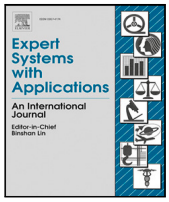
Expert Systems With Applications 259 (2025) 125231

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



OnsitNet: A memory-capable online time series forecasting model incorporating a self-attention mechanism

Hui Liu [a,b,c], Zhengkai Wang [a,*], Xiyao Dong [a], Junzhao Du [a,b,c]

[a] School of Computer Science and Technology, Xidian University, Xi'an, 710126, China
[b] Engineering Research Center of Blockchain Technology Application and Evaluation, Ministry of Education, Xi'an, 710126, China
[c] Key Laboratory of Smart Human–Computer Interaction and Wearable Technology of Shaanxi Province, Xi'an, 710126, China

ARTICLE INFO

Dataset link: https://drive.google.com/drive/folders/1ohGYWWohJlOlb2gsGTeEq3Wii2egnEPR, https://github.com/zhouhaoyi/ETDataset/tree/main/ETT-small, https://drive.google.com/drive/folders/1M3gTc1DSvnUFMI57p70VFH5MHhZh3wC8

Keywords:
Time series forecasting
Online learning
Offline learning
Self-attention mechanism
ITransformer

ABSTRACT

Traditional time series (TS) forecasting models are based on fixed, static datasets and lack scalability when faced with the continuous influx of data in real-world scenarios. Real-time online learning of data streams is crucial for improving forecasting efficiency. However, few studies focus on online TS forecasting, and existing approaches have several limitations. Most current online TS forecasting models merely train on data streams and are ineffective in handling concept drift scenarios. Furthermore, they often fail to adequately consider dependencies between variables and do not leverage the robust modeling capabilities of offline models. Therefore, we propose an innovative online learning method called OnsitNet. It consists of multiple learning modules that progressively expand the receptive field of convolutional kernels within the learning modules using an exponentially growing dilation factor, aiding in the capture of multi-scale data features. Within the learning modules, we propose an online learning strategy focusing on memorizing concept drift scenarios, with a fast learner, memorizer, and Pearson trigger. The Pearson trigger activates dynamic interaction between the fast learner and memorizer by detecting new data patterns, facilitating online rapid learning of data streams. To capture the dependencies between variables, we propose a new model, SITransformer, which is a streamlined version of the offline model ITransformer. Unlike the traditional Transformer, it reverses the roles of the feed-forward network and the attention mechanism. This inverted architecture is more effective at learning the correlations between variables. Experimental results on five real-world datasets show OnsitNet achieves lower online prediction errors, enabling timely and effective forecasting of future trends in TS data.

1. Introduction

Time series (TS) forecasting refers to the utilization of methods to predict future data values based on historical data with or without timestamps. Accurate prediction of TS is highly beneficial in people's lives. For instance, precise forecasts for traffic conditions enable individuals to anticipate congestion in advance, thereby saving time. Accurate predictions of the number of cases in an epidemic allow policymakers to allocate and transport medical supplies efficiently, reducing casualties. Therefore, it is crucial to study efficient and accurate TS prediction models. With the development of technology, several TS forecasting models have already been developed.

Early TS forecasting models predominantly use statistical models. For instance, the AutoRegressive Integrated Moving Average (ARIMA) (Box, Jenkins, Reinsel, & Ljung, 2015) model is a combination of the AutoRegressive (AR), differencing integration (I), and Moving Average (MA). It utilizes linear functions to learn parameters such as autoregressive order and employs them to forecast future values. In recent years, deep learning has experienced rapid advancements, providing excellent tools for the field of TS prediction. Recurrent neural networks (RNNs) (Yadav & Thakkar, 2024) have significantly enhanced predictive ability due to their inherent memory ability. This is because the features processed by RNN not only come from the current input, but also contain hidden states from previous time steps. Currently, the Transformer model based on the self-attention mechanism (Vaswani et al., 2017) and its variants have become the mainstream methods in the field of TS forecasting due to their powerful modeling capabilities and flexibility. Examples include Informer (Zhou et al., 2021) and Autoformer (Wu, Xu, Wang, & Long, 2021). Recent studies suggest that, in certain scenarios, even simple linear models can achieve exceptional forecasting performance, sometimes surpassing Transformer-based models. Consequently, models based on Multilayer Perceptrons

* Corresponding author.
E-mail addresses: liuhui@xidian.edu.cn (H. Liu), 23031110427@stu.xidian.edu.cn (Z. Wang), 21181214453@stu.xidian.edu.cn (X. Dong), dujz@xidian.edu.cn (J. Du).

https://doi.org/10.1016/j.eswa.2024.125231
Received 28 May 2024; Received in revised form 11 August 2024; Accepted 26 August 2024
Available online 29 August 2024

(MLPs) have emerged, such as N-BEATS (Oreshkin, Carpov, Chapados, & Bengio, 2020) and LightTS (Zhang et al., 2022).

However, whether traditional TS prediction models or deep learning-based TS prediction models, despite achieving decent predictive performance in certain scenarios, generally exhibit a significant drawback. The reason is that these methods require a complete and fixed dataset during training, and throughout the entire training process, the input–output patterns remain in a static and unchanging ideal state. It is worth noting that, in real-world scenarios, data is sent in the form of data streams, and the patterns of input and output are dynamically changing. In such cases, traditional TS prediction methods typically require retraining the model with each new data point and the entire historical dataset. As time goes by, the volume of data increases significantly. Considering only the entire historical dataset in each training iteration will result in a significant increase in computational costs and a decrease in prediction efficiency. Based on the above observations, we believe that online learning strategies (Gultekin & Paisley, 2018; Liu, Hoi, Zhao, & Sun, 2016) can effectively alleviate the shortcomings of current mainstream time series prediction research methods. It means that when new data arrives, there is no need to retrain with old data. On the contrary, online prediction models can capture the flow of changes in data features by utilizing new data. However, there are a number of challenges with current approaches to online learning. Some works (Aljundi, Caccia, & Belilovsky, 2019a) simply train on data streams, leading to slow convergence or even failure to converge. In scenarios involving significant changes in data characteristics, known as concept drift, a large number of samples are usually required to capture these trends, adversely affecting the short-term predictive performance following the concept drift. Furthermore, most online methods do not consider the relationships between variables, despite the fact that real-world variables often exhibit varying degrees of correlation, which is a crucial factor in accurate forecasting. Additionally, current offline models already demonstrate strong modeling capabilities and predictive performance. However, existing online TS prediction works only update internal structures and optimization objectives without considering the integration and effective utilization of these powerful offline models.

To effectively address the aforementioned issues and align with real-world TS forecasting scenarios, we propose OnsitNet, an online learning model that integrates self-attention mechanisms and memory capabilities. OnsitNet employs a stacked architecture of learning modules, each utilizing dilated convolutions with an exponentially increasing dilation factor as the number of learning modules grows. This method maintains computational efficiency without increasing the number of parameters or computational load while capturing features at different scales, a proven effective approach (Sahoo, Pham, Lu, & Hoi, 2017). Within the learning modules, we propose an online learning layer to facilitate the online learning of the data stream. This online learning layer possesses memory capabilities and is implemented by a memorizer. It focuses on storing concept drift patterns with high predictive impact, thus ensuring stable predictions when encountering similar situations during the learning process without experiencing a breakdown. Immediate processing of the data stream is achieved through a fast learner, and a Pearson trigger is employed to detect new and old patterns. When the Pearson trigger is activated, the fast learner interacts with the memorizer, and both are synchronized in their updates. To address the shortcomings of current online learning methods, which hardly consider the relationships between variables and do not effectively utilize offline models, we propose SITransformer, a streamlined version of the offline model ITransformer (Liu et al., 2024). SITransformer employs a self-attention mechanism to capture dependencies between variables while ensuring computational efficiency. Unlike traditional Transformers, SITransformer treats the entire TS of each independent variable as a token. This approach is more suitable for feature extraction and modeling of time series, aiding in a more comprehensive understanding of the overall TS's dynamic structure and correlations. Our contributions can be summarized as follows:

- We propose a novel online TS forecasting method, OnsitNet. It utilizes an architecture with exponentially increasing dilation factors to capture both local and long-term features of the data. Unlike previous online forecasting models, it possesses memory capabilities to effectively handle concept drift. Additionally, it leverages the robust modeling capabilities of offline models, not only extracting dependencies between variables effectively but also enhancing online learning capabilities.
- To achieve rapid learning from data streams and mitigate the impact of concept drift scenarios, we propose an online learning layer. It achieves online learning by manipulating gradients through the fast learner, stores concept drift patterns in the memorizer, and triggers dynamic interactions between the fast learner and memorizer through the Pearson Trigger.
- Our proposed SITransformer adopts an inverted TS perspective, reversing the roles of the self-attention mechanism and the feed-forward network. This setup allows a rational and efficient treatment of TS and accomplishes the extraction of dependencies between variables.
- Experiments are conducted on five real-world datasets, and the results show that OnsitNet achieves the best predictive performance, obtaining the highest number of optimal performances. Specifically, in the long-term prediction results on the ETTh2 and WTH datasets, it reduces the MSE by 20% and 14.8% compared to the best baseline method.

2. Related work

2.1. Traditional multivariate time series forecasting

In recent years, researchers have proposed numerous TS forecasting methods. Among them, the Vector Autoregressive model VAR (Kilian & Lütkepohl, 2017) and Autoregressive Integrated Moving Average model ARIMA (Lee & Tong, 2011; Murat, Malinowska, Gos, & Krzyszczak, 2018) are widely applied statistical models for handling linear relationships. Despite their high interpretability, these methods overly rely on certain assumptions and explore only linear relationships between variables. With the development of deep learning, the focus has shifted from studying linear relationships between variables to investigating nonlinear relationships. Current deep learning prediction models can generally be divided into four categories: models based on Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Transformers ((described in Section 2.2)), and Multilayer Perceptrons (MLP). RNN and its variants (Lai, Chang, Yang, & Liu, 2018; Shih, Sun, & Lee, 2019) owing to their memory and parameter-sharing characteristics, have gained certain advantages in handling TS data. Sen, Yu, and Dhillon (2019) proposed the deep model (DeepGlO), which integrated temporal networks and a global matrix factorization model. The temporal network captured attribute features and related covariates for each time series, while the global matrix factorization model was normalized through Temporal Convolutional Networks (TCN). Xu et al. (2020) introduced Tensorized Long Short-Term Memory (TLASM) with adaptive shared memory, which modeled temporal patterns of long-term sequences in multi-task learning settings. CNN models excel at capturing local features in time series, and many studies have utilized them for extracting local dynamic temporal features. For instance, Hewage et al. (2020) achieved higher accuracy in weather forecasting tasks using TCN. MICN (Wang et al., 2023) extracted global and local features through isometric convolution and down-sampled convolution, aiming to capture a comprehensive view. Zhang, Sheng, Zhang, and Wen (2024) effectively fused TCN, lightweight attention mechanism and ensemble learning strategy for short-term runoff prediction and achieved excellent performance. However, models based on RNN and CNN are limited in long-term forecasting tasks due to issues such as limited receptive fields. MLP has also been introduced into the field of time series prediction. Oreshkin et al. (2020) proposed the N-BEATS

framework, which, based on backward and forward residual links and very deep fully connected layers, offered high interpretability. Zeng, Chen, Zhang, and Xu (2023) introduced a simple single-layer linear model, LTSF-Linear, which outperformed Transformer-based models in many cases. Other models, including Nhits (Challu et al., 2023) and TimeMixer (Wang et al., 2024), also achieved good performance in terms of prediction efficiency and accuracy.

2.2. Transformer-based time series forecasting

Transformer is initially proposed for processing natural language text, but due to its powerful learning capabilities, it has been widely applied in various fields. As a result, methods utilizing Transformer and its variants for TS forecasting have emerged. Crossformer (Zhang & Yan, 2023) divided the input sequence into segments and utilized a two-stage attention layer (TSA) to capture dependencies across dimensions and time. This approach was applied to both the encoder and decoder, and the final prediction result was synthesized from each layer's output. PatchTST (Nie, Nguyen, Sinthong, & Kalagnanam, 2023) decomposed time series into patches, adopting a channel-wise independence concept where each input token contained information from a single channel. Additionally, a masked self-supervised learning method was introduced. Fedformer (Zhou et al., 2022), based on the Autoformer framework, incorporated Fourier enhancement and wavelet enhancement modules into the encoder and decoder, achieving lower complexity. Liu et al. (2021) proposed a new pyramid attention-based Transformer (Pyraformer), modeling dependencies at different scales through intra-scale edges and extracting features at different resolutions through inter-scale edges. The Preformer, proposed by Du, Su, and Wei (2023), segmented the embedded feature vector sequence into multiple sections and introduced an efficient multi-scale segment correlation mechanism, effectively encoding TS. While these methods enhance various components within the Transformer architecture, they do not consider the rationality of the overall architecture in handling TS data. ITransformer (Liu et al., 2024) takes this into account and adopts Transformer's inverted architecture, which effectively extracts correlations between variables and meets our requirements.

2.3. Online time series forecasting

A reality is that most current TS forecasting work seldom considers online learning. This is mainly due to a relatively insufficient understanding of online learning, as previous research and practice have focused more on offline learning, with less knowledge about real-time learning tasks in dynamic environments. However, for application scenarios where data continuously arrives, online learning is an effective and necessary approach. In the real world, TS data precisely possesses this characteristic, making the application of online learning to TS forecasting reasonable. Liu et al. (2016) redefined ARIMA as a full-information online optimization task (excluding random noise terms) to estimate the parameters of the ARIMA model. Gultekin and Paisley (2018) introduced a matrix factorization technique that efficiently learned low-dimensional embeddings in an online manner, developing a recursive least mean square estimator based on these embeddings. The solution proposed by Gupta, Narwariya, Malhotra, Vig, and Shroff (2021) included a series of generators and solvers. Generators created data for subsequent tasks, while solvers utilized a core dynamics module (CDM) and a conditioning module (CM) to handle input-variable-dimension data (VID). Kurle, Cseke, Klushyn, Van Der Smagt, and Günnemann (2019) addressed the online inference problem for non-stationary streaming data using Bayesian neural networks and memory-based online variational Bayesian methods. For irregular and uneven TS, Li and Marlin (2020) introduced variational autoencoders and generative adversarial networks for learning. However, these methods lack a fast learning mechanism when facing concept drift scenarios.

Data distributions in real-world scenarios are dynamically changing. Although models may grasp certain patterns or rules through historical data during training, they often fail to adapt when the underlying data distribution changes, a phenomenon known as concept drift. Recently, to address concept drift, researchers have proposed effective online models. Li, Yang, Liu, Xia, and Bian (2022) introduced DDG-DA, which can predict the evolution of data distribution and use it to generate training samples, thereby improving model performance. Pham, Liu, Sahoo, and Hoi (2023) proposed the Fast–Slow Learning Network (FSNet), which utilized adapters to learn from data streams and incorporated a certain memory capability. These models often do not fully consider dependencies between variables and do not leverage the advantages of offline models. Based on this, the model proposed in this paper effectively addresses these shortcomings.

3. Methodology

Problem Formulation: Given a time-series data input $X \in R^{L*D}$, where L represents the sequence length and D indicates the number of dimensions (variables). Our goal is to identify an optimal model with a series of parameters to predict future data values $\hat{Y}$ and to minimize the discrepancy between the predicted values $\hat{Y}$ and the true values Y as much as possible. Within the SITransformer layer, features are denoted as H, with the output represented by H^l. The internal parameters of each online learning layer are denoted by θ_l, the memorizer is represented by M_l, where l indicates the layer number, and Pearson denotes the Pearson trigger.

3.1. General overview

Fig. 1(a) illustrates the design concept and structure of OnsitNet. OnsitNet adopts a primary structure of stacked learning modules. The input X is processed through multiple learning modules, and the processed data features are fed into a prediction layer containing a convolutional layer to obtain the prediction results. The convolutional layer possesses the attribute of parameter sharing, which reduces the complexity of the model and the risk of overfitting. Each learning module contains an SITransformer layer, online learning layers, Gelu layers, and a Weighted Averager (WA). The role of GeLU is to introduce non-linear transformations, helping to better capture features in complex data. Additionally, GeLU provides a smooth non-linear activation, which enhances training stability and facilitates gradient flow. We position a Weighted Averager outside of each online learning layer, assigning different weights to the output of each part to align with real-world scenarios.

Fig. 1(b) illustrates the structure of our SITransformer layer. The role of the SITransformer is to extract the relationships among variables, thereby obtaining richer feature information for the variables and facilitating more effective online learning. It consists of an embedding layer, an encoder, and a decoder, where the encoder includes a self-attention layer, a normalized layer, and a feed-forward layer. Contrary to the traditional embedding approach that takes a single time point of all variables as a token, the embedding layer considers the entire time series of each variable as one token. The self-attention layer uses a multi-head attention mechanism to capture the relationships among all tokens, while the normalized layer and feed-forward layer capture global feature information of the variables. Compared to the original ITransformer, which uses a multi-layer perceptron (MLP) as the feed-forward layer and multiple normalized layers in the encoder, the encoder of our proposed SITransformer simplifies the feed-forward layer to a single linear layer and uses only a single normalized layer. This model design not only effectively learns features but also significantly simplifies the model architecture, making it more efficient. Finally, the decoder, which contains a single linear layer, further transforms the obtained features.

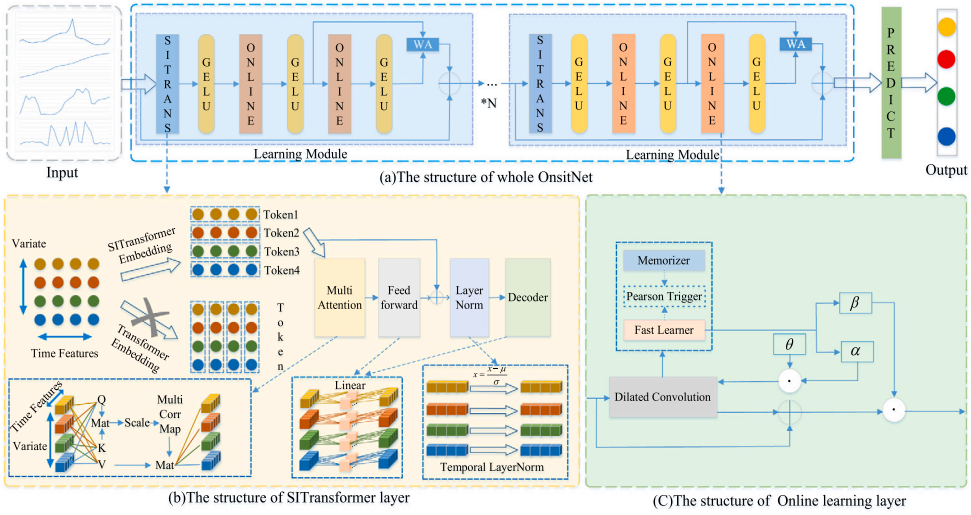


Fig. 1. The overall architecture of OnsitNet. (a) represents the overall flow of OnsitNet, where raw inputs are processed through multiple learning modules and go through a prediction layer to obtain predictions. The learning modules are dominated by the SITransformer layer and the online learning layer. (b) shows the design concept of the SITransformer layer and compares it with the way the traditional Transformer handles time series. The attention mechanism of the SITransformer calculates the relationship between different variables, and the other components handle the global temporal features of each variable. Whereas Transformer's attention mechanism computes relationships between features of different time variables, the other components also process information about all variables at the same timestamp. (C) shows the interaction flow within the online learning layer, which is dominated by the dilated convolution, and the real-time data flow is accomplished by the fast learner, the memorizer, and the Pearson trigger.

Fig. 1(c) illustrates the structure of our online learning layer. We achieve the goal of online rapid learning through the online learning layer, effectively reducing the negative impact of concept drift. It comprises a fast learner, a memorizer, and a Pearson trigger. The fast learner learns data features by manipulating gradients, allowing for quick localization and updating of parameters such as weights and biases within the module during training. The Pearson trigger is responsible for comparing new and old patterns. When there is a significant change in patterns, the fast learner searches for similar patterns from the memorizer and updates itself, while the memorizer stores the new patterns within it. Therefore, the memorizer stores patterns of concept drift, enabling the fast learner to effectively respond when encountering similar scenarios again without the need for time-consuming and inefficient relearning, thus reducing the impact on prediction accuracy.

Consequently, through the design philosophy of OnsitNet, we have realized effective online learning that captures variable relationships and deals with concept drift scenarios, which has been applied to the context of time series prediction. The SITransformer layer and the online learning layer will be introduced in Section 3.2 and Section 3.3, respectively.

3.2. The design of the SITransformer layer

Recent works (Zeng et al., 2023) have demonstrated through extensive experiments that simple linear layers outperform Transformers on certain TS prediction tasks. This is because traditional Transformers, when processing time series data, typically treat single data points of all variables at the same time step as a token. However, time series data is not always perfectly aligned or accurately preprocessed, which can lead to issues such as misaligned timestamps. Using traditional Transformers can make unrelated data points from different timestamps appear similar, resulting in meaningless self-attention calculations and introducing noise. Moreover, the permutation-invariant attention mechanism of Transformers is not well-suited for time series data, which has strong sequential dependencies. Inspired by Liu et al. (2024), the proposed SITransformer inverses the roles of the attention mechanism and the feed-forward layer without changing the Transformer components, thus avoiding the aforementioned issues. This inverted approach aligns precisely with our need to capture the relationships between variables. Therefore, we integrate it into the online learning scenario, with the components as follows.

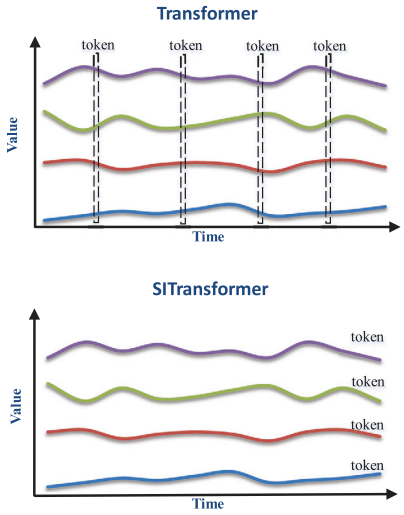


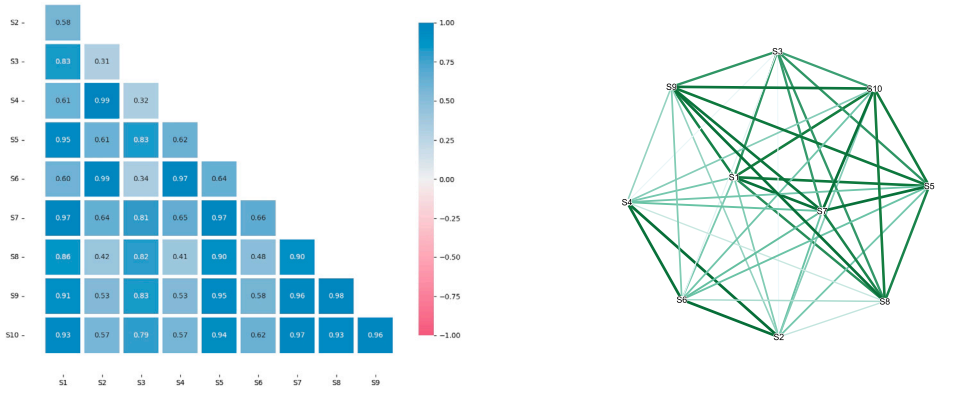
Fig. 2. Comparison of Transformer and SITranformer embedding methods.

3.2.1. The design of the embedding layer

As shown in Fig. 2, unlike the traditional Transformer's embedding approach, which treats a single data point of all variables at each time as one token, SITransformer considers the entire time series $H_d \in R^L$ of each variable as one token. This ensures the global and sequential features of the time series. Here, $H \in R^{L*D}$ represents the original input $X \in R^{L*D}$ or the output of the previous module, d represents the dth variable.

3.2.2. The design of the self-attention layer

In Fig. 3, we visualize the correlation heatmap (as depicted in (a)) and the correlation network graph (as depicted in (b)) for 30 timestamps of data collected from 10 sensors in the Traffic dataset. In (b), edges connecting sensors represent correlations, with darker colors indicating higher correlation strength. Fig. 3 reveals varying degrees



(a) Correlation heatmap (b) Correlation network graph

Fig. 3. Visualization of the correlation among Traffic sensors (10 sensors, 30 timestamps). (a) represents a heatmap. (b) represents a network graph, with edge weights ranging from 0.31 to 0.99. The deeper the color, the higher the weight.

of correlation among different variables (sensors), with some exhibiting extremely high correlations (reaching 0.99). Therefore, employing attention mechanisms to capture the relationships between these variables is highly essential. In a Transformer, the features of all variables at a single time point are treated as a token, and when calculating attention among all tokens, it essentially captures the correlations between data points at different times. This approach remains within the time dimension and does not effectively capture the correlations between variables. In contrast, the self-attention layer here calculates the attention scores between each token (variable), truly extracting the dependencies between variables. By linearly projecting each token, we obtain the representations of Query (Q), Key (K), and Value (V). The self-attention layer uses Q and K to calculate the attention scores between each pair of tokens through a softmax operation and multiplies them with V for feature updates. The calculation is as follows:

$$H_d^A = Attention(Q, K, V) = Softmax(\frac{Qk^T}{\sqrt{d_k}})V \tag{1}$$

where d_k is the dimensionality of the Key.

3.2.3. The design of the normalized layer

The normalized layer stabilizes and accelerates the convergence of the model by standardizing the time series features. Due to the potential issue of misaligned timestamps in the data, when traditional Transformers are used to normalize the data, they may make this misaligned data appear similar, leading to incorrect interaction relationships. The SITransformer addresses this issue by normalizing the entire sequence for each variable, thus mitigating the impact of this problem. As shown in Eq. (2), normalization is performed by operating on the mean and variance of the features.

$$H_d^N = \frac{H_d^A - \overline{H_d^A}}{\sqrt{Var(H_d^A)}} \tag{2}$$

where $\overline{H_d^A}$ represents the mean of H_d^A, $Var(H_d^A)$ represents the variance of H_d^A, and H_d^N represents the features obtained through the normalized layer.

3.2.4. The design of the feed-forward layer

The feed-forward layer processes the global information of the entire sequence for each variable, in contrast to dealing with information from a single moment, allowing for the extraction of richer features. To emphasize, in order to reduce the number of parameters to meet the efficiency requirements of online learning, we do not use a MLP, but instead, we use a single linear layer to simplify the model, as shown in Eq. (3).

$$H_d^F = Linear(H_d^N) \tag{3}$$

where H_d^F represents the features obtained through the feed-forward layer. Finally, $H^I \in R^{L*D}$ is obtained through the single linear layer of the decoder.

3.3. The design of the online learning layer

Traditional offline time series forecasting models need to input both streaming data and overall historical data into the model for training when dealing with real-world data streams, which is difficult to adapt to the dynamic changes of data and real-time requirements, so we need online learning strategies. However, to the best of our knowledge, there is less work considering online learning in the field of time series forecasting, and this research is still in its infancy. Moreover, existing online learning models (Aljundi et al., 2019a) usually learn directly from data streams. When encountering conceptual drift, these models cannot adapt to such significant new changes in a timely manner. Instead, they rely on the continuous accumulation of data to gradually capture emerging trends, which challenges both the efficiency and accuracy of prediction. Therefore, we propose an online learning layer. The online learning layer primarily utilizes dilated convolutions. Dilated convolutions expand the receptive field of a standard convolutional kernel, with the receptive field calculated as $K_c = S * (K_0 - 1) + 1$. Here, S is the dilation factor, K_0 is the size of the standard convolutional kernel, and K_c is the size of the dilated convolutional kernel (receptive field). In contrast, the receptive field of a standard convolution is equal to the size of its kernel K_0. This illustrates that dilated convolutions can capture a larger range of contextual information without increasing computational cost, which is beneficial for capturing long-term sequence features. In this paper's setup, the dilation factors of the dilated convolutions within each learning module of the online learning layer are different, increasing exponentially. This design focuses on multi-scale features simultaneously, thereby improving overall performance. The effectiveness of dilated convolutions will be further demonstrated in Section 4.5. Besides the main part, the online learning layer accomplishes the task of online learning through the following three components.

3.3.1. The design of the fast learner

Since the partial derivatives represent the impact of each layer's parameters on the loss function, the fast learner will learn data samples

through gradient operations to achieve rapid learning. However, gradients from single samples are noisy, and we use Exponential Moving Average (EMA) to smooth them, as shown in Definition 1 and Eq. (4).

Definition 1. The Exponential Moving Average is a type of moving average that weights the moving average by an exponential decay, where the most recent data points carry greater weight, and the older data points have less weight.

$$\vec{g}_l = \tau \vec{g}_l + (1-\tau) g_l^t \tag{4}$$

where $\vec{g}_l$ represents the EMA gradient for the lth layer, and g_l^t represents the gradient for the lth layer at time t. To facilitate processing, the gradients $\vec{g}_l$ are mapped into a set of more compact learning coefficients μ_l using a block-wise operation. First, flatten the $\vec{g}_l$ into a one-dimensional vector and divide it into n blocks. Then, map it to a hidden feature representation. Lastly, the hidden feature representation is mapped to the learning coefficients μ_l, as shown in Eq. (5).

$$\begin{aligned} \vec{g}_l &= Vec(\vec{g}_l) \\ [b_1, b_2, \dots, b_n] &= Divide(\vec{g}_l) \\ [h_1, h_2, \dots, h_n] &= [W_1 b_1, W_1 b_2, \dots, W_1 b_n] \\ [\mu_l^1, \mu_l^2, \dots, \mu_l^n] &= [W_2 h_1, W_2 h_2, \dots, W_2 h_n] \end{aligned} \tag{5}$$

Therefore, μ_l can be expressed as $\mu_l = [\alpha_l, \beta_l]$. Here, α_l is the weight learning coefficient, and β_l is the feature learning coefficient. As shown in Eq. (6), α_l interacts multiple times with the parameters θ_l of this layer to obtain ϑ_l, which is used to update the parameters of the dilated convolution. (Note: here μ_l already includes interactions with the memorizer, which will be explained in the interaction mechanism section.)

$$\vartheta_l = Title(\alpha_l) \odot \theta_l \tag{6}$$

where $Title$ is a weight adapter designed to implement channel-level weight adjustment. After updating the parameters of the dilated convolution, the input of this layer $\widetilde{H^O}$ (acquired from the SITransformer layer as H^I or the output from the previous online learning layer) is added to the features output by the dilated convolution using a residual operation.

$$\widehat{H^O} = \widetilde{H^O} + DilatedConv(\widetilde{H^O}) \tag{7}$$

Finally, by interacting with the feature learning coefficient β_l, the output of the online learning layer H^O is obtained.

$$H^O = Title(\beta_l) \odot \widehat{H^O} \tag{8}$$

3.3.2. The design of the Pearson trigger

We examine the newly arrived data patterns. When certain conditions are met, interaction between the fast learner and the memorizer occurs. We trigger this interaction only when there is a significant change in data patterns, as performing interaction at every step would be costly and introduce noise. The current EMA gradient is $\vec{g}_l$, and by setting the hyperparameter τ in Eq. (4) to a large value, we retain past EMA gradient $\vec{g}_l{}'$. Since the Pearson correlation coefficient is advantageous in capturing linear relationships between gradients and sensitive to changes in trends, we use it to calculate the correlation between $\vec{g}_l$ and $\vec{g}_l{}'$.

$$Pearson(\vec{g}_l, \vec{g}_l{}') = \frac{Cov(\vec{g}_l, \vec{g}_l{}')}{\sigma(\vec{g}_l)\sigma(\vec{g}_l{}')} \tag{9}$$

where Cov represents the covariance between the two, and σ represents the standard deviation. The interaction mechanism is triggered when $Pearson(\vec{g}_l, \vec{g}_l{}') < -\gamma$, where γ is a hyperparameter with a relatively large value.

3.3.3. The design of the interaction mechanism

When encountering concept drift patterns in an online environment, failing to take additional measures can cause the online learning model to struggle to adapt to new data distributions. The model would need a continuously increasing number of samples to capture these significant pattern changes, impacting prediction efficiency and accuracy. Therefore, we set up a memorizer. Its purpose is to store conceptual drift patterns that have already been experienced. When similar concept drift patterns are encountered again, we no longer need to relearn them, but can utilize those already stored patterns to assist the current learning, thus reducing the impact of concept drift patterns. Thus, when the trigger condition is satisfied (meaning that the current is a concept drift pattern), the fast learner will interact with the memorizer. We first use the softmax operation to pick the first k patterns s_l^k from the memorizer that are more similar to the current one, as follows.

$$S_l = Softmax(\overline{\mu_l} M_l) \tag{10}$$

$$S_l^k = Topk_{max}(S_l) \tag{11}$$

where s_l^k represents the top k largest similarity probability values. Then, we obtain the desired information from the memorizer using Eq. (12) and use this information to update and obtain the final learning coefficients μ_l.

$$\widehat{\mu}_l = \sum_{i=1}^{k} S_l^k[i] \cdot M_l[i] \tag{12}$$

$$\mu_l = \delta \mu_l + (1-\delta)\widehat{\mu}_l \tag{13}$$

where δ is a hyperparameter, $S_l^k[i]$ represents the ith element of S_l^k, and $M_l[i]$ represents the ith row of M_l. Finally, to update the information within the memorizer, we effectively write the new knowledge into the most relevant positions indicated by s_l^k and perform a weighted operation.

$$M_l = \varepsilon M_l + (1-\varepsilon)\overline{\mu_l} \otimes S_l^k \tag{14}$$

where ε is a hyperparameter, and $\otimes$ represents the outer product operator. To maintain numerical stability and facilitate computation, a normalization operation is performed.

$$M_l = \frac{M_l}{Max(1, \|M_l\|_2)} \tag{15}$$

We posit that each online learning layer has a distinct promotional effect on predictions (analyzed in Section 4.7). Consequently, we employ a weighted averaging mechanism externally for each online learning layer, assigning unequal weights to obtain the final output of the learning module, where ρ is also a hyperparameter.

$$H^{learn} = \rho H_1^O + (1-\rho) H_2^O \tag{16}$$

4. Experiments

4.1. Experimental setup

Datasets: We assess the proposed method using five real-world datasets. ETTh2 (Electricity Transformer Temperature-hourly) contains data points featuring "oil temperature" and 6 power load characteristics, with a sampling interval of one hour. ETTm1 (Electricity Transformer Temperature-minutely) shares the same data attributes as ETTh2 but with a reduced sampling interval of 15 min. ECL (Electricity Consuming Load) contains the electricity consumption data of 321 clients in two years, recorded hourly. Traffic contains the road occupancy rates as measured by 862 sensors on the freeways in the San Francisco Bay Area from 2015 to 2016. WTH (Weather) contains climate data from 1600 locations across the United States from 2010 to 2013, with an hourly collection interval. The specific details of the dataset are shown in Table 1, and Fig. 4 also presents the real data of

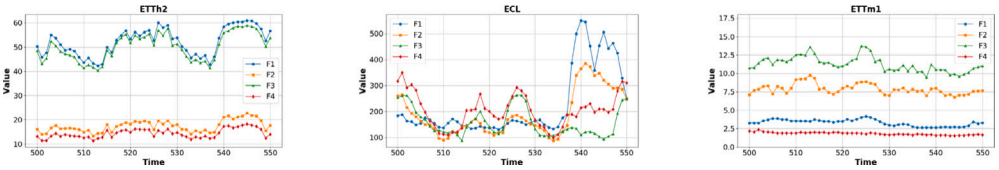


Fig. 4. Real values for ETTh2, ETTm1, and ECL (4 features, 50 timestamps).

Table 1
Details of the five datasets.

Datasets	ETTh2	ETTm1	ECL	Traffic	WTH
Features	7	7	321	862	12
Timestamps	17 420	69 680	26 304	17 544	52 696
Granularity	1 h	15 min	1 h	1 h	1 h

four features at the timestamps between 500–550 in ETTh2, ETTm1, and ECL.

Evaluation Metrics: We employ two evaluation metrics. They are the Mean Squared Error (MSE), given by $MSE = \frac{1}{n}\sum_{i=1}^{n}(y-\hat{y})^2$, and the Mean Absolute Error (MAE), given by $MAE = \frac{1}{n}\sum_{i=1}^{n} \mid y-\hat{y} \mid$.

4.2. Online forecasting results

Baseline: Empirical Replay (ER) (Chaudhry et al., 2019) is a reinforcement learning technique that improves learning efficiency by randomly storing and sampling past experiences; DER++ (Buzzega, Boschini, Porrello, Abati, & Calderara, 2020) combines knowledge distillation and regularization for generalized continual learning; MIR (Aljundi et al., 2019a) replaces random sampling in ER with a selection of samples that causes the greatest forgetting; TFCL (Aljundi, Kelchtermans, & Tuytelaars, 2019b) achieves task-free online learning with memory-aware synapses; OnlineTCN implements a standard TCN backbone with residual convolutional filters (Woo, Liu, Sahoo, Kumar, & Hoi, 2022); Informer (Zhou et al., 2021) is a transformer variant utilizing probabilistic sparse self-attention with a parallel generative decoder; FSNet (Pham et al., 2023) employs a TCN architecture and stacks dilated convolutional layers as the data stream arrives, enabling online learning; Time-TCN (Wen et al., 2024) also uses a TCN architecture, but it only performs convolution operations on the temporal dimension, ignoring the feature dimension.

Online Forecasting Results Analysis: The cumulative MSE and MAE results of the predictions for each model are shown in Tables 2 and 3, with the best results highlighted in bold. From the tables, it can be seen that OnsitNet ranks first in 21 out of 30 results, while the remaining top positions are held by other models. This indicates that OnsitNet is extremely effective in the field of online time series prediction, capable of effectively capturing future trends in the data. Specifically, on the ETTh2, WTH, and ETTm1 datasets, OnsitNet reduces the MSE by an average of 14.9%, 8.5%, and 6.1% respectively compared to the second-best results. Particularly in long-term predictions (predicting the 48th time step), OnsitNet reduces the MSE by 20.5%, 14.8%, and 7.1% respectively compared to the second-best results. As shown in Fig. 4, ETTh2 exhibits more frequent trend fluctuations compared to other datasets, indicating the presence of numerous concept drift scenarios. Nonetheless, OnsitNet significantly reduces the error by 14.9%, fully demonstrating its ability to effectively capture these highly variable pattern shifts. However, OnsitNet does not show significant performance improvement on the Traffic dataset, and its performance is suboptimal on the ECL dataset. This may be due to OnsitNet's limitations in handling datasets with a large number of features. ER and its variants (DER++, MIR) also perform well in certain cases, highlighting the necessity of leveraging past experiences. We also consider non-online learning prediction models such as Informer. However, Informer's results are suboptimal, with large errors and even failing to converge, indicating its inability to adapt to the online learning setup. FSNet and Time-TCN, which possess online learning capabilities, effectively handle the data streams, achieving the third-best and second-best results respectively.

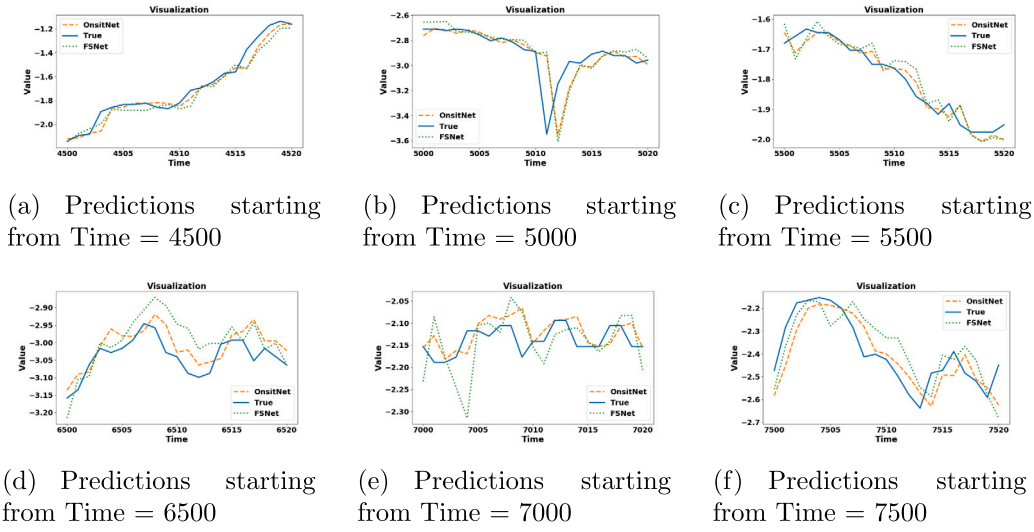
4.3. Visualization results

We visualize the actual values of "oil temperature" in the ETTm1 and the prediction results of OnsitNet and FSNet. To intuitively display the trends in different stages of prediction, we visualize two stages of data: 4500–5500 and 6500–7500. For clarity, each small stage consists of 20 timestamps, as shown in Fig. 5. As this paper focuses on online learning, the data during testing is fed into the model one instance at a time. Within this setup, the predictions made by OnsitNet largely follow the actual value trends, which sufficiently demonstrates that OnsitNet achieves accurate online forecasting. Moreover, even when encountering significant concept drift scenarios (as shown in Fig. 5(b) 5010–5015), OnsitNet do not collapse but instead maintain stable predictions, a feat that conventional time-series forecasting methods are unable to achieve. We also visualize FSNet, and it is evident that in both stages, the results of OnsitNet are closer to the true values compared to FSNet. Furthermore, in some cases (as shown in Fig. 5(e) 7004), FSNet's predicted trend oscillates significantly, whereas OnsitNet consistently maintains a stable forecasting state, which also reflects the commendable stability of OnsitNet.

4.4. The effective detection results of self-attention mechanism and SITransformer layer

As shown in Fig. 6, we visualize the average attention scores generated for the 100th and 200th executions of OnsitNet on the ETTh2, WTH, and Traffic datasets. By assigning the task of multivariate correlations to the attention mechanism, the learned graphs achieve stronger interpretability. The figure reveals different attention scores for different datasets, with particularly larger values observed for the WTH and Traffic dataset. This not only highlights the importance of considering the interrelationships among variables but also confirms the effectiveness of our use of the self-attention mechanism in extracting the relationships between variables.

And we aim to explore the overall contribution of the SITransformer layer to facilitating online learning. To achieve this, we replace the SITransformer within each learning module with a linear layer (OL-Linear), Transformer (OL-Transformer), Temporal Convolutional Network (OL-TCN), and Long Short-Term Memory network (OL-LSTM), while keeping other components unchanged. Experiments are conducted on five datasets, and the results are presented in Table 4. It can be seen that OnsitNet has the lowest error on almost all datasets. It is evident that, in most cases, OL-Transformer exhibits larger errors compared to other methods, surpassing even OL-Linear. This observation, consistent with findings by Zeng et al. (2023), suggests challenges with Transformers in handling time-series data in online learning scenarios. OL-TCN and OL-LSTM also achieve desirable results, but in the vast majority of scenarios, they are still inferior to OnsitNet. This is because the SITransformer can not only learn features within variables but also capture relationships between variables through the self-attention mechanism, thereby facilitating online learning more effectively.



(a) Predictions starting from Time = 4500 (b) Predictions starting from Time = 5000 (c) Predictions starting from Time = 5500

(d) Predictions starting from Time = 6500 (e) Predictions starting from Time = 7000 (f) Predictions starting from Time = 7500

Fig. 5. Data visualization of OnsitNet, FSNet, and actual Values.

Table 2
Cumulative MSE of each model for predicting the 1st, 24th, and 48th future time steps. '–' indicates that the model fails to converge.

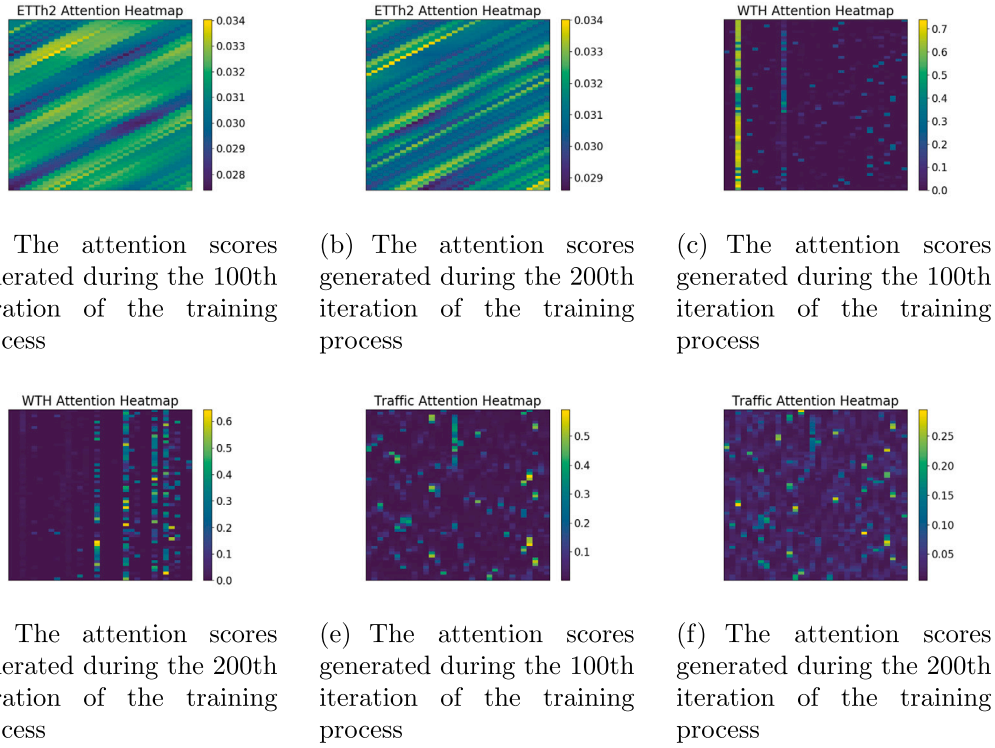
MSE	ETTh2			ETTm1			WTH			Traffic			ECL			Optimal number
	1	24	48	1	24	48	1	24	48	1	24	48	1	24	48	
Informer	7.571	4.629	5.692	0.456	0.478	0.388	0.426	0.380	0.367	–	–	–	–	–	–	0
OnlineTCN	0.502	0.830	1.183	0.085	0.258	0.283	0.206	0.308	0.302	0.329	0.463	0.498	3.309	11.339	11.534	0
TFCL	0.557	0.846	1.208	0.087	0.211	0.236	0.177	0.301	0.323	–	–	–	2.732	12.094	12.110	0
ER	0.508	0.808	1.136	0.086	0.202	0.220	0.180	0.293	0.297	0.291	0.391	0.394	2.579	9.327	9.685	0
MIR	0.486	0.812	1.103	0.085	0.192	0.210	0.179	0.291	0.297	–	–	–	**2.575**	9.265	9.411	1
DER++	0.508	0.828	1.157	0.083	0.196	0.208	0.174	0.287	0.294	0.289	0.387	0.384	2.657	8.996	9.009	0
FSNet	0.466	0.687	0.846	0.085	0.115	0.127	0.162	0.188	0.223	0.288	**0.362**	0.379	3.143	6.051	7.034	1
Time-TCN	0.491	0.779	1.307	0.093	0.281	0.308	0.158	0.311	0.308	0.321	0.469	0.536	4.060	**5.260**	**5.230**	2
OnsitNet	**0.443**	**0.586**	**0.672**	**0.081**	**0.108**	**0.118**	**0.158**	**0.176**	**0.190**	**0.288**	0.367	**0.376**	3.011	5.676	6.901	**11**

Table 3
Cumulative MAE of each model for predicting the 1st, 24th, and 48th future time steps. '–' indicates that the model fails to converge.

MAE	ETTh2			ETTm1			WTH			Traffic			ECL			Optimal number
	1	24	48	1	24	48	1	24	48	1	24	48	1	24	48	
Informer	0.850	0.668	0.752	0.512	0.525	0.460	0.458	0.417	0.419	–	–	–	–	–	–	0
OnlineTCN	0.436	0.547	0.589	0.214	0.381	0.403	0.276	0.367	0.362	0.283	0.363	0.382	0.635	1.196	1.235	0
TFCL	0.472	0.548	0.592	0.198	0.341	0.363	0.240	0.363	0.382	–	–	–	0.524	1.256	1.303	0
ER	0.376	0.543	0.571	0.197	0.333	0.351	0.244	0.356	0.363	0.252	0.302	0.307	0.506	1.057	1.074	0
MIR	0.410	0.541	0.565	0.197	0.325	0.342	0.244	0.355	0.361	–	–	–	0.504	1.066	1.079	0
DER++	0.375	0.540	0.577	0.192	0.326	0.340	0.235	0.351	0.359	0.248	0.295	0.295	0.421	1.035	1.048	0
FSNet	0.368	0.467	0.515	0.191	0.249	0.263	0.216	0.276	0.301	0.253	**0.288**	0.298	0.472	0.997	1.061	1
Time-TCN	0.425	0.544	0.636	0.211	0.395	0.421	**0.204**	0.378	0.378	0.280	0.321	0.351	**0.332**	**0.420**	**0.438**	4
OnsitNet	**0.366**	**0.449**	**0.483**	**0.185**	**0.241**	**0.254**	0.210	**0.268**	**0.282**	**0.246**	0.294	**0.295**	0.440	0.969	1.043	**10**

Table 4
Comparison of the effectiveness of SITransformer with Linear, Transformer, TCN and LSTM.

Method		OnsitNet		OL-Linear		OL-Transformer		OL-TCN		OL-LSTM	
Dataset	H	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh2	1	**0.443**	0.366	0.507	0.378	0.477	0.372	0.490	**0.363**	0.543	0.364
	24	**0.586**	**0.449**	0.679	0.449	1.247	0.569	0.630	0.457	0.674	0.466
	48	**0.681**	**0.480**	0.813	0.487	1.616	0.667	0.793	0.508	0.836	0.508
ETTm1	1	**0.081**	**0.185**	0.082	0.186	0.086	0.191	0.083	0.186	0.086	0.193
	24	0.108	0.241	0.106	0.239	0.123	0.257	**0.105**	**0.238**	0.106	0.239
	48	**0.118**	0.254	0.122	0.257	0.225	0.354	0.118	**0.251**	0.120	0.255
WTH	1	**0.161**	**0.214**	0.162	0.216	0.169	0.225	0.163	0.216	0.161	0.214
	24	**0.176**	**0.268**	0.188	0.273	0.184	0.275	0.189	0.273	0.179	0.269
	48	**0.190**	**0.282**	0.216	0.299	0.211	0.297	0.215	0.299	0.202	0.288
ECL	1	**3.011**	**0.440**	3.205	0.482	3.274	0.456	3.419	0.465	3.880	0.449
	24	**5.676**	**0.969**	9.342	0.969	15.273	0.970	9.466	0.972	10.707	0.982
	48	**6.901**	1.043	11.546	**0.981**	16.672	1.041	11.774	1.009	13.384	0.986
Traffic	1	**0.288**	**0.246**	0.330	0.262	0.346	0.279	0.328	0.262	0.327	0.258
	24	**0.367**	**0.294**	0.435	0.323	0.547	0.394	0.440	0.327	0.444	0.326
	48	**0.376**	**0.295**	0.462	0.340	0.570	0.409	0.472	0.346	0.471	0.343



(a) The attention scores generated during the 100th iteration of the training process

(b) The attention scores generated during the 200th iteration of the training process

(c) The attention scores generated during the 100th iteration of the training process

(d) The attention scores generated during the 200th iteration of the training process

(e) The attention scores generated during the 100th iteration of the training process

(f) The attention scores generated during the 200th iteration of the training process

Fig. 6. Attention heatmaps of OnsitNet when running on ETTh2, WTH and Traffic. (Averaged over attention matrices from all heads).

Further, we conduct experiments related to the internal components of the SITransformer. Meanwhile, for the sake of comparison, we also include w/o SITrans, which indicates the complete removal of the SITransformer. In addition to that, we mainly add two component experiments: (1) w/o SITrans-ATT indicates that the attention mechanism inside the SITransformer is removed and only the linear layer is retained. (2) w/o SITrans-Linear indicates that the linear layer inside the SITransformer is removed and only the attention layer is retained. Because our initial intention is to process under the inverted mechanism of the embedding layer, and the normalized layer only accelerates the training and improves the stability, the embedding and normalized layers are involved throughout these two component experiments. The prediction results on the five datasets are shown in the Table 5.

From the Table 5, it is easy to see that the average errors of w/o SITrans-ATT and w/o SITrans-Linear are between OnsitNet and w/o SITrans. This result indicates that both the attention mechanism and the linear layer within the SITransformer play irreplaceable roles. Specifically, w/o SITrans-Linear achieves larger errors than OnsitNet on each dataset, especially on the ETTh2 and WTH datasets, which increase the MSE values by 11.2% and 6.7% on average over OnsitNet. This indicates that the linear layer can effectively handle the global features of time series and capture the dynamics of long time series over time. Second, the mean error of w/o SITrans-ATT increases by 6.0% MSE values over OnsitNet, especially on the ETTh2 dataset by 31.8% MSE values on average. This suggests that capturing the relationship between variables is essential to improve the prediction performance. In addition, w/o SITrans-ATT achieves a larger average error than w/o SITrans-Linear, which suggests that on the whole the attentional mechanism facilitates the prediction more than the linear layer. However, both have their own areas of strength, for example, the linear layer is more useful than the attentional mechanism on the WTH dataset, and the attentional mechanism is more useful than the linear layer on the ETTh2 dataset.

4.5. The effective detection results of dilated convolution

In this section, we assess the effectiveness of dilated convolution. To this end, we conduct two sets of comparative experiments: one where the dilated convolution within the online learning layer is replaced with standard convolution, resulting in the model OnlineConv, and another where it is replaced with a linear layer, resulting in the model OnlineLinear. We test these models on five datasets, and the results are presented in Table 6.

As shown in Table 6, OnsitNet achieves the lowest error in the vast majority of cases, outperforming both OnlineConv and OnlineLinear. This result indicates that, in this experiment, dilated convolution performs better than both standard convolution and linear layers. Compared to standard convolution, dilated convolution can capture longer-range temporal features more effectively without increasing computational cost. Additionally, we set incremental dilated factors in different learning modules, with exponential growth, to comprehensively extract both local and long-term features of the time series. In contrast, standard convolution can only capture local features within a fixed range and has limited ability to adapt to long-term features. While linear layers are effective for simple or linear relationships, they have limitations in modeling nonlinear and multi-scale temporal features and struggle to capture both local and global complex dependencies simultaneously. In contrast, dilated convolution, by expanding the receptive field, can extract features across multiple time scales, better handling complex temporal patterns. Therefore, linear layers perform significantly worse than dilated convolution. OnlineConv and OnlineLinear exhibit comparable performance, each with its own strengths in specific data domains.

4.6. Ablation study

In this experiment, we evaluate the components of OnsitNet. This includes models without SITransformer (w/o SITrans), models without weighted averaging (w/o WA), models without memorizer (w/o ME),

Table 5
Effectiveness test of attention mechanism and linear within SITransformer.

Method		OnsitNet		w/o SITrans-Linear		w/o SITrans-ATT		w/o SITrans	
Dataset	H	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh2	1	**0.443**	**0.366**	0.533	0.367	0.464	0.371	0.537	0.393
	24	**0.586**	**0.449**	0.661	0.454	0.781	0.477	0.791	0.484
	48	**0.672**	**0.483**	0.699	0.483	0.997	0.558	0.992	0.516
ETTm1	1	**0.081**	**0.185**	0.082	0.186	0.082	0.185	0.083	0.187
	24	**0.108**	**0.241**	0.109	0.242	0.109	0.242	0.118	0.252
	48	**0.118**	**0.254**	0.122	0.258	0.124	0.260	0.128	0.263
WTH	1	**0.158**	**0.210**	0.163	0.215	0.160	0.213	0.162	0.216
	24	**0.176**	**0.268**	0.187	0.275	0.179	0.270	0.188	0.275
	48	**0.190**	**0.282**	0.209	0.295	0.203	0.287	0.209	0.297
ECL	1	**3.011**	**0.440**	3.099	0.529	3.309	0.442	3.321	0.534
	24	**5.676**	**0.969**	5.752	0.972	5.917	0.979	5.952	0.980
	48	**6.901**	**1.043**	6.910	1.049	6.913	1.044	7.053	1.073
Traffic	1	**0.288**	**0.246**	0.290	0.251	0.288	0.250	0.290	0.251
	24	**0.367**	**0.294**	0.369	0.298	0.377	0.310	0.422	0.314
	48	**0.376**	**0.295**	0.385	0.305	0.381	0.300	0.452	0.333
AVG		**1.276**	**0.401**	1.304	0.411	1.352	0.412	1.376	0.422

Table 6
Comparison of the effectiveness of dilated convolution, standard convolution, and linear.

Method		OnsitNet		OnlineConv		OnlineLinear	
Dataset	H	MSE	MAE	MSE	MAE	MSE	MAE
ETTh2	1	**0.443**	0.366	0.587	0.369	0.558	**0.364**
	24	**0.586**	**0.449**	0.600	0.453	0.813	0.493
	48	**0.672**	0.483	1.108	0.555	0.683	**0.470**
ETTm1	1	**0.081**	**0.185**	0.081	0.185	0.087	0.195
	24	**0.108**	**0.241**	0.108	0.242	0.113	0.247
	48	0.118	0.254	**0.116**	**0.252**	0.141	0.279
WTH	1	**0.158**	**0.210**	0.163	0.216	0.160	0.212
	24	**0.176**	**0.268**	0.187	0.276	0.185	0.275
	48	**0.190**	**0.282**	0.200	0.286	0.198	0.286
ECL	1	**3.011**	**0.440**	3.246	0.568	3.367	0.535
	24	**5.676**	**0.969**	5.716	0.977	5.904	0.995
	48	**6.901**	**1.043**	7.059	1.108	7.062	1.050
Traffic	1	**0.288**	**0.246**	0.289	0.253	0.290	0.253
	24	**0.367**	**0.294**	0.370	0.297	0.386	0.310
	48	**0.376**	**0.295**	0.378	0.298	0.378	0.300
AVG		**1.276**	**0.401**	1.347	0.423	1.355	0.417

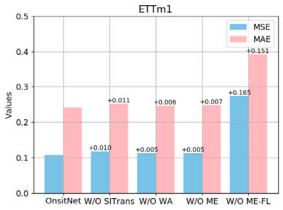
and models without both the memorizer and fast learner (w/o ME-FL). Here, w/o SITrans represents OnsitNet without the SITransformer. w/o WA represents the removal of the external weighted averager from both online learning layers. w/o ME represents OnsitNet without the memorizer. Without the memorizer, the fast learner does not need to interact with the memorizer through the Pearson trigger, thus losing both the Pearson trigger and the interaction mechanism. Hence, the memorizer, Pearson trigger, and interaction mechanism are all absent. w/o ME-FL maintains the fast learning mechanism using gradients but without EMA smoothing of the gradients. Learning is conducted solely through the learning coefficients μ, meaning EMA, memorizer, Pearson trigger, and interaction mechanism are all absent. We test these five models on all datasets, with a prediction timestamp of 24. The results are shown in Table 7 and Fig. 7. Fig. 7(a)–(e) show the results on five datasets, while (f) shows the average result across the five datasets.

It can be seen that OnsitNet achieves the lowest MSE and MAE, which fully demonstrates that the components of OnsitNet contribute to optimization to varying degrees. Among them, w/o ME-FL obtains the highest prediction error on all datasets, which precisely indicates that the memorizer and fast learner together hold the highest importance. This shows the necessity of dynamic interactions between the fast learner and the memorizer to reduce the impact of concept drift patterns. It also highlights the importance of using EMA smoothing for noise reduction in samples. The w/o SITrans model generally achieves the second worst prediction results, demonstrating that the offline model SITransformer significantly enhances online learning, with inter-variable correlations being a critical factor affecting prediction accuracy. Moreover, the impact of SITransformer is more pronounced on the ETTh2 dataset. Combining this observation with Fig. 4, it can be inferred that the SITransformer better assists the online learning layer in dealing with concept drift scenarios. The w/o WA and w/o ME models also perform worse than OnsitNet, indicating that both components contribute to varying degrees of optimization. On the one hand, the necessity of memorizing past patterns is illustrated. Conceptual drift patterns stored in the memorizer can assist in the learning of the current pattern, and do not have to be retrained when a similar scene is encountered again thus reducing the negative effects of conceptual drift patterns. On the other hand, since we set a small ρ value, the results of w/o WA indicate that the second online learning layer plays a more significant role than the first online learning layer in actual predictions.

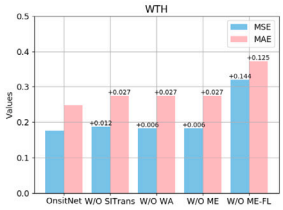
4.7. Hyperparameter analysis

We conduct experiments to evaluate the impact of two hyperparameters: the Pearson trigger coefficient (γ) and the weighted averaging parameter (ρ) outside the online learning layer. Each hyperparameter is tested with four contrast parameters, and experiments are conducted on the five datasets, with results presented in Tables 8 and 9. From

(a) Prediction results of OnsitNet components on the ETTh2 dataset

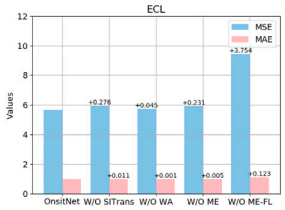


(b) Prediction results of OnsitNet components on the ETTm1 dataset

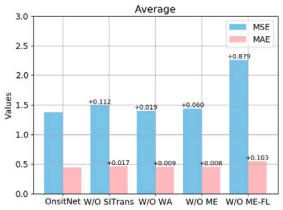


(c) Prediction results of OnsitNet components on the WTH dataset

(d) Prediction results of OnsitNet components on the Traffic dataset



(e) Prediction results of OnsitNet components on the ECL dataset



(f) Average prediction results of OnsitNet components on all datasets

Fig. 7. Ablation study of OnsitNet on five datasets. The prediction horizon is 24 timestamps. The text in the figure represents the error differences between other models and OnsitNet.

Table 7
Detailed ablation experiment results on five datasets. The prediction horizon is 24 timestamps.

Dataset		ETTh2	ETTm1	WTH	Traffic	ECL	AVG
OnsitNet	MSE	**0.586**	**0.108**	**0.176**	**0.367**	**5.676**	**1.382**
	MAE	**0.449**	**0.241**	**0.248**	**0.294**	**0.969**	**0.444**
w/o SITrans	MSE	0.791	0.118	0.188	0.422	5.952	1.494
	MAE	0.484	0.252	0.275	0.314	0.980	0.461
w/o WA	MSE	0.592	0.113	0.182	0.400	5.721	1.401
	MAE	0.453	0.247	0.275	0.323	0.970	0.453
w/o ME	MSE	0.626	0.113	0.182	0.386	5.907	1.442
	MAE	0.452	0.248	0.275	0.314	0.974	0.452
w/o ME-FL	MSE	0.857	0.273	0.320	0.429	9.430	2.261
	MAE	0.545	0.392	0.373	0.337	1.092	0.547

Table 8, it is observed that, for the most part, optimal results are achieved when γ equals 0.75. In most cases, as γ gradually increases or decreases, the error will also become larger. This supports our initial consideration that setting γ too small introduces noise, while setting it too large may lead to the loss of important pattern changes. We also observe that γ has a smaller impact on WTH, ETTm1, and Traffic, and a larger impact on ECL and ETTh2. As previously analyzed, ETTh2 contains more concept drift scenarios, making it more sensitive to changes in γ. The ECL dataset, on the other hand, exhibits greater variability in data patterns, posing higher challenges for the model's adaptability and parameter adjustment. Results from Table 9 indicate that the error for $\rho = 0.1$ is larger than the error for $\rho = 0.2$. As ρ increases, the error generally increases as well. This suggests that the importance of the online learning layer in the first layer is relatively low, emphasizing the critical role of the online learning layer in the second layer. This difference underscores the necessity of weighted averaging and aligns with the trend of increasing impact due to the gradual approach of timestamps in time-series data.

Table 8
Results of predicting future 24th timestamp with different γ values on the five datasets.

γ		0.75	0.8	0.9	0.5	0.3
ECL	MSE	**5.676**	6.318	6.336	6.167	6.207
	MAE	0.969	0.972	0.965	0.967	**0.962**
WTH	MSE	**0.176**	0.186	0.191	0.178	0.185
	MAE	**0.268**	0.275	0.284	0.269	0.274
ETTh2	MSE	**0.586**	0.610	0.617	0.632	0.682
	MAE	**0.449**	0.451	0.453	0.449	0.461
ETTm1	MSE	**0.108**	0.111	0.112	0.112	0.113
	MAE	**0.241**	0.246	0.247	0.246	0.247
Traffic	MSE	**0.367**	0.421	0.423	0.422	0.422
	MAE	**0.294**	0.312	0.312	0.311	0.312

5. Conclusion

Our Work: For the online time series prediction scenario, we propose an online learning model with memory capability and an integrated self-attention mechanism-OnsitNet. OnsitNet adaptively adjusts the receptive field of dilated convolutions within the learning modules to capture complex data features at different scales. The role of the SITransformer is to capture dependencies between variables through mechanisms like self-attention, providing each variable with richer feature information. The online learning layer rapidly processes data streams through the fast learner and interacts with the memorizer via the Pearson trigger, effectively addressing concept drift patterns.

Table 9
Results of predicting future 24th timestamp with different ρ values on the five datasets.

ρ		0.2	0.1	0.4	0.6	0.8
ECL	MSE	**5.676**	6.041	6.282	6.688	6.775
	MAE	**0.969**	0.972	0.974	0.998	1.002
WTH	MSE	**0.176**	0.190	0.176	0.178	0.185
	MAE	**0.268**	0.278	0.268	0.268	0.274
ETTh2	MSE	**0.586**	0.618	0.616	0.634	0.644
	MAE	0.449	0.451	**0.447**	0.447	0.449
ETTm1	MSE	**0.108**	0.109	0.113	0.113	0.115
	MAE	**0.241**	0.243	0.248	0.248	0.250
Traffic	MSE	**0.367**	0.416	0.408	0.417	0.419
	MAE	**0.294**	0.306	0.298	0.307	0.308

Experimental results on five real-world datasets indicate that OnsitNet achieves nearly optimal prediction performance in the scenario of online time series forecasting. In summary, we offer an effective online forecasting solution for real-world scenarios where data streams continuously arrive.

Future Work: In the experiments conducted in this paper, we observe that OnsitNet demonstrates moderate performance in scenarios with an abundance of features, which will inform a key area of our future research. Furthermore, since data in the real world is not static but continuously transmitted as data streams, online learning will persist as a primary focus of our explorative efforts.

CRediT authorship contribution statement

Hui Liu: Conceptualization, Funding acquisition, Resources, Supervision, Validation, Writing – review & editing. **Zhengkai Wang:** Conceptualization, Data curation, Investigation, Formal analysis, Methodology, Software, Visualization, Writing – original draft. **Xiyao Dong:** Data curation, Investigation, Software. **Junzhao Du:** Funding acquisition, Project administration, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The ECL and WTH are available for download at https://drive.google.com/drive/folders/1ohGYWWohJlOlb2gsGTeEq3Wii2egnEPR.

The ETTh2 and ETTm1 are available for download at https://github.com/zhouhaoyi/ETDataset/tree/main/ETT-small.

And Traffic can be download from https://drive.google.com/drive/folders/1M3gTc1DSvnUFMI57p70VFH5MHhZh3wC8.

Acknowledgments

This work is partially supported by a grant from the National Natural Science Foundation of China (No. 62032017, No. 62272368), Key Talent Project of Xidian University, China (No. QTZX24004), the Innovation Capability Support Program of Shaanxi, China (No. 2023-CX-TD-08), Shaanxi Qinchuangyuan "scientists+engineers" team, China (No. 2023KXJ-040), Science and Technology Program of Xi'an, China (No. 23KGDW0005-2022).

Appendix A. Algorithm

The overall design philosophy of OnsitNet is shown in Algorithm 1.

Algorithm 1 OnsitNet

Require: Input feature X, True value Y, interaction threshold γ, backbone θ, memorizer M, trigger = False

for t=1,,,T **do**
  Receive the t-look-back window X_t
  for j=1,,,N **do** *//Execute learning modules*
    $X = X_t$ if j=1 else $X = H_{j-1}^{Learn}$
    $h_0 = SITrans(X)$ *//Through the SITransformer layers*
    for i=1,,,2 **do** *//Execute online learning layers*
      if trigger=True **then**
        $\widehat{\mu_l} \leftarrow Read(\overline{\mu_l}, M_l)$
        $M_l \leftarrow Write(M_l, \overline{\mu_l})$ *//Write to memorizer*
        $\mu_l = \delta\mu_l + (1-\delta)\widehat{\mu_l}$ *//Update learning coefficient*
      $\vartheta_l = Title(\alpha_l) \odot \theta_l$
      $\widehat{H^O} = \widetilde{H^O} + DilatedConv(\widetilde{H^O})$
      $H_i^O = Title(\beta_l) \odot \widehat{H^O}$
    end for
    $H_j^{Learn} = \rho H_1^O + (1-\rho) H_2^O$
  end for
  $\hat{Y} = Predict(H_N^{Learn})$
  Calculate the L = Loss(Y,$\hat{Y}$) and backpropagate
  Updating Predict's parameters via SGD
  for j=1,,,N **do**
    for i=1,,,2 **do**
      Updating the EMA gradient within the online learning layer and updating the internal parameters via SGD, if $Pearson(\vec{g}_l, \vec{g}_l{}') < -\gamma$ then trigger=True
      Updating the parameters of the SITransformer layer via SGD
    end for
  end for
end for

Appendix B. Additional experimental details

Our experiments adhere to the protocol established by Pham et al. (2023), dividing into a warm-up phase and an online training phase, with the experimental data split set at a 1:3 ratio. In the warm-up phase, online training samples are normalized by computing the mean and standard deviation, and hyperparameter cross-validation is performed. We set both the epoch and batch size to 1 to comply with the requirements of online learning. The Adam optimizer and mean squared error loss function are utilized. The look-back window length is fixed at 60, with forecast horizons ranging from 1 to 48. The convolution kernel for the dilated convolution is 3, and the dilation factor doubles with each additional online learning module, starting with an initial dilation factor of 1. There are 11 learning modules, and each learning module contains 2 online learning layers. All experiments are conducted using the PyTorch framework and executed on a GPU NVIDIA GeForce RTX 3060.

References

Aljundi, R., Caccia, L., & Belilovsky, E. (2019a). Online continual learning with maximally interfered retrieval. *Advances in Neural Information Processing Systems*, *32*, 11849–11860, https://proceedings.neurips.cc/paper/2019/hash/15825aee15eb335cc13f9b559f166ee8-Abstract.html.

Aljundi, R., Kelchtermans, K., & Tuytelaars, T. (2019b). Task-free continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11254–11263). https://openaccess.thecvf.com/content_CVPR_2019/html/Aljundi_Task-Free_Continual_Learning_CVPR_2019_paper.html.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., & Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline. *Advances in Neural Information Processing Systems*, *33*, 15920–15930, https://proceedings.neurips.cc/paper/2020/hash/b704ea2c39778f07c617f6b7ce480e9e-Abstract.html.

Challu, C., Olivares, K. G., Oreshkin, B. N., Ramirez, F. G., Canseco, M. M., & Dubrawski, A. (2023). Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 37*, (6), (pp. 6989–6997). http://dx.doi.org/10.1609/aaai.v37i6.25854.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., et al. (2019). On tiny episodic memories in continual learning. http://dx.doi.org/10.48550/arXiv.1902.10486, arXiv preprint arXiv:1902.10486.

Du, D., Su, B., & Wei, Z. (2023). Preformer: predictive transformer with multi-scale segment-wise correlations for long-term time series forecasting. In *ICASSP 2023-2023 IEEE international conference on acoustics, speech and signal processing* (pp. 1–5). IEEE, http://dx.doi.org/10.1109/ICASSP49357.2023.10096881.

Gultekin, S., & Paisley, J. (2018). Online forecasting matrix factorization. *IEEE Transactions on Signal Processing*, *67*(5), 1223–1236. http://dx.doi.org/10.1109/TSP.2018.2889982.

Gupta, V., Narwariya, J., Malhotra, P., Vig, L., & Shroff, G. (2021). Continual learning for multivariate time series tasks with variable input dimensions. In *2021 IEEE international conference on data mining* (pp. 161–170). IEEE, http://dx.doi.org/10.1109/ICDM51629.2021.00026.

Hewage, P., Behera, A., Trovati, M., Pereira, E., Ghahremani, M., Palmieri, F., et al. (2020). Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, *24*, 16453–16482. http://dx.doi.org/10.1007/s00500-020-04954-0.

Kilian, L., & Lütkepohl, H. (2017). *Structural vector autoregressive analysis*. Cambridge University Press.

Kurle, R., Cseke, B., Klushyn, A., Van Der Smagt, P., & Günnemann, S. (2019). Continual learning with bayesian neural networks for non-stationary data. In *International conference on learning representations*. https://openreview.net/forum?id=SJlsFpVtDB.

Lai, G., Chang, W.-C., Yang, Y., & Liu, H. (2018). Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval* (pp. 95–104). http://dx.doi.org/10.1145/3209978.3210006.

Lee, Y.-S., & Tong, L.-I. (2011). Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming. *Knowledge-Based Systems*, *24*(1), 66–72. http://dx.doi.org/10.1016/j.knosys.2010.07.006.

Li, S. C.-X., & Marlin, B. (2020). Learning from irregularly-sampled time series: A missing data perspective. In *International conference on machine learning* (pp. 5937–5946). PMLR, https://proceedings.mlr.press/v119/li20k.html.

Li, W., Yang, X., Liu, W., Xia, Y., & Bian, J. (2022). Ddg-da: Data distribution generation for predictable concept drift adaptation. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 36*, (4), (pp. 4092–4100). http://dx.doi.org/10.1609/aaai.v36i4.20327.

Liu, C., Hoi, S. C., Zhao, P., & Sun, J. (2016). Online arima algorithms for time series prediction. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 30*, (1), http://dx.doi.org/10.1609/aaai.v30i1.10257, Article 10257.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., et al. (2024). Itransformer: Inverted transformers are effective for time series forecasting. In *International conference on learning representations*. https://openreview.net/forum?id=JePfAI8fah.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., et al. (2021). Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*. https://openreview.net/forum?id=0EXmFzUn5I.

Murat, M., Malinowska, I., Gos, M., & Krzyszczak, J. (2018). Forecasting daily meteorological time series using ARIMA and regression models. *International agrophysics*, *32*(2), http://dx.doi.org/10.1515/intag-2017-0007, Article 0007.

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. In *International conference on learning representations*. https://openreview.net/forum?id=oigW0ASMNz.

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International conference on learning representations*. https://iclr.cc/virtual_2020/poster_r1ecqn4YwB.html.

Pham, Q., Liu, C., Sahoo, D., & Hoi, S. C. (2023). Learning fast and slow for online time series forecasting. In *International conference on learning representations*. https://openreview.net/forum?id=RNMIIZwzCg.

Sahoo, D., Pham, Q., Lu, J., & Hoi, S. C. (2017). Online deep learning: Learning deep neural networks on the fly. http://dx.doi.org/10.48550/arXiv.1711.03705, arXiv preprint arXiv:1711.03705.

Sen, R., Yu, H.-F., & Dhillon, I. S. (2019). Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, *32*, https://proceedings.neurips.cc/paper/2019/hash/3a0844cee4fcf57de0c71e9ad3035478-Abstract.html.

Shih, S.-Y., Sun, F.-K., & Lee, H.-y. (2019). Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, *108*, 1421–1441. http://dx.doi.org/10.1007/s10994-019-05815-0.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*, https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., & Xiao, Y. (2023). Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*. https://openreview.net/forum?id=zt53IDUR1U.

Wang, S., Wu, H., Shi, X., Hu, T., Luo, H., Ma, L., et al. (2024). Timemixer: Decomposable multiscale mixing for time series forecasting. In *International conference on learning representations*. https://openreview.net/forum?id=7oLshfEIC2.

Wen, Q., Chen, W., Sun, L., Zhang, Z., Wang, L., Jin, R., et al. (2024). Onenet: Enhancing time series forecasting models under concept drift by online ensembling. *Advances in Neural Information Processing Systems*, *36*, https://proceedings.neurips.cc/paper_files/paper/2023/hash/dd6a47bc0aad6f34aa5e77706d90cdc4-Abstract-Conference.html.

Woo, G., Liu, C., Sahoo, D., Kumar, A., & Hoi, S. (2022). Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. http://dx.doi.org/10.48550/arXiv.2202.01575, arXiv preprint arXiv:2202.01575.

Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, *34*, 22419–22430, https://proceedings.neurips.cc/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html.

Xu, D., Cheng, W., Zong, B., Song, D., Ni, J., Yu, W., et al. (2020). Tensorized LSTM with adaptive shared memory for learning trends in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 34*, (02), (pp. 1395–1402). http://dx.doi.org/10.1609/aaai.v34i02.5496.

Yadav, H., & Thakkar, A. (2024). NOA-LSTM: An efficient LSTM cell architecture for time series forecasting. *Expert Systems with Applications*, *238*, Article 122333. http://dx.doi.org/10.1016/j.eswa.2023.122333.

Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2023). Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 37*, (9), (pp. 11121–11128). http://dx.doi.org/10.1609/aaai.v37i9.26317.

Zhang, C., Sheng, Z., Zhang, C., & Wen, S. (2024). Multi-lead-time short-term runoff forecasting based on ensemble attention temporal convolutional network. *Expert Systems with Applications*, *243*, Article 122935. http://dx.doi.org/10.1016/j.eswa.2023.122935.

Zhang, Y., & Yan, J. (2023). Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International conference on learning representations*. https://openreview.net/forum?id=vSVLM2j9eie.

Zhang, T., Zhang, Y., Cao, W., Bian, J., Yi, X., Zheng, S., et al. (2022). Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures. http://dx.doi.org/10.48550/arXiv.2207.01186, arXiv preprint arXiv:2207.01186.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., & Jin, R. (2022). Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning* (pp. 27268–27286). PMLR, https://proceedings.mlr.press/v162/zhou22g.html.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., et al. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 35*, (12), (pp. 11106–11115). http://dx.doi.org/10.1609/aaai.v35i12.17325.