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

Donghao Luo, Xue Wang

Department of Precision Instrument, Tsinghua University, Beijing, China

ICLR 2024 spotlight

BrAIn Lab. Brain and Artificial Intelligence Lab.

SEOULTECH SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

Author

Donghao Luo

PhD student, Department of Precision Instrument, Tsinghua University, Tsinghua University

Joined November 2022

Names

Donghao Luo (Preferred), Luo donghao

Emails

****@mails.tsinghua.edu.cn (Confirmed), ****@163.com (Confirmed), ****@163.com (Confirmed)

Personal Links

Homepage Google Scholar

Career & Education History

PhD student Department of Precision Instrument, Tsinghua University, Tsinghua University *2021 – Present*
(mails.tsinghua.edu.cn)

Advisors, Relations & Conflicts

PhD Advisor Xue Wang *Present*

Expertise

No areas of expertise listed

Publications

AnomalyTCN: Dual-branch Convolution with Contrastive Representation for Efficient Time Series Anomaly Detection
Donghao Luo, Xue Wang
23 Sept 2024 (modified: 24 Nov 2024)
ICLR 2025 Conference Withdrawn Submission

DeformableTST: Transformer for Time Series Forecasting without Over-reliance on Patching
Donghao Luo, Xue Wang
Published: 26 Sept 2024, Last Modified: 06 Nov 2024
NeurIPS 2024 poster

DeformableTST: Transformer for Time Series Forecasting without Over-reliance on Patching
Donghao Luo, Xue Wang
Published: 01 Jan 2024, Last Modified: 13 May 2025
NeurIPS 2024

ModernTCN: A Modern Pure Convolution Structure for General Time Series Analysis
Donghao Luo, Xue Wang
Published: 01 Jan 2024, Last Modified: 13 May 2025 ICLR 2024

ModernTCN: A Modern Pure Convolution Structure for General Time Series Analysis
Luo donghao, wang xue
Published: 16 Jan 2024, Last Modified: 22 Mar 2024
ICLR 2024 spotlight

Introduction

- **Convolution-based models for time-series** 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 are widely-used in many time series tasks.
- But in **2020s**, **Transformer**-based models and **MLP**-based models have emerged rapidly and achieved impressive performance. They can better capture the **long-term temporal dependency** and outperform TCNs by a significant margin.
- As a result, convolution-based models are losing steam nowadays due to their **limited effective receptive fields (ERFs)**.

Introduction

- Some previous convolution-based models mainly **focus on designing extra sophisticated structures** to work with the traditional convolution, **ignoring the importance of updating the convolution itself**, and still cannot achieve comparable performance to the state-of-the-art models.
- The reason is previous convolution-based models still have limited ERFs, which prevents their further performance improvements.

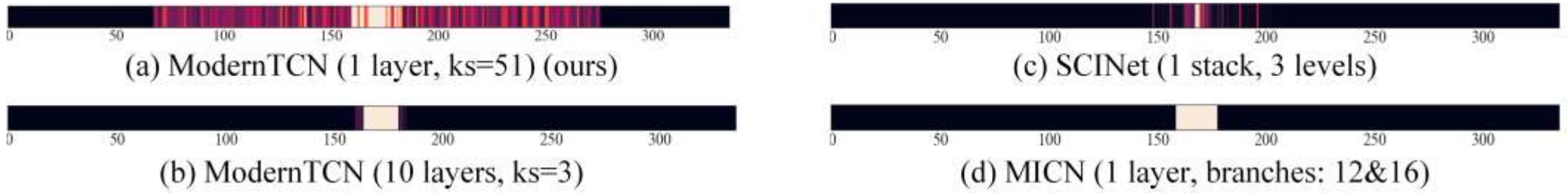
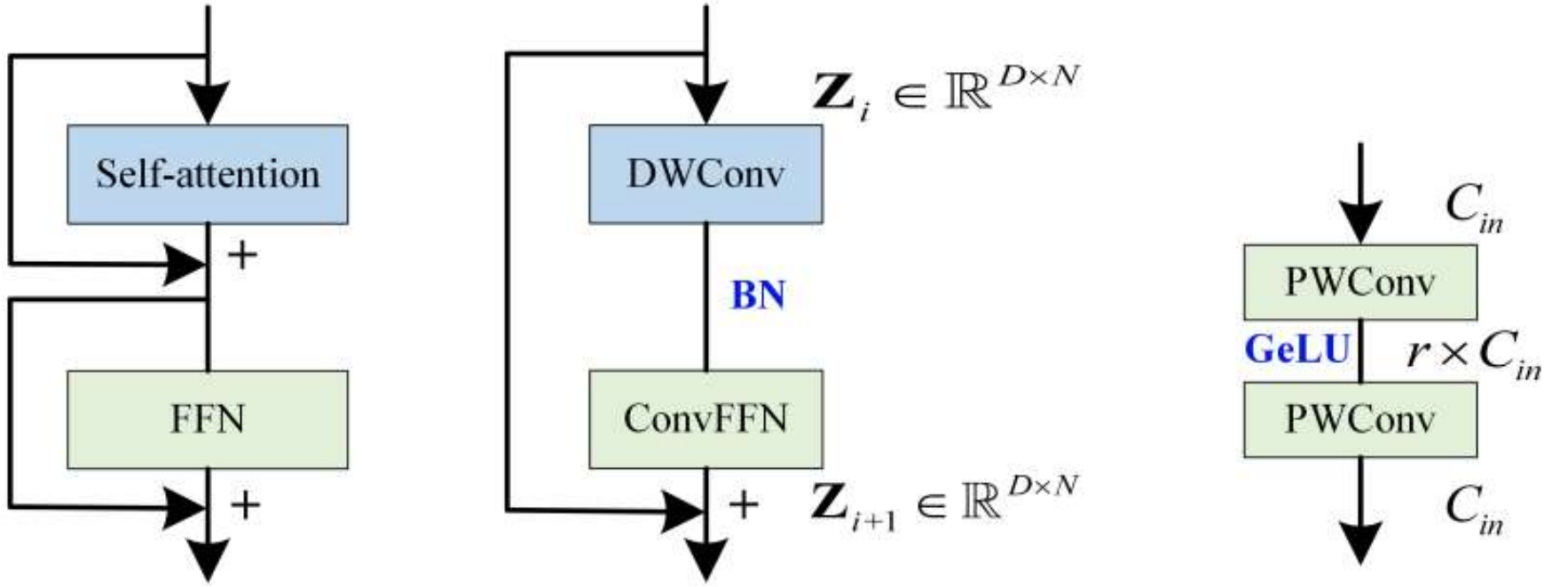


Figure 1: The Effective Receptive Field (ERF) of ModernTCN and previous convolution-based methods. A more widely distributed light area indicates a larger ERF. Our ModernTCN can obtain a much larger ERF than previous convolution-based methods. Meanwhile, enlarging the kernel size is a more effective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

Introduction

- Computer vision (CV) took a very different path to explore the convolution. Unlike recent studies in time series community, latest studies in CV focus on optimizing the convolution itself and propose **modern convolution**.



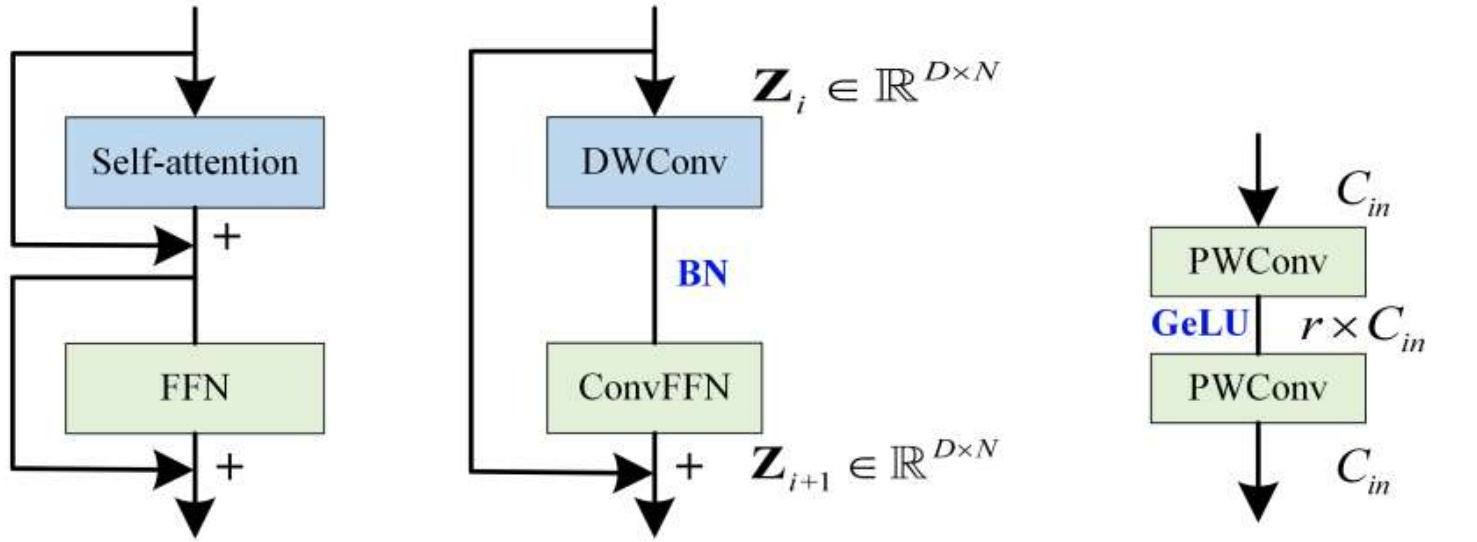
(a) Transformer block (b) modern convolution block (c) Structure of ConvFFN

Introduction

- Objective of the study: dive into the question of how to better use convolution in time series and propose a novel solution.
- Approach
    1. Modernize the 1D convolution block
    2. Make time series related modification

Modernize the 1D convolution block

- **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.
- **ConvFFN** is similar to the FFN module in Transformer. It consists of two PWConvs. This module is to learn the new feature representation of each token independently.



(a) Transformer block (b) modern convolution block (c) Structure of ConvFFN

Time-series Related Modification

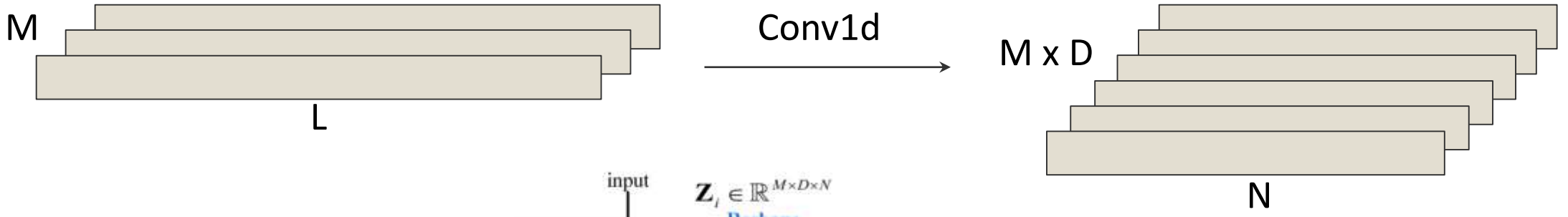
Maintaining the Variable Dimension

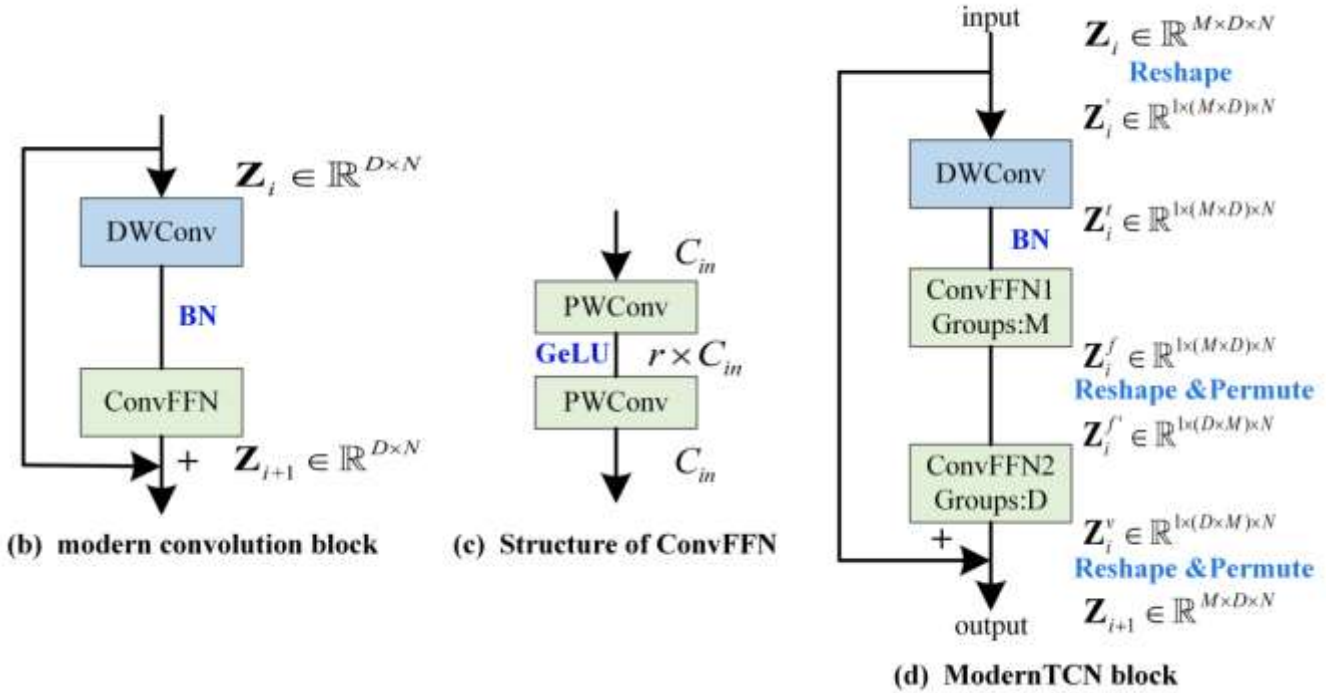
- **In CV**, before the backbone, we embed **3 channel RGB** features at each pixel into a D-dimensional vector to mix the information from RGB channels via the embedding layer.
- But the **difference among variables in time series is much greater** than that among RGB channels in a picture.
- Such embedding design also leads to the **discard of variable dimension**, making it unable to further study the cross-variable dependency.
- Proposed method: **patchify variable-independent embedding**.

Time-series Related Modification

Maintaining the Variable Dimension

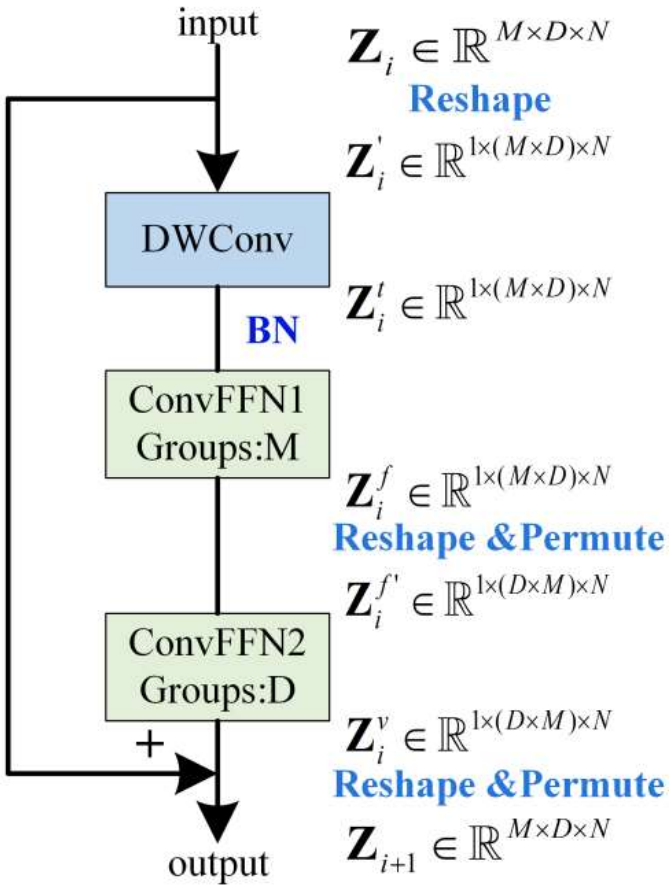
- Patchify embedding: Transforming time-series input **M x L** into **(M x D) x N**





- **DWConv** is modified to be **feature and variable independent**, making it learn the temporal dependency of each univariate time series independently.
- A **large kernel** is adopted in DWConv to **increase ERFs** and improve the temporal modeling ability.

Time-series Related Modification



(d) ModernTCN block

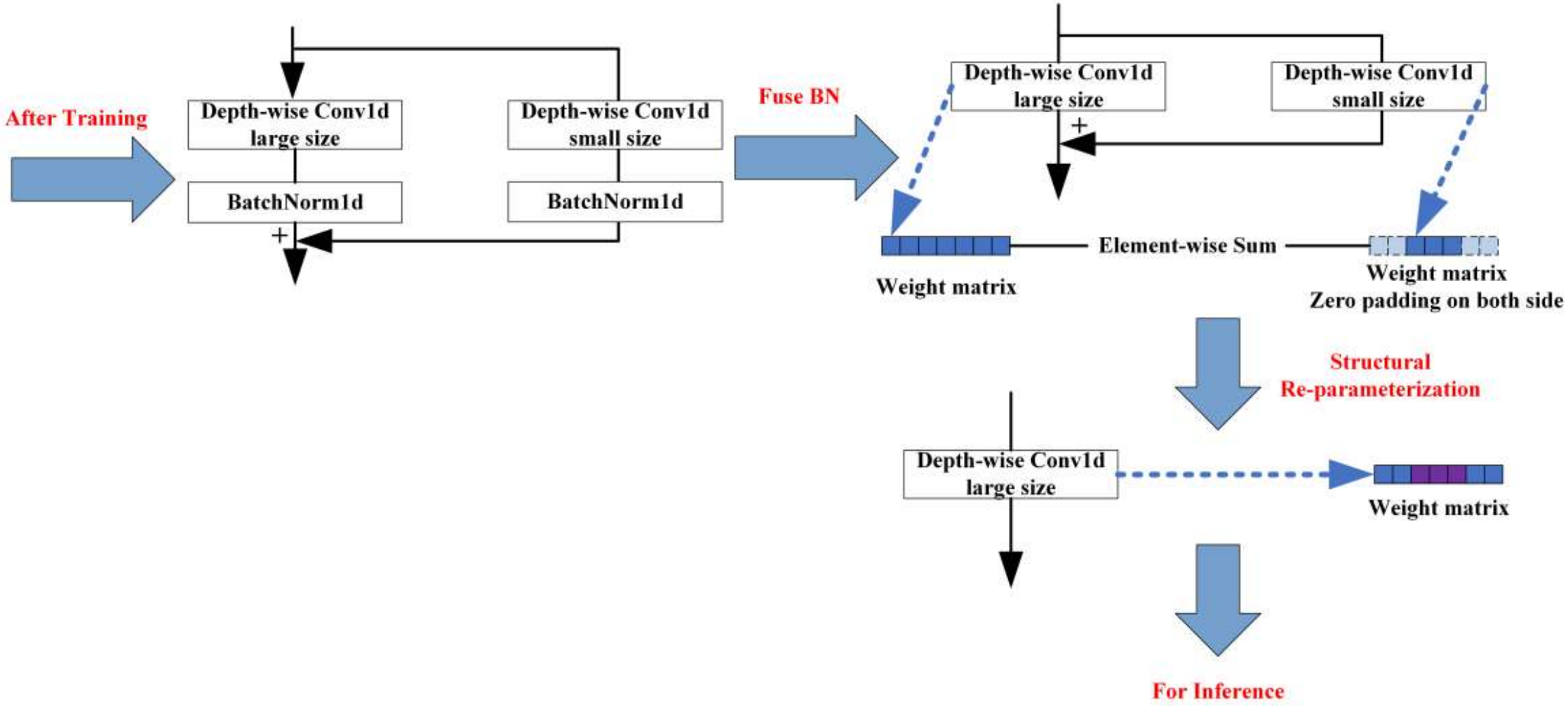
ConvFFN

```
#convffn1
self.ffn1pw1 = nn.Conv1d(in_channels=nvars * dmodel, out_channels=nvars * dff, kernel_size=1, stride=1,
                         padding=0, dilation=1, groups=nvars)
self.ffn1act = nn.GELU()
self.ffn1pw2 = nn.Conv1d(in_channels=nvars * dff, out_channels=nvars * dmodel, kernel_size=1, stride=1,
                         padding=0, dilation=1, groups=nvars)
self.ffn1drop1 = nn.Dropout(drop)
self.ffn1drop2 = nn.Dropout(drop)

#convffn2
self.ffn2pw1 = nn.Conv1d(in_channels=nvars * dmodel, out_channels=nvars * dff, kernel_size=1, stride=1,
                         padding=0, dilation=1, groups=dmodel)
self.ffn2act = nn.GELU()
self.ffn2pw2 = nn.Conv1d(in_channels=nvars * dff, out_channels=nvars * dmodel, kernel_size=1, stride=1,
                         padding=0, dilation=1, groups=dmodel)
self.ffn2drop1 = nn.Dropout(drop)
self.ffn2drop2 = nn.Dropout(drop)
```

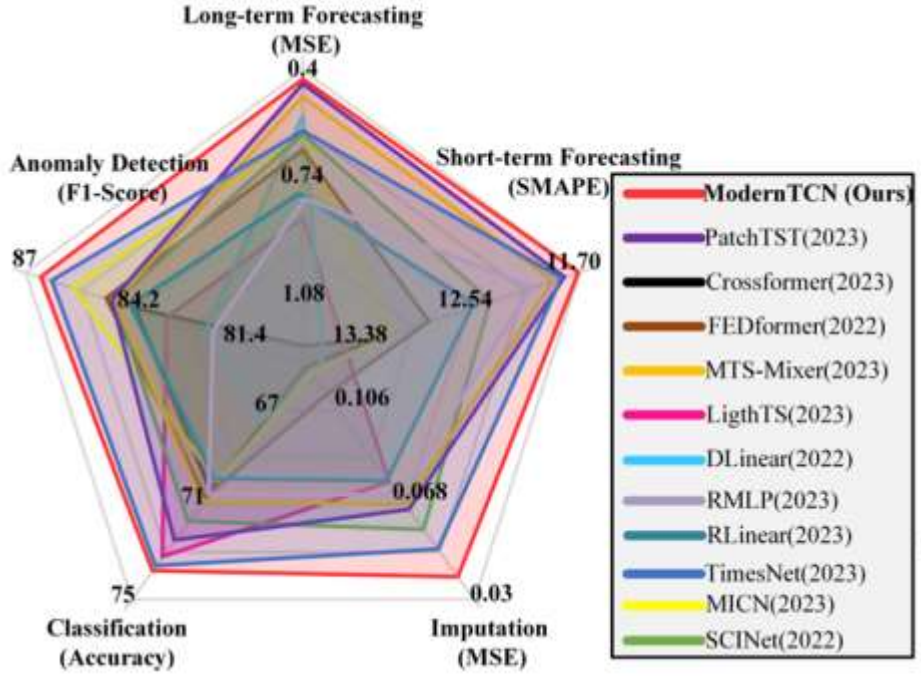
Technique to Better Train A Large Kernel

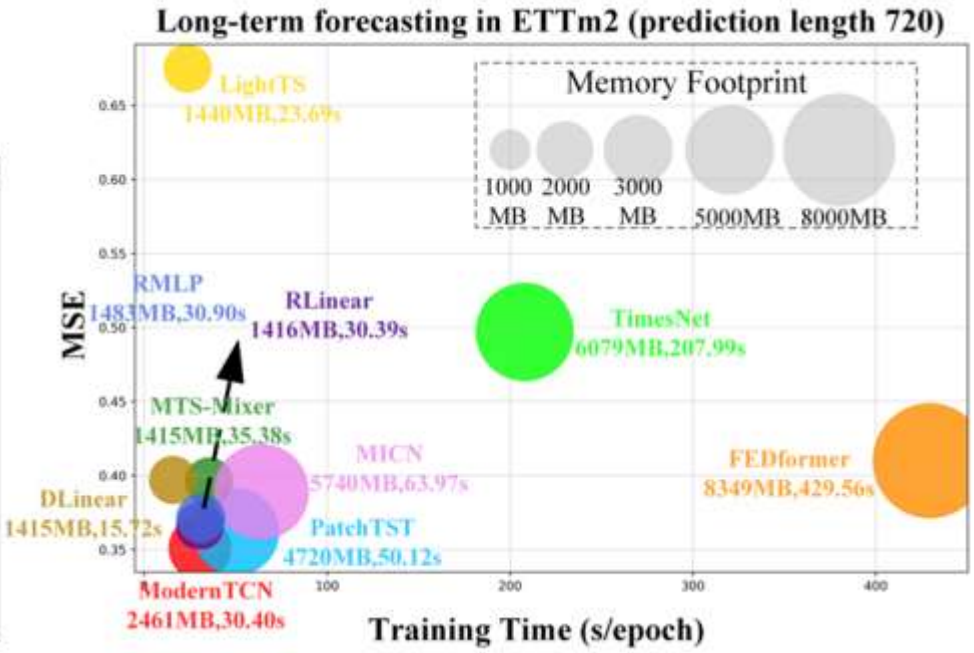
Structural Re-parameterization



Experiments

- ModernTCN was evaluated on five analysis tasks, including long-term and short-term forecasting, imputation, classification and anomaly detection.





- ModernTCN achieves consistent state-of-the-art performance on five mainstream analysis tasks with higher efficiency.

Long-term 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

Short-term forecasting

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

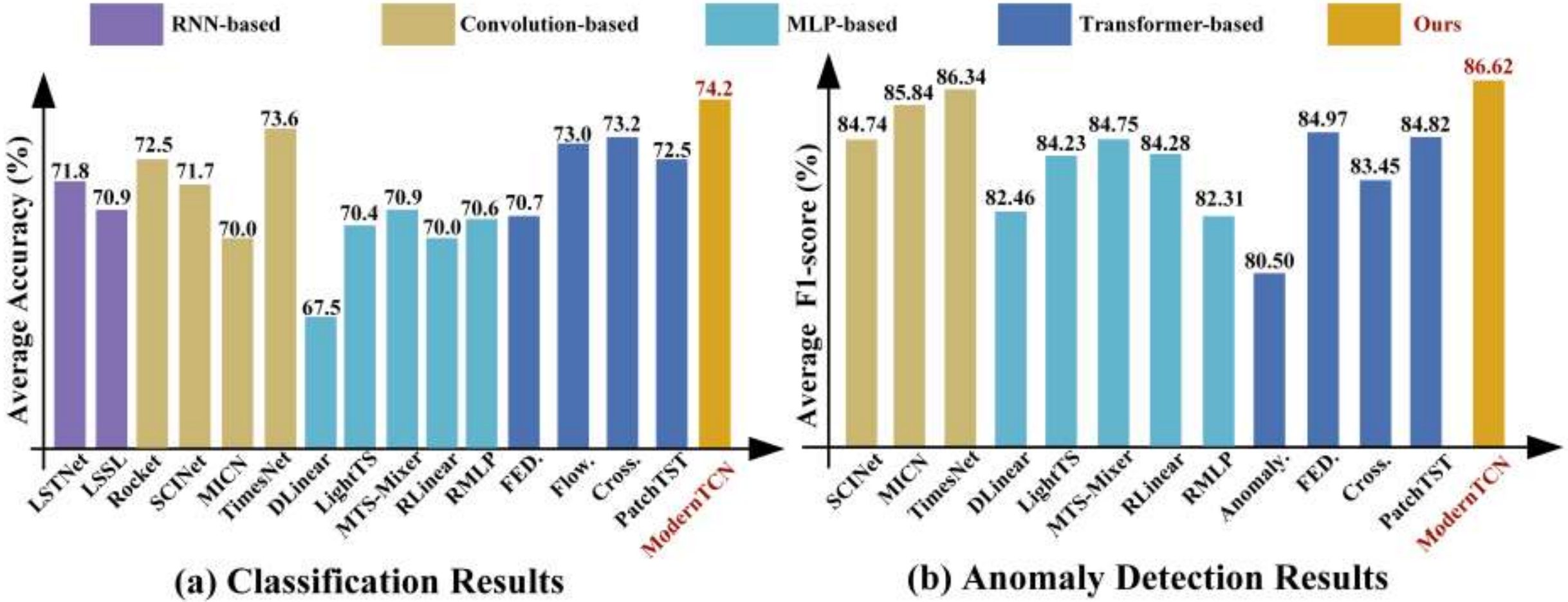
Time-series Related Modification

- Imputation task aims to impute the missing values based on the partially observed time series.

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

Classification and anomaly detection



(a) Classification Results

(b) Anomaly Detection Results

Impact of kernel size

Table 16: Impact of kernel size. We compare three different kernel sizes ranging from small to large. A lower MSE or MAE indicates a better performance. The best results are highlighted in **blod**.

Datasets		ILI				ETTh1				Electricity			
Prediction length		24	36	48	60	96	192	336	720	96	192	336	720
kernel size = 3	MSE	1.906	1.546	1.754	1.893	0.381	0.416	0.403	0.460	0.143	0.155	0.175	0.203
	MAE	0.862	0.841	0.891	0.900	0.405	0.423	0.419	0.470	0.237	0.249	0.270	0.299
kernel size = 31	MSE	1.687	1.486	1.376	1.855	**0.367**	**0.405**	**0.389**	**0.449**	0.133	0.147	**0.159**	0.193
	MAE	0.848	0.855	0.797	0.929	**0.393**	**0.413**	**0.410**	**0.460**	0.228	0.243	**0.257**	0.288
kernel size = 51	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**

Summary and Findings

- ModernTCN achieves consistent state-of-the-art performance on five mainstream analysis tasks, demonstrating its **excellent task-generality** and highlighting the **potential of convolution in time series analysis**.
- 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**.
- ModernTCN outperforms all MLP-based baselines in all five tasks. 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.**
- In a pure convolution structure, enlarging the kernel size is a much more effective way to increases ERF. Therefore, **by enlarging the kernel size, ModernTCN can easily obtain a larger ERF and further bring performance improvement.**