EffiCANet: Efficient Time Series Forecasting with Convolutional Attention

Xinxing Zhou
College of CS
Nankai University
zhouxinxing@dbis.nankai.edu.cn

Jiaqi Ye
College of CS
Nankai University
yjq@mail.nankai.edu.cn

Shubao Zhao
Digital Research Institute
of ENN Group
zhaoshubao@enn.cn

Ming Jin
School of ICT
Griffith University
mingjinedu@gmail.com

Chengyi Yang
Digital Research Institute
of ENN Group
yangchengyia@enn.cn

Yanlong Wen*
College of CS
Nankai University
wenyl@nankai.edu.cn

Xiaojie Yuan
College of CS
Nankai University
yuanxj@nankai.edu.cn

ABSTRACT

The exponential growth of multivariate time series data from sensor networks in domains like industrial monitoring and smart cities requires efficient and accurate forecasting models. Current deep learning methods often fail to adequately capture long-range dependencies and complex inter-variable relationships, especially under real-time processing constraints. These limitations arise as many models are optimized for either short-term forecasting with limited receptive fields or long-term accuracy at the cost of efficiency. Additionally, dynamic and intricate interactions between variables in real-world data further complicate modeling efforts. To address these limitations, we propose EffiCANet, an Efficient Convolutional Attention Network designed to enhance forecasting accuracy while maintaining computational efficiency. EffiCANet integrates three key components: (1) a Temporal Large-kernel Decomposed Convolution (TLDC) module that captures long-term temporal dependencies while reducing computational overhead; (2) an Inter-Variable Group Convolution (IVGC) module that captures complex and evolving relationships among variables; and (3) a Global Temporal-Variable Attention (GTVA) mechanism that prioritizes critical temporal and inter-variable features. Extensive evaluations across nine benchmark datasets show that EffiCANet achieves the maximum reduction of 10.02% in MAE over state-of-the-art models, while cutting computational costs by 26.2% relative to conventional large-kernel convolution methods, thanks to its efficient decomposition strategy.

PVLDB Reference Format:
Xinxing Zhou, Jiaqi Ye, Shubao Zhao, Ming Jin, Chengyi Yang, Yanlong Wen, and Xiaojie Yuan. EffiCANet: Efficient Time Series Forecasting with Convolutional Attention. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

*Corresponding author

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

1 INTRODUCTION

As we advance into an era dominated by AI-driven systems and ubiquitous Internet of Things (IoT), time series data has become a cornerstone of modern industries [4, 35, 49]. From intelligent manufacturing and smart cities to precision medicine, vast sensor networks continuously generate complex multivariate data streams, driving real-time insights for proactive monitoring, resource scheduling, and predictive maintenance [10, 33, 44, 48]. This data surge underscores the need for advanced time series forecasting methods to enable data-driven decision-making and enhance operational intelligence across diverse emerging applications.

Traditional time series forecasting techniques, including Transformer-based methods [31, 36, 52, 53], have demonstrated strong performance across various domains. However, while they excel at capturing long-term dependencies, they often struggle to balance computational efficiency with real-time processing in resource-constrained environments. In contrast, temporal convolutional networks (TCNs) [19] are more efficient but suffer from limited receptive fields, restricting their ability to capture long-term patterns. Linear models [21, 46], on the other hand, offer simplicity and computational efficiency but lack the capacity to capture complex, non-linear dependencies. These trade-offs reveal a crucial gap in current research: there is a critical need for models that can balance efficiency with accurate forecasting over short and long horizons, especially for real-time decision-making. Bridging this gap is crucial for transforming the potential of IoT and AI-driven systems into actionable, timely, and reliable insights that optimize operations, enhance resource allocation, and foster innovation in sectors such as smart cities, precision manufacturing, and healthcare [5, 32, 37].

In real-world forecasting, time series data rarely exhibit independence. Interactions among variables are often complicated by asynchrony and lag effects. For example, Figure 1(a) shows three variables with asynchronous behavior due to measurement discrepancies, manifested as slight random shifts at specific timestamps. A common example can be found in climate monitoring, where temperature and humidity readings from different sensors may not align due to sensor calibration errors and environmental conditions, leading to misunderstandings regarding underlying environmental

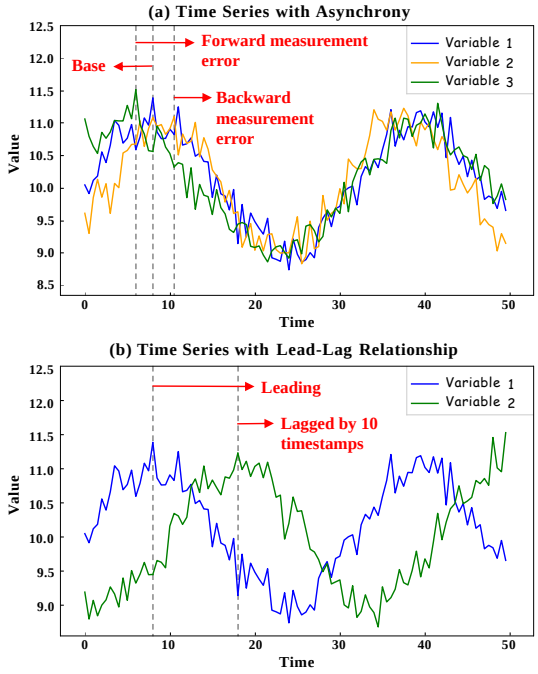


Figure 1: Illustration of asynchrony and lead-lag relationship in multiple time series: (a) Three variables exhibiting asynchronous behavior due to measurement errors, marked by vertical dashed lines. (b) A lead-lag relationship between two variables, with one leading the other by 10 timestamps.

interactions. Besides, Figure 1(b) illustrates a lead-lag relationship, where one variable is influenced by the historical values of another. This scenario is often observed in supply chain management, where inventory levels may respond to changes in demand with a noticeable delay. Ignoring these relationships can cause models to miss important latent connections, reducing their ability to capture the dynamics of complex systems. Addressing these challenges is crucial for developing robust and reliable forecasting models that accurately reflect real-world behaviors.

In this paper, we study the problem of multivariate time series forecasting, focusing on efficiently capturing both short- and long-term dependencies, as well as the dynamic interactions between variables. However, existing approaches [31, 43, 50, 51] often struggle to balance accuracy, computational efficiency, and the complexity of inter-variable dynamics. Specifically, we identify two primary challenges in current methods that hinder their effectiveness in complex, real-time forecasting tasks.

Challenge 1: Efficiently capturing long-term dependencies. While mainstream architectures like Transformers are highly effective at learning long-term dependencies, their high computational cost limits their feasibility in real-time or resource-constrained scenarios. In contrast, convolutional models, such as TCNs, offer computationally efficient local pattern recognition but suffer from limited receptive fields, making long-term dependency capture challenging. Expanding the convolutional receptive field through large kernels [27] partially alleviates this issue but increases computational complexity. The challenge lies in expanding the receptive field efficiently, enabling both short- and long-term forecasting without compromising speed or scalability.

Challenge 2: Modeling dynamic inter-variable dependencies with asynchronous and lagged relationships. In multivariate time series forecasting, variable dependencies are often complex and evolve dynamically. These relationships are further complicated by issues like sensor noise, time lags, and the heterogeneous nature of the variables. Existing models frequently emphasize either temporal dependencies or inter-variable relationships in isolation, failing to account for asynchronous or lagged interactions. This limitation leads to an incomplete understanding of variable interdependencies, especially when variables influence one another with time offsets. Developing methods that dynamically model these evolving relationships is crucial for accurate and robust forecasting.

To address these challenges, we propose EffiCANet, an Efficient Convolutional Attention Network designed to capture both short- and long-term dependencies, while dynamically modeling complex inter-variable relationships.

Addressing Challenge 1: To overcome the limitations of convolutional models in capturing long-term dependencies efficiently, we introduce the Temporal Large-kernel Decomposed Convolution (TLDC) module. This module innovatively decomposes large kernels into a series of smaller, computationally efficient convolutions, significantly expanding the receptive field without a proportional increase in computational cost. Combined with an efficiently designed global temporal attention mechanism in the Global Temporal-Variable Attention (GTVA) module that adaptively highlights relevant time steps, our model proficiently captures both short- and long-term dependencies. This hybrid approach of convolution and attention effectively mitigates the computational bottlenecks of traditional large-kernel methods, making the model highly suitable for scenarios requiring low-latency predictions.

Addressing Challenge 2: To tackle the complexities of modeling dynamic inter-variable relationships, we present the Inter-Variable Group Convolution (IVGC) module. This module leverages group convolutions [28] to capture inter-variable interactions over adjacent time windows, organizing variable-time window pairs into shared kernel groups for efficient processing. By sharing kernels across these groups, the model adapts to evolving dependencies. Additionally, a global variable attention mechanism in GTVA module assigns adaptive weights to each variable, dynamically responding to shifts in variable importance. This combined approach not only enhances the model's ability to learn intricate relationships in multivariate time series data but also ensures efficient computation, making it well-suited for real-time applications.

In summary, the key contributions of this paper are as follows:

- We identify the challenges of capturing long-term temporal dependencies and complex inter-variable relationships in resource-constrained environments and propose EffiCANet, a novel convolutional attention network designed to address these issues.
- EffiCANet incorporates a large-kernel decomposed convolution module for efficient long-term temporal dependency modeling, a group convolution module for dynamic inter-variable dependency capture, and a global attention mechanism to enhance temporal and variable feature extraction.

- Extensive experiments on nine public datasets demonstrate that EffiCANet consistently outperforms state-of-the-art methods in both forecasting accuracy and computational efficiency.

The remainder of this paper is organized as follows: Section 2 reviews the relevant literature. Section 3 details our proposed method. Section 4 describes the experimental setup and results. Finally, Section 5 concludes the paper.

2 RELATED WORK

2.1 Convolution-based time series forecasting

Convolutional neural networks (CNNs) have long been utilized in time series forecasting due to their ability to capture local temporal patterns efficiently. Despite the rise of Transformer-based models, CNNs remain relevant for their computational efficiency and adaptability [13, 26, 30, 38, 40, 41].

Convolutional techniques have evolved substantially over time. For instance, MLCNN [2] utilizes convolutional layers to generate multi-level representations for near and distant future predictions. SCINet [23] employs a recursive downsample-convolve-interact architecture, using multiple convolutional filters to extract temporal features from downsampled sequences. MICN [39] integrates multi-scale down-sampled and isometric convolutions to capture both local and global temporal patterns. TimesNet [42] transforms 1D time series into 2D tensors and applies inception blocks with 2D convolutional kernels to model period variations. Recently, the large-kernel convolution [6] and separable convolution [14], originating from computer vision, have proven effective for time series analysis. ModernTCN [27] extends the receptive field of traditional TCN architectures to enhance long-term dependency modeling, while ConvTimeNet [3] leverages depthwise and pointwise convolutions to model both global sequence and cross-variable dependencies.

While large kernels or deep architectures improve long-term dependency capture, they often lead to excessive parameter growth, computational overhead, and increased memory consumption. EffiCANet addresses these issues by using large-kernel decomposed convolutions, efficiently modeling long-term dependencies without sacrificing computational efficiency.

2.2 Attention-based time series forecasting

Attention mechanisms, originally developed for tasks such as machine translation, have gained widespread adoption in time series forecasting. Their strength lies in dynamically focusing on relevant temporal patterns, enabling the modeling of both short-term and long-term dependencies.

2.2.1 Transformer Models. Transformer-based architectures [36] have become foundational in attention-based time series forecasting due to their proficiency in capturing long-range dependencies. Early Transformer models struggled with scalability in time series applications, prompting the development of specialized variants [34]. LogTrans [20] introduces sparse attention to reduce computational overhead, while Informer [52] incorporates ProbSparse attention for enhanced scalability. Autoformer [43] and FEDformer [53] adopt decomposition-based methods to separate trend and seasonal components. Pyraformer [24] leverages hierarchical structures to balance efficiency and accuracy. Crossformer [47] and PatchTST [31] employs cross-dimensional attention and patch-based approaches to improve multivariate time series modeling. Recently, iTransformer [25] inverts the standard Transformer architecture to enhance multivariate correlation modeling by attending to variate tokens rather than temporal ones. Pathformer [1] adopts multi-scale modeling with adaptive pathways to capture temporal dynamics at various resolutions, while SAMformer [16] addresses overfitting issues through sharpness-aware optimization. TSLANet [7] further refines attention mechanisms by incorporating adaptive spectral and interactive convolution blocks to better manage noise and multi-scale dependencies.

2.2.2 Other Attention-Based Methods. Beyond the Transformer paradigm, other attention-based models have also shown promise in time series forecasting. These models combine attention mechanisms with various techniques to enhance the extraction of temporal patterns. MH-TAL [8] integrates temporal attention within RNNs to discover hidden patterns. RGDAN [9] incorporates a graph diffusion attention module for learning spatial dependencies and a temporal attention module to capture time-related correlations. FMamba [29] merges the selective state space model of Mamba [11] with fast-attention techniques, optimizing temporal feature extraction while maintaining linear computational complexity.

Despite their success, attention-based models face challenges such as high computational cost and reliance on positional encodings, which may not fully capture complex temporal dynamics. EffiCANet addresses these limitations by integrating a more efficient attention mechanism that dynamically captures both temporal and inter-variable dependencies.

2.3 Linear models for time series forecasting

Recently, linear models have re-emerged as powerful tools for time series forecasting due to their simplicity, interpretability, and effectiveness in capturing linear dependencies. Models such as DLinear and NLinear [46] first show that straightforward linear approaches can rival and occasionally surpass complex Transformer architectures in long-term forecasting. [21] explore the impact of reversible normalization (RevIN) [17] and channel independence (CI) on linear models, and propose RLinear, which consists of RevIN and a single linear layer that maps the input to the output time series. FITS [45] applies a real discrete Fourier transform (RFT) and optional high-frequency filtering in the frequency domain.

However, linear models are inherently limited in their ability to capture the complex non-linear dependencies present in multivariate datasets. While they perform well on low-dimensional datasets, their performance deteriorates significantly on datasets with a larger number of variables, highlighting the need for more well-designed models like EffiCANet that can effectively capture both linear and non-linear dependencies.

3 METHODOLOGY

3.1 Problem Definition

We consider a multivariate time series $\mathbf{X} \in \mathbb{R}^{M \times T}$, where M represents the number of variables and T denotes the length of the

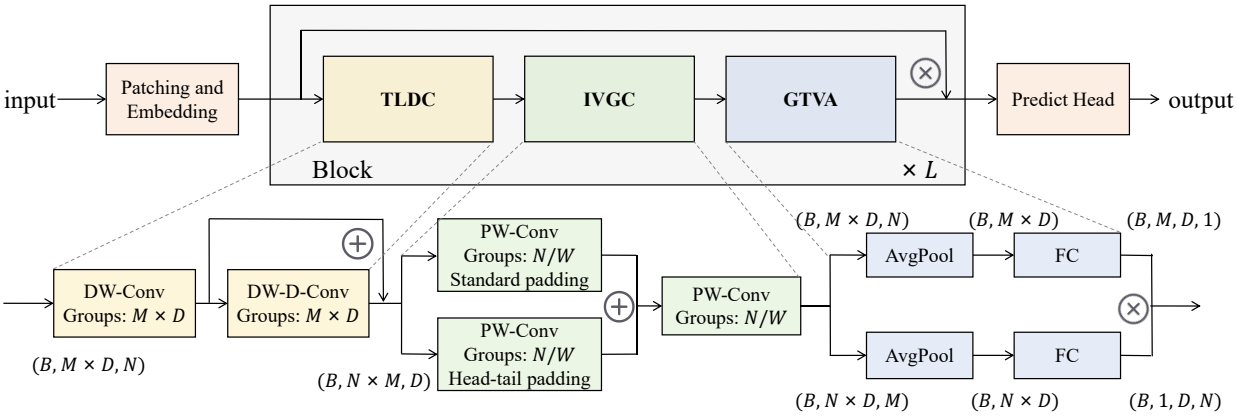


Figure 2: Overview of the EffiCANet architecture. The input is first processed through a patching and embedding layer, transforming raw data into a suitable feature space. The core structure consists of L stacked Blocks, each containing three key modules: TLDC, IVGC, and GTVA. Within each Block, the output is iteratively refined by element-wise multiplication with its respective input, enhancing feature representations. Finally, the output is passed through a prediction head to generate the forecast.

time series. Each element $\mathbf{X}_m^t \in \mathbb{R}$ corresponds to the value of the m-th variable at time step t. The sequence $\mathbf{X}_m \in \mathbb{R}^T$ represents the full time series for the m-th variable. Given a time interval Δt, the subsequence $\mathbf{X}^{t_0:t_0+\Delta t} \in \mathbb{R}^{M\times\Delta t}$ refers to the segment of all variables between time steps t_0 and $t_0 + \Delta t$.

Our goal is to define a predictive model $\mathcal{F}$ that, given a historical window of the time series, can forecast the values for future time steps. Specifically, the model $\mathcal{F}$ takes a historical time window $\mathbf{X}^{t_0-H:t_0}$, where H is the window length, and predicts the next τ time steps, denoted as $\hat{\mathbf{X}}^{t_0:t_0+\tau}$. The relationship can be formalized as:

$$\hat{\mathbf{X}}^{t_0:t_0+\tau} = \mathcal{F}_\Phi(\mathbf{X}^{t_0-H:t_0}) \tag{1}$$

where Φ denotes the parameters of the model $\mathcal{F}$.

3.2 Model Overview

EffiCANet is designed to efficiently capture both temporal dependencies and inter-variable dynamics in multivariate time series forecasting. As shown in Figure 2, the model processes input data through a sequence of stacked blocks, each integrating large-kernel convolutions, group convolutions, and global attention mechanisms to capture complex relationships across both temporal and variable dimensions.

Given the multivariate time series input $\mathbf{X}_{\text{in}} = \mathbf{X}^{t_0-H:t_0} \in \mathbb{R}^{M\times H}$, we adopt a patching and embedding strategy inspired by [27]. Initially, the input is reshaped via an unsqueeze operation into $\mathbf{X}_{\text{in}} \in \mathbb{R}^{M\times 1\times H}$, introducing a channel dimension that enables independent processing of each variable. A convolutional stem layer then partitions the sequence into patches using a kernel size of P and a stride of S, producing $N = \lfloor \frac{H-P}{S} \rfloor + 2$ overlapping patches. This convolution maps the single input channel to D output channels, resulting in an embedded tensor $\mathbf{X}_{\text{emb}} \in \mathbb{R}^{M\times D\times N}$, where N represents the number of patches.

The model extracts features through a series of L stacked blocks, each designed to iteratively refine the feature representations by capturing both localized and global temporal-variable patterns.

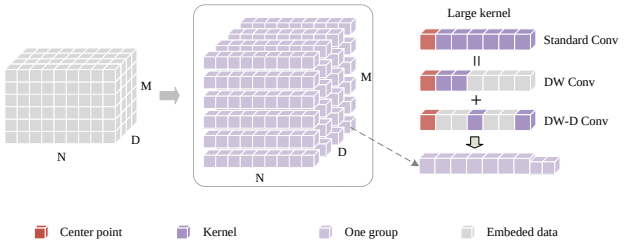


Figure 3: Illustration of the TLDC module. The input tensor, shaped as (M, D, N), is processed through group convolution, segregating it into $M \times D$ groups corresponding to each variable-channel pair. Each group undergoes a decomposition of standard convolution into two stages: a depth-wise convolution (DW Conv) and a depth-wise dilated convolution (DW-D Conv). In this example, a DW Conv with a kernel size of 5 is followed by a DW-D Conv with the same kernel size and a dilation rate of 3, together simulating the effect of a large kernel size of 13. The combined outputs capture both short-term and long-term temporal dependencies effectively.

Formally, the output of the l-th block, denoted as $\mathbf{Z}^{(l)}$, is computed as:

$$\mathbf{Z}^{(l)} = f_{\text{Block}}^{(l)}(\mathbf{Z}^{(l-1)}) \tag{2}$$

where $\mathbf{Z}^{(0)} = \mathbf{X}_{\text{emb}}$, and $f_{\text{Block}}^{(l)}(\cdot)$ denotes the operations within the l-th block.

After processing through all L blocks, the refined feature representation is passed to generate the final forecast $\hat{\mathbf{X}}^{t_0:t_0+\tau} \in \mathbb{R}^{M\times\tau}$, predicting the values for the future time steps.

3.3 Temporal Large-kernel Decomposed Convolution

The Temporal Large-kernel Decomposed Convolution (TLDC) module is designed to capture both short- and long-range temporal

features in multivariate time series through an efficient decomposition of large-kernel convolutions. Instead of directly applying computationally expensive large kernels, TLDC approximates their impact using a sequence of smaller, more efficient convolutions. This approach significantly reduces the computational overhead while preserving the ability to model extensive temporal patterns.

Given the input tensor $\mathbf{X}_{\text{emb}}$ in shape (M, D, N), it is first reshaped to $(M{\times}D, N)$ to facilitate group convolutions. This grouping structure divides the input into $M \times D$ separate groups, each corresponding to a unique variable-channel pair, as illustrated in Figure 3. By processing temporal dependencies independently within each group, this configuration allows for parallel computation while preserving the distinct characteristics of each variable.

To approximate a large convolution kernel of size K, the module applies a two-stage hierarchical convolutional process. First, a Depth-Wise Convolution (DW Conv) [14] with a kernel size of $(2d - 1)$ is applied, capturing short-term dependencies within a local temporal window:

$$\mathbf{X}_{\text{local}} = \text{DW Conv}(\mathbf{X}_{\text{emb}}) \tag{3}$$

This operation focuses on adjacent temporal points, enabling the capture of localized patterns while maintaining computational efficiency. The depth-wise approach ensures that each variable-channel pair is processed individually, preserving feature-specific information.

To extend the receptive field, a Depth-Wise Dilated Convolution (DW-D Conv) [12] is subsequently applied to $\mathbf{X}_{\text{local}}$. Using a kernel size of $\lceil \frac{K}{d} \rceil$ and dilation rate d, this convolution broadens the temporal scope, enabling the model to capture long-range dependencies by sampling across wider intervals:

$$\mathbf{X}_{\text{dilated}} = \text{DW-D Conv}(\mathbf{X}_{\text{local}}) \tag{4}$$

To incorporate both local and distant temporal features, the outputs of the DW Conv and DW-D Conv are combined via element-wise addition:

$$\mathbf{X}_{\text{combined}} = \mathbf{X}_{\text{dilated}} + \mathbf{X}_{\text{local}} \tag{5}$$

This summation enables the module to capture information across varying temporal scales, effectively balancing the complexity of large-kernel operations with the ability to extract detailed and broader temporal patterns.

The complexity reduction achieved by the TLDC can be quantified by comparing its parameter count and computational cost in terms of floating-point operations per second (FLOPs), to that of a direct large-kernel convolution. Typically, a single convolution with kernel size K requires:

$$\text{Params}_{\text{standard}} = (M \times D) \times (K + 1) \tag{6}$$

$$\text{FLOPs}_{\text{standard}} = 2 \times M \times D \times K \times N \tag{7}$$

In contrast, the two-stage TLDC requires:

$$\text{Params}_{\text{TLDC}} = M \times D \times (2d + 1 + \lceil \frac{K}{d} \rceil) \tag{8}$$

$$\text{FLOPs}_{\text{TLDC}} = 2 \times M \times D \times (2d - 1 + \lceil \frac{K}{d} \rceil) \times N \tag{9}$$

The reduction in complexity can be approximated by the ratios:

$$\frac{\text{Params}_{\text{TLDC}}}{\text{Params}_{\text{standard}}} = \frac{2d + 1 + \lceil \frac{K}{d} \rceil}{K + 1} \tag{10}$$

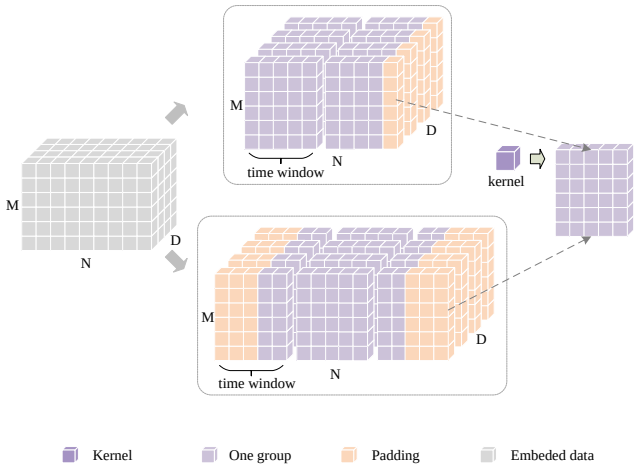


Figure 4: Illustration of the IVGC process. Initially, the data is padded to ensure divisibility by the time window size. Two padding strategies are applied: standard padding (top) and head-tail padding (bottom), creating distinct segmentations of the time dimension. Each group, represented by non-overlapping time windows, undergoes a convolution with a kernel size of 1 to capture inter-variable interactions within localized temporal patches. The outputs from both padding strategies are later aligned, merged, and further processed to produce the final integrated representation.

$$\frac{\text{FLOPs}_{\text{TLDC}}}{\text{FLOPs}_{\text{standard}}} = \frac{2d - 1 + \lceil \frac{K}{d} \rceil}{K} \tag{11}$$

When d is moderately smaller than K, these ratios simplify to approximately:

$$\frac{\text{Params}_{\text{TLDC}}}{\text{Params}_{\text{standard}}} \approx \frac{2d}{K} + \frac{1}{d}, \quad \frac{\text{FLOPs}_{\text{TLDC}}}{\text{FLOPs}_{\text{standard}}} \approx \frac{2d}{K} + \frac{1}{d} \tag{12}$$

When $\frac{2d}{K}$ is negligible for $d \ll K$, the dominant term in both cases is $O(1/d)$. This indicates that the TLDC reduces parameter and computational complexity by a factor inversely proportional to the dilation rate d. For instance, with $K = 55$ and $d = 5$, the parameter reduction factor is approximately 0.39, and the FLOPs reduction factor is around 0.36, leading to a 61% decrease in parameters and a 64% decrease in FLOPs.

Through this decomposition of large kernels, the TLDC module effectively captures long-range temporal dependencies while significantly reducing computational cost, enabling efficient temporal feature extraction. This approach is particularly advantageous for data-intensive time series analysis tasks, as it maintains model complexity within practical limits while delivering strong predictive performance and adaptability.

3.4 Inter-Variable Group Convolution

The Inter-Variable Group Convolution (IVGC) module is designed to efficiently model dependencies among variables in multivariate time series data. By convolving across structured groups of temporal patches, this module captures intricate inter-variable interactions, particularly in scenarios where variables exhibit asynchrony or

subtle lags. The group convolution framework focus on local temporal patterns and enhances computational efficiency, making it suitable for data with complex temporal dynamics and variable dependencies.

The input tensor $\mathbf{X}_{\text{combined}}$ in size (M, D, N) is first reshaped into a matrix of size $(N \times M, D)$. To enable group convolutions over multiple time windows, the temporal dimension N is padded to be divisible by the predefined window size W. This padding ensures the group convolution to operate on non-overlapping time windows, with the number of groups defined as $\frac{N_{padded}}{W}$, where N_{padded} represents the padded temporal length. Each group spans consecutive time points for all variables, allowing a shared convolutional kernel across the entire window. This configuration captures dynamic inter-variable dependencies within localized windows while preserving computational efficiency.

To ensure consistent alignment of temporal windows across groups, we calculate the necessary padding length, denoted as N_{pad1}, as follows:

$$N_{\text{pad1}} = \begin{cases} 0, & \text{if } N \equiv 0 \pmod{W} \\ W - (N \bmod W), & \text{otherwise} \end{cases} \tag{13}$$

This formula adjusts the temporal dimension to the nearest multiple of W, aligning the time windows for all groups.

To capture a diverse range of dynamic patterns, an additional head-tail padding strategy is applied. This involves shifting the window segmentation by adding padding of $\lfloor \frac{W}{2} \rfloor$ to the start of the temporal dimension and $W - \lfloor \frac{W}{2} \rfloor$ to the end. This configuration slightly offsets the time windows, expanding the convolutional receptive field and allowing the model to capture variable relationships across varying temporal shifts. The additional padding lengths are defined as:

$$\begin{aligned} N_{\text{left\_pad2}} &= \lfloor \frac{W}{2} \rfloor \\ N_{\text{right\_pad2}} &= W - \lfloor \frac{W}{2} \rfloor + N_{\text{pad1}} \end{aligned} \tag{14}$$

After applying both padding strategies, we obtain two versions of the input tensor: $\mathbf{X}_{\text{padded1}}$ for standard padding and $\mathbf{X}_{\text{padded2}}$ for head-tail padding. Each tensor undergoes a 1-dimensional group convolution along the channel dimension with a kernel size of 1, aggregating inter-variable information within each time window while preserving localized dependencies.

The group convolution for each padded tensor is formulated as follows:

$$\begin{aligned} \mathbf{Y}_{\text{padded1}} &= \text{Conv}(\mathbf{X}_{\text{padded1}}) \\ \mathbf{Y}_{\text{padded2}} &= \text{Conv}(\mathbf{X}_{\text{padded2}}) \end{aligned} \tag{15}$$

These convolutions are performed separately to capture both standard and shifted temporal patterns. The outputs, $\mathbf{Y}_{\text{padded1}}$ and $\mathbf{Y}_{\text{padded2}}$, are then aligned by removing any extra channels resulting from padding.

Subsequently, the outputs from both convolution paths are merged and processed by a final convolution to further refine the extracted temporal and inter-variable interactions:

$$\mathbf{Y} = \text{Conv}(\mathbf{Y}_{\text{padded1}} + \mathbf{Y}_{\text{padded2}}) \tag{16}$$

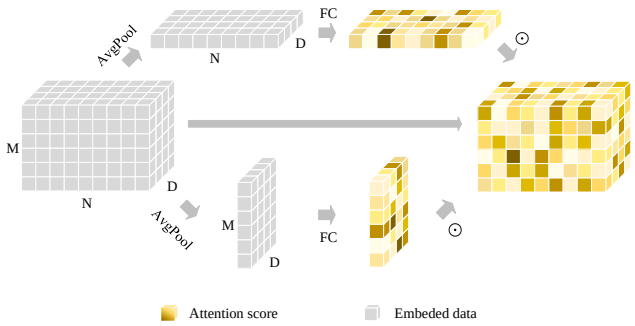


Figure 5: Illustration of the GTVA mechanism using Squeeze-and-Excitation (SE) blocks. Temporal and variable attention pathways separately generate attention weights, which are then multiplied with the convolution outputs to enhance feature learning in both dimensions.

Finally, the output tensor $\mathbf{Y}$ is reshaped back to its original dimensions (M, D, N), restoring its compatibility with subsequent model layers.

The IVGC module is based on the principle that variables are more likely to exhibit strong interactions within close temporal proximity, rather than over distant time points. By segmenting the data into localized time windows and applying group convolutions, the IVGC captures dynamic inter-variable dependencies that evolve over time, adapting efficiently to these temporal shifts. This design enhances the module's focus on relevant, time-varying relationships while minimizing computational costs by constraining the convolutional operations within compact groups, making it both scalable and effective for complex multivariate time series analysis.

3.5 Global Temporal-Variable Attention

The Global Temporal-Variable Attention (GTVA) mechanism leverages the Squeeze-and-Excitation (SE) principle [15] to selectively enhance feature learning across temporal and variable dimensions in multivariate time series. By decoupling the SE operation into dual-path attention — one focused on temporal dependencies and the other on inter - variable relationships — GTVA extends beyond standard SE applications. This design complements the convolutional layers' local feature extraction by integrating broader temporal and inter-variable contexts, which is critical for effectively modeling both long-range dependencies and complex variable interactions.

Given the input tensor $\mathbf{Y} \in \mathbb{R}^{M \times D \times N}$, the GTVA module independently generates attention weights for the temporal and variable dimensions. This dual-path approach preserves the unique characteristics of each dimension, avoiding the conflation that may arise from a single-path attention structure. By emphasizing channel-sensitive attention, the mechanism refines feature learning across these axes, further augmenting the model's ability to capture intricate temporal and inter-variable relationships.

Temporal Attention To capture global temporal dependencies, the input tensor is reshaped to $\mathbf{Y}_{\text{temp}} \in \mathbb{R}^{(N \times D) \times M}$. Global average pooling is then applied along the variable dimension to distill a temporal-centric representation:

$$\mathbf{T}_{\text{pool}} = \text{AvgPool}(\mathbf{Y}_{\text{temp}}) \in \mathbb{R}^{(N \times D)} \tag{17}$$

This representation is then processed through a two-layer fully connected (FC) network with a reduction ratio r, where r is set to balance computational cost and representation capability:

$$\mathbf{T}_{\text{atten}} = \sigma\left(W_2 \cdot \text{ReLU}(W_1 \cdot \mathbf{T}_{\text{pool}})\right) \tag{18}$$

Here, $W_1 \in \mathbb{R}^{(N\times D)\times \frac{N\times D}{r}}$ and $W_2 \in \mathbb{R}^{\frac{N\times D}{r}\times (N\times D)}$ are weight matrices, and σ denotes the sigmoid activation function. The resulting attention weights $\mathbf{T}_{\text{atten}}$, reshaped back to $(1, D, N)$, provide a temporal attention mask that selectively enhances or suppresses features along the time axis, emphasizing relevant temporal patterns in the input tensor.

Variable Attention In parallel, variable attention captures inter-variable dependencies. The tensor is reshaped to $\mathbf{Y}_{\text{var}} \in \mathbb{R}^{(M\times D)\times N}$, and global average pooling over the temporal axis produces a variable-centric representation:

$$\mathbf{V}_{\text{pool}} = \text{AvgPool}(\mathbf{Y}_{\text{var}}) \in \mathbb{R}^{(M\times D)} \tag{19}$$

This representation is similarly processed through a FC network:

$$\mathbf{V}_{\text{atten}} = \sigma\left(W_4 \cdot \text{ReLU}(W_3 \cdot \mathbf{V}_{\text{pool}})\right) \tag{20}$$

where $W_3 \in \mathbb{R}^{(M\times D)\times \frac{M\times D}{r}}$ and $W_4 \in \mathbb{R}^{\frac{M\times D}{r}\times (M\times D)}$. The attention weights $\mathbf{V}_{\text{atten}}$, reshaped to $(M, D, 1)$, allow the model to adaptively emphasize or suppress features based on cross-variable dependencies.

The dual attention weights are then combined with the convolution output using the Hadamard product, applying both temporal and variable attention simultaneously:

$$\mathbf{Y}_{\text{out}} = \sigma\left(\mathbf{T}_{\text{atten}} \odot \mathbf{V}_{\text{atten}} \odot \mathbf{Y}\right) \tag{21}$$

where $\odot$ denotes the element-wise Hadamard product. This fusion allows the model to emphasize time-sensitive and variable-relevant features concurrently, effectively capturing long-term dependencies and variable-specific interactions.

Feedback Integration To further refine feature representations, the GTVA output $\mathbf{Y}_{\text{out}}$ is incorporated with the block's input via element-wise multiplication:

$$\mathbf{X}'_{\text{emb}} = \mathbf{Y}_{\text{out}} \odot \mathbf{X}_{\text{emb}} \tag{22}$$

Here, $\mathbf{X}'_{\text{emb}}$ is the input of the next Block. This design introduces a feedback mechanism that allows each block's refined output to dynamically interact with its input, fostering progressive feature enhancement across layers. Unlike conventional residual connections, this feedback strategy cumulatively recalibrates feature representations, continually amplifying relevant information throughout the network, which is particularly beneficial for long-horizon forecasting.

The GTVA module extends traditional SE mechanisms by explicitly considering channel-level interactions within the context of both temporal and variable dimensions. Building on prior convolutions, this design refines forecasting through global temporal and variable attention, enabling precise adjustments at a comprehensive level to capture fine-grained dependencies across the entire time series.

Algorithm 1: Training of EffiCANet

Input: Training set $\mathcal{D} = \{(\mathbf{X}, \mathbf{Y})\}$; number of blocks L; historical and forecasting horizon H, τ; patch size P, stride S; large-kernel size K; dilation rate d; time window size W; reduction ratio r

Output: Trained EffiCANet

1 **Initialize** model with L blocks and patch embedding layer
2 **repeat**
3 Sample $(\mathbf{X}, \mathbf{Y})$ from $\mathcal{D}$
4 $\mathbf{X}_{\text{emb}} \leftarrow \text{PatchEmbed}(\mathbf{X}; P, S)$
5 $\mathbf{Z}^{(0)} \leftarrow \mathbf{X}_{\text{emb}}$
6 **for** $i = 1$ **to** L **do**
7 **// TLDC module**
8 $\mathbf{X}_{\text{local}} \leftarrow \text{DW Conv}(\mathbf{Z}^{(i-1)})$
9 $\mathbf{X}_{\text{dilated}} \leftarrow \text{DW-D Conv}(\mathbf{X}_{\text{local}})$
10 $\mathbf{X}_{\text{combined}} \leftarrow \mathbf{X}_{\text{dilated}} + \mathbf{X}_{\text{local}}$
11 **//IVGC module**
12 Pad $\mathbf{X}_{\text{combined}}$ to match window size W
13 $\mathbf{Y} \leftarrow \text{Group Conv}(\mathbf{X}_{\text{padded}}; W)$
14 **// GTVA module**
15 $\mathbf{T}_{\text{att}} \leftarrow \sigma(W_2 \cdot \text{ReLU}(W_1 \cdot \text{AvgPool}(\mathbf{Y}_{\text{temp}})))$
16 $\mathbf{V}_{\text{att}} \leftarrow \sigma(W_4 \cdot \text{ReLU}(W_3 \cdot \text{AvgPool}(\mathbf{Y}_{\text{var}})))$
17 $\mathbf{Y}_{\text{out}} \leftarrow \sigma(\mathbf{T}_{\text{att}} \odot \mathbf{V}_{\text{att}} \odot \mathbf{Y})$
18 Update $\mathbf{Z}^{(i)} \leftarrow \mathbf{Y}_{\text{out}} \odot \mathbf{Z}^{(i-1)}$
19 **end**
20 $\hat{\mathbf{Y}} \leftarrow \text{PredictHead}(\mathbf{Z}^{(L)})$
21 Compute loss $\mathcal{L} \leftarrow \mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})$
22 Update model parameters $\boldsymbol{\Theta}$ via backpropagation
23 **until** *convergence*;
24 **return** Trained EffiCANet

3.6 Summary

Algorithm algorithm 1 outlines the training process of EffiCANet, which incorporates three primary modules within each of the L blocks: TLDC, IVGC, and GTVA. First, the input data $\mathbf{X}$ undergoes patch embedding and is initialized as $\mathbf{Z}^{(0)}$ (lines 4-5). In each block, the temporal features are captured using DW Conv and DW-D Conv in the TLDC module (lines 7-10), while inter-variable dependencies are extracted in the IVGC module through group convolutions with a specified time window size W (lines 11-13). The GTVA module then applies temporal-variable attention pooling, producing an adaptive output that adjusts to both temporal and variable dependencies (lines 14-18). This sequence concludes with an aggregation step, and the final representation $\mathbf{Z}^{(L)}$ is passed to the prediction head for output generation (lines 20). The model parameters are iteratively optimized until convergence by minimizing the prediction loss $\mathcal{L}$ (lines 21-22).

4 EXPERIMENTS

4.1 Experimental Setup

4.1.1 Datasets and Evaluation Metrics. We evaluate EffiCANet's performance across nine widely-adopted multivariate time series datasets.

- **ETT:** The Electricity Transformer Temperature (ETT) dataset [52] contains transformer oil temperature data, including four subsets — ETTm1, ETTm2, ETTh1, and ETTh2, each representing different temporal granularities and historical spans.
- **Electricity:** The Electricity dataset [43] consists of hourly electricity consumption data from 321 households.
- **Weather:** The Weather dataset [43] includes meteorological variables such as temperature, humidity, wind speed, and pressure, collected from multiple weather stations.
- **Traffic:** The Traffic dataset [43] contains hourly occupancy rates of roadways in the San Francisco Bay Area.
- **Exchange:** The Exchange dataset [18] records daily exchange rates of eight major world currencies.
- **ILI:** The Influenza-like Illness (ILI) dataset [43] tracks the weekly percentage of doctor visits for influenza-like illnesses in the United States.

Table 1 provides a detailed overview of each dataset, including the number of timesteps, features, and sampling frequency.

Table 1: Summary of each dataset.

Datasets	Timesteps	Features	Frequency
ETTh1	17420	7	Hourly
ETTh2	17420	7	Hourly
ETTm1	69680	7	15 mins
ETTm2	69680	7	15 mins
Electricity	26304	321	Hourly
Weather	52696	21	10 mins
Traffic	17544	862	Hourly
Exchange	7207	8	Daily
ILI	966	7	Weekly

For evaluation, we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as performance metrics, where lower values reflect better forecasting accuracy.

4.1.2 Baselines. We evaluate our model against a range of baselines across three categories: convolution-based, Transformer-based, and MLP-based models.

- **Convolution-based Baselines.** ConvTimeNet [3] utilizes deep-wise and pointwise convolutions to capture both global sequence and cross-variable dependence. ModernTCN [27] modernizes the traditional TCN by expanding the receptive field and offers a pure convolution structure for time series analysis. TimesNet [42] focuses on multi-scale temporal feature extraction through specialized convolutions, while MICN [39] integrates information across various time scales for improved prediction accuracy.
- **Transformer-based Baselines.** PatchTST [31] introduces a patching mechanism with channel-independence, and Crossformer [47] applies cross-dimensional attention to capture both temporal and inter-variable dependencies.
- **MLP-based Baselines.** MTS-mixer [22] replaces attention mechanisms with factorized MLP modules to separately model temporal and feature dependencies. DLinear [46], RLinear, RMLP [21] demonstrate the effectiveness of simple linear layers and MLP structures in long-term forecasting.

4.1.3 Implementation Details. The experiments were implemented on an NVIDIA GeForce RTX 4090 GPU. We used the Adam optimizer with learning rates ranging from $1e^{-4}$ to $1e^{-2}$, adjusting for each dataset. The batch sizes were set to 32, 128, 256, or 512 depending on the dataset size. The model consisted of 1 to 3 blocks, with training running up to 100 epochs and early stopping triggered if validation loss stagnated for 20 epochs. A channel dimension size of 64 was employed consistently throughout all experiments. The simulated temporal large-kernel convolution size was set to 55 with a dilation rate of 5. Input sequence lengths were optimized per dataset, with values of 128, 336, 512, 720, and 960. Baselines followed the original configurations to ensure fair comparisons.

4.2 Main Results

In this study, we evaluated EffiCANet against a comprehensive set of baseline models across nine datasets, using prediction lengths of 24, 36, 48, 60 for the ILI dataset, and 96, 192, 336, 720 for the others. Each model was evaluated under optimized conditions, ensuring fair and accurate comparisons across different methodologies in time series forecasting.

EffiCANet consistently delivered strong results, as summarized in Table 2. Across 72 experimental evaluations, it ranked first in 51 cases and second in 13, demonstrating superior performance in most scenarios. For instance, on the ETTh2 dataset, EffiCANet achieved a 4.7% reduction in MSE and a 2.53% reduction in MAE compared to the second-best model. Similarly, on the ILI dataset, it achieved a 10.02% reduction in MSE and a 3.34% reduction in MAE. These consistent gains underscore the model's robustness and adaptability.

Analyzing the baseline models, convolution-based approaches like ModernTCN excel on datasets with shorter sequences or periodic patterns, such as Exchange and ILI, where local temporal dependencies dominate. However, they struggle with longer sequences due to limited capacity in capturing extended dependencies. Transformer-based models like PatchTST perform well on datasets with more variables or longer sequences, such as Traffic and Electricity, owing to their capability to model intricate interactions and long-term dependencies, but may underperform with irregular or shorter time series. MLP-based models, like RLinear, show efficiency on simpler datasets with fewer variables, such as the ETT series. However, they tend to fall short when dealing with complex temporal patterns and inter-variable relationships, particularly in datasets with a high number of variables, such as Traffic.

EffiCANet outperforms these models by effectively capturing both long-range dependencies and short-term dynamics through large-kernel decomposed convolutions. The inter-variable group convolution further optimizes variable interactions over close time intervals, while global temporal-variable attention dynamically focuses on relevant features. This holistic design allows EffiCANet to excel in scenarios requiring nuanced analysis of both temporal and cross-variable relationships.

Table 2: Performance comparison of our method against various baseline models across different datasets and prediction lengths. Red indicates the best performance, and blue indicates the second-best performance.

Models		EffiCANet		ConvTimeNet (2024)		ModernTCN (2024)		PatchTST (2023)		Crossformer (2023)		MTS-mixer (2023)		DLinear (2023)		TimesNet (2023)		MICN (2023)		RLinear (2023)		RMLP (2023)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.288**	**0.337**	0.292	0.345	0.292	0.346	0.290	0.342	0.316	0.373	0.314	0.358	0.299	0.343	0.338	0.375	0.314	0.360	0.301	0.342	0.298	0.345
	192	0.331	0.364	**0.329**	0.368	0.332	0.368	0.332	0.369	0.377	0.411	0.354	0.386	0.335	0.365	0.371	0.387	0.359	0.387	0.335	**0.363**	0.344	0.375
	336	**0.362**	0.384	0.363	0.390	0.365	0.391	0.366	0.392	0.431	0.442	0.384	0.405	0.369	0.386	0.410	0.411	0.398	0.413	0.370	**0.383**	0.390	0.410
	720	**0.407**	**0.413**	0.427	0.428	0.416	0.417	0.416	0.420	0.600	0.547	0.427	0.432	0.425	0.421	0.478	0.450	0.459	0.464	0.425	0.414	0.445	0.441
	Avg	**0.347**	**0.375**	0.353	0.383	0.351	0.381	0.351	0.381	0.431	0.443	0.370	0.395	0.357	0.379	0.400	0.406	0.383	0.406	0.358	0.376	0.369	0.393
ETTm2	96	0.165	0.254	0.167	0.257	0.166	0.256	0.165	0.255	0.296	0.352	0.177	0.259	0.167	0.260	0.187	0.267	0.178	0.273	**0.164**	**0.253**	0.174	0.259
	192	**0.217**	**0.289**	0.222	0.295	0.222	0.293	0.220	0.292	0.342	0.385	0.241	0.303	0.224	0.303	0.249	0.309	0.245	0.316	0.219	0.290	0.236	0.303
	336	**0.271**	0.327	0.276	0.329	0.272	**0.324**	0.274	0.329	0.410	0.425	0.297	0.338	0.281	0.342	0.321	0.351	0.295	0.350	0.273	0.326	0.291	0.338
	720	**0.340**	**0.377**	0.358	0.381	0.351	0.381	0.362	0.385	0.563	0.538	0.396	0.398	0.397	0.421	0.497	0.403	0.389	0.406	0.366	0.385	0.371	0.391
	Avg	**0.248**	**0.312**	0.256	0.316	0.253	0.314	0.255	0.315	0.402	0.425	0.277	0.325	0.267	0.332	0.291	0.333	0.277	0.336	0.256	0.314	0.268	0.322
ETTh1	96	**0.361**	**0.391**	0.368	0.394	0.368	0.394	0.370	0.399	0.386	0.429	0.372	0.395	0.375	0.399	0.384	0.402	0.396	0.427	0.366	**0.391**	0.390	0.410
	192	**0.399**	**0.412**	0.406	0.414	0.405	0.413	0.413	0.421	0.419	0.444	0.416	0.426	0.405	0.416	0.557	0.436	0.430	0.453	0.404	**0.412**	0.430	0.432
	336	**0.374**	**0.407**	0.405	0.420	0.391	0.412	0.422	0.436	0.440	0.461	0.455	0.449	0.439	0.443	0.491	0.469	0.433	0.458	0.420	0.423	0.441	0.441
	720	**0.415**	**0.444**	0.442	0.457	0.450	0.461	0.447	0.466	0.519	0.524	0.475	0.472	0.472	0.490	0.521	0.500	0.474	0.508	0.442	0.456	0.506	0.495
	Avg	**0.387**	**0.414**	0.405	0.421	0.404	0.420	0.413	0.431	0.441	0.465	0.430	0.436	0.423	0.437	0.458	0.450	0.433	0.462	0.408	0.421	0.442	0.445
ETTh2	96	**0.251**	**0.323**	0.264	0.330	0.263	0.332	0.274	0.336	0.383	0.420	0.307	0.354	0.289	0.353	0.340	0.374	0.289	0.357	0.262	0.331	0.288	0.352
	192	**0.299**	**0.362**	0.316	0.368	0.320	0.374	0.339	0.379	0.421	0.450	0.374	0.399	0.383	0.418	0.402	0.414	0.409	0.438	0.320	0.374	0.343	0.387
	336	**0.295**	**0.359**	0.315	0.378	0.313	0.376	0.329	0.380	0.449	0.459	0.398	0.432	0.448	0.465	0.452	0.452	0.417	0.452	0.325	0.386	0.353	0.402
	720	**0.372**	**0.419**	0.382	0.425	0.392	0.433	0.379	0.422	0.472	0.497	0.463	0.465	0.605	0.551	0.462	0.468	0.426	0.473	**0.372**	0.421	0.410	0.440
	Avg	**0.304**	**0.366**	0.319	0.375	0.322	0.379	0.330	0.379	0.431	0.457	0.386	0.413	0.431	0.447	0.414	0.427	0.385	0.430	0.320	0.378	0.349	0.395
Electricity	96	**0.129**	0.226	0.132	0.226	**0.129**	0.226	**0.129**	**0.222**	0.187	0.283	0.141	0.243	0.153	0.237	0.168	0.272	0.159	0.267	0.140	0.235	**0.129**	0.224
	192	0.145	0.241	0.148	0.241	**0.143**	**0.239**	0.147	0.240	0.258	0.330	0.163	0.261	0.152	0.249	0.184	0.289	0.168	0.279	0.154	0.248	0.147	0.240
	336	**0.160**	**0.257**	0.165	0.259	0.161	0.259	0.163	0.259	0.323	0.369	0.176	0.277	0.169	0.267	0.198	0.300	0.196	0.308	0.171	0.264	0.164	**0.257**
	720	**0.187**	**0.285**	0.205	0.293	0.191	0.286	0.197	0.290	0.404	0.423	0.212	0.308	0.233	0.344	0.220	0.320	0.203	0.312	0.209	0.297	0.203	0.291
	Avg	**0.156**	**0.252**	0.163	0.255	**0.156**	0.253	0.159	0.253	0.293	0.351	0.173	0.272	0.177	0.274	0.192	0.295	0.182	0.292	0.169	0.261	0.161	0.253
Weather	96	**0.145**	**0.197**	0.155	0.205	0.149	0.200	0.149	0.198	0.153	0.217	0.156	0.206	0.152	0.237	0.172	0.220	0.161	0.226	0.175	0.225	0.149	0.202
	192	**0.189**	**0.239**	0.200	0.249	0.196	0.245	0.194	0.241	0.197	0.269	0.199	0.248	0.220	0.282	0.219	0.261	0.220	0.283	0.218	0.260	0.194	0.242
	336	**0.234**	0.279	0.252	0.287	0.238	**0.277**	0.245	0.282	0.252	0.311	0.249	0.291	0.265	0.319	0.280	0.306	0.275	0.328	0.265	0.294	0.243	0.282
	720	**0.309**	**0.331**	0.321	0.335	0.314	0.334	0.314	0.334	0.318	0.363	0.336	0.343	0.323	0.362	0.365	0.359	0.311	0.356	0.329	0.339	0.316	0.333
	Avg	**0.219**	**0.262**	0.232	0.269	0.224	0.264	0.226	0.264	0.230	0.290	0.235	0.272	0.240	0.300	0.259	0.287	0.242	0.298	0.247	0.279	0.225	0.265
Traffic	96	**0.354**	0.261	0.376	0.265	0.368	0.253	0.360	**0.249**	0.512	0.290	0.462	0.332	0.410	0.282	0.593	0.321	0.508	0.301	0.496	0.375	0.430	0.327
	192	**0.371**	0.269	0.392	0.271	0.379	0.261	0.379	**0.256**	0.523	0.297	0.488	0.354	0.423	0.287	0.617	0.336	0.536	0.315	0.503	0.377	0.451	0.340
	336	**0.388**	0.279	0.405	0.277	0.397	0.270	0.392	**0.264**	0.530	0.300	0.498	0.360	0.436	0.296	0.629	0.336	0.525	0.310	0.517	0.382	0.470	0.351
	720	0.435	0.301	0.436	0.294	0.440	0.296	**0.432**	**0.286**	0.573	0.313	0.529	0.370	0.466	0.315	0.640	0.350	0.571	0.323	0.555	0.398	0.513	0.372
	Avg	**0.387**	0.278	0.402	0.277	0.396	0.270	0.391	**0.264**	0.535	0.300	0.494	0.354	0.434	0.295	0.620	0.336	0.535	0.312	0.518	0.383	0.466	0.348
Exchange	96	**0.080**	**0.196**	0.086	0.204	**0.080**	**0.196**	0.093	0.214	0.186	0.346	0.083	0.201	0.081	0.203	0.107	0.234	0.102	0.235	0.083	0.201	0.083	0.201
	192	0.166	**0.288**	0.184	0.303	0.166	**0.288**	0.192	0.312	0.467	0.522	0.174	0.296	**0.157**	0.293	0.226	0.344	0.172	0.316	0.170	0.293	0.170	0.292
	336	0.305	**0.396**	0.341	0.421	0.307	0.398	0.350	0.432	0.783	0.721	0.336	0.417	0.305	0.414	0.367	0.448	**0.272**	0.407	0.309	0.401	0.309	0.401
	720	**0.636**	**0.569**	0.879	0.701	0.656	0.582	0.911	0.716	1.367	0.943	0.900	0.715	0.643	0.601	0.964	0.746	0.714	0.658	0.817	0.680	0.816	0.680
	Avg	**0.297**	**0.362**	0.373	0.407	0.302	0.366	0.387	0.419	0.701	0.633	0.373	0.407	**0.297**	0.378	0.416	0.443	0.315	0.404	0.345	0.394	0.345	0.394
ILI	24	**1.263**	0.742	1.469	0.800	1.347	**0.717**	1.319	0.754	3.040	1.186	1.472	0.798	2.215	1.081	2.317	0.934	2.684	1.112	4.337	1.507	4.445	1.536
	36	**1.203**	**0.703**	1.450	0.845	1.250	0.778	1.430	0.834	3.356	1.230	1.435	0.745	1.963	0.963	1.972	0.920	2.507	1.013	4.205	1.481	4.409	1.519
	48	**1.227**	**0.763**	1.572	0.875	1.388	0.781	1.553	0.815	3.441	1.223	1.474	0.822	2.130	1.024	2.238	0.940	2.423	1.012	4.257	1.484	4.388	1.507
	60	1.490	0.831	1.741	0.920	1.774	0.868	**1.470**	**0.788**	3.608	1.302	1.839	0.912	2.368	1.096	2.027	0.928	2.653	1.085	4.278	1.487	4.306	1.502
	Avg	**1.296**	**0.760**	1.558	0.860	1.440	0.786	1.443	0.798	3.361	1.235	1.555	0.819	2.169	1.041	2.139	0.931	2.567	1.055	4.269	1.490	4.387	1.516
1st Count		**52**		1		9		9		0		0		1		0		1		7		2	

Table 3: Ablation study of removing temporal and variable-specific modules. The best results are shown in bold.

Dataset	ETTm2		ETTh2		Weather		Traffic		ILI	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Ours	**0.248**	**0.312**	**0.304**	**0.366**	**0.219**	**0.262**	**0.387**	**0.278**	**1.296**	**0.760**
w/o TD	0.251	0.315	0.307	0.371	0.225	0.268	0.417	0.297	1.664	0.857
w/o VD	0.249	0.314	0.307	0.368	0.223	0.263	0.407	0.283	1.600	0.807
w TLDC	0.250	0.316	0.308	0.369	0.228	0.263	0.409	0.285	1.671	0.820
w IVGC	0.256	0.318	0.317	0.372	0.230	0.274	0.418	0.299	2.202	0.977

Table 4: Ablation study comparing the performance and computational efficiency between our TLDC and the standard TLC. Green highlights denote instances where TLDC outperforms TLC, while blue highlights indicate comparable performance.

Method	TLDC				TLC			
Metric	MSE	MAE	FLOPs	params	MSE	MAE	FLOPs	params
ETTh1	0.387	0.414	0.4666	0.5250	0.388	0.414	0.6322	0.5447
ECL	0.156	0.252	0.6114	1.8452	0.156	0.252	0.8283	2.7491
Weather	0.219	0.262	0.0213	0.8114	0.219	0.260	0.0289	0.8706
Exchange	0.302	0.366	0.0240	0.0341	0.302	0.366	0.0330	0.0567

4.3 Ablation Study

To evaluate the contributions of key components in our model, we conducted an ablation study, as shown in Table 3 and Table 4, across four datasets: ETTh1, Electricity, Weather, and Exchange. The analysis covers two aspects: the effect of removing temporal and variable-specific modules and a comparison between large-kernel decomposed convolution (TLDC) and standard large-kernel convolution (LKC). Results are averaged across four prediction horizons for a comprehensive assessment.

In Table 3, we analyze the impact of removing key modules. Replacing the temporal-specific components, including the TLDC and global temporal attention, with standard convolution (w/o TD) results in a noticeable performance drop across all datasets, underscoring the importance of capturing temporal dependencies. Similarly, when the variable-specific components, including the IVGC and global variable attention, are replaced by standard convolution (w/o VD), the model's performance degrades, particularly on datasets with numerous correlated variables like Traffic, highlighting the need for cross-variable interaction modeling.

We then test the effect of using only the TLDC or IVGC modules. While the TLDC alone maintains some stability in temporally dependent datasets, it cannot fully compensate for the lack of variable interactions. Conversely, using only the IVGC leads to a sharper performance decline, demonstrating the interdependence of temporal and variable modeling, especially in complex datasets. This also emphasizes the role of global attention, which enhances both temporal and variable convolution by refining their dependencies.

In Table 4, we compare the TLDC with the standard LKC. For this comparison, we calculated FLOPs (in Gigaflops) and parameter counts (in Millions) using batch sizes of 100 for the ETTh1 and Exchange datasets, and 1 for the ECL and Weather datasets. The prediction accuracy between the two methods remains nearly identical, with deviations in MAE and MSE within 0.8%. However, the computational advantages of TLDC are substantial. For instance, on the ECL dataset, TLDC reduces parameters by 32.9% and FLOPs by 26.2%. This highlights the efficiency of TLDC, especially in resource-constrained environments where computational costs are crucial. The slight accuracy trade-off is offset by significant improvements in efficiency, making TLDC the superior choice for balancing performance and deployability.

4.4 Model Efficiency

We evaluate the computational efficiency and prediction performance of EffiCANet against various baselines on three datasets: ETTm1 with a prediction horizon of 96, and ETTh2 and Weather with horizons of 336. The evaluation considers three key metrics: FLOPs, parameter count, and MSE. As shown in Figure 6, the results are presented using logarithmic scales for FLOPs and parameters to clearly visualize computational differences.

MLP-based models, including DLinear, RLinear, RMLP, and MTS-Mixer, exhibit extremely low FLOPs and parameter counts. While DLinear and RLinear achieve relatively competitive MSE on simpler datasets like ETTm1, their performance declines significantly on more complex datasets with higher variable counts, such as the Weather dataset. Other MLP-based models similarly underperform compared to more advanced architectures, underscoring their limitations in capturing complex inter-variable dependencies, which is crucial for multivariate time series forecasting.

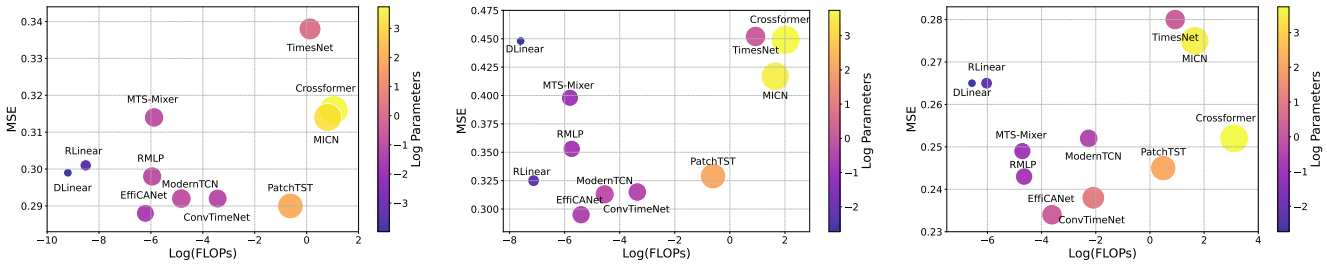
Transformer-based models, such as PatchTST and Crossformer, show a substantial increase in both FLOPs and parameters, reflecting their more complex architectures. However, this complexity does not always lead to better performance. For instance, Crossformer has high computational demands but produces relatively less accurate predictions, suggesting that its architecture may be over-parameterized for the given task. PatchTST demonstrates a better balance between complexity and performance but still suffers from high computational costs.

Convolution-based models, such as ModernTCN, ConvTimeNet, and EffiCANet, achieve a more favorable balance between efficiency and accuracy. Notably, EffiCANet achieves the lowest MSE among all models while maintaining a significantly lower computational footprint compared to Transformer models. This efficiency makes our approach particularly advantageous for real-time applications or scenarios with limited computational resources. Other convolutional models, like ModernTCN and ConvTimeNet, also show strong performance with relatively low FLOPs, further underscoring the suitability of convolutional architectures for multivariate forecasting tasks.

4.5 Parameter Sensitivity

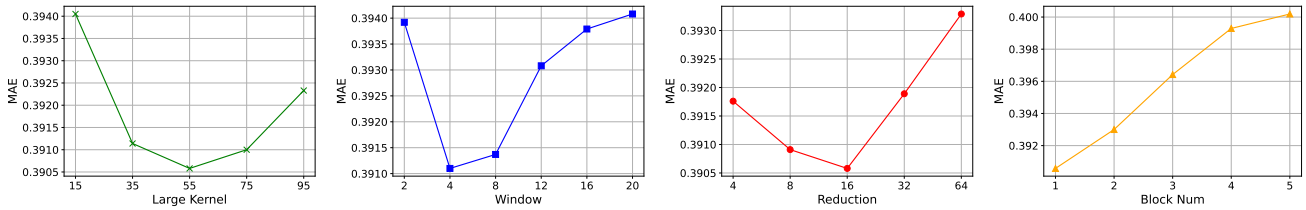
We perform parameter sensitivity analysis on the ETTh1 dataset with a prediction horizon of 96, using MAE as the evaluation metric. The focus is on four key parameters: the large-kernel size K and its corresponding dilation rate d in the TLDC module, the time window size W in the IVGC module, the reduction ratio r in the GTVA module, and the number of blocks L in the model's architecture.

For the large-kernel size, as shown in Figure 7a, we vary the K from 15 to 95, with d set accordingly to 1, 3, 5, 7, and 9. The



(a) ETTm1 dataset (b) ETTh2 dataset (c) Weather dataset

Figure 6: Comparison of model efficiency on the ETTm1 dataset. The x-axis represents log-transformed FLOPs, while the y-axis shows MSE. Bubble size and color indicate the log-transformed number of parameters.



(a) MAE vs Large Kernel (b) MAE vs Window (c) MAE vs Reduction (d) MAE vs Block Numbers

Figure 7: Parameter Sensitivity.

MAE decreases as K increases, reaching its minimum at $K = 55$. Beyond this, performance declines, likely due to over-smoothing of the temporal features. Figure 7b explores the time window size, revealing that $W = 4$ is optimal. Smaller windows may miss temporal dependencies, while larger windows appear to introduce noise or redundancy, resulting in diminished performance.

In Figure 7c, we find that a moderate $r = 16$ provides the best balance, suggesting it effectively reduces dimensionality without losing key information. Both lower and higher ratios degrade performance, either due to retaining too much irrelevant information or over-compressing the feature space. Figure 7d shows that using more than one block leads to overfitting, with minimal accuracy gains. A single block captures core dependencies, and limiting block count ensures model simplicity and better generalization. While these results provide optimal settings for the ETTh1 dataset, different datasets may require adjustments.

4.6 Model Scalability

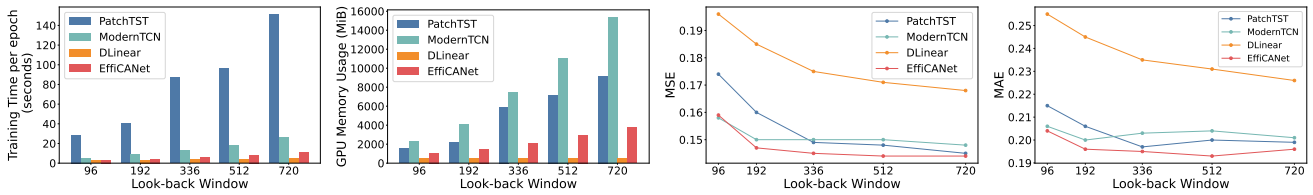
We evaluate EffiCANet's scalability on the Weather dataset with sequence lengths of 96, 192, 336, 512, and 720, and a fixed prediction horizon of 96. We compare EffiCANet to three baseline models — ModernTCN, PatchTST, and DLinear — representing convolutional, Transformer, and linear architectures. The comparison includes training time per epoch, GPU memory usage, MSE, and MAE.

In Figure 8a and Figure 8b, EffiCANet consistently achieves shorter training times and lower GPU memory usage than PatchTST and ModernTCN, with the gap widening as sequence length increases. DLinear, while resource-efficient, exhibits higher error rates, highlighting its limitations in handling complex temporal dependencies. Figure 8c and Figure 8d show the MSE and MAE across varying sequence lengths. EffiCANet maintains low and stable error rates as sequence length increases, outperforming the baselines in predictive accuracy. PatchTST also shows improving accuracy with longer sequences but at a significantly higher computational cost. Overall, these results demonstrate EffiCANet's balance of computational efficiency and accuracy across different input lengths.

4.7 Visualization

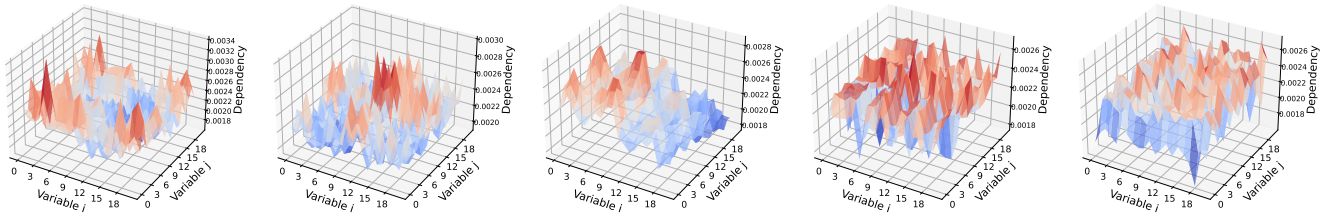
4.7.1 Visualization of variable dependency. To interpret the inter-variable dependencies captured by EffiCANet, we conducted a visualization analysis on the Weather dataset with a prediction horizon of 96. Using convolutional weights from the IVGC module across five consecutive time windows, we computed dependency values that represent relationships between variables within each window. Specifically, the product of convolutional weights for each variable pair reflects their mutual influence during the convolution operation. Summing these products provides an aggregated measure of inter-variable dependency, capturing how variables interact over time. Each subplot in Figure 9 illustrates the dependency distribution across variables within a specific time window, providing insights into how interactions evolve dynamically.

The visualizations reveal that inter-variable dependencies fluctuate across time windows, indicating dynamic relationships that change with recent history. For example, we observe strong correlations between certain variable pairs in earlier time windows, likely reflecting short-term interactions. As time progresses, these relationships evolve, with some weakening while others emerging, revealing complex temporal patterns. This behavior mirrors real-world weather dynamics, where factors like temperature, humidity, and wind speed exhibit variable interdependence across time.



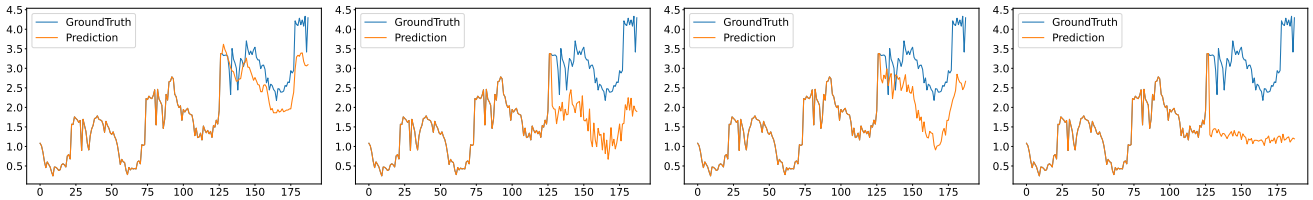
(a) Training time vs H (b) GPU memory vs H (c) MSE vs H (d) MAE vs H

Figure 8: Model Scalability across difference look-back windows.



(a) Time window 1 (b) Time window 2 (c) Time window 3 (d) Time window 4 (e) Time window 5

Figure 9: Variable dependency across various time windows.



(a) EffiCANet (b) ModernTCN (c) PatchTST (d) DLinear

Figure 10: Forecasting visualization comparison on ILI dataset.

The group convolutions in the IVGC module are key to this adaptive modeling, as they localize dependency modeling within small, time-varying groups of variables. Unlike conventional approaches that assume static dependencies, our model dynamically responds to changes in variable interactions across time windows, effectively capturing both transient and persistent dependencies. This adaptability improves the model's ability to handle time-sensitive dependencies, contributing to more accurate predictions.

4.7.2 Visualization of forecasting results. We visualize the forecasting performance of EffiCANet and three baseline models — ModernTCN, PatchTST, and DLinear — on the ILI dataset with a prediction horizon of 60 timesteps. This assessment aims to assess each model's ability to capture trends and fluctuations relative to the ground truth.

As shown in Figure 10, EffiCANet closely aligns with the ground truth, accurately capturing both smooth and abrupt changes in the data, especially during periods of rapid variation. While ModernTCN and PatchTST follow the general trends, they struggle to adapt to short-term fluctuations, resulting in underestimation of key peaks and troughs. DLinear exhibits the weakest performance, failing to capture key patterns and over-smoothing most of the predicted values.

5 CONCLUSION

In this work, we propose a novel approach for multivariate time series forecasting through three specialized modules that effectively capture both temporal and inter-variable dependencies. The Temporal Large-kernel Decomposed Convolution (TLDC) module efficiently models short- and long-term temporal dependencies by decomposing large kernels, enabling scalable handling of extended sequences. The Inter-Variable Group Convolution (IVGC) module adapts to dynamic inter-variable relationships by learning localized interactions within time windows, capturing critical time-sensitive dependencies. The Global Temporal-Variable Attention (GTVA) module further enhances the model by jointly attending to temporal and variable dimensions, providing a refined contextual understanding that improves both forecasting accuracy and adaptability to complex patterns. Future work will focus on optimizing the model for non-stationary time series and enhancing interpretability to broaden its application to complex, dynamic systems.

REFERENCES

[1] Peng Chen, Yingying ZHANG, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. 2023. Pathformer: Multi-scale Transformers with Adaptive Pathways for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

[2] Jiezhu Cheng, Kaizhu Huang, and Zibin Zheng. 2020. Towards better forecasting by fusing near and distant future visions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3593–3600.

[3] Mingyue Cheng, Jiqian Yang, Tingyue Pan, Qi Liu, and Zhi Li. 2024. Convtimenet: A deep hierarchical fully convolutional model for multivariate time series analysis. *arXiv preprint arXiv:2403.01493* (2024).

[4] Yunyao Cheng, Peng Chen, Chenjuan Guo, Kai Zhao, Qingsong Wen, Bin Yang, and Christian S Jensen. 2023. Weakly guided adaptation for robust time series forecasting. *Proceedings of the VLDB Endowment* 17, 4 (2023), 766–779.

[5] Jia Ding, Yuxuan Zhao, and Junyang Jin. 2023. Forecasting natural gas consumption with multiple seasonal patterns. *Applied Energy* 337 (2023), 120911.

[6] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. 2022. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11963–11975.

[7] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, and Xiaoli Li. [n.d.]. TSLANet: Rethinking Transformers for Time Series Representation Learning. In *Forty-first International Conference on Machine Learning*.

[8] Chenyou Fan, Yuze Zhang, Yi Pan, Xiaoyue Li, Chi Zhang, Rong Yuan, Di Wu, Wensheng Wang, Jian Pei, and Heng Huang. 2019. Multi-horizon time series forecasting with temporal attention learning. In *Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery & data mining*. 2527–2535.

[9] Jin Fan, Wenchao Weng, Hao Tian, Huifeng Wu, Fu Zhu, and Jia Wu. 2024. RGDAN: A random graph diffusion attention network for traffic prediction. *Neural networks* 172 (2024), 106093.

[10] Mojtaba A Farahani, MR McCormick, Ramy Harik, and Thorsten Wuest. 2025. Time-series classification in smart manufacturing systems: An experimental evaluation of state-of-the-art machine learning algorithms. *Robotics and Computer-Integrated Manufacturing* 91 (2025), 102839.

[11] Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752* (2023).

[12] Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. 2023. Visual attention network. *Computational Visual Media* 9, 4 (2023), 733–752.

[13] Pradeep Hewage, Ardhendu Behera, Marcello Trovati, Ella Pereira, Morteza Ghahremani, Francesco Palmieri, and Yonghuai Liu. 2020. Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing* 24 (2020), 16453–16482.

[14] Andrew G Howard. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[15] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[16] Romain Ilbert, Ambroise Odonnat, Vasilii Feofanov, Aladin Virmaux, Giuseppe Paolo, Themis Palpanas, and Ievgen Redko. [n.d.]. SAMformer: Unlocking the Potential of Transformers in Time Series Forecasting with Sharpness-Aware Minimization and Channel-Wise Attention. In *Forty-first International Conference on Machine Learning*.

[17] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

[18] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 95–104.

[19] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. 2017. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 156–165.

[20] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems* 32 (2019).

[21] Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu. 2023. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv preprint arXiv:2305.10721* (2023).

[22] Zhe Li, Zhongwen Rao, Lujia Pan, and Zenglin Xu. 2023. Mts-mixers: Multivariate time series forecasting via factorized temporal and channel mixing. *arXiv preprint arXiv:2302.04501* (2023).

[23] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. 2022. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems* 35 (2022), 5816–5828.

[24] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.

[25] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. [n.d.]. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

[26] Ioannis E Livieris, Emmanuel Pintelas, and Panagiotis Pintelas. 2020. A CNN–LSTM model for gold price time-series forecasting. *Neural computing and applications* 32 (2020), 17351–17360.

[27] Donghao Luo and Xue Wang. 2024. Moderntcn: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*.

[28] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*. 116–131.

[29] Shusen Ma, Yu Kang, Peng Bai, and Yun-Bo Zhao. 2024. Fmamba: Mamba based on fast-attention for multivariate time-series forecasting. *arXiv preprint arXiv:2407.14814* (2024).

[30] Sidra Mehtab and Jaydip Sen. 2022. Analysis and forecasting of financial time series using CNN and LSTM-based deep learning models. In *Advances in Distributed Computing and Machine Learning: Proceedings of ICADCML 2021*. Springer, 405–423.

[31] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. [n.d.]. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *The Eleventh International Conference on Learning Representations*.

[32] Francesco Piccialli, Fabio Giampaolo, Edoardo Prezioso, David Camacho, and Giovanni Acampora. 2021. Artificial intelligence and healthcare: Forecasting of medical bookings through multi-source time-series fusion. *Information Fusion* 74 (2021), 1–16.

[33] Dalin Qin, Guobing Liu, Zengxiang Li, Weicheng Guan, Shubao Zhao, and Yi Wang. 2023. Federated deep contrastive learning for mid-term natural gas demand forecasting. *Applied Energy* 347 (2023), 121503.

[34] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. [n.d.]. Scaling to Billion Parameters for Time Series Foundation Models with Mixture of Experts. In *NeurIPS Workshop on Time Series in the Age of Large Models*.

[35] Emmanouil Sylligardos, Paul Boniol, John Paparrizos, Panos Trahanias, and Themis Palpanas. 2023. Choose wisely: An extensive evaluation of model selection for anomaly detection in time series. *Proceedings of the VLDB Endowment* 16, 11 (2023), 3418–3432.

[36] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[37] Jessica Walther and Matthias Weigold. 2021. A systematic review on predicting and forecasting the electrical energy consumption in the manufacturing industry. *Energies* 14, 4 (2021), 968.

[38] Renzhuo Wan, Shuping Mei, Jun Wang, Min Liu, and Fan Yang. 2019. Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics* 8, 8 (2019), 876.

[39] Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. 2023. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*.

[40] Aji Prasetya Wibawa, Agung Bella Putra Utama, Hakkun Elmunsyah, Utomo Pujianto, Felix Andika Dwiyanto, and Leonel Hernandez. 2022. Time-series analysis with smoothed Convolutional Neural Network. *Journal of big Data* 9, 1 (2022), 44.

[41] Harya Widiputra, Adele Mailangkay, and Elliana Gautama. 2021. Multivariate CNN-LSTM Model for Multiple Parallel Financial Time-Series Prediction. *Complexity* 2021, 1 (2021), 9903518.

[42] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. [n.d.]. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *The Eleventh International Conference on Learning Representations*.

[43] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems* 34 (2021), 22419–22430.

[44] Haixu Wu, Hang Zhou, Mingsheng Long, and Jianmin Wang. 2023. Interpretable weather forecasting for worldwide stations with a unified deep model. *Nature Machine Intelligence* 5, 6 (2023), 602–611.

[45] Zhijian Xu, Ailing Zeng, and Qiang Xu. [n.d.]. FITS: Modeling Time Series with $10k$ Parameters. In *The Twelfth International Conference on Learning Representations*.

[46] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting?. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 11121–11128.

[47] Yunhao Zhang and Junchi Yan. 2023. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh*

international conference on learning representations.

[48] Ziyi Zhang, Shaogang Ren, Xiaoning Qian, and Nick Duffield. 2024. Towards invariant time series forecasting in smart cities. In *Companion Proceedings of the ACM on Web Conference 2024*. 1344–1350.

[49] Kai Zhao, Chenjuan Guo, Yunyao Cheng, Peng Han, Miao Zhang, and Bin Yang. 2023. Multiple time series forecasting with dynamic graph modeling. *Proceedings of the VLDB Endowment* 17, 4 (2023), 753–765.

[50] Shubao Zhao, Ming Jin, Zhaoxiang Hou, Chengyi Yang, Zengxiang Li, Qingsong Wen, and Yi Wang. 2024. HiMTM: Hierarchical Multi-Scale Masked Time Series Modeling with Self-Distillation for Long-Term Forecasting. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 3352–3362.

[51] Shubao Zhao, Xinxing Zhou, Ming Jin, Zhaoxiang Hou, Chengyi Yang, Zengxiang Li, Qingsong Wen, Yi Wang, Yanlong Wen, and Xiaojie Yuan. 2024. Rethinking self-supervised learning for time series forecasting: A temporal perspective. *Knowledge-Based Systems* (2024), 112652.

[52] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.

[53] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*. PMLR, 27268–27286.