Pattern Recognition 158 (2025) 110985

Contents lists available at ScienceDirect

Pattern Recognition

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

Dynamic convolutional time series forecasting based on adaptive temporal bilateral filtering

Dandan Zhang [a], Zhiqiang Zhang [a], Nanguang Chen [b], Yun Wang [a,*]

[a] School of Computer Science and Engineering, Southeast University, Nanjing, China
[b] College of Design and Engineering, National University of Singapore, Singapore

ARTICLE INFO

Keywords:
Nonlinear feature
Adaptive temporal bilateral filtering
Gated deformable convolution
Time series forecasting

ABSTRACT

Time series data typically contain complex dynamic patterns, which not only include linear trends and seasonal variations but also significant nonlinear changes and complex dependencies. Currently, feature extraction methods for time series data primarily employ mixed-mode extraction within the temporal domain, neglecting the effective extraction and analysis of nonlinear characteristics between different observation points and the interdependencies between variables. To address these issues, we designed an adaptive temporal bilateral filtering module that effectively preserves and highlights the nonlinear features and patterns in time series while filtering out noise and redundant information. We also designed a nonlinear feature adaptive extraction module that integrates a gating mechanism and deformable convolutions. This design allows the model to adaptively adjust the shape of the convolutional kernels according to different time steps and conditions. This enables accurate capture and extraction of nonlinear features between various time observations and dependencies between different variables. Additionally, we use stacked convolutional layers to extract local contextual features, addressing fluctuations in local features caused by changes in data distribution in real-world scenarios. In summary, we propose DCNet, a dynamic convolutional network based on an adaptive temporal bilateral filter, and evaluated its performance on twelve real-world datasets. The results indicate that DCNet consistently achieves state-of-the-art performance in both short-term and long-term forecasting tasks, with favorable runtime efficiency.

1. Introduction

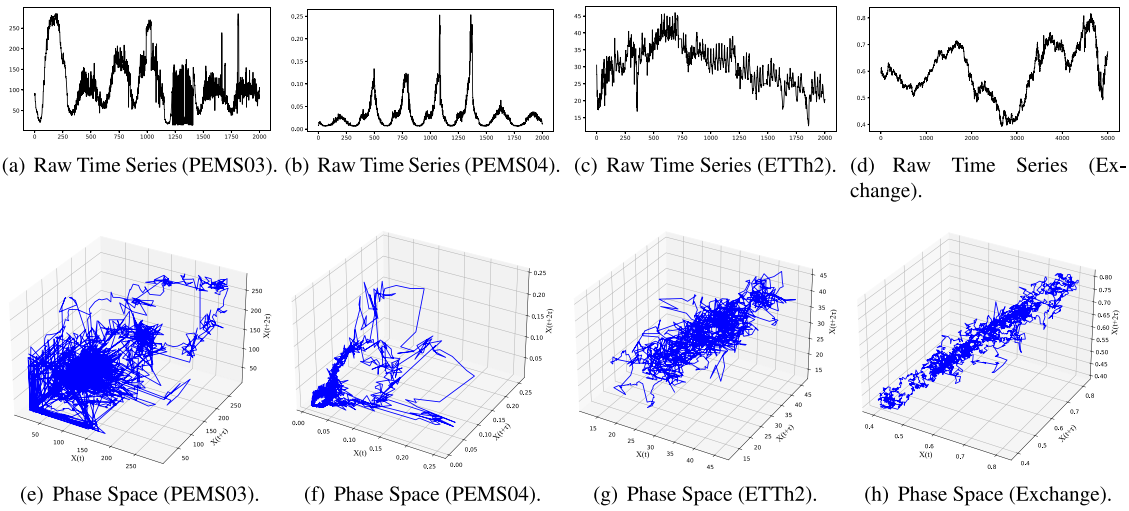
Time-series forecasting (TSF) is one of the most critical challenges in time series analysis, with wide-ranging applications in fields such as energy [1], financial [2,3], traffic [4], weather [5], and more. Its core objective is to leverage past time series data to forecast changing trends over a future time horizon. Effective feature extraction from time series data is essential for improving forecasting accuracy. However, in real-world scenarios, time series data often exhibit strong nonlinear features due to factors such as environmental conditions and equipment usage, which pose significant challenges for effective feature extraction. Specifically, we face the following three challenges.

Firstly, extracting and characterizing complex nonlinear patterns is very difficult. The data collected by sensors are highly sensitive to environmental changes and external interference, leading to significant increases in abrupt information and uncertainty within time series. This situation exacerbates the complexity of relationships between different observation points, making effective feature recognition extremely difficult. **Secondly, extracting dependencies between different variables is challenging.** Time series data in real-world scenarios may exhibit dynamic patterns. These data not only contain simple temporal dependencies but may also involve interaction effects between multiple variables, nonlinear trends, and more. These complex features significantly increase the difficulty of data analysis and modeling. **Thirdly, the effective extraction of local dynamic features in complex environments is challenging**. The distribution of time series data in real-world scenarios may change over time (such as the introduction of new trends or the alteration of existing ones), which can affect the model's understanding of local features.

To further illustrate the complex nonlinear characteristics of time series data in real-world scenarios, we performed phase space reconstruction on real datasets from different fields. Fig. 1 shows the phase space reconstruction diagrams of these datasets. The following observations can be made: (1) the trajectory shapes are complex and variable; (2) the trajectories cluster in multiple specific regions; (3) the trajectories are close to each other and intricately intertwined. These characteristics indicate that there are strong nonlinear relationships

* Corresponding author.
E-mail address: ywang_cse@seu.edu.cn (Y. Wang).

https://doi.org/10.1016/j.patcog.2024.110985
Received 16 June 2024; Received in revised form 11 August 2024; Accepted 5 September 2024
Available online 10 September 2024



(a) Raw Time Series (PEMS03). (b) Raw Time Series (PEMS04). (c) Raw Time Series (ETTh2). (d) Raw Time Series (Exchange).

(e) Phase Space (PEMS03). (f) Phase Space (PEMS04). (g) Phase Space (ETTh2). (h) Phase Space (Exchange).

Fig. 1. Nonlinear features of data.

and dynamic behaviors between different time observations in the time series, which greatly limit the effective extraction of time series data features. Additionally, we find that the PEMS03 and PEMS04 datasets [6] exhibit significant noise effects, causing data points to be scattered in the phase space, with chaotic and disordered trajectories. This makes it challenging to identify any periodicity or clear dynamic patterns, thereby complicating the feature extraction process.

To effectively extract complex features and patterns in multivariate time series, a series of Transformer-driven models have been proposed, including PatchTST [7], iTransformer [8], and Crossformer [9]. These models leverage the self-attention mechanism to capture complex long-term dependencies in multivariate time series but often overlook the understanding of local contextual information. Furthermore, long-term correlation information is highly sensitive to changes in data distribution, which affects the self-attention mechanism's ability to capture and understand long-term dependencies. As demonstrated by DLinear [10] and the experiments in Section 4.8, Transformer-based models generally perform poorly in capturing global temporal correlations in time series. Additionally, the use of the self-attention mechanism as a core feature extraction method in Transformer-based models results in high computational costs [10].

To enhance computational efficiency, TSMixer [11], LightTS [12], and DLinear [10] employ simple linear structures for extracting and analyzing time series data, outperforming most Transformer-based models in TSF. However, simple linear models have limited capability in understanding nonlinear features, hindering their performance on nonlinear datasets. The PatchTST [7] model reduces the computational complexity by dividing the input sequence into subsequences. These models mainly focus on extracting global correlation information in the horizontal time domain, neglecting the understanding of local contextual information and the correlation of features between different variables, which are crucial for TSF.

Benefiting from their ability to extract local contextual information, convolutional network-based prediction models have demonstrated outstanding performance in TSF [13,14]. These models typically utilize standard convolutional modules to extract local features from time series, effectively preserving local contextual information and the interdependencies between different variables to some extent. However, the symmetric receptive fields of these convolutional layers [15] limit their ability to perceive asymmetric features, posing challenges in capturing the nonlinear characteristics and dynamic patterns of sequences.

To address these challenges, we focus on the effective extraction of nonlinear features, building on the foundation of effectively extracting linear features in complex time series. This includes capturing the nonlinear dependencies between different time observations and the interdependencies among variables. Specifically, to extract and characterize the nonlinear features and patterns of complex time series, we propose an Adaptive Temporal Bilateral Filtering (ATBF) module. This module dynamically weights temporal and spatial information to highlight and preserve the nonlinear characteristics and edge pattern information in the time series, thereby effectively addressing Challenge 1. To capture the nonlinear features between different time observations and the complex interdependencies among variables in time series, we designed a Nonlinear Feature Adaptive Extraction (NFAE) module that integrates gating mechanisms with deformable convolutions. This module adaptively deforms the convolutional kernels, enhancing the model's flexibility and capability in feature extraction. It helps the model better capture complex abrupt patterns and dependencies in the time series, thereby addressing Challenge 2. To enhance the model's understanding of local features across different data distributions, we stacked convolutional layers with varying receptive field sizes into a Temporal Feature Extraction (TFE) module to extract local contextual features and patterns in the time series. This approach effectively captures local fluctuations and short-term changes in the time series, thereby addressing Challenge 3.

Based on the above analysis, we propose DCNet, a dynamic convolutional network architecture based on an adaptive temporal bilateral filter. The main contributions are as follows:

- A simple yet effective dynamic convolutional network, DCNet, based on adaptive temporal bilateral filtering, designed for long-term and short-term time series forecasting, aims to accurately extract diverse features and patterns. Superior predictive performance compared to state-of-the-art (SOTA) approaches on multiple publicly available datasets.
- We designed an ATBF module that dynamically weights the importance of temporal features, allowing for more accurate preservation and highlighting of nonlinear changes and dynamic patterns in the data.
- We designed an NFAE module that endows convolutional kernels with the ability to more flexibly adapt to different patterns and features through a gating mechanism. This helps to accurately capture nonlinear features and complex variable dependencies in time series while significantly reducing model redundancy.
- We used the TFE module to extract multi-granularity contextual information from the time series, capturing dynamic changes in local features while reducing the impact of data distribution changes on feature extraction.

The remaining structure of this paper is as follows: In Section 2, we introduce existing methods for time series forecasting and the

application of deformable convolutions. Section 3 provides a detailed description of the DCNet architecture. Section 4 presents experimental results validating the effectiveness and efficiency of DCNet. Finally, in Section 5, we provide a comprehensive summary of the predictive research on the DCNet model and outline prospects for future research.

2. Related work

In this section, we delineate the related work into two distinct modules. Section 2.1 provides an overview of prominent contributions within the field of time series forecasting, while Section 2.2 delves into the applications of deformable convolutions in other fields.

2.1. Time modeling in time series forecasting

Deep learning models have achieved significant success in time series forecasting, and they can be primarily categorized into three paradigms: CNN-based models, Transformer-based models, and MLP-based models.

CNN-based models utilize convolutional kernels along the temporal dimension to effectively capture local temporal patterns. For instance, MICN [14] employs multi-scale convolution to extract both global and local contextual relationships. SCINet [16] decomposes time series into multiple resolutions dynamically through SCI-Blocks in a recursive downsampling fashion to extract features at multiple scales. TSLANet [17] learns long-term and short-term relationships in the data through convolutional operations. ConvTimeNet [13] adaptively segments time series into patches and integrates deepwise convolution and pointwise convolution operations to capture global sequence dependencies and cross-variable interactions. However, the aforementioned models are limited to extracting nonlinear features within a fixed receptive field, which hinders their ability to capture asymmetric nonlinear features.

Transformer-based models have garnered widespread attention due to their capability in capturing long-range dependencies through attention mechanisms. For instance, models such as Robformer [18], ETSformer [19] and FEDformer [20] decompose time series data to extract complex patterns. CrossFormer [9] captures temporal and cross-dimensional dependencies in multivariate time series data by first transforming it into a 2D array and then employing two-stage attention layers. PatchTST [7] employs a channel independence strategy specifically designed to extract correlations of each channel in multivariate data. Pathformer [21] divides the time series into patches of different scales and designs attention mechanisms within and between patches. iTransformer [8] effectively captures multivariate correlations by independently embedding each variable in the time series as variable sub-tokens. However, as the forecasting horizon increases, the attention mechanism's extraction capability may diminish.

Models based on MLPs utilize simple linear models to extract abstract representations of time series, such as LightTS [12], DLinear [10] decomposes time series and utilizes linear network layers for modeling to achieve prediction. TSMixer [11] combines Patch and MLP, enabling it to extract both temporal correlations and inter-channel correlations. TimeMixer [22] conduct long-term and short-term forecasting by decoupling historical information of complex patterns in multiscale time series. However, when confronted with high-dimensional data, the expressive capacity of linear networks is constrained, thereby impeding their ability to accurately capture nonlinear features within complex and noisy time series.

2.2. Application of deformable convolutions

Deformable convolutions have found widespread application in the field of image recognition. Recent research has increasingly combined deformable convolutions with attention mechanisms. For instance, in [23], researchers utilized deformable convolutions to capture significant offset information in specific spatial structures, thereby enhancing recognition precision. In [24], due to the complex geometric transformations and feature blurring present in the data, A2-DCNet modules were employed to capture remote spatial context information from a global perspective.

Furthermore, research utilizing deformable convolutions in object detection tasks has yielded promising results In [25], a background-guided deformable convolutional autoencoder network was proposed, effectively separating anomalies from complex backgrounds and enhancing anomaly detection capabilities. Moreover, [26] showed that stacking deformable convolutions and integrating semantic segmentation improved the understanding of contextual relationships. Finally, [27] illustrated that deformable convolutions, akin to spatial attention, could be employed for multiscale feature extraction, while channel attention was used to identify significant features.

3. Methodology

In this section, we provide a detailed overview of the DCNet architecture. As shown in Fig. 2, the DCNet model aims to effectively extract temporal pattern features from multidimensional historical data, capture complex nonlinear information between different observations, and identify correlations among different variables. These capabilities enable DCNet to achieve accurate time series forecasting. The prediction process of the DCNet model mainly consists of three stages: 3.2 Data processing, 3.3 Feature acquisition, and 3.4 Model prediction.

3.1. Problem statement

In a rolling prediction setup with a fixed-size window, we consider an input sequence denoted as $X = \{x_1, \ldots, x_i, \ldots, x_M\} \in \mathbb{R}^{M \times N}$, where $x_i = \left\{x_i^1, \ldots, x_i^j, \ldots, x_i^N\right\}$. Here, M denotes the number of time points in the input, N represents the number of input variables, and x_i^j represents the value of variable j at the ith time point. The objective of long-term prediction for multivariate time series is to forecast $x_{M+L_y} = \left\{x_{M+1}, \ldots, x_{M+L_y}\right\}$, where the output length L_y represents an extended future time period.

3.2. Data processing

Time series data in real-world scenarios contain a significant amount of nonlinear features. These nonlinear features can mislead the linear feature extraction process, making it difficult for the model to capture the true patterns and relationships in the data, which leads to a decline in prediction performance. Additionally, due to the complexity and variability of nonlinear features, the model may select incorrect features for training, severely affecting its effectiveness and accuracy. Therefore, effectively identifying and handling nonlinear features has become a key challenge in improving the performance of time series prediction models.

To enhance the robustness of the model, we normalize the data and convert it into time series windows for training and testing. We employ a normalization method to adjust the time series data to a unified scale, expressed as

$$F(i) = \frac{x_i - \mu}{\delta} \quad , \tag{1}$$

where μ and δ are the mode-wise mean and variance vectors in the training time series.

To effectively highlight and preserve important nonlinear features and edge information in complex time series, we developed an ATBF module, as shown in Fig. 2. This filter adapts to different local statistical characteristics by calculating spatial weights between time steps and

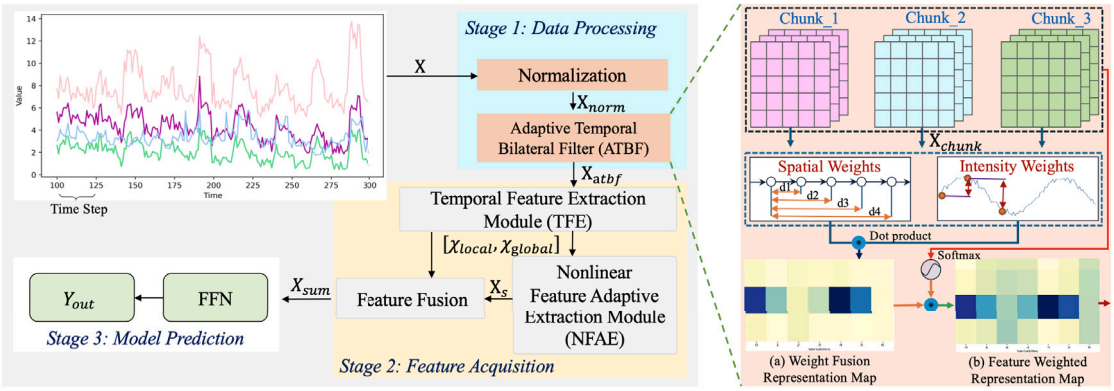


Fig. 2. Framework of the DCNet. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

dynamically adjusting the parameters of intensity weights based on the differences between adjacent time values, thereby effectively preserving nonlinear features. The main steps of this process are as follows:

Multi-chunk parallel processing. Traditional temporal bilateral filtering employs sequential point-by-point computation of spatial and intensity weights, resulting in high computational complexity ($O(M \times W)$, where W represents the window size of the neighborhood range) [28]. To reduce computational complexity, we employ a data chunking parallel processing method. First, we process the data into batch size (B) format, then input it into the $Split$ function to divide it into multiple data chunks along the time dimension, serving as the input units for the model. By leveraging data chunking and parallelism, we can reduce M by a factor of the chunk size P: M/P, thus significantly reducing computational complexity ($O(\frac{M}{P} \times W)$). For the derivation process, please refer to Appendix A.

$$
\begin{aligned}
&X_{norm} = [[F(x_1), F(x_2), \dots, F(x_M)], \dots, \\
&\quad [F(x_{M*(B-1)+1}), F(x_{M*(B-1)+2}), \dots, F(x_{M*B})]] \in \mathbb{R}^{B\times M\times N}, \\
&X_{split} = Split(X_{norm}, dim = 1) \\
&\quad = [X^1_{chunk} \in \mathbb{R}^{B\times P\times N}, X^2_{chunk} \in \mathbb{R}^{B\times P\times N}, \dots, X^C_{chunk} \in \mathbb{R}^{B\times P\times N}], \\
&X_{chunks} = Stack(X_{split}, dim = 1) \in \mathbb{R}^{B\times C\times P\times N},
\end{aligned} \tag{2}
$$

where, the function of $Split$ is to divide the input X_{norm} into C data chunks along the first dimension ($dim = 1$), each with a length of P. The function of the $Stack$ function is to aggregate the input list of data along the first dimension ($dim = 1$). X_{norm} representing the input batch data after standard normalization.

Spatial weights are applied based on the temporal distance between samples, primarily to account for the temporal correlation between samples, as shown in the right image of Fig. 2. This ensures that adjacent time steps have a greater influence on the current time step. By doing so, it is possible to smooth the data while preserving the local structure and nonlinear patterns within the time series.

Since the spatial position information at the same location in all time blocks is identical, we only need to compute the spatial position information in one block and then broadcast ($Repeat$) it to all chunks. Within a neighborhood range of length W, the expression for spatial weights $\omega_S(X_{chunks})$ is as follows:

$$
\begin{aligned}
&\omega_{chunk}(X^t_{chunk}) = e^{(-\frac{d^2}{2\sigma_s^2})} \in \mathbb{R}^{1\times L\times 1}, \\
&L = len(arange(-\frac{W}{2}, \frac{W}{2} + 1)), t \in [0, P-1], \\
&\omega_S(X_{chunks}) = Repeat(Unsqueeze(\omega_{chunk}(X^t_{chunk}), dim = 1), [B, C, 1, N]) \\
&\quad \in \mathbb{R}^{B\times C\times L\times N},
\end{aligned} \tag{3}
$$

where $\omega_{chunk}(X^t_{chunk})$ represents the spatial weight information at the t-th time point within a time block ω_{chunk}. The $Unsqueeze$ function is used to add a dimension at dim. The $Repeat$ function copies the positional weight information from a subsequence block multiple times across the specified dimension. d represents the distance between the current point and its neighboring points in $[-\frac{W}{2}, \frac{W}{2} + 1]$. σ_s is the standard deviation factor of the spatial weights.

Intensity weights are applied based on the similarity between sample values (as shown in the right image of Fig. 2), giving greater weight to neighboring sample values that are similar to the current sample value. This allows important nonlinear features and edge patterns to be preserved. These nonlinear features often exhibit similarity between adjacent time steps, and through the application of intensity weights, the model can better capture these features.

The process of calculating intensity weights at each time point t in the input sequence can be expressed by the following mathematical formula:

1. Define the start (ls) and end positions (le) of the local window (lw).

$$
\begin{aligned}
&X^t_{pad} = Pad(X^t_{chunk}, pad\_size = ps, mode = reflect) \in \mathbb{R}^{B\times(P+2\cdot ps)\times N}, \\
&X_{pad} = Stack([X^1_{pad}, \dots, X^t_{pad}, \dots, X^C_{pad}], dim = 1) \in \mathbb{R}^{B\times C\times(P+2\cdot ps)\times N}, \\
&ls = Max(0, t - \frac{W}{2}), \quad le = Min(P, t + \frac{W}{2} + 1), \\
&lw = \Theta(X_{pad}, index = [ls + ps : le + ps), dim = 2) \in \mathbb{R}^{B\times C\times L'\times N}, \\
&L' = \begin{cases} (t - \frac{W}{2}) \text{ if } t \le \frac{W}{2} \\ (W + 1) \text{ if } t > \frac{W}{2}. \end{cases}
\end{aligned} \tag{4}
$$

Where, padding (Pad) is applied to each input chunk (X^t_{chunk}) along the temporal dimension using reflected data at the tensor's edges, with a padding size of ps ($ps = 4$). The padded data chunks are then stacked along the dim dimension to form X_{pad}. The purpose of data padding is to preserve contextual information by adding reflected data at the edges of data chunks, reducing boundary effects, and addressing potential boundary issues that may arise after data chunking. Max represents taking the maximum of two values, and Min represents taking the minimum of two values. The Θ function represents the operation of extracting data within specified indices ($index$) along the time dimension (dim).

2. Calculate the standard deviation(σ_r).

$$
lv = Var(lw, dim = 2) + \xi, \quad \sigma_r = \sigma' \sqrt{lv}. \tag{5}
$$

In this process, the variance (Var) is calculated along the time dimension (dim) of the local window, and to prevent division by zero, a small constant of ξ is added to the result. σ' represent a constant. Adjust the parameters of the intensity weights according to the local variance, allowing the filter to perform smoothing based on local characteristics.

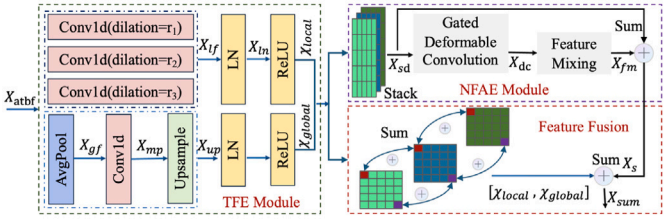


Fig. 3. Feature acquisition module.

3. Calculate the intensity weights $\omega_I(X^t_{pad})$ at the tth moment in the time series dimension.

$$\omega_I(X^t_{pad}) = e^{-\frac{(X^{t+ps}_{pad}-lw)^2}{2\sigma_r^2}} \in \mathbb{R}^{B\times C\times L'\times N}. \tag{6}$$

Feature weighting combines spatial weights $\omega_S(X_{chunks})$ and intensity weights $\omega_I(X^t_{pad})$, capturing the nonlinear features and edge information between adjacent time steps more effectively through dynamic weighting. This enables the model to recognize and utilize complex patterns in the data, and adapt to different local statistical characteristics of the time series, thereby enhancing the model's efficiency and accuracy in handling complex time series.

$$\begin{aligned}\omega(X^{t'}_{chunks}) &= \Theta(\omega_S(X_{chunks}), index = [0:L'), dim = 2)\\ &\cdot \omega_I(X^t_{pad}) \in \mathbb{R}^{B\times C\times L'\times N}, t' \in [0, L').\end{aligned} \tag{7}$$

To highlight the differences between weights and further mitigate the impact of extreme values on normalization results, we use the $Softmax$ function to map the inputs to a probability distribution. This approach helps the model better capture significant local abrupt changes and nonlinear features in the time series.

Notably, after standard normalization, while the different pattern information within the data is highlighted, the traditional temporal bilateral filter, due to its linear smoothing characteristics, remains insensitive to significant local variations [29]. This can result in local abrupt changes and complex nonlinear features being mistakenly identified as edges during processing, leading to their weakening or blurring [30]. In contrast, incorporating adaptive adjustment into the bilateral filtering process allows the smoothing degree to be adaptively adjusted based on local data characteristics [31]. This adaptive mechanism enables the filter to better emphasize nonlinear features, preserve abrupt local changes, and complex patterns, and reduce dependence on extreme values, leading to superior performance when processing data with significant nonlinear variations [32]. In this study, we use the Softmax function to dynamically adjust the weights of different feature patterns, helping the model better capture significant local changes and nonlinear features in the time series.

The formula for calculating the feature weighting matrix ($\tilde{\omega}^t_{sum}$) at time t within the local window (L') is as follows:

$$\begin{aligned}\tilde{\omega}(X^{t'}_{chunks}) &= Softmax(\omega(X^{t'}_{chunks})) = \frac{e^{\omega(X^{t'}_{chunks})}}{\sum_{k=1}^{L'} e^{\omega(X^{L'}_{chunks})}} \in \mathbb{R}^{B\times C\times L'\times N},\\ \tilde{\omega}^t_{sum} &= Sum(\tilde{\omega}(X^{t'}_{chunks}) \cdot lw, dim = 2) \in \mathbb{R}^{B\times C\times 1\times N},\end{aligned} \tag{8}$$

where, the Sum function operates by summing along the dimension dim. Then, aggregate (Cat) the weighted feature information of each time point along the time dimension (dim) to form the final weighted feature information (X_{agg}) of the time series. Finally, we use the $Reshape$ function to change the shape of the tensor (X_{agg}) so that it matches the shape of the input matrix, serving as the final output (X_{atbf}) of the time series processing stage. This operation involves merging the first and second dimensions ($C \times P = M$).

$$\begin{aligned}X_{agg} &= Cat([\tilde{\omega}^0_{sum}, \ldots, \tilde{\omega}^t_{sum}, \ldots, \tilde{\omega}^P_{sum}], dim = 2) \in \mathbb{R}^{B\times C\times P\times N},\\ X_{atbf} &= Reshape(X_{agg}) \in \mathbb{R}^{B\times M\times N}.\end{aligned} \tag{9}$$

In summary, through the above operations, the nonlinear features and patterns in the time series are preserved and highlighted, laying a foundation for the effective extraction of subsequent nonlinear features.

Additionally, in the right image of Fig. 2 illustrating the ATBF module, the two figures below show the feature weight distributions (the color intensity represents the magnitude of the weights). Fig. 2(a) represents the original weight distribution extracted by bilateral filtering, while Fig. 2(b) shows the weight distribution extracted by bilateral filtering with Softmax added. From the figures, we can see that after adding the Softmax function, the differences between the weight values are significantly amplified, which better highlights the complex patterns and nonlinear relationships in the time series data. For example, in Fig. 2(b), the weight values of variables 1, 2, 3, and 6 are significantly higher compared to the weights without Softmax, making these abrupt changes and nonlinear features more prominent and clearer in the feature representation map. For more information about the spatial weights and intensity weights, please refer to Ref. [29].

3.3. Feature acquisition

To comprehensively and effectively extract features from time series, we designed the feature acquisition module, which consists of two components: the TFE module and the NFAE module, as shown in Fig. 3. The TFE module captures temporal dependency information of local and global patterns within the time series. In contrast, the NFAE module effectively extracts nonlinear information between different time observations and correlations among different variables in the time series.

TFE Module. Inspired by the concept of SPP [33], the TFE module employs one-dimensional (1D) convolutions with varying dilation factors to capture features and patterns at different resolutions within the input multivariate time series data. This approach equips the model with the capability to discern local feature variations: with small dilation factors, the model convolves within a small receptive field, facilitating more precise identification of local patterns and structures, thus aiding in enhancing the network's comprehension of complex patterns. Conversely, with large dilation factors, the model employs larger convolutional kernels to capture features within a broader receptive field, thereby facilitating the identification of local contextual relationships within the input sequence.

Additionally, by aggregating feature and pattern information extracted at different resolutions at the same location, it ensures positional invariance of the data during the processing. To encapsulate global trends and patterns, our model incorporates adaptive average pooling operations. This technique is instrumental in distilling the overarching trends and patterns from the time series data, thus enriching the model's comprehensive grasp of the temporal dynamics presented.

When data flows through this module, it is mapped into two tasks: (1) Local feature extraction. By selecting different dilation factors for multi-resolution dilated convolution, the receptive field of the convolutional kernel can be expanded without losing resolution. This module utilizes convolutions of different sizes to learn patterns and local contextual information at different resolutions. (2) Global feature extraction. Introducing global adaptive average pooling effectively captures global information from the entire feature map, aiding the model in better understanding the global patterns and structural information of the entire input sequence. In summary, by overlaying features of different resolutions at the same location, this not only enhances the model's translational invariance but also improves the model's generalization capability through the integration of complementary information. Furthermore, through the designed weight sharing mechanism, the same set of weights is reused across inputs of different resolutions. This not only reduces the number of parameters in the model but also helps to enhance the model's efficiency.

The calculation formula for specific operational steps is as follows:

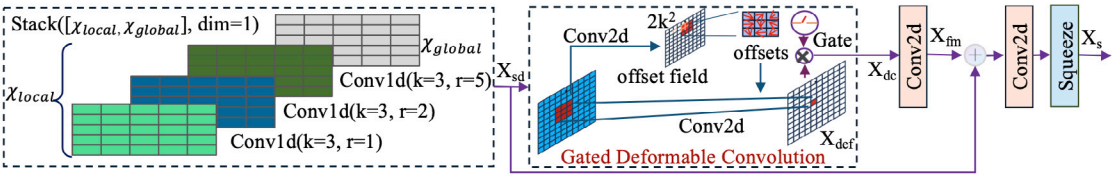


Fig. 4. NFAE module architecture.

(1) Local feature extraction. Given a time series X_{atbf} with a shape of $\mathbb{R}^{B\times M\times N}$, a convolution *kernel* of length k, a *dilation* factor r, and a *padding* size $\mathfrak{p}$ (by default, $\mathfrak{p} = (\frac{k-1}{2}) \cdot r, r \in R$). The formula to capture the local feature of a time series can be described as follows:

$$
\begin{aligned}
&X_{lf} = Conv1d(X_{atbf}, kernel = k, dilation = r, padding = \mathfrak{p}) \in \mathbb{R}^{B\times M\times N},\\
&X_{ln} = LN(X_{lf}) \in \mathbb{R}^{B\times M\times N},\\
&\chi_{local} = ReLU(X_{ln}) \in \mathbb{R}^{B\times M\times N}.
\end{aligned}
\tag{10}
$$

The function of the $Conv1d$ is to extract local features from the input data, capturing information at different granularities in the time series based on various dilation factors r, and to integrate (Sum) them into X_{lf} at the same location. Linear normalization (LN) is used to linearly scale the output of $Conv1d$ to a specific range, which helps accelerate training speed and enhance model stability. The $ReLU$ activation function introduces nonlinearity into the model, enabling it to capture more complex data structures. To balance performance and efficiency, we set the dilated rate $r = \{1, 2, 5\}$ as suggested in [34].

(2) Global feature extraction. Given a time series X_{atbf} with a shape of $\mathbb{R}^{B\times M\times N}$, The formula to capture the global feature of a time series can be described as follows:

$$
\begin{aligned}
&X_{gf} = AdaptiveAvgpool1d(X_{atbf}) \in \mathbb{R}^{B\times 1\times N},\\
&X_{mp} = Conv1d(X_{gf}, kernel = 1) \in \mathbb{R}^{B\times 1\times N},\\
&X_{up} = Upsample(X_{mp}, mode = linear) \in \mathbb{R}^{B\times M\times N},\\
&\chi_{global} = ReLu(LN(X_{up})) \in \mathbb{R}^{B\times M\times N}.
\end{aligned}
\tag{11}
$$

Where, the $AdaptiveAvgPool1d$ function performs an adaptive average pooling operation on the input. It compresses the input feature tensor into a single average value for each feature channel, thereby extracting global features (X_{gf}) from the input sequence. We employ a one-dimensional convolution operation with a kernel size of 1, which acts as a linear transformation layer, mapping the features after adaptive average pooling into a new feature space (X_{mp}). The $Upsample$ function is used to restore the output size to the original input shape using linear interpolation. Finally, after normalization and the $ReLU$ activation function, the global features of the time series are captured into χ_{global}.

NFAE Module. Fig. 1 demonstrates that dependencies between different time observations in time series data can manifest as nonlinear and non-uniform distributions. To learn the nonlinear information between different observations and position offset information in feature graphs with different resolutions, we designed a gated deformable convolution (GDC) module. This module incorporates a gating mechanism into the deformable convolution, thereby enhancing the model's flexibility and capacity to capture intricate nonlinear features. By allowing adjustments to the positions of convolutional kernels, deformable convolution enhances the model's ability to capture local nonlinear features, better handling irregular features and patterns. Consequently, this approach enhances the model's sensitivity to local nonlinear patterns, as shown in Fig. 4. This capability is particularly important for time series data with significant nonlinear and dynamic feature changes.

Specifically, we construct a four-dimensional tensor X_{sd} by stacking multi-resolution feature maps along with the output maps from adaptive average pooling. This tensor serves as the input to the NFAE module. We then utilize the GDC module to extract nonlinear information and position offset information from the input tensor across different resolution feature maps. This module employs a twisted sampling network approach to focus on regions with richer information. This methodology enables the model to more accurately capture and understand nonlinear features and relationships among variables that vary between short-term and long-term in time series, while also removing redundant feature information.

The mathematical expression for extracting nonlinear features using the NFAE module is as follows:

$$
\begin{aligned}
&X_{sd} = Stack([\chi_{local}, \chi_{global}], dim = 1) \in \mathbb{R}^{B\times D\times M\times N}, D = len(r) + 1,\\
&X_{offset} = Conv2d(X_{sd}, kernel = k) \in \mathbb{R}^{B\times(2*k^2)\times M\times N},\\
&X_{offsets} = Sum(Reshape(X_{offset}, [B, M, N, k^2, 2]), dim = 3)\\
&\in \mathbb{R}^{B\times M\times N\times 2},\\
&X_{sampling} = Grid\_sample(X_{sd}, X_{offsets}) \in \mathbb{R}^{B\times D\times M\times N},\\
&X_{def} = Conv2d(X_{sampling}, kernel = k) \in \mathbb{R}^{B\times D\times M\times N},\\
&Gate = Sigmoid(Conv2d(X_{sd}), kernel = k) \in \mathbb{R}^{B\times D\times M\times N},\\
&X_{dc} = Gate * X_{def} \in \mathbb{R}^{B\times D\times M\times N}.
\end{aligned}
\tag{12}
$$

First, χ_{local} and χ_{global} are stacked along the first dimension (dim), with the number of stacks equal to the number of dilation factors ($len(r)$) used plus the number of times global information is extracted (1). Calculate the offsets (X_{offset}) for each convolutional kernel position, where the output channels of this convolutional layer are $2 * k^2$ because it needs to compute the offsets in both x and y directions for each kernel position. The shape is then adjusted using the $Reshape$ and Sum functions to obtain the final offsets ($X_{offsets}$) for the convolutional kernel. Next, the $Grid\_sample$ function performs bilinear interpolation sampling on the input tensor X_{sd} based on the generated sampling grid. After passing through a standard 2D convolution, the final feature map X_{def} is produced. The gating mechanism ($Gate$) uses a 2D convolution with an output size of D and a $Sigmoid$ activation function to independently adjust the feature maps at each granularity. Finally, the convolution result (X_{def}) is element-wise multiplied by the gating values ($Gate$) to implement the adaptive gated convolution mechanism.

The above steps aim to enhance the adaptability and effectiveness of convolution operations, especially for complex multivariate data. For more detailed theoretical explanations regarding deformable convolutions, please refer to Ref. [35].

Additionally, we use a 2D convolution with D output channels to mix features, aiming to shuffle fine-grained, lower-level time series information with coarse-grained, higher-level time series information. Then, another 2D convolution with 1 output channel is used to integrate features across different resolutions. Finally, the function $Squeeze$ is applied to remove the dim dimension, resulting in the final output (X_s) of the NFAE module. The process is described as follows:

$$
\begin{aligned}
&X_{fm} = Conv2d(X_{dc}, kernel = k) \in \mathbb{R}^{B\times D\times M\times N},\\
&X_s = Squeeze(Conv2d(X_{sd} + X_{fm}, kernel = k), dim = 1) \in \mathbb{R}^{B\times M\times N}.
\end{aligned}
\tag{13}
$$

After these steps, the correlations between different dimensions and the nonlinear relationships between different observations in the time series are captured in X_s.

Feature Fusion. Through the aforementioned operations, we capture the local features and pattern information of the time series at

different resolutions in χ_{local}, the global information of the time series in χ_{global}, and the information extracted by the NFAE module in X_s. Then, at the corresponding positions, features from different patterns at the same time point are fused (Sum) to obtain the final feature representation X_{sum}. The formula for feature fusion is as follows:

$$X_{sum} = Sum(\chi_{local}, \chi_{global}, X_s) \in \mathbb{R}^{B\times M\times N}. \tag{14}$$

3.4. Model prediction

In the downstream prediction tasks of DCNet, we employ a simple FFN architecture. To address potential underfitting issues in regression tasks and improve the computational efficiency of the model, we were inspired by [36] to use the Moore–Penrose (MP) pseudo-inverse ($Pinv$) matrix to compute the parameters of the model network. The mathematical expression for the downstream regression task of the model is as follows:

$$\begin{aligned} X_f &= Linear(Reshape(X_{sum})) \in \mathbb{R}^{B\times H}, \\ H(X_f) &= Sigmoid(X_f) \in \mathbb{R}^{B\times H}. \end{aligned} \tag{15}$$

First, the last two dimensions of the input data (X_{sum}) are merged using the $Reshape$ function. Then, a $Linear$ function with an output dimension of H is applied to the reshaped data. Finally, the activation function ($Sigmoid$) is applied to obtain the hidden layer output matrix. We use the least squares method to calculate the output layer weight matrix β :

$$\min_{\beta} MP(\beta) = \|[H^T(X_f)@H(X_f)\beta - H^T(X_f)@\mathcal{Y}]\|^2 + \lambda \cdot I\|\beta\|_1, \tag{16}$$

$$\begin{aligned} \beta &= Pinv(H^T(X_f)@H(X_f) + \lambda \cdot I)@H^T(X_f)@\mathcal{Y} \\ &= [H^T(X_f)@H(X_f) + \lambda \cdot I]^{\dagger}@H^T(X_f)@\mathcal{Y} \in \mathbb{R}^{H\times(L_y\times N)}, \end{aligned} \tag{17}$$

where, $H^T(X_f)$ represents the transpose of the matrix $H(X_f)$, @ represents matrix operations, $\cdot$ represents element-wise operations, $\lambda > 0$ is the regularization parameter, I represents the identity matrix, $\mathcal{Y}$ is the target matrix for the training data. $\|[H^T(X_f)@H(X_f)\beta - H^T(X_f)@\mathcal{Y}]\|$ is the least squares convex term, and $\|\cdot\|_1$ is the L1 norm non-smooth regularization term.

Finally, given an input data $\hat{X} \in \mathbb{R}^{B\times M\times N}$, the prediction result $\hat{Y}$ is:

$$\hat{Y} = Reshape(H(\hat{X}) \cdot \beta) \in \mathbb{R}^{B\times L_y\times N}. \tag{18}$$

To provide a clearer description of the specific steps and processes involved in model prediction, we present the pseudocode for this procedure in Algorithm 1.

Algorithm 1: Model prediction.

Input:

χ_{local}, χ_{global}, X_s, $\mathcal{Y}$, a batch size of test data $\hat{X}$, regularization parameter $\lambda = 0.1$.

1: Compute the hidden layer output matrix $H(\cdot)$ with Eq. (15).
2: Compute the pseudo-inverse matrix MP with Eq. (16).
3: Compute the output layer weight matrix β with Eq. (17).
4: Compute the feature representations $\hat{\chi}_{local}$, $\hat{\chi}_{global}$ and $\hat{X}_s$ for the input data $\hat{X}$ using Eq. (1) - Eq.(13).
5: Calculate the predicted value $\hat{Y}$ for $\hat{X}$ using Eq. (18).

Output: $\hat{Y}$.

4. Experiments

In this section, we will delve into the effectiveness and efficiency of the DCNet network. Through rigorous experiments on long-term and short-term forecasting using twelve real-world datasets and comparing against fifteen SOTA models, our goal is to address the following research questions: RQ1 (Effectiveness): Can DCNet outshine the present SOTA baseline models when applied to real datasets (Sections 4.4–4.9)? RQ2 (Efficiency): Does DCNet exhibit superior performance in terms of resource utilization compared to the current baseline models (Section 4.10)? RQ3 (Ablation): To what extent do the distinct components of DCNet influence its overall performance in time series forecasting tasks (Section 4.11)?

Table 1
The key features of the twelve time series datasets.

Tasks	Datasets	Dim	Timesteps	Granularity	Information
Long-term forecasting	[a]ETTh1/h2	7	17 420	1 h	Electricity
	[a]ETTm1/m2	7	69 680	15 min	Electricity
	[b]Electricity	321	26 304	1 h	Electricity
	[c]Traffic	862	52 560	1 h	Traffic
	[d]Exchange	8	7558	1 day	Financial
	[e]ILI	7	966	1 week	Medical
Short-term forecasting	PEMS03	358	26 208	5 min	Traffic
	PEMS04	307	16 992	5 min	Traffic
	PEMS07	883	28 224	5 min	Traffic
	PEMS08	170	17 856	5 min	Traffic

[a] https://github.com/zhouhaoyi/ETDataset.
[b] https://archieve.ics.uci.deu/ml/datasets/ElectricityLoadDiagrams 20112014.
[c] http://pems.dot.ca.gov.
[d] https://github.com/laiguo-kun/multivariate-time-series-data.
[e] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html.

4.1. Datasets

Table 1 summarizes the key features of these datasets.

For a more detailed introduction to the datasets used in this work, please refer to Appendix C.

4.2. Baselines

We selected 15 representative SOTA models that have demonstrated outstanding performance in the field of TSF as benchmarks: Patch-based TSMixer [11], PatchTST [7], and PDF [37]; MLP-based TimeMixer [22], FITS [38], and DLinear [10]; Transformer-based architectures iTransformer [8], Crossformer [9], ETSformer [19], FEDformer [20], and Non-Stationary Transformer [39]; and CNN-based models ConvTimeNet [13], TSLANet [17], MICN [14], and TimesNet [40].

4.3. Implementation details

Our approach is trained with an initial learning rate of 1×10^{-3}, the convolution dilated rate r for multi-resolution dilated convolution was set to $\{1, 2, 5\}$. In long-term forecasting, the input sequence length for the ILI dataset is $L = 36$, with prediction sequence lengths of $\{24, 36, 48, 60\}$. For other datasets, the input sequence length is $L = 96$, with prediction sequence lengths of $\{96, 192, 336, 720\}$. In short-term forecasting, the input sequence length for the PEMS datasets is $L = 96$, with a prediction sequence length of 12. To verify the robustness of our results, we conducted long-term forecasting by training the DCNet model with three different random seeds. For each seed, we calculated the mean squared error (MSE) and mean absolute error (MAE). The average results are presented in Tables 2 and 3. For short-term forecasting, we employed MAE, mean absolute percentage error (MAPE), and root mean squared error (RMSE) as evaluation metrics. Additionally, we assessed the model's efficiency based on floating point operations (FLOPs), number of parameters, training time, and inference time. Experiments are implemented using PyTorch on a single NVIDIA GeForce RTX 3090 24 GB GPU, Intel(R) Core(TM) i7-10700k CPU and 32 GB RAM.

Table 2
Long-term multivariate forecasting results with different prediction lengths on the ETT Dataset. The best results are in bold numbers. Avg represents the average value across the four prediction lengths.

Categories	Models	Metric	ETTm1					ETTm2					ETTh1					ETTh2					Count
			96	192	336	720	Avg	96	192	336	720	Avg	96	192	336	720	Avg	96	192	336	720	Avg	
Ours	DCNet	MSE	**0.142**	**0.162**	**0.174**	**0.184**	**0.166**	**0.111**	**0.129**	**0.14**	**0.152**	**0.133**	**0.203**	**0.228**	**0.238**	**0.245**	**0.229**	**0.147**	**0.168**	**0.192**	**0.223**	**0.183**	32
		MAE	**0.263**	**0.282**	**0.295**	**0.304**	**0.286**	**0.219**	**0.236**	**0.247**	**0.255**	**0.239**	**0.326**	**0.347**	**0.355**	**0.363**	**0.348**	**0.259**	**0.279**	**0.295**	**0.310**	**0.286**	
CNNs	ConvTimeNet (2024)	MSE	0.264	0.316	0.315	0.382	0.319	0.183	0.248	0.311	0.401	0.286	0.338	0.377	0.400	0.465	0.395	0.385	0.438	0.451	0.470	0.436	0
		MAE	0.330	0.368	0.378	0.425	0.375	0.265	0.305	0.345	0.396	0.328	0.368	0.385	0.404	0.439	0.399	0.396	0.425	0.437	0.461	0.430	
	MICN (2023)	MSE	0.316	0.363	0.408	0.481	0.392	0.179	0.307	0.325	0.502	0.328	0.398	0.430	0.460	0.491	0.445	0.332	0.422	0.447	0.442	0.411	0
		MAE	0.362	0.390	0.426	0.476	0.414	0.275	0.376	0.388	0.490	0.382	0.427	0.453	0.460	0.509	0.462	0.377	0.441	0.474	0.467	0.440	
	TimesNet (2023)	MSE	0.338	0.374	0.410	0.478	0.400	0.187	0.249	0.321	0.408	0.291	0.384	0.436	0.491	0.521	0.458	0.340	0.402	0.452	0.462	0.414	0
		MAE	0.375	0.387	0.411	0.450	0.406	0.267	0.309	0.351	0.403	0.333	0.402	0.429	0.469	0.500	0.450	0.374	0.414	0.452	0.468	0.427	
MLPs	TimeMixer (2024)	MSE	0.320	0.361	0.390	0.454	0.381	0.175	0.237	0.298	0.391	0.275	0.375	0.429	0.484	0.498	0.447	0.289	0.372	0.386	0.412	0.364	0
		MAE	0.357	0.381	0.404	0.441	0.395	0.258	0.299	0.340	0.396	0.323	0.400	0.421	0.458	0.482	0.440	0.341	0.392	0.414	0.434	0.395	
	FITS (2024)	MSE	0.351	0.392	0.424	0.485	0.413	0.181	0.246	0.306	0.407	0.285	0.381	0.434	0.474	0.464	0.438	0.290	0.375	0.414	0.419	0.375	0
		MAE	0.370	0.393	0.413	0.448	0.406	0.264	0.304	0.341	0.397	0.327	0.391	0.422	0.446	0.463	0.431	0.339	0.388	0.425	0.437	0.397	
	DLinear (2023)	MSE	0.345	0.380	0.413	0.474	0.403	0.193	0.284	0.369	0.554	0.350	0.386	0.437	0.481	0.519	0.456	0.333	0.477	0.594	0.831	0.559	0
		MAE	0.372	0.389	0.413	0.453	0.407	0.292	0.362	0.427	0.522	0.401	0.400	0.432	0.459	0.516	0.452	0.387	0.476	0.541	0.657	0.515	
Transformers	iTransformer (2024)	MSE	0.334	0.377	0.426	0.491	0.407	0.180	0.250	0.311	0.412	0.288	0.386	0.441	0.487	0.503	0.454	0.297	0.380	0.428	0.427	0.383	0
		MAE	0.368	0.391	0.420	0.459	0.410	0.264	0.309	0.348	0.407	0.332	0.405	0.436	0.458	0.491	0.448	0.349	0.400	0.432	0.445	0.407	
	Crossformer (2023)	MSE	0.428	0.445	0.533	0.728	0.534	0.197	0.326	0.372	0.410	0.326	0.429	0.494	0.706	0.750	0.595	0.632	0.876	0.924	1.390	0.956	0
		MAE	0.444	0.468	0.519	0.655	0.522	0.321	0.375	0.421	0.448	0.391	0.440	0.482	0.625	0.689	0.559	0.547	0.663	0.702	0.863	0.694	
	ETSformer (2022)	MSE	0.375	0.408	0.435	0.499	0.429	0.189	0.253	0.314	0.414	0.293	0.494	0.538	0.574	0.562	0.542	0.340	0.430	0.485	0.500	0.439	0
		MAE	0.398	0.410	0.428	0.462	0.425	0.280	0.319	0.357	0.413	0.342	0.479	0.504	0.521	0.535	0.510	0.391	0.439	0.479	0.497	0.452	
	FEDformer (2022)	MSE	0.379	0.426	0.445	0.543	0.448	0.203	0.269	0.325	0.421	0.305	0.376	0.420	0.459	0.506	0.440	0.358	0.429	0.496	0.463	0.437	0
		MAE	0.419	0.441	0.459	0.490	0.452	0.287	0.328	0.366	0.415	0.349	0.419	0.448	0.465	0.507	0.460	0.397	0.439	0.487	0.474	0.449	
	Non-Sta (2022)	MSE	0.386	0.459	0.495	0.585	0.481	0.192	0.280	0.334	0.417	0.306	0.513	0.534	0.588	0.643	0.570	0.476	0.512	0.552	0.562	0.526	0
		MAE	0.398	0.444	0.464	0.516	0.456	0.274	0.339	0.361	0.413	0.347	0.491	0.504	0.535	0.616	0.537	0.458	0.493	0.551	0.560	0.516	
Patchs	PDF (2024)	MSE	0.320	0.375	0.411	0.464	0.393	0.181	0.242	0.302	0.403	0.282	0.369	0.414	0.451	0.483	0.429	0.292	0.377	0.419	0.424	0.378	0
		MAE	0.351	0.376	0.399	0.431	0.389	0.264	0.302	0.341	0.396	0.326	0.387	0.419	0.438	0.475	0.430	0.341	0.392	0.429	0.441	0.401	
	TSMixer (2023)	MSE	0.319	0.369	0.402	0.464	0.389	0.175	0.240	0.301	0.404	0.280	0.381	0.433	0.472	0.485	0.443	0.289	0.375	0.425	0.435	0.381	0
		MAE	0.358	0.384	0.406	0.442	0.398	0.258	0.230	0.338	0.398	0.306	0.391	0.420	0.441	0.471	0.431	0.338	0.391	0.435	0.449	0.403	
	PatchTST (2023)	MSE	0.324	0.362	0.390	0.461	0.384	0.177	0.248	0.304	0.403	0.283	0.394	0.446	0.485	0.495	0.455	0.302	0.388	0.426	0.431	0.387	0
		MAE	0.361	0.383	0.402	0.438	0.396	0.260	0.306	0.342	0.397	0.326	0.408	0.438	0.455	0.474	0.444	0.348	0.400	0.433	0.446	0.407	

Note: Count represents the total number of optimal metric values.

4.4. Main results and analysis

Long-term forecasting. In the field of long-term forecasting, the DCNet framework demonstrates excellent performance on almost all datasets. Analysis of the results in Tables 2 and 3 leads to the following conclusions:

- The DCNet model significantly improves inference performance on nearly all datasets. As the forecasting horizon increases, the prediction error of DCNet shows a gradually stable upward trend, indicating its ability to maintain high long-term robustness in practical applications.
- Compared to the current SOTA forecasting model ConvTimeNet, DCNet exhibits notable improvements across various datasets. Specifically, DCNet gives 47.9% (0.319 → 0.166) MSE reduction in ETTm1, 53.5% (0.286 → 0.133) in ETTm2, 42.0% (0.395 → 0.229) in ETTh1, 58% (0.436 → 0.183) in ETTh2, 32.1% (0.202 → 0.137) in Electricity, 39.8% (0.402 → 0.242) in Traffic, 65.7% (0.347 → 0.119) in exchange and 82.9% (1.866 → 0.319) in ILI. Overall, DCNet yields a 64.1% averaged MSE reduction among above settings. Notably, DCNet has achieved significant success on the Exchange and ILI datasets, which contain a large number of nonlinear features. Compared to the ConvTimeNet model, which uses depthwise separable convolutions to extract time series features, DCNet demonstrates greater advantages in long-term time series forecasting. This enhancement is attributed to design of the ATBF and NFAE modules, which effectively extract and utilize nonlinear features within the time series.
- DCNet outperforms both the iTransformer and TSMixer models significantly. Compared to iTransformer, DCNet reduces the MSE across all datasets by 66.8% (0.576 → 0.191). Similarly, compared to TSMixer, DCNet reduces the MSE across all datasets by 49.6% (0.379 → 0.191). iTransformer and TSMixer focus on extracting features between different variables in time series. Although they outperform models like PatchTST, which only focus on temporal information, they may suffer from information redundancy due to strong correlations between variables, leading to reduced predictive accuracy. DCNet uses GDC to extract correlations between variables and employs a flexible warp network to focus on important features, remove redundancies, and enhance the model's predictive performance.
- The DCNet model achieves significantly better results than the SOTA MLP-based models. DCNet has an average MSE decrease of 67.5% (0.587 → 0.191) on all datasets in TimeMixer. This enhanced performance can be attributed to the comprehensive feature extraction capabilities of DCNet. Unlike MLP-based models, which struggle to extract sufficient feature information from complex time series for accurate future trend prediction, DCNet excels in this regard. It effectively captures and utilizes both local contextual information and global correlation information within the time series.
- When processing the high-dimensional Traffic dataset (862 dimensions), although DCNet outperforms ConvTimeNet in terms of MSE, its performance on the MAE metric is not as good as ConvTimeNet. One possible reason is that DCNet utilizes an adaptive bilateral filter during data preprocessing, which may lead to an imbalanced weight distribution among different variables and features. In high-dimensional data, certain variables and features may have a greater impact on the prediction of the target value, while others may have a smaller impact. The adaptive bilateral filter might have handled these important variables and features more precisely, thereby reducing MSE. However, for those variables and features that have a smaller but still important impact, the model may have overlooked their errors, leading to a higher MAE.

In summary, the superior performance of DCNet in handling long time series can be attributed to its comprehensive consideration of local and global information, nonlinear dynamics, and interdependencies among different variables often overlooked by many current TSF models. This holistic approach enables DCNet to excel in a wide range of real-world time series forecasting tasks.

Short-term forecasting. Short-term forecasting results for the time series are presented in Table 4. Analysis of this table yields the following conclusions:

Table 3
Long-term multivariate forecasting results with different prediction lengths. The best results are in bold numbers. Avg represents the average value across the four prediction lengths.

Categories	Models	Metric	Electricity					Traffic					Exchange					ILI					Count
			96	192	336	720	Avg	96	192	336	720	Avg	96	192	336	720	Avg	24	36	48	60	Avg	
Ours	DCNet	MSE	**0.131**	**0.135**	**0.138**	**0.142**	**0.137**	**0.228**	**0.239**	**0.243**	**0.256**	**0.242**	0.097	**0.103**	**0.117**	**0.158**	**0.119**	**0.290**	**0.305**	**0.311**	**0.368**	**0.319**	26
		MAE	**0.233**	**0.239**	**0.245**	**0.250**	**0.242**	0.286	0.292	0.300	0.307	0.296	0.209	**0.217**	**0.232**	**0.279**	**0.234**	**0.311**	**0.336**	**0.342**	**0.390**	**0.345**	
CNNs	ConvTimeNet (2024)	MSE	0.179	0.185	0.201	0.242	0.202	0.376	0.392	0.405	0.436	0.402	0.083	0.170	0.316	0.817	0.347	1.701	1.941	1.847	1.976	1.866	5
		MAE	0.263	0.269	0.285	0.318	0.284	**0.265**	**0.271**	**0.277**	**0.294**	0.277	**0.196**	0.291	0.404	0.679	0.393	0.823	0.888	0.893	0.952	0.889	
	MICN (2023)	MSE	0.164	0.177	0.193	0.212	0.187	0.519	0.537	0.534	0.577	0.542	0.102	0.172	0.272	0.714	0.315	2.684	2.667	2.558	2.747	2.664	0
		MAE	0.269	0.285	0.304	0.321	0.295	0.309	0.315	0.313	0.325	0.316	0.235	0.316	0.407	0.658	0.404	1.112	1.068	1.052	1.110	1.086	
	TimesNet (2023)	MSE	0.168	0.184	0.198	0.220	0.193	0.593	0.617	0.629	0.640	0.620	0.107	0.226	0.367	0.964	0.416	2.317	1.972	2.238	2.027	2.139	0
		MAE	0.272	0.289	0.300	0.320	0.295	0.321	0.336	0.336	0.350	0.336	0.234	0.344	0.448	0.746	0.443	0.934	0.920	0.940	0.928	0.931	
MLPs	TimeMixer (2024)	MSE	0.153	0.166	0.185	0.225	0.182	0.462	0.473	0.498	0.506	0.484	0.100	0.212	0.379	0.904	0.399	2.122	2.289	2.165	2.085	2.165	0
		MAE	0.247	0.256	0.277	0.310	0.272	0.285	0.296	0.296	0.313	0.297	0.222	0.326	0.442	0.715	0.426	0.874	0.931	0.908	0.909	0.906	
	FITS (2024)	MSE	0.293	0.268	0.355	0.416	0.333	0.898	0.763	0.894	1.019	0.894	0.089	0.182	0.327	0.852	0.363	3.340	4.016	4.541	4.406	4.076	0
		MAE	0.401	0.378	0.452	0.498	0.432	0.572	0.522	0.608	0.646	0.587	0.210	0.303	0.414	0.696	0.406	1.299	1.453	1.554	1.517	1.456	
	DLinear (2023)	MSE	0.197	0.196	0.209	0.245	0.212	0.650	0.598	0.605	0.645	0.625	0.088	0.176	0.313	0.839	0.354	2.398	2.646	2.614	2.804	2.616	0
		MAE	0.282	0.285	0.301	0.333	0.300	0.396	0.370	0.373	0.394	0.383	0.218	0.315	0.427	0.695	0.414	1.040	1.088	1.086	1.146	1.090	
Transformers	iTransformer (2024)	MSE	0.148	0.162	0.178	0.225	0.178	0.395	0.417	0.433	0.467	0.428	0.086	0.177	0.331	0.847	0.360	2.014	2.115	2.188	2.114	2.108	0
		MAE	0.240	0.253	0.269	0.317	0.270	0.268	0.276	0.283	0.302	0.282	0.206	0.299	0.417	0.691	0.403	0.899	0.943	0.972	0.968	0.946	
	Crossformer (2023)	MSE	0.254	0.261	0.273	0.303	0.273	0.558	0.572	0.587	0.652	0.592	0.329	0.544	1.017	1.239	0.782	3.041	3.406	3.459	3.640	3.387	0
		MAE	0.347	0.353	0.364	0.388	0.363	0.320	0.331	0.342	0.359	0.338	0.440	0.586	0.786	0.912	0.681	1.186	1.232	1.221	1.305	1.236	
	ETSformer (2022)	MSE	0.187	0.199	0.212	0.233	0.208	0.607	0.621	0.622	0.632	0.621	0.085	0.182	0.348	1.025	0.410	2.527	2.615	2.359	2.487	2.497	0
		MAE	0.304	0.315	0.329	0.345	0.323	0.392	0.399	0.396	0.396	0.396	0.204	0.303	0.428	0.774	0.427	1.020	1.007	0.972	1.016	1.004	
	FEDformer (2022)	MSE	0.193	0.201	0.214	0.246	0.214	0.587	0.604	0.621	0.626	0.610	0.148	0.271	0.460	1.195	0.519	3.228	2.679	2.622	2.857	2.847	0
		MAE	0.308	0.315	0.329	0.355	0.327	0.366	0.373	0.383	0.382	0.376	0.237	0.335	0.476	0.769	0.454	0.945	0.848	0.900	0.963	0.914	
	Non-Sta (2022)	MSE	0.169	0.182	0.200	0.222	0.193	0.612	0.613	0.618	0.653	0.624	0.111	0.219	0.421	1.092	0.461	2.294	1.825	2.010	2.178	2.077	0
		MAE	0.273	0.286	0.304	0.321	0.296	0.338	0.340	0.328	0.355	0.340	0.237	0.335	0.476	0.769	0.454	0.945	0.848	0.900	0.963	0.914	
Patchs	PDF (2024)	MSE	0.165	0.181	0.197	0.238	0.195	0.463	0.469	0.484	–	0.472	0.083	0.175	0.327	0.849	0.359	2.585	2.623	2.465	2.386	2.515	0
		MAE	0.248	0.266	0.282	0.315	0.278	0.297	0.298	0.305	–	0.300	0.201	0.296	0.413	0.694	0.401	1.065	1.085	1.035	1.029	1.054	
	TSMixer (2023)	MSE	0.146	0.163	0.180	0.216	0.176	0.483	0.490	0.506	0.538	0.504	**0.082**	0.176	0.337	0.909	0.376	0.432	0.446	0.477	0.578	0.483	1
		MAE	0.244	0.259	0.279	0.307	0.272	0.321	0.321	0.330	0.341	0.328	0.199	0.297	0.418	0.725	0.410	0.487	0.479	0.510	0.564	0.510	
	PatchTST (2023)	MSE	0.195	0.199	0.215	0.256	0.216	0.544	0.540	0.551	0.586	0.555	0.082	0.187	0.345	0.887	0.375	1.724	1.536	1.821	1.923	1.751	0
		MAE	0.285	0.289	0.305	0.337	0.304	0.359	0.354	0.358	0.375	0.362	0.201	0.307	0.427	0.708	0.411	0.843	0.752	0.832	0.842	0.817	

Note: Count represents the total number of optimal metric values.
The '–' indicates failure for the out-of-memory.

- The predictions generated by DCNet surpass those of other baseline models, with the Transformer-based Crossformer model achieving the second-best results. The success of DCNet can be attributed to its comprehensive feature extraction approach, which effectively captures and utilizes local, global, and nonlinear features, significantly enhancing the model's predictive performance. Crossformer demonstrates its effectiveness in capturing complex patterns within real-world time series by explicitly extracting the interdependencies between different variables in the time series.
- Transformer-based models (e.g., iTransformer) exhibit limited capability when dealing with time series data that contain a large number of nonlinear features (e.g., PEMS03, PEMS04). This limitation may stem from their reliance on self-attention mechanisms to capture global information from the input sequence while overlooking the extraction of local dynamic features.
- The MICN model, which is based on multi-scale dilated convolution, shows the poorest performance across all datasets. This outcome suggests that traditional convolutions with fixed-size receptive fields are inadequate for real-world time series datasets containing numerous nonlinear features. The fixed nature of these receptive fields restricts their flexibility in handling nonlinear features at varying scales, leading to a decline in predictive performance.

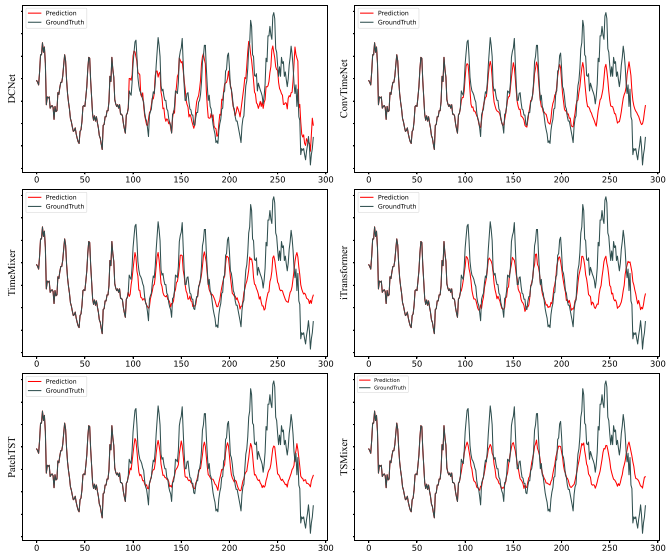


Fig. 5. Visualize the ETTh1 dataset, where the first 96 data points represent the input length.

4.5. Visualization of forecasting

To evaluate the predictive performance of DCNet compared to SOTA TSF models (such as ConvTimeNet, TimeMixer, iTransformer, TSMixer and PatchTST) in real-world scenarios, we qualitatively compared the prediction results of the last dimension on the test set of the ETTh1 dataset, as shown in Fig. 5. The input length was set to 96, and the prediction length was set to 192 to assess the fitting between the predicted sequences and the actual sequences.

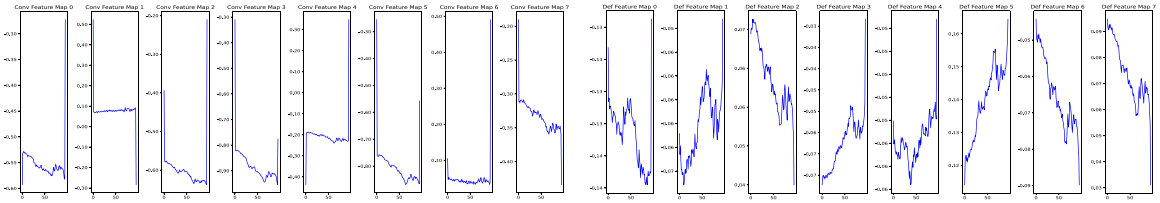
From Fig. 5, it can be observed that DCNet is capable of adapting to the dynamic changes in time series data, capturing periodic, trend, and oscillation information. While other models effectively capture the periodic information of the time series, they neglect to acquire trend and oscillation information. For instance, around the 280th time point, the sequence suddenly oscillates downward. DCNet successfully captured the trend information of such nonlinear oscillatory changes, whereas other models retained the trend information from the previous period, indicating that their predictive performance did not benefit from the module that extracts long-term feature information.

4.6. Validation of capturing nonlinear features

To further verify the effectiveness of our proposed NFAE module in extracting features from real-world time series data, we selected

Table 4
Short-term multivariate forecasting results with different metrics. All input lengths are 96 and prediction lengths are 12. The best results are in bold numbers, and the second best are highlighted with an underline.

Categories		Ours	CNNs				MLPs		Transformers		Patchs		
Models		DCNet	ConvTimeNet (2024)	TSLANet (2024)	MICN (2023)	TimesNet (2023)	TimeMixer (2024)	FITS (2024)	iTransformer (2024)	Crossformer (2023)	PDF (2024)	TSMixer (2023)	PatchTST (2023)
PEMS03	MAE	**0.145**	0.245	0.185	0.462	0.190	0.187	0.348	0.178	0.166	0.195	0.198	0.212
	MAPE	**1.295**	1.898	1.459	2.948	1.455	1.466	2.656	1.364	1.321	2.794	1.567	1.636
	RMSE	**0.208**	0.361	0.278	0.606	0.290	0.277	0.468	0.266	0.253	0.293	0.277	0.310
PEMS04	MAE	**0.183**	0.307	0.272	0.935	0.269	0.247	0.448	0.235	0.206	0.260	0.238	0.269
	MAPE	**1.550**	2.635	2.179	7.799	2.208	2.039	3.658	1.931	1.682	2.131	1.985	2.177
	RMSE	**0.311**	0.566	0.526	1.298	0.534	0.487	0.623	0.481	0.430	0.509	0.434	0.516
PEMS07	MAE	**0.130**	0.257	0.197	0.622	0.199	0.187	0.382	0.181	0.150	0.194	0.186	0.211
	MAPE	**1.349**	2.561	1.948	4.773	1.904	1.907	3.509	1.849	1.667	2.004	1.970	2.058
	RMSE	**0.191**	0.368	0.297	0.800	0.302	0.279	0.501	0.274	0.232	0.290	0.273	0.303
PEMS08	MAE	**0.124**	0.313	0.256	0.797	0.302	0.259	0.473	0.255	0.238	0.268	0.259	0.285
	MAPE	1.594	2.115	1.709	5.239	2.077	1.649	3.235	1.705	**1.497**	1.704	1.685	1.800
	RMSE	**0.204**	0.582	0.517	1.189	0.664	0.523	0.723	0.516	0.497	0.531	0.481	0.540



(a) Standard Converlutional Feature Map. (b) GDC Feature Map.

Fig. 6. Visualization of feature maps from different convolutions.

the Exchange dataset for validation. We processed the data using both standard convolutional kernels and our proposed GDC. We randomly selected a batch of data and extracted feature maps from the intermediate layers to observe the activations of the input data after passing through the convolutional layers. This helps in understanding how different convolutional modules capture nonlinear relationships. We visualized the feature maps from the first eight layers, as shown in Fig. 6.

From Fig. 6, we can draw the following conclusions:

(1) Amplitude of variations in feature maps. The feature maps from standard convolution exhibit smaller amplitudes of change with fewer fluctuations, indicating a potential inadequacy in handling sudden and nonlinear changes in the input data. The feature maps from GDC exhibit larger amplitudes of change with more noticeable fluctuations, indicating a greater flexibility in capturing sudden and nonlinear changes in the input data.

(2) Diversity in feature maps. The feature maps from standard convolution show more consistent patterns with relatively smooth changes. The feature maps from GDC show more diverse patterns and variations, indicating a stronger adaptability in handling nonlinear features.

(3) Response to sudden changes. The feature maps from standard convolution exhibit smooth transitions at points of sudden changes in the data, lacking sensitivity to such changes. The feature maps from GDC exhibit noticeable fluctuations and variations at points of sudden changes in the data, better capturing these sudden features.

Specifically, from the standard convolutional feature maps (as shown in Fig. 6(a)), we can see that most feature maps show relatively smooth and stable changes with almost no significant fluctuations or sudden changes. In feature map 5, some fluctuations can be observed, but the overall changes remain relatively stable. From the GDC feature maps (as shown in Fig. 6(b)), we can see that feature maps 1, 3, 4, and 5 exhibit significant fluctuations and sudden changes, indicating that GDC can capture dynamic sudden features in the input data. Feature maps 0 and 2 exhibit more details and fluctuations, suggesting that GDC performs well in handling non-stationary features.

4.7. Verification of capturing correlations among different variables

To further verify the effectiveness of our proposed NFAE module in extracting correlations between different variables from real-world time series data, we selected the Exchange dataset for validation. We used standard convolutional kernels, self-attention mechanisms, and our proposed GDC to extract correlations between different variables in the time series. We randomly selected a batch of data and extracted feature maps from the intermediate layers to observe the correlations of the input data after passing through the feature extraction modules. This helps in understanding the differences and correlations between different variable extraction methods. The heatmaps of the correlations between different variables in the time series extracted by different methods are shown in Fig. 7. From Fig. 7, we can draw the following conclusions:

(1) Regular convolution has certain limitations in extracting correlations between different variables in time series. While it captures the correlations between some pairs of features well (e.g., the correlation between feature 0 and feature 3 is 0.67, and the correlation between feature 0 and feature 6 is 0.57), the overall absolute values of the correlations are relatively small, limiting the overall ability to capture correlations.

(2) In the heatmap of the NFAE, the correlations between features are significantly enhanced, indicating that NFAE can better capture the high correlations between certain variables. Additionally, the NFAE model captures the correlations between features more uniformly and strongly, allowing it to better capture the complex relationships between features.

(3) The correlation map extracted by the self-attention mechanism shows smoother and more extensive correlations. For example, the correlation distribution between variable 0 and other variables is more uniform, without extreme correlation values (such as close to 1 or close to −1).

Overall, both NFAE and the self-attention mechanism perform excellently in capturing correlations between complex time series features. On the Exchange dataset, which contains a large number of nonlinear features, NFAE slightly outperforms in capturing correlations between different variables.

4.8. Long-term information utilization

To validate DCNet's ability to capture long-term correlations, we compared the performance of different models on the ETTh1 and

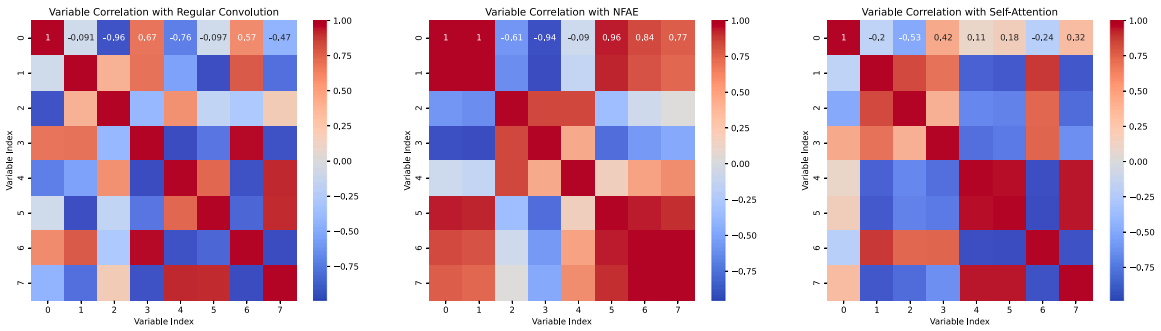


Fig. 7. Heatmaps visualizing inter-variable relationships in the Exchange dataset.

ETTm1 datasets, with output lengths set to $\{96, 720\}$ and input sequence windows set to $\{24, 48, 96, 192, 336\}$. In theory, using longer input sequence windows can increase the receptive field and potentially improve prediction performance. However, as evidenced by the experimental results in Fig. 8, the performance of Transformer-based models, TimeMixer, and MICN models does not continuously improve with increasing model input length. This suggests that these models do not benefit from longer input sequence windows, i.e., they lack in capturing long-term correlations. In contrast, DCNet, ConvTimeNet, and TSMixer show continuous improvement in model performance (decreasing MSE values) with increasing input sequence length. This indicates that these models can leverage more features from longer input windows and effectively capture long-term correlations in the data.

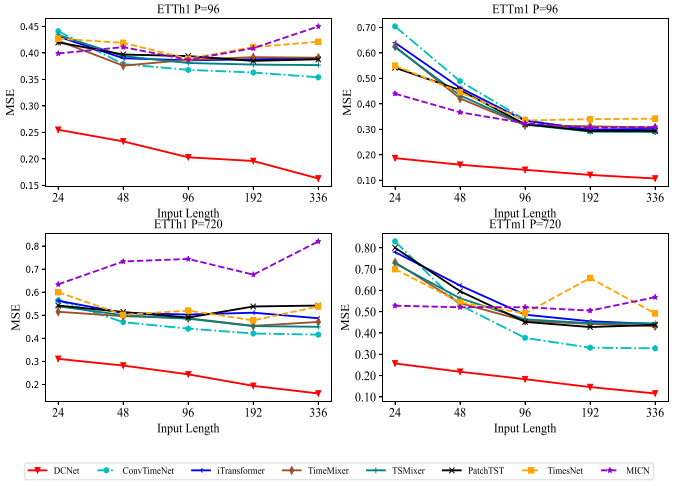


Fig. 8. The predictive performance (MSE) of different input sequence windows.

4.9. T-tests

To further validate the effectiveness of the DCNet model compared to other baseline models, we conducted T-tests to assess the significant differences between the prediction results of DCNet and the baseline models. As shown in Fig. 9, we can draw the following conclusions: (1) The T-statistic for DCNet relative to all baseline models significantly exceeds $T' = 1.782$ (by looking up the critical value in the T-distribution table based on the P-value, we obtain the T-value), indicating that the average MSE of DCNet is significantly lower than that of the baseline models. (2) The calculated P-values are all less than the threshold value of $P = 0.05$, indicating statistically significant differences in prediction results among different models. Based on the above analysis, we can conclude that the observed performance differences between DCNet and the baseline models are indeed the result of true differences, rather than random factors.

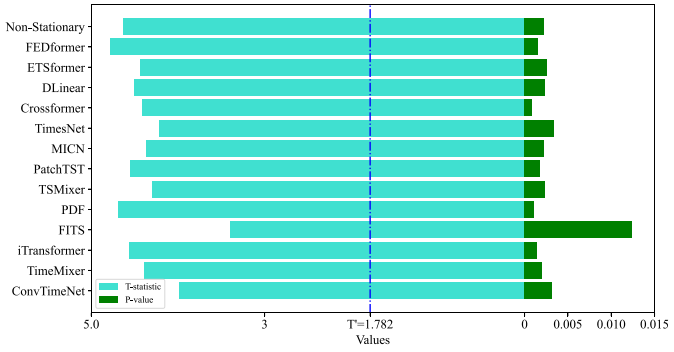


Fig. 9. The result of the statistical significance test between DCNet and fourteen baseline models.

4.10. Efficiency analysis

To evaluate the efficiency of DCNet, we selected the ETTh1 dataset and compared it with six SOTA models. We conducted efficiency analysis of the models from four aspects: computational complexity (Flops), model size (Parameters), training time, and inference time. To facilitate a clearer comparison and analysis of model efficiency, we integrated these four metrics and computed their average values as the model efficiency scores, as shown in Table 5. Table 5 distinctly demonstrates the efficiency advantages of DCNet, which are particularly crucial for practical applications of TSF. This advantage primarily stems from the data segmentation mechanism, parallel processing, parameter sharing in models, gated deformable convolutions, and the ability to quickly update network weights using the pseudoinverse algorithm. Additionally, we find that the DCNet model has a relatively large number of parameters and a longer training time. This is because DCNet uses multiple convolutional layers, which, while enhancing the model's expressive power and feature extraction capability, also inevitably increase the model's size and training time. We also observe a decreasing trend in the training time of iTransformer as the input length increases. This is because iTransformer employs an early termination strategy during training, where training halts when the loss value decreases continuously for a certain number of iterations (n = 3 in iTransformer). According to our experiments, as the input length increases, the model is more likely to meet this condition during training.

4.11. Ablation study

To evaluate the effectiveness of each component in DCNet, we conducted two different modes of ablation experiments on the DCNet architecture: module replacement and removal. Table 6 presents the results of the ablation experiments on the DCNet architecture. The variants of DCNet are as follows: (1) DCNet†: replacing the GDC module in the NFAE module of DCNet with a standard deformable convolution

Table 5
Results of the efficiency analysis.

Categories	Metrics		Flops	Parameters	Training time	Inference time	Ranking	
	Models		(G)	(M)	(s)	(s)	Four metrics	Avg ranking
Ours	DCNet	192	0.064	0.949	20.308	0.091	(1,4,3,1)	2.25
		384	0.128	1.898	35.999	0.124		
		768	0.254	3.700	49.618	0.217		
		1536	0.510	7.589	73.758	0.436		
CNNs	ConvTimeNet	192	4.745	0.265	14.978	7.977	(4,1,2,7)	3.50
		384	9.491	0.335	24.781	8.975		
		768	18.981	0.476	38.070	10.97		
		1536	37.962	0.757	48.436	13.962		
	MICN	192	108.741	27.542	116.441	2.926	(7,7,6,5)	6.25
		384	187.220	36.559	143.929	3.174		
		768	364.311	48.594	174.319	3.823		
		1536	799.024	76.665	273.936	5.757		
MLPs	TimeMixer	192	4.329	0.245	14.209	6.979	(6,5,1,6)	4.50
		384	14.895	0.902	17.310	7.978		
		768	54.757	3.464	27.918	8.576		
		1536	209.391	13.571	54.004	9.474		
Transformers	iTransformer	192	0.305	0.866	1056.093	2.493	(2,2,7,3)	3.50
		384	0.322	0.915	1036.082	2.658		
		768	0.356	1.014	422.610	2.986		
		1536	0.426	1.210	407.591	3.155		
Patchs	TSMixer	192	0.027	0.100	38.392	1.723	(3,3,4,2)	3.00
		384	0.087	0.339	41.811	1.931		
		768	0.306	1.239	56.190	1.976		
		1536	1.139	4.727	69.689	2.200		
	PatchTST	192	17.253	4.140	51.121	3.649	(5,6,5,4)	5.00
		384	34.505	5.265	80.290	3.936		
		768	69.010	7.515	87.804	4.057		
		1536	138.021	12.015	101.786	3.955		

Table 6
Results of the ablation study.

Models	Metric	ETTh2					Exchange					Electricity				
		96	192	336	720	Avg	96	192	336	720	Avg	96	192	336	720	Avg
DCNet	MSE	0.147	0.168	0.192	0.223	0.183	0.097	0.103	0.117	0.158	0.119	0.131	0.135	0.138	0.142	0.137
	MAE	0.259	0.279	0.295	0.310	0.286	0.209	0.217	0.232	0.279	0.234	0.233	0.239	0.245	0.250	0.242
DCNet†	MSE	0.149	0.172	0.196	0.230	0.187	0.115	0.124	0.138	0.179	0.139	0.134	0.138	0.151	0.153	0.144
	MAE	0.260	0.281	0.313	0.329	0.296	0.236	0.240	0.256	0.295	0.257	0.236	0.243	0.264	0.266	0.252
DCNet‡	MSE	0.151	0.174	0.204	0.231	0.190	0.105	0.111	0.127	0.166	0.127	0.140	0.144	0.146	0.153	0.146
	MAE	0.261	0.281	0.300	0.312	0.289	0.219	0.227	0.246	0.280	0.243	0.249	0.254	0.258	0.265	0.257
w/o NFAE	MSE	0.156	0.180	0.205	0.237	0.195	0.123	0.134	0.152	0.183	0.148	0.136	0.144	0.163	0.165	0.152
	MAE	0.267	0.287	0.304	0.320	0.295	0.242	0.255	0.273	0.298	0.267	0.238	0.249	0.273	0.277	0.259
w/o TFE	MSE	0.153	0.176	0.204	0.232	0.191	0.108	0.114	0.128	0.175	0.131	0.141	0.146	0.149	0.155	0.148
	MAE	0.262	0.282	0.300	0.313	0.289	0.230	0.233	0.251	0.293	0.252	0.251	0.256	0.261	0.267	0.259
w/o ATBF	MSE	0.161	0.187	0.215	0.250	0.203	0.114	0.123	0.138	0.185	0.140	0.139	0.141	0.147	0.152	0.145
	MAE	0.272	0.293	0.310	0.327	0.301	0.230	0.240	0.257	0.301	0.257	0.240	0.243	0.250	0.257	0.248
w/o (ATBF+NFAE)	MSE	0.162	0.188	0.217	0.251	0.205	0.144	0.152	0.169	0.233	0.175	0.152	0.160	0.162	0.167	0.160
	MAE	0.272	0.293	0.311	0.327	0.301	0.257	0.268	0.289	0.341	0.289	0.252	0.262	0.266	0.271	0.263
w/o (ATBF+TFE)	MSE	0.166	0.202	0.235	0.272	0.219	0.135	0.137	0.147	0.204	0.156	0.148	0.154	0.156	0.161	0.155
	MAE	0.272	0.299	0.319	0.336	0.307	0.263	0.267	0.273	0.315	0.280	0.248	0.255	0.261	0.265	0.257
w/o (NFAE+TFE)	MSE	0.169	0.209	0.250	0.302	0.233	0.139	0.142	0.151	0.210	0.161	0.155	0.162	0.164	0.165	0.162
	MAE	0.275	0.303	0.326	0.351	0.314	0.268	0.266	0.275	0.320	0.282	0.268	0.275	0.280	0.279	0.276
w/o ALL	MSE	0.191	0.230	0.274	0.331	0.257	0.156	0.161	0.174	0.247	0.185	0.160	0.165	0.170	0.174	0.167
	MAE	0.290	0.317	0.339	0.366	0.328	0.286	0.292	0.304	0.351	0.308	0.261	0.267	0.274	0.279	0.270

module; (2) DCNet‡: replacing the TFE module with a standard 1D convolution module; (3) w/o NFAE: removing the module designed to extract nonlinear features between different observations and inter-dependencies among different variables; (4) w/o TFE: removing the module designed to extract local contextual information and global patterns in time series; (5) w/o ATBF: removing the module designed to highlight nonlinear features; (6) w/o (ATBF+NFAE): removing the modules for highlighting and extracting nonlinear features; (7) w/o

(ATBF+TFE): removing the modules for highlighting nonlinear features and extracting temporal correlations; (8) w/o (NFAE+TFE): removing the modules for extracting temporal and spatial correlations. (9) w/o ALL: removing all modules and using only the FFN layer with the pseudoinverse algorithm to perform time series forecasting. We conducted ablation studies on the ETTh2, Exchange, and Electricity datasets, which exhibit significant nonlinear characteristics, with an input length set to 96 and prediction lengths set to {96,192,336,720}.

The results in Table 6 indicate that each component is essential. Specifically, across all test datasets: (1) The performance metrics of the DCNet† variant are inferior to those of DCNet, indicating that the gated deformable convolution module better extracts nonlinear features compared to using a deformable convolution module alone. (2) The performance of the DCNet‡ variant indicates that using a standard CNN to extract local features and patterns in time series is less effective than using multi-scale convolution, highlighting the effectiveness of multi-scale convolution in capturing local features and patterns in time series. (3) The removal of the (ATBF+TFE), (ATBF+NFAE), and (NFAE+TFE) modules all lead to performance degradation, indicating that the feature information extracted by a single module is limited and lacks sufficient feature representation capacity. (4) Removing NFAE (w/o NFAE) will weaken the ability to extract nonlinear information from time series, and relying solely on the temporal information extracted by the model will reduce prediction accuracy, especially on the ETTh2 dataset. (5) Removing TFE (w/o TFE) prevents the model from extracting local contextual information and global patterns from time series, leading to a decline in prediction performance, particularly at longer prediction lengths (336 and 720). (6) Removing ATBF (w/o ATBF) results in a significant reduction in prediction accuracy. This reduction is likely due to the inability to effectively extract important nonlinear features and edge information from the time series, which weakens the ability to capture complex dynamic patterns and nonlinear features. (7) The experimental results using only the FFN layer (w/o ALL) are poor, further proving that the FFN layer lacks the capability to effectively extract complex features from time series data.

5. Conclusion and future work

This paper introduces DCNet, a dynamic convolutional network based on adaptive temporal bilateral filters, designed for the effective extraction and utilization of time series features. DCNet employs adaptive bilateral filters to preserve nonlinear features, utilizes convolutional kernels with varying dilation factors to capture local patterns at different resolutions, and adopts adaptive average pooling to learn global temporal patterns. The gated mechanism-based deformable convolutions further capture nonlinear features and interdependencies between different observations and variables. In real-world scenarios, data often exhibit rapid drift, leading to significant differences in the distribution between training data and test data, which can result in a performance decline for traditional models, including DCNet, on test data. In future research, we plan to explore test-time training methods to adjust the model in real-time, addressing dynamically changing data environments, and enhancing the practicality and reliability of the model in real-world applications.

CRediT authorship contribution statement

Dandan Zhang: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Conceptualization. **Zhiqiang Zhang:** Writing – review & editing, Writing – original draft, Software, Methodology, Conceptualization. **Nanguang Chen:** Writing – review & editing, Supervision, Conceptualization. **Yun Wang:** Writing – review & editing, Funding acquisition.

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

Data will be made available on request.

Acknowledgments

This work is partially supported by National Hi-Tech Project, China with grant No. WDZC20215250117.

Appendix A. Adaptive temporal bilateral filtering computational complexity

For a time series of length M, the computational complexity of bilateral filtering primarily depends on the size of the filtering window W. At each time step, bilateral filtering needs to consider all data points within the window. For each data point, the distance (both spatial and intensity) to other data points within the window must be calculated. The operations for calculating the distance and the weighted average require $O(W)$ time. Therefore, for the entire time series, the total computational complexity of bilateral filtering is: $O(M \times W)$.

To reduce computational complexity, data chunking and parallel computing can be employed:

(1) Data Chunking: The time series data can be divided into multiple chunks, and bilateral filtering can be independently applied to each chunk. This reduces the window size for each computation and balances the computational load.

(2) Parallel Computing: Utilizing multi-core processors can significantly speed up the filtering process.

Assuming the time series is divided into P chunks for parallel processing, each chunk would have a size of M/P. The total complexity can then be reduced to: $O(\frac{M}{P} \times W)$.

Appendix B. Model dynamic updates

As time progresses, the data distribution in real-world applications may change, resulting in a significant deterioration of the model's predictive performance. To address this issue, we extend the model training process to a dynamic update stage. We assume that when the model's predictive performance decreases by 5%, we adopt newly collected samples to retrain the DCNet model. It is noteworthy that during this process, we maintain a fixed training set size, which means that when introducing new data, an equal amount of the earliest historical data is removed. This strategy ensures the speed of model training and reduces the required time.

Algorithm 2 outlines the details of the dynamic prediction process.

Appendix C. Supplementary information on the datasets

We extensively evaluated the proposed DCNet model on twelve benchmark datasets spanning four real-world fields: energy, finance, medical, and traffic. Table 1 summarizes the key features of these datasets.

- **ETT**[1]**:** The dataset records continuous operation data of power resources over a period of two years, including seven indicators such as oil temperature and load.

[1] ETT dataset was acquired at https://github.com/zhouhaoyi/ETDataset.

Algorithm 2: Dynamic prediction.

Input:

$X = \{x_1, \ldots, x_i, \ldots, x_M\}$: X denotes the training dataset, where x_i represents the N variable factors at time i.

1: **Phase I.** Training the proposed DCNet model.
2: **Randomly generated:** The weight ω and the bias vector b between the input layer and the hidden layer in FFN.
3: Data processing for X_{atbf} is performed based on Eqs.(1) – (9).
4: Extract the temporal correlation χ_{local} and χ_{global} based on Eqs.(10) - (11).
5: Extract the inter-variable correlation X_s based on Eqs.(12) - (13).
6: Input X_{sum} into the FFN and calculate β.
7: **Phase II**. Dynamically forecasting the values of a time series.
8: Input the obtained new samples x_j and training data set X into the trained DCNet model. If the MSE value exceeds the threshold, update β.

Output: Predicting the trend of changes over a future time period.

- **Electricity**[2]**:** The dataset comprises hourly electricity consumption data for 321 customers over two years.
- **Traffic**[3]**:** The dataset consists of measurements collected every hour from 862 sensors located on the highways in the San Francisco Bay Area.
- **Exchange**[4]**:** The dataset records the daily exchange rates of eight different countries over a period of 26 years.
- **ILI**[5]**:** The dataset collects seven indicators, including the proportion of influenza patients and the total number of patients, from the Centers for Disease Control and Prevention in the United States on a weekly basis.
- **PEMS** [6]**:** This dataset records traffic network data collected every 5 min on California highways, including metrics such as traffic flow, speed, and occupancy.

References

[1] Y. Pang, X. Zhou, J. Zhang, Q. Sun, J. Zheng, Hierarchical electricity time series prediction with cluster analysis and sparse penalty, Pattern Recognit. 126 (2022) http://dx.doi.org/10.1016/j.patcog.2022.108555.

[2] M. Jiang, W. Chen, H. Xu, Y. Liu, A novel interval dual convolutional neural network method for interval-valued stock price prediction, Pattern Recognit. 145 (2024) http://dx.doi.org/10.1016/j.patcog.2023.109920.

[3] L. Yang, T. Gao, W. Wei, M. Dai, C. Fang, J. Duan, Multi-task meta label correction for time series prediction, Pattern Recognit. 150 (2024) 110319, http://dx.doi.org/10.1016/j.patcog.2024.110319.

[4] Z. Wang, H. Ran, J. Ren, M. Sun, PWDformer: Deformable transformer for long-term series forecasting, Pattern Recognit. 147 (2024) http://dx.doi.org/10.1016/j.patcog.2023.110118.

[5] H. Hu, Y. Li, X. Zhang, M. Fang, A novel hybrid model for short-term prediction of wind speed, Pattern Recognit. 127 (2022) http://dx.doi.org/10.1016/j.patcog.2022.108623.

[6] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, Z. Jia, Freeway performance measurement system: mining loop detector data, Transp. Res. Rec. 1748 (1) (2001) 96–102.

[7] Y. Nie, N.H. Nguyen, P. Sinthong, J. Kalagnanam, A time series is worth 64 words: Long-term forecasting with transformers, in: The Eleventh International Conference on Learning Representations, 2023.

[8] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, M. Long, iTransformer: Inverted transformers are effective for time series forecasting, in: The Twelfth International Conference on Learning Representations, 2024.

[2] https://archieve.ics.uci.deu/ml/datasets/ElectricityLoadDiagrams 20112014.

[3] http://pems.dot.ca.gov.

[4] https://github.com/laiguo-kun/multivariate-time-series-data.

[5] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html.

[9] M. Hassanin, A. Khamiss, M. Bennamoun, F. Boussaid, I. Radwan, Crossformer: Cross spatio-temporal transformer for 3d human pose estimation, 2022, arXiv preprint arXiv:2203.13387.

[10] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting? in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, 2023, pp. 11121–11128.

[11] V. Ekambaram, A. Jati, N. Nguyen, P. Sinthong, J. Kalagnanam, TSMixer: Lightweight MLP-mixer model for multivariate time series forecasting, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '23, 2023, pp. 459–469.

[12] T. Zhang, J. Bian, Y. Zhang, X. Yi, J. Li, W. Cao, S. Zheng, Less is more: Fast multivariate time series forecasting with light sampling-oriented MLP structures, 2022.

[13] M. Cheng, J. Yang, T. Pan, Q. Liu, Z. Li, ConvTimeNet: A deep hierarchical fully convolutional model for multivariate time series analysis, 2024, arXiv preprint arXiv:2403.01493.

[14] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, Y. Xiao, Micn: Multi-scale local and global context modeling for long-term series forecasting, in: The Eleventh International Conference on Learning Representations, 2022.

[15] W. Zeng, C. Lin, K. Liu, J. Lin, A.K.H. Tung, Modeling spatial nonstationarity via deformable convolutions for deep traffic flow prediction, IEEE Trans. Knowl. Data Eng. 35 (3) (2023) 2796–2808.

[16] M. Liu, A. Zeng, M. Chen, Z. Xu, Q. Lai, L. Ma, Q. Xu, SCINet: Time series modeling and forecasting with sample convolution and interaction, 35 (2022).

[17] E. Eldele, M. Ragab, Z. Chen, M. Wu, X. Li, TSLANet: Rethinking transformers for time series representation learning, 2024, arXiv:2404.08472.

[18] Y. Yu, R. Ma, Z. Ma, Robformer: A robust decomposition transformer for long-term time series forecasting, Pattern Recognit. 153 (2024) 110552, http://dx.doi.org/10.1016/j.patcog.2024.110552.

[19] G. Woo, C. Liu, D. Sahoo, A. Kumar, S. Hoi, Etsformer: Exponential smoothing transformers for time-series forecasting, 2022, arXiv preprint arXiv:2202.01381.

[20] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, in: International Conference on Machine Learning, 2022, pp. 27268–27286.

[21] P. Chen, Y. Zhang, Y. Cheng, Y. Shu, Y. Wang, Q. Wen, B. Yang, C. Guo, Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting, in: The Twelfth International Conference on Learning Representations, 2024.

[22] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J.Y. Zhang, J. ZHOU, TimeMixer: Decomposable multiscale mixing for time series forecasting, in: The Twelfth International Conference on Learning Representations, 2024.

[23] S. Zhuo, J. Zhang, Attention-based deformable convolutional network for Chinese various dynasties character recognition, Expert Syst. Appl. 238 (B) (2024).

[24] J. Du, W. Fan, C. Gong, J. Liu, F. Zhou, Aggregated-attention deformable convolutional network for few-shot SAR jamming recognition, Pattern Recognit. 146 (2024) http://dx.doi.org/10.1016/j.patcog.2023.109990.

[25] Z. Wu, M.E. Paoletti, H. Su, X. Tao, L. Han, J.M. Haut, A. Plaza, Background-guided deformable convolutional autoencoder for hyperspectral anomaly detection, IEEE Trans. Geosci. Remote Sens. 61 (2023) http://dx.doi.org/10.1109/TGRS.2023.3334562.

[26] L. Yu, X. Zhi, J. Hu, S. Zhang, R. Niu, W. Zhang, S. Jiang, Improved deformable convolution method for aircraft object detection in flight based on feature separation in remote sensing images, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 17 (2024) 8313–8323, http://dx.doi.org/10.1109/JSTARS.2024.3386696.

[27] Y. Shi, C. Wang, S. Xu, M.-D. Yuan, F. Liu, L. Zhang, Deformable convolution-guided multiscale feature learning and fusion for UAV object detection, IEEE Geosci. Remote Sens. Lett. 21 (2024) http://dx.doi.org/10.1109/LGRS.2024.3362890.

[28] M.M. Bronstein, Lazy sliding window implementation of the bilateral filter on parallel architectures, IEEE Trans. Image Process. 20 (6) (2011) 1751–1756, http://dx.doi.org/10.1109/TIP.2010.2095020.

[29] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, in: Sixth International Conference on Computer Vision, 1998, pp. 839–846, http://dx.doi.org/10.1109/ICCV.1998.710815.

[30] H. Lv, P. Shan, H. Shi, L. Zhao, An adaptive bilateral filtering method based on improved convolution kernel used for infrared image enhancement, Signal Image Video Process. 16 (8) (2022) 2231–2237.

[31] M.M.U. Faiz, Adaptive bilateral filter, Image Process. Appl. (2023) 32.

[32] H. Salehi, J. Vahidi, A novel hybrid filter for image despeckling based on improved adaptive wiener filter, bilateral filter and wavelet filter, Int. J. Image Graph. 21 (03) (2021) 2150036.

[33] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (9) (2015) 1904–1916.

[34] C. Li, Z. Qiu, X. Cao, Z. Chen, H. Gao, Z. Hua, Hybrid dilated convolution with multi-scale residual fusion network for hyperspectral image classification, Micromachines 12 (5) (2021) 545.

[35] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable convolutional networks, in: 2017 IEEE International Conference on Computer Vision, ICCV, 2017.

[36] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541), Vol. 2, Ieee, 2004, pp. 985–990.
[37] T. Dai, B. Wu, P. Liu, N. Li, J. Bao, Y. Jiang, S.-T. Xia, Periodicity decoupling framework for long-term series forecasting, in: The Twelfth International Conference on Learning Representations, 2024.
[38] Z. Xu, A. Zeng, Q. Xu, FITS: Modeling time series with $10k$ parameters, in: The Twelfth International Conference on Learning Representations, 2024.
[39] Y. Liu, H. Wu, J. Wang, M. Long, Non-stationary transformers: Exploring the stationarity in time series forecasting, Adv. Neural Inf. Process. Syst. 35 (2022) 9881–9893.
[40] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, M. Long, Timesnet: Temporal 2d-variation modeling for general time series analysis, in: The Eleventh International Conference on Learning Representations, 2022.

Dandan Zhang received a BS and MS degree in Information and Communication Engineering from North University of China, in 2015 and 2019, respectively. She is currently working toward a Ph.D. degree in the School of Computer Science and Engineering, Southeast University, China. Her research interests include data mining, artificial intelligence, edge computing, computer network, machine learning and deep learning.

Zhiqiang Zhang received a BS and MS degree in automation from Shandong University of Science and Technology and control engineering from the Northeastern University, in 2013 and 2019, respectively. He is currently working toward a Ph.D. degree in the School of Computer Science and Engineering, Southeast University, China. His research interests include data mining, artificial intelligence, big data analysis, cloud computing, machine learning and deep learning.

Nanguang Chen is an Associate Professor of Biomedical Engineering at the National University of Singapore (NUS). He received his Ph.D. in Biomedical Engineering in 2000 from Tsinghua University. He also received his M.Sc. in Physics (Peking University) and B.Sc. in Electrical Engineering (Hunan University) in 1994 and 1988, respectively. He joined the Optical and Ultrasound Imaging Lab at the University of Connecticut in 2000 as a postdoctoral fellow and became an Assistant Research Professor in 2002. Since 2004, he has been a faculty member with NUS. His research interests include diffuse optical tomography, optical coherence tomography, and multi-dimensional fluorescence microscopy, and laser speckle imaging. He has published more than 200 papers and holds five international patents.

Yun Wang received the B.S. degree in computer software from Nanjing University, China, in 1989 and the M.E. degree and Ph.D. degree in computer networking from Southeast University, China, in 2004 and 2007, respectively. She was a post-doctoral researcher in INRIA/IRISA, France and senior researcher at University of Texas at Dallas, USA from 1999 to 2002. She joined in Southeast University, China in 1997, and is a full professor at School of Computer Science and Engineering. Her research interests include distributed systems, edge computing and computer network. She was PI for more than 20 research projects supported by national and international grants. She obtains 3 Science and Technology Awards from Ministry of Education, China. She has published more than 120 peer-reviewed journal and conference papers, including TOC, JPDC, InfoCom, ICDCS, and IPDPS.