MODERNTCN: A MODERN PURE CONVOLUTION STRUCTURE FOR GENERAL TIME SERIES ANALYSIS

Donghao Luo, Xue Wang
Department of Precision Instrument, Tsinghua University, Beijing 100084, China
ldh21@mails.tsinghua.edu.cn, wangxue@mail.tsinghua.edu.cn

ABSTRACT

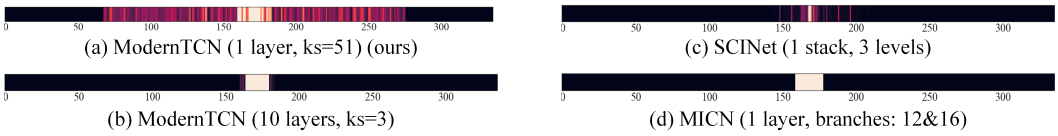
Recently, Transformer-based and MLP-based models have emerged rapidly and won dominance in time series analysis. In contrast, convolution is losing steam in time series tasks nowadays for inferior performance. This paper studies the open question of how to better use convolution in time series analysis and makes efforts to bring convolution back to the arena of time series analysis. To this end, we modernize the traditional TCN and conduct time series related modifications to make it more suitable for time series tasks. As the outcome, we propose **ModernTCN** and successfully solve this open question through a seldom-explored way in time series community. As a pure convolution structure, ModernTCN still achieves the consistent state-of-the-art performance on five mainstream time series analysis tasks while maintaining the efficiency advantage of convolution-based models, therefore providing a better balance of efficiency and performance than state-of-the-art Transformer-based and MLP-based models. Our study further reveals that, compared with previous convolution-based models, our ModernTCN has much larger effective receptive fields (ERFs), therefore can better unleash the potential of convolution in time series analysis. Code is available at this repository: https://github.com/luodhhh/ModernTCN.

1 INTRODUCTION

Time series analysis is widely used in extensive applications, such as industrial forecasting (Zhou et al., 2021), missing value imputation (Friedman, 1962), action recognition (Ye & Keogh, 2009), and anomaly detection (Xu et al., 2021). Because of the immense practical value, the past few years have witnessed the rapid development in time series analysis(Wen et al., 2022; Lim & Zohren, 2021). Among them, the rise of Transformer-based methods and MLP-based models is especially compelling (Nie et al., 2023; Zhang & Yan, 2023; Zhou et al., 2022; Cirstea et al., 2022; Wu et al., 2021; Liu et al., 2021a; Li et al., 2019b; Kitaev et al., 2020; Vaswani et al., 2017) (Li et al., 2023b; Zhang et al., 2022; Zeng et al., 2022). **But around the same time, convolution-based models have received less attention for a long time.**

It's non-trivial to use convolution in time series analysis for it provides a better balance of efficiency and performance. Date back to the 2010s, TCN and its variants (Bai et al., 2018; Sen et al., 2019) are widely-used in many time series tasks. But things have changed in 2020s. Transformer-based models and MLP-based models have emerged rapidly and achieved impressive performance in recent years. Thanks to their global effective receptive fields (ERFs), they can better capture the long-term temporal (cross-time) dependency and thus outperform traditional TCNs by a significant margin. As a result, convolution-based models are losing steam nowadays due to their limited ERFs.

Some previous convolution-based models (Wang et al., 2023; Liu et al., 2022a) try to bring convolution back to the arena of time series analysis. But they mainly focus on designing extra sophisticated structures to work with the traditional convolution, ignoring the importance of updating the convolution itself. And they still cannot achieve comparable performance to the state-of-the-art Transformer-based and MLP-based models. The reason behind can be explained by Figure 1. Increasing the ERF is the key to bringing convolution back to time series analysis. But previous convolution-based models



(a) ModernTCN (1 layer, ks=51) (ours)

(b) ModernTCN (10 layers, ks=3)

(c) SCINet (1 stack, 3 levels)

(d) MICN (1 layer, branches: 12&16)

Figure 1: The Effective Receptive Field (ERF) of ModernTCN and previous convolution-based methods. A more widely distributed light area indicates a larger ERF. Our ModernTCN can obtain a much larger ERF than previous convolution-based methods. Meanwhile, enlarging the kernel size is a more effective way to obtain large ERF than stacking more small kernels.

still have limited ERFs, which prevents their further performance improvements. **How to better use convolution in time series analysis is still a non-trivial and open question.**

As another area where convolution is widely used, computer vision (CV) took a very different path to explore the convolution. Unlike recent studies in time series community, lastest studies in CV focus on optimizing the convolution itself and propose modern convolution (Liu et al., 2022d; Ding et al., 2022; Liu et al., 2022b). Modern convolution is a new convolution paradigm inspired by Transformer. Concretely, modern convolution block incorporates some architectural designs in Transformer and therefore has a similar structure to Transformer block (Figure 2 (a) and (b)). Meanwhile, to catch up with the gloabal ERF in Transformer, modern convolution usually adopat a large kernel as it can effectively increase the ERF (Figure 1). Although the effectiveness of modern convolution has been demonstrated in CV, it has still received little attention from the time series community. Based on above findings, we intend to first modernize the convolution in time series analysis to see whether it can increase the ERF and bring performance improvement.

Besides, convolution is also a potentially efficient way to capture cross-variable dependency. Cross-variable dependency is another critical dependency in time series in addition to the cross-time one. It refers to the dependency among variables in multivariate time series. And early study(Lai et al., 2018b) has already tried to use convolution in variable dimension to capture the cross-variable dependency. Although its performance is not that competitive nowadays, it still demonstrates the feasibility of convolution in capturing cross-variable dependency. Therefore, it's reasonable to believe that convolution can become an efficient and effective way to capture cross-variable dependency after proper modifications and optimizations.

Based on above motivations, we take a seldom-explored way in time series community to successfully bring convolution-based models back to time series analysis. Concretely, we modernize the traditional TCN and conduct some time series related modifications to make it more suitable for time series tasks. As the outcome, we propose a modern pure convolution structure, namely **ModernTCN**, to efficiently utilize cross-time and cross-variable dependency for general time series analysis. We evaluate ModernTCN on five mainstream analysis tasks, including long-term and short-term forecasting, imputation, classification and anomaly detection. Surprisingly, as a pure convolution-based model, ModernTCN still achieves the consistent state-of-the-art performance on these tasks. Meanwhile, ModernTCN also maintains the efficiency advantage of convolution-based models, therefore providing a better balance of efficiency and performance. **Our contributions are as follows:**

- We dive into the question of how to better use convolution in time series and propose a novel solution. Experimental results show that our method can better unleash the potential of convolution in time series analysis than other existing convolution-based models.
- ModernTCN achieves the consistent state-of-the-art performance on multiple mainstream time series analysis tasks, demonstrating the excellent task-generalization ability.
- ModernTCN provides a better balance of efficiency and performance. It maintains the efficiency advantage of convolution-based models while competing favorably with or even better than state-of-the-art Transformer-based models in terms of performance.

2 Related Work

2.1 Convolution in Time Series Analysis

Convolution used to be popular in time series analysis in 2010s. For example, TCN and its variants (Bai et al., 2018; Sen et al., 2019; Franceschi et al., 2019) adopt causal convolution to model the

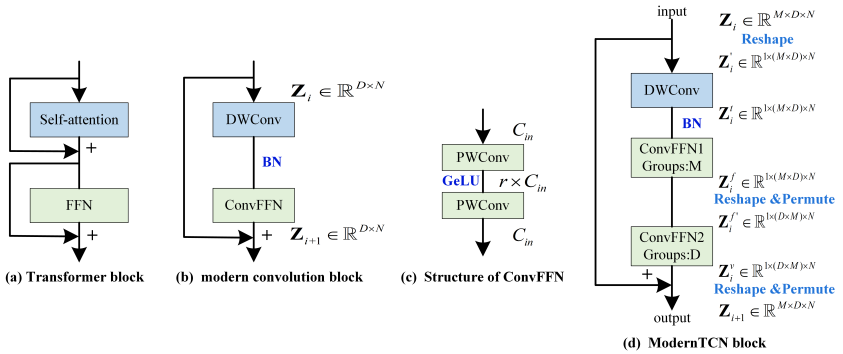


Figure 2: ModernTCN block design. M,N,D are sizes of variable, temporal and feature dimensions. DWConv and PWConv are short for depth-wise and point-wise convolution (2017). $Groups$ is the group number in group convolution (2018). BN and GeLU (2015; 2016) are adpoted in our design.

temporal causality. But they suffer from the limited ERFs. With the rapid development of Transformer-based and MLP-based models, convolution has received less attention in recent years. Some studies try to bring convolution back to time series community. MICN (Wang et al., 2023) goes beyond causal convolution and proposes a multi-scale convolution structure to combine local features and global correlations in time series. SCINet (Liu et al., 2022a) removes the idea of causal convolution and introduces a recursive downsample-convolve-interact architecture to model time series with complex temporal dynamics. But they still have difficulty in modeling long-term dependency due to the limited ERFs. TimesNet (Wu et al., 2023) is special in the family of convolution-based models. Different from other models that mainly use 1D convolution, it transforms 1D time series into 2D-variations and uses 2D convolution backbones in CV to obtain informative representations.

2.2 Modern Convolution in Computer Vision

Convolutional neural networks (ConvNets) (Krizhevsky et al., 2017; Simonyan & Zisserman, 2014; He et al., 2015; Xie et al., 2017; Huang et al., 2017) used to be the dominant backbone architectures in CV. But in 2020s, Vision Transformers (ViTs) (Dosovitskiy et al., 2020; Liu et al., 2021b) are proposed and outperform previous standard ConvNets. To catch up with the performance of ViTs, modern convolution in 2020s are introduced. Inspried by the architectural designs in Transformers, ConvNeXt(Liu et al., 2022d) re-design the convolution block to make it more similar to the Transformer block. To further catch up with the global ERF of Transformers, RepLKNet (Ding et al., 2022) scales the kernel size to 31×31 with the help of Structural Reparameter technique. Further more, SLaK (Liu et al., 2022b) enlarges the kernel size to 51×51 by decomposing a large kernel into two rectangular, parallel kernels and by using dynamic sparsity. Inspired by above studies, we modernize and modify 1D convolution in time series community to make it more suitable for time series analysis tasks.

3 ModernTCN

In this section, we first provide a design roadmap for ModernTCN block to introduce how we modernize and optimize the traditional 1D convolution block in time series community. Then we introduce the overall structure of ModernTCN. And more related details are in Appendix G.

3.1 Modernize the 1D Convolution Block

Following the idea of (Liu et al., 2022d), we firstly re-design the 1D convolution block as shown in Figure 2 (b). DWConv is responsible for learning the temporal information among tokens on a per-feature basis, which plays the same role as the self-attention module in Transformer. ConvFFN is similar to the FFN module in Transformer. It consists of two PWConvs and adopts an inverted bottleneck structure, where the hidden channel of the ConvFFN block is r times wider than the input channel. This module is to learn the new feature representation of each token independently.

Above design leads to a separation of temporal and feature information mixing. Each of DWConv and ConvFFN only mixes information across one of the temporal or feature dimension, which is

differnet from the traditional convolution that jointly mixes information on both dimensions. This decoupling design can make the object tasks easier to learn and reduce the computational complexity.

Based on above design, we borrow from the success of CV and modernize the 1D convolution. But we find that simply modernizing the convolution in the same way as CV brings little to no performance improvement in time series tasks. In fact, above design does not take into account the characteristics of time series. In addition to feature dimension and temporal dimension, time series also has a variable dimension. But the backbone stacked by convolution blocks as designed in Figure 2 (b) cannot handle the variable dimension properly. Since cross-variable information is also critical in multivariate time series (Zhang & Yan, 2023; Li et al., 2023b), more time series related modifications are still needed to make the modern 1D convolution more suitable for time series analysis.

3.2 Time Series Related Modifications

Maintaining the Variable Dimension In CV, before the backbone, we embed 3 channel RGB features at each pixel into a D-dimensional vector to mix the information from RGB channels via the embedding layer. But the similar variable-mixing embedding (e.g., simply embed M variables into a D-dimensional vector per time step) is not suitable for time series. Firstly, the difference among variables in time series is much greater than that among RGB channels in a picture (Cirstea et al., 2022). Just an embedding layer fails to learn the complex dependency across variables and even loses the independent characteristics of variables for not considering their different behaviors. Secondly, such embedding design leads to the discard of variable dimension, making it unable to further study the cross-variable dependency. To this issue, we propose patchify variable-independent embedding.

We denote $\mathbf{X}_{in} \in \mathbb{R}^{M \times L}$ as the M variables input time series of length L and will further divide it into N patches of patch size P after proper padding (Padding details are in Appendix B). The stride in the patching process is S, which also serves as the length of non overlapping region between two consecutive patches. Then the patches will be embedded into D-dimensional embedding vectors:

$$\mathbf{X}_{emb} = \mathrm{Embedding}(\mathbf{X}_{in}) \tag{1}$$

$\mathbf{X}_{emb} \in \mathbb{R}^{M \times D \times N}$ is the input embedding. Different from previous studies (Nie et al., 2023; Zhang & Yan, 2023), we conduct this patchify embedding in an equivalent fully-convolution way for a simpler implementation. After unsqueezing the shape to $\mathbf{X}_{in} \in \mathbb{R}^{M \times 1 \times L}$, we feed the padded $\mathbf{X}_{in}$ into a 1D convolution stem layer with kernel size P and stride S. And this stem layer maps 1 input channel into D output channels. In above process, each of the M univariate time series is embedded independently. Therefore, we can keep the variable dimension. And followings are modifications to make our structure able to capture information from the additional variable dimension.

DWConv DWConv is originally designed for learning the temporal information. Since it's more difficult to jointly learn the cross-time and cross-variable dependency only by DWConv, it's inappropriate to make DWConv also responsible for mixing information across variable dimension. Therefore, we modify the original DWConv from only feature independent to both feature and variable independent, making it learn the temporal dependency of each univariate time series independently. And we adopt large kernel in DWConv to increase ERFs and improve the temporal modeling ability.

ConvFFN Since DWConv is feature and variable independent, ConvFFN should mix the information across feature and variable dimensions as a complementary. A naive way is to jointly learn the dependency among features and variables by a single ConvFFN. But such method leads to higher computational complexity and worse performance. Therefore, we further decouple the single ConvFFN into ConvFFN1 and ConvFFN2 by replacing the PWConvs with grouped PWConvs and setting different group numbers. The ConvFFN1 is responsible for learning the new feature representations per variable and the ConvFFN2 is in charge of capturing the cross-variable dependency per feature.

After above modifications, we have the final **ModernTCN block** as shown in Figure 2 (d). And each of DWConv, ConvFFN1 and ConvFFN2 only mixes information across one of the temporal, feature or variable dimension, which maintains the idea of the decoupling design in modern convolution.

3.3 Overall Structure

After embedding, $\mathbf{X}_{emb}$ is fed into the backbone to capture both the cross-time and cross-variable dependency and learn the informative representation $\mathbf{Z} \in \mathbb{R}^{M \times D \times N}$:

$$\mathbf{Z} = \mathrm{Backbone}(\mathbf{X}_{emb}) \tag{2}$$

$\text{Backbone}(\cdot)$ is the stacked ModernTCN blocks. Each ModernTCN block is organized in a residual way (He et al., 2015). The forward process in the i-th ModernTCN block is:

$$\mathbf{Z}_{i+1} = \text{Block}(\mathbf{Z}_i) + \mathbf{Z}_i \tag{3}$$

Where $\mathbf{Z}_i \in \mathbb{R}^{M \times D \times N}$, $i \in \{1, ..., K\}$ is the i-th block's input,

$$\mathbf{Z}_i = \begin{cases} \mathbf{X}_{emb} & , i = 1 \\ \text{Block}(\mathbf{Z}_{i-1}) + \mathbf{Z}_{i-1} & , i > 1 \end{cases} \tag{4}$$

$\text{Block}(\cdot)$ denotes the ModernTCN block. Then the final represnetation $\mathbf{Z} = \text{Block}(\mathbf{Z}_K) + \mathbf{Z}_K$ will be further used for multiple time series analysis tasks. See Appendix B for pipelines of each task.

4 Experiments

We evaluate ModernTCN on five mainstream analysis tasks, including long-term and short-term forecasting, imputation, classification and anomaly detection to verify the generality of ModernTCN.

Baselines Since we attempt to propose a foundation model for time series analysis, we extensively include the latest and advanced models in time series community as basic baselines, which includes the Transformer-based models: PatchTST (2023), Crossformer (2023) and FEDformer (2022); MLP-based models: MTS-Mixer (2023b), LightTS (2022), DLinear (2022), RLinear and RMLP (2023a); Convolution-based Model: TimesNet (2023), MICN (2023) and SCINet (2022a). We also include the state-of-the-art models in each specific task as additional baselines for a comprehensive comparison.

Main Results As shown in Figure 3, **ModernTCN achieves consistent state-of-the-art performance on five mainstream analysis tasks with higher efficiency**. Detailed discussions about experimental results are in Section 5.1. We provide the experiment details and results of each task in following subsections. In each table, the best results are in **bold** and the second best are underlined.

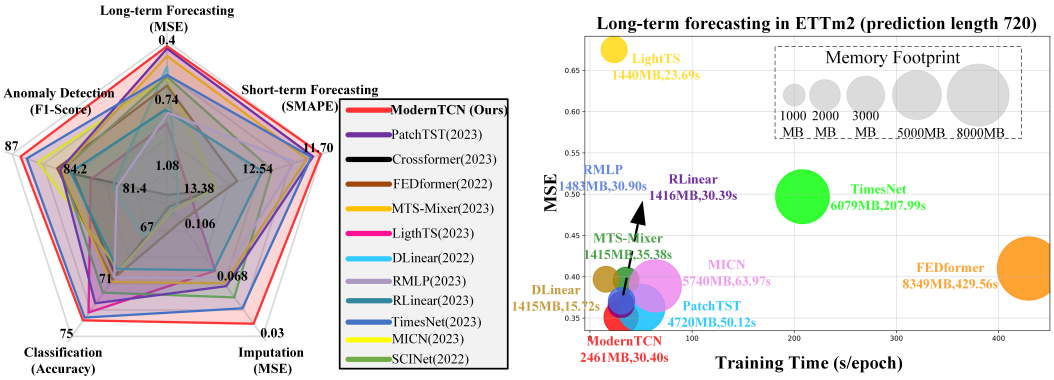


Figure 3: Model performance comparison (left) and efficiency comparison (right).

4.1 Long-term Forecasting

Setups We conducted long-term forecasting experiments on 9 popular real-world benchmarks, including Weather (Wetterstation), Traffic (PeMS), Electricity (UCI), Exchange (Lai et al., 2018a), ILI (CDC) and 4 ETT datasets (Zhou et al., 2021). Following (Nie et al., 2023; Zhang & Yan, 2023), we re-run all baselines with various input lengths and choose the best results to avoid under-estimating the baselines and provide a fairer comparison. We calculate the MSE and MAE of multivariate time series forecasting as metrics.

Results Table 1 shows the excellent performance of ModernTCN in long-term forecasting. Concretely, ModernTCN gains most of the best performance in above 9 cases, surpassing extensive state-of-the-art MLP-based and Transformer-based models. It competes favorably with the best Transformer-based model PatchTST in terms of performance while having faster speed and less memory usage (Figure 3 right), therefore providing a better balance of performance and efficiency. It's notable that ModernTCN surpasses existing convolution-based models by a large margin (27.4% reduction on MSE and 15.3% reduction on MAE), indicating that our design can better unleash the potential of convolution in time series forecasting.

Table 1: Long-term forecasting task. All the results are averaged from 4 different prediction lengths, that is $\{24, 36, 48, 60\}$ for ILI and $\{96, 192, 336, 720\}$ for the others. A lower MSE or MAE indicates a better performance. See Table 27 in Appendix for the full results with more baselines.

Models	**ModernTCN (Ours)**		PatchTST (2023)		Crossformer (2023)		FEDformer (2022)		MTS-Mixer (2023b)		RLinear (2023a)		DLinear (2022)		TimesNet (2023)		MICN (2023)		SCINet (2022a)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.404**	**0.420**	0.413	0.431	0.441	0.465	0.428	0.454	0.430	0.436	0.408	0.421	0.423	0.437	0.458	0.450	0.433	0.462	0.460	0.462
ETTh2	0.322	0.379	0.330	0.379	0.835	0.676	0.388	0.434	0.386	0.413	**0.320**	**0.378**	0.431	0.447	0.414	0.427	0.385	0.430	0.371	0.410
ETTm1	**0.351**	0.381	**0.351**	0.381	0.431	0.443	0.382	0.422	0.370	0.395	0.358	0.376	0.357	**0.379**	0.400	0.406	0.383	0.406	0.387	0.411
ETTm2	**0.253**	**0.314**	0.255	0.315	0.632	0.578	0.292	0.343	0.277	0.325	0.256	**0.314**	0.267	0.332	0.291	0.333	0.277	0.336	0.294	0.355
Electricity	**0.156**	**0.253**	0.159	**0.253**	0.293	0.351	0.207	0.321	0.173	0.272	0.169	0.261	0.177	0.274	0.192	0.295	0.182	0.292	0.195	0.281
Weather	**0.224**	**0.264**	0.226	**0.264**	0.230	0.290	0.310	0.357	0.235	0.272	0.247	0.279	0.240	0.300	0.259	0.287	0.242	0.298	0.287	0.317
Traffic	0.396	0.270	**0.391**	**0.264**	0.535	0.300	0.604	0.372	0.494	0.354	0.518	0.383	0.434	0.295	0.620	0.336	0.535	0.312	0.587	0.378
Exchange	0.302	**0.366**	0.387	0.419	0.701	0.633	0.478	0.478	0.373	0.407	0.345	0.394	**0.297**	0.378	0.416	0.443	0.315	0.404	0.435	0.445
ILI	**1.440**	**0.786**	1.443	0.798	3.361	1.235	2.597	1.070	1.555	0.819	4.269	1.490	2.169	1.041	2.139	0.931	2.567	1.055	2.252	1.021

Table 2: Short-term forecasting task. Results are weighted averaged from several datasets under different sample intervals. Lower metrics indicate better performance. See Table 28 for full results.

Models	**ModernTCN (Ours)**	CARD (2023)	PatchTST (2023)	Crossformer (2023)	FEDformer (2022)	MTS-Mixer (2023b)	RLinear (2023a)	DLinear (2022)	TimesNet (2023)	MICN (2023)	SCINet (2022a)	N-HiTS (2023)	N-BEATS (2019)
SMAPE	**11.698**	11.815	11.807	13.474	12.840	11.892	12.473	13.639	11.829	13.130	12.369	11.927	11.851
MASE	**1.556**	1.587	1.590	1.866	1.701	1.608	1.677	2.095	1.585	1.896	1.677	1.613	1.599
OWA	**0.838**	0.850	0.851	0.985	0.918	0.859	0.898	1.051	0.851	0.980	0.894	0.861	0.855

4.2 Short-term Forecasting

Setups We adopt M4 dataset (Makridakis et al., 2018) as the short-term forecasting benchmark. Following (Wu et al., 2023), we fix the input length to be 2 times of prediction length and calculate Symmetric Mean Absolute Percentage Error (SMAPE), Mean Absolute Scaled Error (MASE) and Overall Weighted Average (OWA) as metrics. We include additional baselines like CARD (2023), N-BEATS (2019) and N-HiTS (2023) for this specific task. Since the M4 dataset only contains univariate time series, we remove the cross-variable component in ModernTCN and Crossformer.

Results The results are summarized in Table 2. Short-term forecasting in M4 dataset is a much more challenging task because the time series samples are collected from different sources and have quite different temporal properties. Our ModernTCN still achieves the consistent state-of-the-art in this difficult task, demonstrating its excellent temporal modeling ability.

4.3 Imputation

Setups Imputation task aims to impute the missing values based on the partially observed time series. Due to unexpected accidents like equipment malfunctions or communication error, missing values in time series are very common. Since missing values may harm the performance of downstream analysis, imputation task is of high practical value. Following (Wu et al., 2023), we mainly focus on electricity and weather scenarios where the data-missing problem happens commonly. We select the datasets from these scenarios as benchmarks, including ETT (Zhou et al., 2021), Electricity (UCI) and Weather (Wetterstation). We randomly mask the time points in ratios of $\{12.5\%, 25\%, 37.5\%, 50\%\}$ to compare the model capacity under different proportions of missing data.

Results Table 3 shows the compelling performance of ModernTCN in imputation tasks. ModernTCN achieves 22.5% reduction on MSE and 12.9% reduction on MAE compared with previous state-of-the-art baseline TimesNet (2023). Due to the missing values, the remaining observed time series is irregular, making it more difficult to capture cross-time dependency. Our ModernTCN still

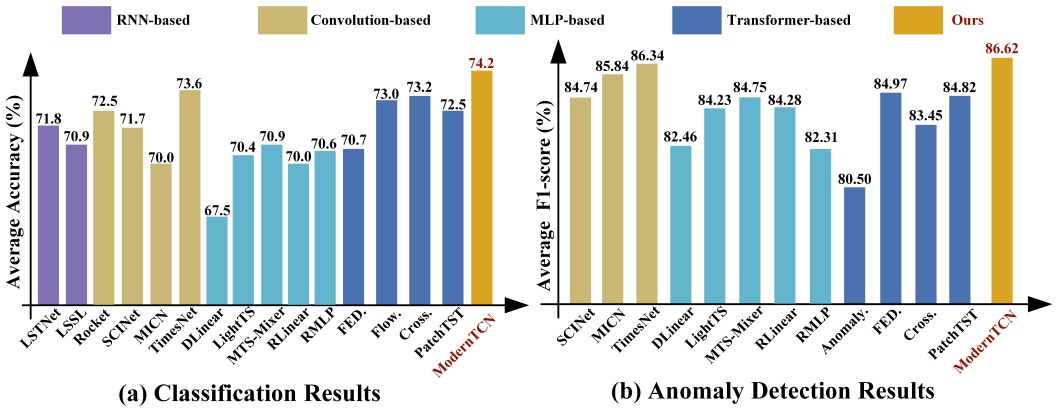
Table 3: Imputation task. We randomly mask $\{12.5\%, 25\%, 37.5\%, 50\%\}$ time points in length-96 time series. The results are averaged from 4 different mask ratios. A lower MSE or MAE indicates a better performance. See Table 29 in Appendix for full results with more baselines.

Models	**ModernTCN (Ours)**		PatchTST (2023)		Crossformer (2023)		FEDformer (2022)		MTS-Mixer (2023b)		RLinear (2023a)		DLinear (2022)		TimesNet (2023)		MICN (2023)		SCINet (2022a)	
Averaged	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.020**	**0.093**	0.045	0.133	0.041	0.143	0.062	0.177	0.056	0.154	0.070	0.166	0.093	0.206	0.027	0.107	0.070	0.182	0.039	0.129
ETTm2	**0.019**	**0.082**	0.028	0.098	0.046	0.149	0.101	0.215	0.032	0.107	0.032	0.108	0.096	0.208	0.022	0.088	0.144	0.249	0.027	0.102
ETTh1	**0.050**	**0.150**	0.133	0.236	0.132	0.251	0.117	0.246	0.127	0.236	0.141	0.242	0.201	0.306	0.078	0.187	0.125	0.250	0.104	0.216
ETTh2	**0.042**	**0.131**	0.066	0.164	0.122	0.240	0.163	0.279	0.069	0.168	0.066	0.165	0.142	0.259	0.049	0.146	0.205	0.307	0.064	0.165
Electricity	**0.073**	**0.187**	0.091	0.209	0.083	0.199	0.130	0.259	0.089	0.208	0.119	0.246	0.132	0.260	0.092	0.210	0.119	0.247	0.086	0.201
Weather	**0.027**	**0.044**	0.033	0.057	0.036	0.090	0.099	0.203	0.036	0.058	0.034	0.058	0.052	0.110	0.030	0.054	0.056	0.128	0.031	0.053

achieves the best performance in this challenging task, verifying the model capacity in capturing temporal dependency under extremely complicated situations.

It's also notable that cross-variable dependency plays a vital role in imputation task. Since in some time steps, only part of the variables are missing while others are still remaining, utilizing the cross-variable dependency between missing variables and remaining variables can help to effectively impute the missing values. Therefore, some variable-independent methods like PatchTST (2023) and DLinear (2022) fail in this task for not taking cross-variable dependency into consideration.

4.4 Classification and Anomaly Detection



(a) Classification Results **(b) Anomaly Detection Results**

Figure 4: Results of classification and anomaly detection. The results are averaged from several datasets. Higher accuracy and F1 score indicate better performance. $*$. in the Transformer-based models indicates the name of $*$former. See Table 30 and 31 in Appendix for full results.

Setups For classification, we select 10 multivariate datasets from UEA Time Series Classification Archive (Bagnall et al., 2018) for benchmarking and pre-process the datasets following (Wu et al., 2023). We include some task-specific state-of-the-art methods like LSTNet (2018b), Rocket (2020) and Flowformer (2022) as additional baselines.

For anomaly detection, we compare models on five widely-used benchmarks: SMD (Su et al., 2019), SWaT (Mathur & Tippenhauer, 2016), PSM (Abdulaal et al., 2021), MSL and SMAP (Hundman et al., 2018). We include Anomaly transformer (2021) as additional baselines. Following it, we adopt the classical reconstruction task and choose the reconstruction error as the anomaly criterion.

Results **Time series classification** is a classic task in time series community and reflects the model capacity in high-level representation. As shown in Figure 4, ModernTCN achieves the best performance with an average accuracy of 74.2% . It's notable that some MLP-based models fail in classification tasks. This is because MLP-based models prefer to discard the feature dimension to

obtain a lightweight backbone, which leads to the insufficient representation capability and inferior classification performance. **Anomaly detection** results are shown in Figure 4. ModernTCN achieves competitive performance with previous state-of-the-art baseline TimesNet (2023). Meanwhile, compared with TimesNet, ModernTCN saves 55.1% average training time per epoch (3.19s vs 7.10s) in classification task and saves 57.3% average training time per epoch (132.65s vs 310.62s) in anomaly detection task, providing a better balance of efficiency and performance in both tasks.

5 MODEL ANALYSIS

5.1 COMPREHENSIVE COMPARISON OF PERFORMANCE AND EFFICIENCY

Summary of Experimental Results ModernTCN achieves consistent state-of-the-art performance on five mainstream analysis tasks compared with other task-specific models or previous state-of-the-art baselines, demonstrating its excellent task-generality and highlighting the potential of convolution in time series analysis (Figure 3 left). ModernTCN also has more advantage in efficiency, therefore providing a better balance of efficiency and performance (Figure 3 right). It's worth noting that our method surpasses existing convolution-based models by a large margin, indicating that our design can provide a better solution to the problem of how to better use convolution in time series analysis.

Compared with Transformer-based and MLP-based Models Unlike previous convolution-based models, ModernTCN competes favorably with or even better than state-of-the-art Transformer-based models in terms of performance. Meanwhile, as a pure convolution model, ModernTCN has higher efficiency than Transformer-based models. As shown in Figure 3 right, ModernTCN has faster training speed and less memory usage, which demonstrates the efficiency superiority of our model.

ModernTCN outperforms all MLP-based baselines in all five tasks thanks to the better representation capability in ModernTCN blocks. In contrast, MLP-based models prefer to adopt a lightweight backbone for a smaller memory usage. But such design in MLP-based models also leads to the insufficient representation capability and inferior performance. Although ModernTCN is sightly inferior in memory usage, it still has almost the same running time efficiency as some MLP-based baselines thanks to the fast floating point operation speed in convolution. **Considering both performance and efficiency, ModernTCN has more advantage in general time series analysis.**

Compared with TimesNet (2023) In addition to ModernTCN, TimesNet also demonstrates excellent generality in five mainstream tasks. It's worth noting that both models are convolution-based models, which further reveals that convolution has a better comprehensive ability in time series analysis. **Meanwhile, both methods are inspired by CV and intend to make the time series analysis take advantage of the development of CV community.** But the two methods take different paths to accomplish this goal. TimesNet makes efforts to transform the 1D time series into 2D space, making the time series can be modeled by the 2D ConvNets in CV community. But the additional data transformation and aggregation modules also bring extra memory usage and slower training speed. Different from TimesNet, our ModernTCN maintains the 1D time series and turns to modernize and optimize the 1D convolution in time series community. Therefore, we design a modern pure convolution structure that without any additional modules. The fully-convolutional nature in our design brings higher efficiency and makes it extremely simple to implement, therefore leading to the both performance and efficiency superiority than TimesNet (Figure 3 left and right).

5.2 ANALYSIS OF EFFECTIVE RECEPTIVE FIELD (ERF)

Enlarging the ERF is the key to bring convolution back to time series analysis. In this section, we will discuss why ModernTCN can provide better performance than previous convolution-based models from the perspective of ERF. Firstly, rather than stacking more layers like other traditional TCNs (2018), ModernTCN increases the ERF by enlarging the kernel size. And in a pure convolution structure, enlarging the kernel size is a much more effective way to increases ERF. According to the theory of ERF in pure convolution-based models (Luo et al., 2016), ERF is proportion to $O(ks \times \sqrt{nl})$, where ks and nl refers to the kernel size and the number of layers respectively. ERF grows linearly with the kernel size while sub-linearly with the layer number. Therefore, by enlarging the kernel size, ModernTCN can easily obtain a larger ERF and further bring perfomance improvement.

Except for enlarging the kernel size and stacking more layers, some previous convolution-based methods in time series community (MICN (2023) and SCINet (2022a)) prefer to adopt some sophisticated structures to cooperate with the traditional convolution, intending to enlarge their ERFs. Since they are not pure convolution structures, it's hard to analyse their ERFs theoretically. Therefore, we visualize the ERFs of these methods for intuitive comparision. Following (Kim et al., 2023), we sample 50 length-336 input time series from the validation set in ETTh1 for the visualization. The idea behind is to visualize how many points in the input series can make contribution to the middle point of the final feature map. As shown in Figure 1, our method can obtain a much larger ERF than previous convolution-based methods. Therefore our method can better unleash the potential of convolution in time series and sucessfully bring performance improvements in multiple time series analysis tasks.

5.3 Ablation Study

Ablation of ModernTCN Block Design To validate the effectiveness of our design in ModernTCN block, we conduct ablation study in long-term forecasting tasks. Results are shown on Table 4. *Discard Variable Dimension* cannot provide ideal performance, which confirms our arguement that simply modernizing the convolution in the same way as CV could not bring performance improvement for omitting the importance of variable dimension. To better handle the variable dimension, we decouple a single ConvFFN into ConvFFN1 and ConvFFN2 in our design. As shown in Table 4, the *undecoupled ConvFFN* provide the worst performance and the combination of our decoupled two ConvFFNs (*ConvFFN1+ConvFFN2*) achieve the best, which proves the necessity and effectiveness of our decouple modification to ConvFFN module. Please see Appendix H for more details.

Table 4: Ablation of ModernTCN block. We list the averaged MSE/MAE of different forecast lengths.

Dataset	ETTh1		ETTh2		ETTm1		ETTm2		ECL		Weather		Exchange		ILI	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ConvFFN1 + ConvFFN2	**0.404**	**0.420**	**0.322**	**0.379**	**0.351**	**0.381**	**0.253**	**0.314**	**0.156**	**0.253**	**0.224**	**0.264**	**0.302**	**0.366**	**1.440**	**0.786**
Undecoupled ConvFFN	0.425	0.440	0.341	0.391	0.370	0.395	0.278	0.332	0.169	0.269	0.260	0.288	0.323	0.378	1.813	0.875
only ConvFFN1	0.418	0.434	0.329	0.384	0.360	0.386	0.267	0.323	0.163	0.264	0.241	0.278	0.310	0.371	1.667	0.866
only ConvFFN2	0.416	0.438	0.330	0.384	0.359	0.385	0.268	0.325	0.164	0.265	0.249	0.283	0.312	0.371	1.874	0.882
ConvFFN1 + ConvFFN1	0.420	0.435	0.329	0.385	0.363	0.386	0.267	0.323	0.165	0.265	0.242	0.279	0.311	0.371	1.710	0.866
ConvFFN2 + ConvFFN2	0.417	0.437	0.330	0.383	0.362	0.384	0.269	0.325	0.165	0.265	0.254	0.285	0.311	0.371	1.831	0.884
Discard Variable Dimension	0.590	0.560	0.382	0.430	0.508	0.494	0.319	0.367	0.441	0.486	0.300	0.331	0.361	0.412	1.932	0.936

Ablation of Cross-variable Component As an important time series related modification in our design, we design the ConvFFN2 as a cross-variable component to capture the cross-variable dependency. We conduct ablation studies in imputation tasks and anomaly detection tasks. As shown in Table 5, without the ConvFFN2 will cause severe performance degradation in these two tasks, which emphasizes the importance of cross-variable dependency in time series analysis.

Table 5: Ablation of Cross-variable component.

Dataset		ETTm1		ETTm2		ETTh1		ETTh2		ECL		Weather	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Imputation	**Ours**	**0.020**	**0.093**	**0.019**	**0.082**	**0.050**	**0.150**	**0.042**	**0.131**	**0.073**	**0.187**	**0.027**	**0.044**
	w/o Cross-variable	0.038	0.121	0.028	0.097	0.089	0.195	0.059	0.150	0.096	0.203	0.030	0.048
	Promotion	47.4%	23.1%	32.1%	15.5%	43.8%	23.1%	28.8%	12.7%	24.0%%	7.9%	10.0%	8.3%

Dataset		SMD	MSL	SMAP	SWaT	PSM
Metric		F1-score	F1-score	F1-score	F1-score	F1-score
Anomaly Detection	**Ours**	**85.81**	**84.92**	**71.26**	**93.86**	**97.23**
	w/o Cross-variable	81.33	72.13	64.93	83.46	96.08
	Promotion	5.5%	17.7%	9.7%	12.5%	1.2%

6 Conclusion and Future Work

In this paper, we take a seldom-explored way in time series community to solve the question of how to better use convolution in time series analysis. By modernizing and modifying the traditional TCN block with time series related modifications, we propose ModernTCN and successfully bring convolution back to the arena of time series analysis. Experimental results show the great task generality of ModernTCN. While performing on par with or better than state-of-the-art Transformer-based models in terms of performance, ModernTCN maintains the efficiency advantage of convolution-based models, therefore providing a better balance of performance and efficiency. Since convolution-based models have received less attention in time series analysis for a long time, we hope that the new results reported in this study will bring some fresh perspectives to time series community and prompt people to rethink the importance of convolution in time series analysis.

ACKNOWLEDGMENT

This work was supported by the Guangdong Key Area Research and Development Program (2019B010154001) and Hebei Innovation Plan (20540301D).

REPRODUCIBILITY STATEMENT

The model architecture is introduced in details with equations and figures in the main text. And all the implementation details are included in the Appendix, including dataset descriptions, metrics of each task, model configurations and experiment settings. Code is available at this repository: https://github.com/luodhhh/ModernTCN.

REFERENCES

Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. Practical approach to asynchronous multivariate time series anomaly detection and localization. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 2485–2494, 2021.

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

CDC. Illness. `https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html`.

Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6989–6997, 2023.

Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting–full version. *arXiv preprint arXiv:2204.13767*, 2022.

Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13733–13742, 2021.

Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. 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*, pp. 11963–11975, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. In *NeurIPS*, 2019.

Milton Friedman. The interpolation of time series by related series. *J. Amer. Statist. Assoc*, 1962.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *ICLR*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning. *Image Recognition*, 7, 2015.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387–395, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Bum Jun Kim, Hyeyeon Choi, Hyeonah Jang, Dong Gu Lee, Wonseok Jeong, and Sang Woo Kim. Dead pixel test using effective receptive field. *Pattern Recognition Letters*, 167:149–156, 2023.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *SIGIR*, 2018a.

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

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *NeurIPS*, 2019a.

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

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

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

Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

Minhao Liu, Ailing Zeng, Z Xu, Q Lai, and Q Xu. Scinet: time series modeling and forecasting with sample convolution and interaction. In *36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022a.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Mykola Pechenizkiy, Decebal Mocanu, and Zhangyang Wang. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. *arXiv preprint arXiv:2207.03620*, 2022b.

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

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Rethinking the stationarity in time series forecasting. In *NeurIPS*, 2022c.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021b.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022d.

Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

Aditya P Mathur and Nils Ole Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pp. 31–36. IEEE, 2016.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2023.

Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

PeMS. Traffic. `http://pems.dot.ca.gov/`.

Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2828–2837, 2019.

UCI. Electricity. `https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023.

Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

Wetterstation. Weather. `https://www.bgc-jena.mpg.de/wetter/`.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Flowformer: Linearizing transformers with conservation flows. *arXiv preprint arXiv:2202.06258*, 2022.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2023.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. *arXiv preprint arXiv:2110.02642*, 2021.

Wang Xue, Tian Zhou, QingSong Wen, Jinyang Gao, Bolin Ding, and Rong Jin. Make transformer great again for time series forecasting: Channel aligned robust dual transformer. *arXiv preprint arXiv:2305.12095*, 2023.

Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 947–956, 2009.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*, 2022.

Tianping Zhang, Yizhuo Zhang, Wei Cao, Jiang Bian, Xiaohan Yi, Shun Zheng, and Jian Li. Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures. *arXiv preprint arXiv:2207.01186*, 2022.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*, 2023.

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

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pp. 27268–27286. PMLR, 2022.

Table 6: Dataset descriptions of long-term forecasting and imputation.

Dataset	Weather	Traffic	Exchange	Electricity	ILI	ETTh1	ETTh2	ETTm1	ETTm2
Dataset Size	52696	17544	7207	26304	966	17420	17420	69680	69680
Variable Number	21	862	8	321	7	7	7	7	7
Sampling Frequency	10 mins	1 hour	1 day	1 hour	1 week	1 hour	1 hour	15 mins	15 mins

A DATASETS

A.1 LONG-TERM FORECASTING AND IMPUTATION DATASETS

We evaluate the long-term forecasting performance on 9 popular real-world datasets, including Weather, Traffic, Electricity, Exchange, ILI and 4 ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2). And for imputation tasks, we choose Weather, Electricity and 4 ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2) for benchmarking. These datasets have been extensively utilized for benchmarking and cover many aspects of life.

The dataset size (total timesteps), variable number and sampling frequency of each dataset are summarized in Table 6 . We follow standard protocol (Zhou et al., 2021) and split all datasets into training, validation and test set in chronological order by the ratio of 6:2:2 for the ETT dataset and 7:1:2 for the other datasets. And training, validation and test sets are zero-mean normalized with the mean and standard deviation of training set. Each of above datasets only contains one continuous long time series, and we obtain samples by sliding window.

More introduction of the datasets are as follow:

1) **Weather**[1] contains 21 meteorological indicators such as humidity and air temperature for 2020 whole year in Germany.
2) **Traffic**[2] contains the road occupancy rates measured by 862 different sensors on San Francisco Bay area freeways in 2 years. Data is collected from California Department of Transportation.
3) **Electricity**[3] contains hourly electricity consumption of 321 clients from 2012 to 2014.
4) **Exchange**[4] the daily exchange rates of eight different countries ranging from 1990 to 2016.
5) **ILI**(Influenza-Like Illness)[5] contains the weekly recorded influenza-like illness (ILI) patients data in the United States between 2002 and 2021. It contains 7 indicators like the numbers of ILI patients under different age ranges and the ratio of ILI patients to the total patients. Data is provided by Centers for Disease Control and Prevention of the United States.
6) **ETT**(Electricity Transformer Temperature)[6] contains the data collected from electricity transformers with 7 sensors, including load, oil temperature, etc. It contains two sub-dataset labeled with 1 and 2, corresponding to two different electric transformers from two separated counties in China. And each of them contains 2 different resolutions (15 minutes and 1 hour) denoted with m and h. Thus, in total we have 4 ETT datasets: ETTh1, ETTh2, ETTm1, ETTm2.

A.2 SHORT-TERM FORECASTING DATASETS

M4 involves 100,000 different time series samples collected in different domains with different frequencies, covering a wide range of economic, industrial, financial and demographic areas.

It's notable that M4 dataset is different from the long-term forecasting datasets. Each of long-term forecasting dataset only contains one continuous long time series, and we obtain samples by sliding window. Therefore all samples are come from the same source time series and more likely to have similar temporal property. But the samples in M4 datasets are collected from different sources.

[1] https://www.bgc-jena.mpg.de/wetter/

[2] https://pems.dot.ca.gov/

[3] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

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

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

[6] https://github.com/zhouhaoyi/ETDataset

Therefore they may have quite different temporal property, making the forecasting tasks in M4 datasets more difficult. Table 7 summarizes details of statistics of short-term forecasting M4 datasets.

Table 7: Datasets and mapping details of M4 dataset (Makridakis et al., 2018).

Dataset	Sample Numbers (train set,test set)	Variable Number	Prediction Length
M4 Yearly	(23000, 23000)	1	6
M4 Quarterly	(24000, 24000)	1	8
M4 Monthly	(48000, 48000)	1	18
M4 Weekly	(359, 359)	1	13
M4 Daily	(4227, 4227)	1	14
M4 Hourly	(414, 414)	1	48

A.3 Classification Datasets

UEA dataset involves many time series samples collected in different domains for classification, covering the recognition tasks based on face, gesture, action and audio as well as other practical tasks like industry monitoring, health monitoring and medical diagnosis based on heartbeat. Most of them have 10 classes. Table 8 summarizes details of statistics of classification UEA datasets.

Table 8: Datasets and mapping details of UEA dataset (Bagnall et al., 2018).

Dataset	Sample Numbers (train set,test set)	Variable Number	Series Length
EthanolConcentration	(261, 263)	3	1751
FaceDetection	(5890, 3524)	144	62
Handwriting	(150, 850)	3	152
Heartbeat	(204, 205)	61	405
JapaneseVowels	(270, 370)	12	29
PEMS-SF	(267, 173)	963	144
SelfRegulationSCP1	(268, 293)	6	896
SelfRegulationSCP2	(200, 180)	7	1152
SpokenArabicDigits	(6599, 2199)	13	93
UWaveGestureLibrary	(120, 320)	3	315

A.4 Anomaly Detection Datasets

We adopt datasets from different domains like server machine, spacecraft and infrastructure for benchmarking. Each dataset is divided into training, validation and testing sets. Each dataset contains one continuous long time series, and we obtain samples from the continuous long time series with a fixed length sliding window. Table 9 summarizes details of statistics of the datasets.

Table 9: Datasets and mapping details of anomaly detection dataset.

Dataset	Dataset sizes(train set,val set, test set)	Variable Number	Sliding Window Length
SMD	(566724, 141681, 708420)	38	100
MSL	(44653, 11664, 73729)	55	100
SMAP	(108146, 27037, 427617)	25	100
SWaT	(396000, 99000, 449919)	51	100
PSM	(105984, 26497, 87841)	25	100

B Pipeline

B.1 Padding Details in Patchify Variable-independent Embedding

Before patching and embedding, we adopt a padding operation on the original time series $\mathbf{X}_{in}$ to keep $N = L//S$. Specifically, we repeat $\mathbf{X}_{in}$'s last value $(P - S)$ times and then pad them back to the end of $\mathbf{X}_{in}$.

Denoted $\mathbf{X}_{in} \in \mathbb{R}^{M \times L}$ as the M variables input time series of length L, the overall process of Patchify Variable-independent Embedding is as follows:

1) Unsqueezing its shape to $\mathbf{X}_{in} \in \mathbb{R}^{M \times 1 \times L}$.
2) Adopting above padding operation on it.
3) Feeding the padded $\mathbf{X}_{in}$ to the 1D convolution stem layer for patching and embedding.

B.2 Pipeline for Regression Tasks

The pipeline for forecasting, imputation and anomaly detection is shown as Figure 5.

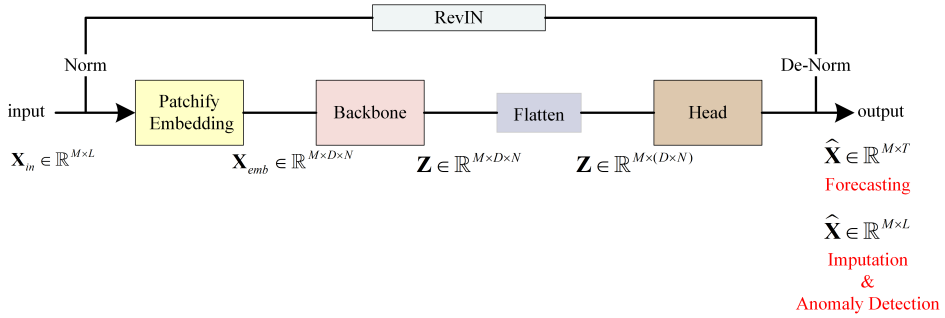


Figure 5: Pipeline for Regression Tasks.

After the backbone, we have $\mathbf{Z} \in \mathbb{R}^{M \times D \times N}$. Then the linear head with a flatten layer is used to obtain the final prediction:

$$\widehat{\mathbf{X}} = \mathrm{Head}(\mathrm{Flatten}(\mathbf{Z})) \tag{5}$$

Where $\widehat{\mathbf{X}} \in \mathbb{R}^{M \times T}$ is the prediction of length T with M variables. $\mathrm{Flatten}(\cdot)$ denotes a flatten layer that changes the final representation's shape to $\mathbf{Z} \in \mathbb{R}^{M \times (D \times N)}$. $\mathrm{Head}(\cdot)$ indicates the linear projection layer that maps the final representation to the final prediction.

Stationary Technique RevIN (Kim et al., 2021) is a special instance normalization for time series to mitigate the distribution shift between the training and testing data. In norm phase, we normalize the input time series per variable with zero mean and unit standard deviation before patching and embedding. Then in de-norm phase, we add the mean and deviation back to the final prediction per variable after the forward process.

Low Rank Approximation for Traffic Datasets To Traffic dataset that contains much more variables than others, directly applying our model to Traffic dataset leads to heavy memory usage. Since the variables in multivariate times series have dependency on each other, a possible way to solve this problem is to find a low rank approximation of these M variables when M is a very big number. For example, FEDformer (2022) uses a low rank approximated transformation in frequency domain for better memory efficiency. And Crossformer (2023) also uses a small fixed number of routers to aggregate messages from all variables to save memory usage.

In this paper, we design a bottleneck structure as a simple and direct method to achieve this goal. In details, before fed into the ConvFFN1 and ConvFFN2, the variable number will be projected to M' by a projection layer, where M' is much smaller than M. Then after the ConvFFN1 and ConvFFN2 process, another projection layer is used to project the variable number back to M.

Table 10: Results with different M' in Traffic dataset.

Models	w/o		$M' = 256$		$M' = 64$		$M' = 16$	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Traffic	0.393	0.267	0.396	0.273	0.395	0.271	0.396	0.270
Memory Usage (%)	100%		82%		77%		72%	

And we conduct experiments with different M' to verify this solution. As shown in Table 10, our method can significantly reduce memory usage with only a little performance degradation. This result proves the fact that there is redundancy between the 862 variables in Traffic dataset. Therefore we can learn a low rank approximation of these 862 variables based on their dependency on each other. And such low rank approximation can help to reduce memory usage without too much performance degradation.

Input The input is M variables time series with input length L. In imputaiton tasks, the input series will further element-wise multipy with a mask matrix to represent the randomly missing values.

Output In forecastiong tasks, the output is the prediction time series with prediction length T. In impuatation tasks, the output is the imputed input time series with input length L. In anomaly detection tasks, the output is the reconstructed input time series with input length L.

B.3 Pipeline for Classification Tasks

The pipeline for classification is shown as Figure 6.

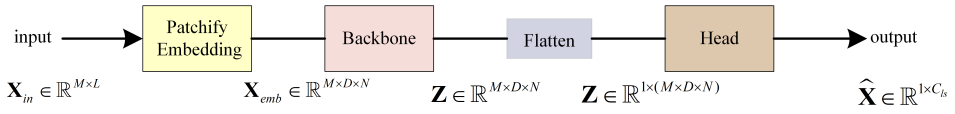


Figure 6: Pipeline for Classification Tasks.

There are two difference: (1) We remove the RevIN; (2) The flatten layer is different. In classification tasks, the flatten layer changes the final representation's shape to $\mathbf{Z} \in \mathbb{R}^{1 \times (M \times D \times N)}$. Then a projection layer with SoftMax activation is to map the final representation to the final classification result $\widehat{\mathbf{X}} \in \mathbb{R}^{1 \times C_{ls}}$, where C_{ls} is the number of classes.

Followings are implementation details and model parameters of each tasks.

C Experiment Details

C.1 Long-term Forecasting

Implementation Details Our method is trained with the L2 loss, using the ADAM (Kingma & Ba, 2014) optimizer with an initial learning rate of 10^{-4}. The default training process is 100 epochs with proper early stopping. The mean square error (MSE) and mean absolute error (MAE) are used as metrics. All the experiments are repeated 5 times with different seeds and the means of the metrics are reported as the final results. All the deep learning networks are implemented in PyTorch(Paszke et al., 2019) and conducted on NVIDIA A100 40GB GPU.

All of the models are following the same experimental setup with prediction length $T \in \{24, 36, 48, 60\}$ for ILI dataset and $T \in \{96, 192, 336, 720\}$ for other datasets as (Nie et al., 2023). We collect some baseline results from (Nie et al., 2023) where all the baselines are re-run with various input length L and the best results are chosen to avoid under-estimating the baselines. For other baselines, we follow the official implementation and run them with vary input length $L \in \{24, 36, 48, 60, 104, 144\}$ for ILI dataset and $L \in \{96, 192, 336, 512, 672, 720\}$ for other

datasets and choose the best results. All experiments are repeated five times. We calculate the MSE and MAE of multivariate time series forecasting as metrics.

Model Parameter By default, ModernTCN contains 1 ModernTCN block with the channel number (dimension of hidden states) $D = 64$ and FFN ratio $r = 8$. The kernel size is set as $large\ size = 51$ and $small\ size = 5$. Patch size and stride are set as $P = 8, S = 4$ in the patchify embedding process. For bigger datasets (ETTm1 and ETTm2), we stack 3 ModernTCN blocks to improve the representation capability. For small datasets (ETTh1, ETTh2, Exchange and ILI), we recommend a small FFN ratio $r = 1$ to mitigate the possible overfitting and for better memory efficiency.

For baseline models, if the original papers conduct long-term forecasting experiments on the dataset we use, we follow the recommended model parameters in the original papers, including the number of layers, dimension of hidden states, etc. But we re-run them with vary input lengths as mentioned in Section 4.1 and choose the best results to obtain a strong baseline.

Metric We adopt the mean square error (MSE) and mean absolute error (MAE) for long-term forecasting.

$$\text{MSE} = \frac{1}{T}\sum_{i=0}^{T}(\widehat{\mathbf{X}}_i - \mathbf{X}_i)^2$$

$$\text{MAE} = \frac{1}{T}\sum_{i=0}^{T}\left|\widehat{\mathbf{X}}_i - \mathbf{X}_i\right|$$

where $\widehat{\mathbf{X}}, \mathbf{X} \in \mathbb{R}^{T\times M}$ are the M variables prediction results of length T and corresponding ground truth. $\mathbf{X}_i$ means the i-th time step in the prediction result.

C.2 Short-term Forecasting

Implementation Details Our method is trained with the SMAPE loss, using the ADAM (Kingma & Ba, 2014) optimizer with an initial learning rate of 5×10^{-4}. The default training process is 100 epochs with proper early stopping. The symmetric mean absolute percentage error (SMAPE), mean absolute scaled error (MASE) and overall weighted average (OWA) are used as metrics. All the experiments are repeated 5 times with different seeds and the means of the metrics are reported as the final results.

Following (Wu et al., 2023), we fix the input length to be 2 times of prediction length for all models. Since the M4 dataset only contains univariate time series, we remove the cross-variable component in ModernTCN and Crossformer.

Model Parameter By default, ModernTCN contains 2 ModernTCN blocks with the channel number (dimension of hidden states) $D = 2048$ and FFN ratio $r = 1$. The kernel size is set as $large\ size = 51$ and $small\ size = 5$. For datasets of less samples (M4 Weekly, M4 Daily and M4 Hourly), we use a smaller channel number $D = 1024$. Patch size and stride are set as $P = 8, S = 4$ in the patchify embedding process. For datasets with shorter input length, we reduce the patch size and stride (e.g., $P = 3, S = 3$ in M4 Yearly and $P = 2, S = 2$ in M4 Quarterly).

Metric For the short-term forecasting, following (Oreshkin et al., 2019), we adopt the symmetric mean absolute percentage error (SMAPE), mean absolute scaled error (MASE) and overall weighted average (OWA) as the metrics, which can be calculated as follows:

$$\text{SMAPE} = \frac{200}{T}\sum_{i=1}^{T}\frac{|\mathbf{X}_i - \widehat{\mathbf{X}}_i|}{|\mathbf{X}_i| + |\widehat{\mathbf{X}}_i|}, \qquad \text{MAPE} = \frac{100}{T}\sum_{i=1}^{T}\frac{|\mathbf{X}_i - \widehat{\mathbf{X}}_i|}{|\mathbf{X}_i|},$$

$$\text{MASE} = \frac{1}{T}\sum_{i=1}^{T}\frac{|\mathbf{X}_i - \widehat{\mathbf{X}}_i|}{\frac{1}{T-p}\sum_{j=p+1}^{T}|\mathbf{X}_j - \mathbf{X}_{j-p}|}, \qquad \text{OWA} = \frac{1}{2}\left[\frac{\text{SMAPE}}{\text{SMAPE}_{\text{Naïve2}}} + \frac{\text{MASE}}{\text{MASE}_{\text{Naïve2}}}\right],$$

where p is the periodicity of the data. $\widehat{\mathbf{X}}, \mathbf{X} \in \mathbb{R}^{T \times M}$ are the M variables prediction results of length T and corresponding ground truth. $\mathbf{X}_i$ means the i-th time step in the prediction result.

C.3 IMPUTATION

Implementation Details Our method is trained with the L2 loss, using the ADAM (Kingma & Ba, 2014) optimizer with an initial learning rate of 10^{-3}. The default training process is 100 epochs with proper early stopping. The mean square error (MSE) and mean absolute error (MAE) are used as metrics. All the experiments are repeated 5 times with different seeds and the means of the metrics are reported as the final results.

We use a mask matrix $\mathbf{C} \in \mathbb{R}^{L \times M}$ to represent the missing values in input time series $\mathbf{X}_{in}$.

$$c_l^m = \begin{cases} 0 & \textit{if } x_l^m \textit{ is not observed} \\ 1 & \textit{otherwise} \end{cases}.$$

$\mathbf{X}_{in} \in \mathbb{R}^{L \times M}$ is the M variables input time series of length L. And L is set as 96 in imputation tasks. x_l^m is the value at l-th timestep in the m-th univariate time series.

The input is the partially observed time series $\mathbf{C} \odot \mathbf{X}_{in}$ and the output is the imputed time series of the input. $\odot$ indicates the element-wise multiplication between the two tensors $\mathbf{X}_{in}$ and $\mathbf{C}$. And we only calculate MSE loss on masked tokens.

Model Parameter By default, ModernTCN has 1 ModernTCN block with channel number $D = 128$ and FFN ratio $r = 1$. The kernel size is set as $large\ size = 71$ and $small\ size = 5$. Patch size and stride are set as $P = 1, S = 1$ to avoid mixing the masked and un-maskded tokens.

Metric We adopt the mean square error (MSE) and mean absolute error (MAE) for imputation.

C.4 CLASSIFICATION

Implementation Details Our method is trained with the Cross Entropy Loss, using the ADAM (Kingma & Ba, 2014) optimizer with an initial learning rate of 10^{-3}. The default training process is 30 epochs with proper early stopping. The classification accuracy is used as metrics. All the experiments are repeated 5 times with different seeds and the means of the metrics are reported as the final results.

Model Parameter By default, ModernTCN has 2 ModernTCN blocks. The channel number D is decided by $\min\{\max\{2^{\lceil \log M \rceil}, d_{\text{min}}\}, d_{\text{max}}\}$ (d_{min} is 32 and d_{max} is 512) following (Wu et al., 2023). The FFN ratio is $r = 1$. Patch size and stride are set as $P = 1, S = 1$ in the patchify embedding process.

Metric For classification, we calculate the accuracy as metric.

C.5 ANOMLY DETECTION

Implementation Details We takes the classical reconstruction task and train it with the L2 loss. We use the ADAM (Kingma & Ba, 2014) optimizer with an initial learning rate of 3×10^{-4}. The default training process is 10 epochs with proper early stopping. We use the reconstruction error (MSE) as the anomaly criterion. The F1-Score is used as metric. All the experiments are repeated 5 times with different seeds and the means of the metrics are reported as the final results.

Model Parameter By default, ModernTCN has 1 ModernTCN block. The channel number D is decided by $\min\{\max\{2^{\lceil \log M \rceil}, d_{\text{min}}\}, d_{\text{max}}\}$ (d_{min} is 8 and d_{max} is 256) following (Wu et al., 2023). The FFN ratio is $r = 1$. The kernel size is set as $large\ size = 51$ and $small\ size = 5$. Patch size and stride are set as $P = 8, S = 4$ in the patchify embedding process.

Table 11: Impact of channel number. We conduct experiments with three different channel numbers ranging from $D = \{32, 64, 128\}$. A lower MSE or MAE indicates a better performance.

Datasets		ILI Default $K = 1, r = 1$				ETTh1 Default $K = 1, r = 1$				Electricity Default $K = 1, r = 8$			
Prediction length		24	36	48	60	96	192	336	720	96	192	336	720
$D = 32$	MSE	1.772	1.598	1.725	1.976	0.377	0.412	0.395	0.453	0.132	0.145	0.159	0.193
	MAE	0.857	0.873	0.866	0.953	0.401	0.419	0.416	0.464	0.227	0.241	0.256	0.286
$D = 64$	MSE	1.347	1.250	1.388	1.774	0.368	0.405	0.391	0.450	0.129	0.143	0.161	0.191
	MAE	0.717	0.778	0.781	0.868	0.394	0.413	0.412	0.461	0.226	0.239	0.259	0.286
$D = 128$	MSE	2.010	1.751	1.378	1.806	0.364	0.402	0.387	0.449	0.135	0.147	0.168	0.196
	MAE	0.913	0.932	0.792	0.935	0.390	0.410	0.407	0.459	0.236	0.247	0.265	0.290

Table 12: Impact of FFN ratio. We conduct experiments with four different FFN ratios ranging from $r = \{1, 2, 4, 8\}$. A lower MSE or MAE indicates a better performance.

Datasets		ILI Default $K = 1, D = 64$				Exchange Default $K = 1, D = 64$				ETTh1 Default $K = 1, D = 64$				ETTm1 Default $K = 3, D = 64$			
Prediction length		24	36	48	60	96	192	336	720	96	192	336	720	96	192	336	720
$r = 1$	MSE	1.347	1.250	1.388	1.774	0.080	0.166	0.307	0.656	0.368	0.405	0.391	0.450	0.294	0.335	0.369	0.419
	MAE	0.717	0.778	0.781	0.868	0.196	0.288	0.398	0.582	0.394	0.413	0.412	0.461	0.346	0.369	0.392	0.421
$r = 2$	MSE	2.083	1.480	1.940	1.758	0.080	0.167	0.306	0.657	0.368	0.407	0.392	0.450	0.291	0.332	0.365	0.417
	MAE	0.943	0.820	0.946	0.890	0.196	0.289	0.397	0.583	0.395	0.415	0.413	0.461	0.345	0.368	0.392	0.415
$r = 4$	MSE	1.877	1.589	1.401	2.042	0.080	0.167	0.308	0.659	0.367	0.411	0.395	0.453	0.292	0.333	0.365	0.416
	MAE	0.887	0.853	0.790	0.980	0.197	0.289	0.399	0.586	0.393	0.419	0.415	0.463	0.346	0.368	0.390	0.417
$r = 8$	MSE	2.038	1.657	1.577	1.937	0.080	0.167	0.309	0.660	0.369	0.409	0.401	0.459	0.292	0.332	0.365	0.416
	MAE	0.932	0.869	0.858	0.960	0.196	0.289	0.400	0.586	0.395	0.417	0.421	0.465	0.346	0.368	0.391	0.417

Metric For anomaly detection, we adopt the F1-score, which is the harmonic mean of precision and recall.

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

D More Ablation Studies

We conduct more ablation studies in long-term forecasting tasks.

D.1 Results with Different Model Parameters

To see whether ModernTCN is sensitive to the choice of model parameters, we perform experiments with varying model parameters, including number of layers (number of ModernTCN blocks) ranging from $K = \{1, 2, 3, 4, 5\}$, channel number (dimension of hidden states) ranging from $D = \{32, 64, 128\}$ and FFN ratio ranging from $r = \{1, 2, 4, 8\}$. In general, except ILI dataset reveals high variance with different model parameter settings, other datasets are robust to the choice of model parameters. We conduct three experiments to figure out the impact of above three model parameters respectively. Detailed results are described in following paragraphs.

Results with Different Channel Numbers Table 11 shows the impact of different channel numbers D. Considering both the parameter efficiency and forecasting performance, we set the default channel number as $D = 64$. And the default channel number $D = 64$ works well for most of the datasets.

Results with Different FFN Ratios Table 12 shows the impact of different FFN Ratios r. Except for ILI dataset, our model is robust to the choice of the FFN ratio r in other datasets. We recommend $r = 8$ for most of the datasets. And for small datasets like ETTh1, ETTh2, Exchange and ILI, we recommend a small FFN ratio like $r = 1$ to mitigate the possible overfitting and for better memory efficiency.

Results with Different Numbers of Layers Table 13 shows the impact of different numbers of layers (numbers of ModernTCN blocks) K. Considering both performance and efficiency, one ModernTCN block is enough for most of the datasets. But for bigger datasets like ETTm1 and ETTm2, we recommend to stack more ModernTCN blocks like $K = 3$ for better representation capability.