iterative Multi-Scale Refining Transformers for Time Series Forecasting. *arXiv* **2022**, arXiv:2206.04038.
64. Chen, W.; Wang, W.; Peng, B.; Wen, Q.; Zhou, T.; Sun, L. Learning to Rotate: Quaternion Transformer for Complicated Periodical Time Series Forecasting. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; ACM: Washington, DC, USA, 2022; pp. 146–156.
65. Haugsdal, E.; Aune, E.; Ruocco, M. Persistence Initialization: A Novel Adaptation of the Transformer Architecture for Time Series Forecasting. *Appl. Intell.* **2023**, *53*, 26781–26796. [CrossRef]
66. Sasal, L.; Chakraborty, T.; Hadid, A. W-Transformers: A Wavelet-Based Transformer Framework for Univariate Time Series Forecasting. In Proceedings of the 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA), Nassau, Bahamas, 12–14 December 2022; IEEE: Nassau, Bahamas, 2022; pp. 671–676.
67. Zhang, Y.; Yan, J. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
68. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
69. Zhou, Z.; Zhong, R.; Yang, C.; Wang, Y.; Yang, X.; Shen, W. A K-Variate Time Series Is Worth K Words: Evolution of the Vanilla Transformer Architecture for Long-Term Multivariate Time Series Forecasting. *arXiv* **2022**, arXiv:2212.02789.
70. Li, Y.; Lu, X.; Xiong, H.; Tang, J.; Su, J.; Jin, B.; Dou, D. *Towards Long-Term Time-Series Forecasting: Feature, Pattern, and Distribution*; IEEE Computer Society: Washington, DC, USA, 2023; pp. 1611–1624.
71. Wang, X.; Zhou, T.; Wen, Q.; Gao, J.; Ding, B.; Jin, R. CARD: Channel Aligned Robust Blend Transformer for Time Series Forecasting. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
72. Chen, Y.; Liu, S.; Yang, J.; Jing, H.; Zhao, W.; Yang, G. A Joint Time-Frequency Domain Transformer for Multivariate Time Series Forecasting. *Neural Netw.* **2024**, *176*, 106334. [CrossRef] [PubMed]
73. Gao, J.; Hu, W.; Chen, Y. Client: Cross-Variable Linear Integrated Enhanced Transformer for Multivariate Long-Term Time Series Forecasting. *arXiv* **2023**, arXiv:2305.18838.

74. Nivron, O.; Parthipan, R.; Wischik, D. Taylorformer: Probabalistic Modelling for Random Processes Including Time Series. In Proceedings of the ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems, Hawaii, HI, USA, 28 July 2023.
75. Zhao, Y.; Ma, Z.; Zhou, T.; Ye, M.; Sun, L.; Qian, Y. GCformer: An Efficient Solution for Accurate and Scalable Long-Term Multivariate Time Series Forecasting. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, Birmingham, UK, 21–25 October 2023; Association for Computing Machinery: New York, NY, USA, 2023; pp. 3464–3473.
76. Zhang, Z.; Meng, L.; Gu, Y. SageFormer: Series-Aware Framework for Long-Term Multivariate Time Series Forecasting. *IEEE Internet Things J.* **2024**, *11*, 18435–18448. [CrossRef]
77. Li, B.; Cui, W.; Zhang, L.; Zhu, C.; Wang, W.; Tsang, I.W.; Zhou, J.T. DifFormer: Multi-Resolutional Differencing Transformer With Dynamic Ranging for Time Series Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 13586–13598. [CrossRef]
78. Yu, C.; Wang, F.; Shao, Z.; Sun, T.; Wu, L.; Xu, Y. DSformer: A Double Sampling Transformer for Multivariate Time Series Long-Term Prediction. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, Birmingham, UK, 21–25 October 2023; Association for Computing Machinery: New York, NY, USA, 2023; pp. 3062–3072.
79. Gorbett, M.; Shirazi, H.; Ray, I. Sparse Binary Transformers for Multivariate Time Series Modeling. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Long Beach, CA, USA, 6 August 2023; ACM: Long Beach, CA, USA, 2023; pp. 544–556.
80. Lin, S.; Lin, W.; Wu, W.; Wang, S.; Wang, Y. PETformer: Long-Term Time Series Forecasting via Placeholder-Enhanced Transformer. *arXiv* **2023**, arXiv:2308.04791.
81. Drouin, A.; Marcotte, É.; Chapados, N. TACTiS: Transformer-Attentional Copulas for Time Series. In Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022.
82. Ashok, A.; Marcotte, É.; Zantedeschi, V.; Chapados, N.; Drouin, A. TACTiS-2: Better, Faster, Simpler Attentional Copulas for Multivariate Time Series. *arXiv* **2023**, arXiv:2310.01327.
83. Le, C.P.; Cannella, C.; Hasan, A.; Ng, Y.; Tarokh, V. PrACTiS: Perceiver-Attentional Copulas for Time Series. *arXiv* **2023**, arXiv:2310.01720.
84. 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.
85. Ni, Z.; Yu, H. BasisFormer: Attention-Based Time Series Forecasting with Learnable and Interpretable Basis. *Adv. Neural Inf. Process. Syst.* **2024**, 36.
86. Zhang, Y.; Ma, L.; Pal, S.; Zhang, Y.; Coates, M. Multi-Resolution Time-Series Transformer for Long-Term Forecasting. In Proceedings of the 27th International Conference on Artificial Intelligence and Statistics, Valencia, Spain, 2–4 May 2024; pp. 4222–4230.
87. Olivares, K.G.; Challu, C.; Marcjasz, G.; Weron, R.; Dubrawski, A. Neural Basis Expansion Analysis with Exogenous Variables: Forecasting Electricity Prices with NBEATSx. *Int. J. Forecast.* **2023**, *39*, 884–900. [CrossRef]
88. Challu, C.; Olivares, K.G.; Oreshkin, B.N.; Ramirez, F.G.; Canseco, M.M.; Dubrawski, A. NHITS: Neural Hierarchical Interpolation for Time Series Forecasting. *Proc. AAAI Conf. Artif. Intell.* **2023**, *37*, 6989–6997. [CrossRef]
89. Fan, W.; Zheng, S.; Yi, X.; Cao, W.; Fu, Y.; Bian, J.; Liu, T.-Y. DEPTS: Deep Expansion Learning for Periodic Time Series Forecasting. *arXiv* **2022**, arXiv:2203.07681.
90. Sun, F.-K.; Boning, D.S. FreDo: Frequency Domain-Based Long-Term Time Series Forecasting. *arXiv* **2022**, arXiv:2205.12301.
91. Chen, S.-A.; Li, C.-L.; Yoder, N.C.; Arık, S.Ö.; Pfister, T. TSMixer: An All-MLP Architecture for Time Series Fore- Casting. *arXiv* **2023**, arXiv:2303.06053.
92. Liu, Y.; Li, C.; Wang, J.; Long, M. Koopa: Learning Non-Stationary Time Series Dynamics with Koopman Predictors 2023. *Adv. Neural Inf. Process. Syst.* **2024**, *36*.
93. Xu, Z.; Zeng, A.; Xu, Q. FITS: Modeling Time Series with 10k Parameters. *arXiv* **2023**, arXiv:2307.03756.
94. Luo, Y.; Lyu, Z.; Huang, X. TFDNet: Time-Frequency Enhanced Decomposed Network for Long-Term Time Series Forecasting. *arXiv* **2023**, arXiv:2308.13386.
95. Yi, K.; Zhang, Q.; Fan, W.; Wang, S.; Wang, P.; He, H.; An, N.; Lian, D.; Cao, L.; Niu, Z. Frequency-Domain MLPs Are More Effective Learners in Time Series Forecasting. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 76656–76679.
96. 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 International Conference on Learning Representations, Virtual Event, 25–29 April 2022.
97. Du, Y.; Wang, J.; Feng, W.; Pan, S.; Qin, T.; Xu, R.; Wang, C. AdaRNN: Adaptive Learning and Forecasting of Time Series. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, Queensland, Australia, 1–5 November 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 402–411.
98. Han, L.; Ye, H.-J.; Zhan, D.-C. The Capacity and Robustness Trade-off: Revisiting the Channel Independent Strategy for Multivariate Time Series Forecasting. Available online: https://arxiv.org/abs/2304.05206v1 (accessed on 9 October 2023).
99. Passalis, N.; Tefas, A.; Kanniainen, J.; Gabbouj, M.; Iosifidis, A. Deep Adaptive Input Normalization for Time Series Forecasting. *IEEE Trans. Neural. Netw. Learn. Syst.* **2020**, *31*, 3760–3765. [CrossRef] [PubMed]
100. Fan, W.; Wang, P.; Wang, D.; Wang, D.; Zhou, Y.; Fu, Y. Dish-TS: A General Paradigm for Alleviating Distribution Shift in Time Series Forecasting. *AAAI* **2023**, *37*, 7522–7529. [CrossRef]

101. Chen, Y.; Ren, K.; Wang, Y.; Fang, Y.; Sun, W.; Li, D. ContiFormer: Continuous-Time Transformer for Irregular Time Series Modeling. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 47143–47175.
102. Li, Z.; Li, S.; Yan, X. Time Series as Images: Vision Transformer for Irregularly Sampled Time Series. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 49187–49204.
103. Schirmer, M.; Eltayeb, M.; Lessmann, S.; Rudolph, M. Modeling Irregular Time Series with Continuous Recurrent Units. In Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; pp. 19388–19405.
104. Gruver, N.; Finzi, M.; Qiu, S.; Wilson, A.G. Large Language Models Are Zero-Shot Time Series Forecasters. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 19622–19635.
105. Jin, M.; Wang, S.; Ma, L.; Chu, Z.; Zhang, J.Y.; Shi, X.; Chen, P.-Y.; Liang, Y.; Li, Y.-F.; Pan, S.; et al. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models. *arXiv* **2023**, arXiv:2310.01728.
106. Zhou, T.; Niu, P.; Wang, X.; Sun, L.; Jin, R. One Fits All: Power General Time Series Analysis by Pretrained LM. *Adv. Neural Inf. Process. Systems* **2023**, *36*, 43322–43355.
107. Chang, C.; Peng, W.-C.; Chen, T.-F. LLM4TS: Two-Stage Fine-Tuning for Time-Series Forecasting with Pre-Trained LLMs. *arXiv* **2023**, arXiv:2308.08469.
108. Sun, C.; Li, H.; Li, Y.; Hong, S. TEST: Text Prototype Aligned Embedding to Activate LLM's Ability for Time Series. *arXiv* **2023**, arXiv:2308.08241.
109. Cao, D.; Jia, F.; Arik, S.O.; Pfister, T.; Zheng, Y.; Ye, W.; Liu, Y. TEMPO: Prompt-Based Generative Pre-Trained Transformer for Time Series Forecasting. *arXiv* **2023**, arXiv:2310.04948.
110. Liu, H.; Zhao, Z.; Wang, J.; Kamarthi, H.; Prakash, B.A. LSTPrompt: Large Language Models as Zero-Shot Time Series Forecasters by Long-Short-Term Prompting. *arXiv* **2024**, arXiv:2402.16132.
111. Liu, Y.; Qin, G.; Huang, X.; Wang, J.; Long, M. AutoTimes: Autoregressive Time Series Forecasters via Large Language Models. *arXiv* **2024**, arXiv:2402.02370.
112. Garza, A.; Mergenthaler-Canseco, M. TimeGPT-1. *arXiv* **2023**, arXiv:2310.03589.
113. Rasul, K.; Ashok, A.; Williams, A.R.; Khorasani, A.; Adamopoulos, G.; Bhagwatkar, R.; Biloš, M.; Ghonia, H.; Hassen, N.V.; Schneider, A.; et al. Lag-Llama: Towards Foundation Models for Time Series Forecasting. *arXiv* **2023**, arXiv:2310.08278.
114. Woo, G.; Liu, C.; Kumar, A.; Xiong, C.; Savarese, S.; Sahoo, D. Unified Training of Universal Time Series Forecasting Transformers. *arXiv* **2024**, arXiv:2402.02592.
115. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models Are Few-Shot Learners. In Proceedings of the Advances in Neural Information Processing Systems, Virtual, 6–12 December 2020; Curran Associates, Inc.: Glasgow, UK, 2020; Volume 33, pp. 1877–1901.
116. Koochali, A.; Schichtel, P.; Dengel, A.; Ahmed, S. Probabilistic Forecasting of Sensory Data With Generative Adversarial Networks—ForGAN. *IEEE Access* **2019**, *7*, 63868–63880. [CrossRef]
117. Rasul, K.; Seward, C.; Schuster, I.; Vollgraf, R. Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting. In Proceedings of the 38th International Conference on Machine Learning, Virtual Event, 18–24 July 2021; pp. 8857–8868.
118. Tashiro, Y.; Song, J.; Song, Y.; Ermon, S. CSDI: Conditional Score-Based Diffusion Models for Probabilistic Time Series Imputation. In Proceedings of the Advances in Neural Information Processing Systems, Virtual Event, 6–14 December 2021; Curran Associates, Inc.: Glasgow, UK, 2021; Volume 34, pp. 24804–24816.
119. Kollovieh, M.; Ansari, A.F.; Bohlke-Schneider, M.; Zschiegner, J.; Wang, H.; Wang, Y. (Bernie) Predict, Refine, Synthesize: Self-Guiding Diffusion Models for Probabilistic Time Series Forecasting. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 28341–28364.
120. Chang, P.; Li, H.; Quan, S.F.; Lu, S.; Wung, S.-F.; Roveda, J.; Li, A. A Transformer-Based Diffusion Probabilistic Model for Heart Rate and Blood Pressure Forecasting in Intensive Care Unit. *Comput. Methods Programs Biomed.* **2024**, *246*, 108060. [CrossRef] [PubMed]
121. Rasul, K.; Sheikh, A.-S.; Schuster, I.; Bergmann, U.; Vollgraf, R. Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows. *arXiv* **2020**, arXiv:2002.06103.
122. Feng, S.; Miao, C.; Xu, K.; Wu, J.; Wu, P.; Zhang, Y.; Zhao, P. Multi-Scale Attention Flow for Probabilistic Time Series Forecasting. *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 2056–2068. [CrossRef]
123. Li, Y.; Lu, X.; Wang, Y.; Dou, D. Generative Time Series Forecasting with Diffusion, Denoise, and Disentanglement. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 23009–23022.
124. Rangapuram, S.S.; Seeger, M.W.; Gasthaus, J.; Stella, L.; Wang, Y.; Januschowski, T. Deep State Space Models for Time Series Forecasting. In Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; Curran Associates, Inc.: Glasgow, UK, 2018; Volume 31.
125. Gu, A.; Dao, T. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv* **2023**, arXiv:2312.00752.