ModernTCN Revisited: A Reproducibility Study with Extended Benchmarks

Anonymous authors
Paper under double-blind review

Abstract

This study presents a reproducibility analysis of ModernTCN, a recently proposed convolutional architecture for time series analysis. ModernTCN aims to address the limitations of traditional Temporal Convolutional Networks (TCNs) by enhancing the effective receptive field (ERF) and capturing long-range dependencies. We validate the experimental setup and performance claims of the original paper, and extend the evaluation to include additional datasets and tasks, such as short-term forecasting on ETT, classification on Speech Commands and PhysioNet, and ablation studies on the cross-variable component. Our results show that while ModernTCN achieves competitive performance, its state-of-the-art claims are tempered by sensitivity to experimental settings and data handling. Furthermore, ModernTCN's performance on Speech Commands lags behind convolutional methods with global receptive fields, and it exhibits less parameter efficiency. However, ablation studies on the PhysioNet dataset confirm the importance of the cross-variable component in handling missing data. This study provides a comprehensive evaluation of ModernTCN's contributions, reproducibility, and generalizability in time series analysis.

1 Introduction

Time series analysis is a fundamental problem with broad applications across various domains, including weather forecasting (Bi et al., 2023), anomaly detection in spacecraft monitoring (Su et al., 2019b), medical symptom classification (Kiyasseh et al., 2021), and missing data imputation (Luo et al., 2018). ModernTCN (Luo & Wang, 2024) is a general time series model that can be applied across these diverse tasks. It modernizes the traditional Temporal Convolutional Network (TCN) by enlarging kernel size, drawing inspiration from computer vision advances (Liu et al., 2022a; Ding et al., 2022a). Additionally, it proposes a block structure inspired by Transformer architectures (Vaswani et al., 2017). The model also incorporates techniques from transformer-based time series models, such as patching and variable independence approaches (Nie et al., 2023).

ModernTCN (Luo & Wang, 2024) claims state-of-the-art performance across five time series tasks: long-term forecasting, short-term forecasting, imputation, classification, and anomaly detection. The primary purpose of this study is to validate these claims through comprehensive reproducibility experiments, examining the model's effectiveness and generalizability

Luo & Wang (2024) posit that convolution-based models have lost prominence in the 2020s due to their limited effective receptive fields (ERFs), which restrict their ability to capture long-term dependencies. However, this perspective overlooks significant advancements in convolutional models that emerged earlier than ModernTCN. Works such as CKConv (Romero et al., 2021b), S4 (Gu et al., 2021), and others (Romero et al., 2021a; Knigge et al., 2023; Li et al., 2023; Poli et al., 2023; Shi et al., 2023; Qin et al., 2023) have demonstrated convolutional models with global receptive fields, extending even to 4 million tokens (Fu et al., 2023; Nguyen et al., 2023). To bridge this gap, our study includes comparisons with several of these overlooked convolutional approaches (specifically CKConv, FlexConv, CCNN, and S4) to evaluate how ModernTCN's performance and receptive field capabilities compare in practice.

The contributions of this study are:

- Validating the experimental setup and performance claims of ModernTCN across multiple time series tasks, identifying issues with data handling and experimental methodology.
- Extending experiments to bridge the gap between ModernTCN and convolutional models with global receptive fields.
- Conducting ablation studies on irregularly sampled data to verify the importance of the cross-variable component in handling missing values.
- Providing implementations[1] for effective receptive field (ERF) visualizations that were missing in the original source code[2].

2 Scope of Reproducibility

This study focuses on the reproducibility and validation of the claims made by the ModernTCN paper (Luo & Wang, 2024). The scope of this study includes:

- **State-of-the-Art Performance** - ModernTCN claims to achieve state-of-the-art results across time series tasks. To validate, we scrutinize the experimental setup and reproduce the results.
- **Enhanced Effective Receptive Fields (ERFs)** - ModernTCN claims that it effectively models long-range dependencies by enlarging the kernel size, thereby significantly increasing the ERFs compared to previous convolution-based models. To validate, we visualize the ERF and compare it with a convolutional model that has a global receptive field.
- **Efficiency and Performance Balance** - ModernTCN claims to provide a better balance of efficiency and performance compared to transformer-based and MLP-based models. We extend the comparison to convolution-based models with global receptive field.
- **Handling Missing Data** - ModernTCN claims its cross-variable component (ConvFFN2) effectively captures dependencies between missing and remaining variables in imputation tasks. To validate, we ablate the component on irregularly sampled time series classification.

By addressing these claims, the study aims to provide a comprehensive evaluation of ModernTCN's contributions to the field of time series analysis, ensuring the reproducibility and generalizability of its reported results.

3 Methodology

3.1 ModernTCN

Figure 1: Diagram of the ModernTCN block. Adapted from Luo & Wang (2024).

ModernTCN (Luo & Wang, 2024) introduces a convolutional approach with large kernel sizes inspired by advances in computer vision (Liu et al., 2022c; Ding et al., 2022b; Liu et al., 2022b). It leverages modern convolution techniques, incorporating depthwise and pointwise convolution layers organized similarly to Transformer blocks (Vaswani et al., 2017). The depthwise convolution (DWConv) layer is responsible for learning temporal information among tokens on a per-channel basis, akin to the self-attention module in

[1]The official source code for this study can be found in the supplementary materials.
[2]The official source code for ModernTCN can be found at `https://github.com/luodhhh/ModernTCN`

Transformers. A large kernel is used in DWConv to enhance the effective receptive field, allowing the model to capture long-term dependencies more effectively. The ConvFFN module consists of two pointwise convolution layers that adopt an inverted bottleneck structure. This module is applied twice in the architecture, first along feature dimensions and then between variables, to learn new feature representations.

The pipeline of ModernTCN is as follows:

1. **Input**: The process begins with the input time series $\mathbf{X}_{\text{in}} \in \mathbb{R}^{M \times L}$, where M is the number of variables and L is the input length.

2. **Patchify Embeddings**: The input is transformed into a higher-dimensional space by patching in a variable-independent manner, resulting in $\mathbf{X}_{\text{emb}} \in \mathbb{R}^{M \times D \times N}$, where D is the feature dimension and N is the number of patches.

3. **Backbone**: As shown in Figure 1 each ModernTCN block consists of:

   - **DWConv Layer**: Processes temporal information independently for each variable and each feature, maintaining dimensions $M \times D$.
   - **ConvFFN Module**: This module consists of two pointwise convolution layers and adopts an inverted bottleneck structure. It is applied twice:
     - **ConvFFN1**: Processes each variable independently with groups=M, learning new feature representations independently for each variable. Until this point, the entire pipeline maintains variable independence.
     - **ConvFFN2**: Processes across variables with groups=D, capturing dependencies between different variables. This is the first component in the pipeline that allows cross-variable information mixing.

   These blocks are stacked and organized in a residual manner as shown in Figure 1.

4. **Flatten and Prediction Head**:

   - **Forecasting**: The output is reshaped and passed through a prediction head to produce the final time series output $\hat{\mathbf{X}} \in \mathbb{R}^{M \times T}$, where T is the prediction length.
   - **Imputation and Anomaly Detection**: The output is reshaped to $\hat{\mathbf{X}} \in \mathbb{R}^{M \times L}$.
   - **Classification**: The representation is flattened to and passed through a projection layer with a SoftMax activation to yield the classification result $\hat{\mathbf{X}} \in \mathbb{R}^{1 \times \text{Cls}}$, where Cls is the number of classes.

5. **RevIN**: For all tasks except classification, Stationary Technique RevIN (Kim et al., 2021) is applied. It normalizes the input time series per variable with zero mean and unit standard deviation before patching and embedding. After the forward process, the mean and deviation are added back to the final prediction per variable.

This architecture leverages the decoupling of temporal, feature, and variable dimensions to improve both performance and efficiency in time series analysis, while the enlarged kernel size in DWConv enhances its ability to capture long-term dependencies.

3.2 Tasks and Datasets

Time series analysis encompasses a variety of tasks, each designed to extract different insights from sequential data. This study focuses on five primary tasks: long-term forecasting, short-term forecasting, imputation, classification, and anomaly detection. An overview of all the datasets used is given in Table 1.

Long-term forecasting involves predicting future values over an extended time horizon, often requiring the model to capture complex temporal dependencies. ModernTCN (Luo & Wang, 2024) uses datasets such as ETT (Zhou et al., 2021), Electricity (UCI), Traffic (PeM), Weather (Wet), Exchange (Lai et al., 2018),

Table 1: Dataset descriptions. The dataset size is organized in (train, validation, test). The column length refers to prediction length for forecasting tasks, input length for imputation, series length for classification, and sliding window length for anomaly detection. Partially adapted from Wu et al. (2023).

Tasks	Dataset	Dim	Length	Dataset Size	Information (Frequency)
Long-term forecasting	ETTm1, ETTm2	7	{96, 192, 336, 720}	(34465, 11521, 11521)	Electricity (15 mins)
	ETTh1, ETTh2	7	{96, 192, 336, 720}	(8545, 2881, 2881)	Electricity (hourly)
	Electricity	321	{96, 192, 336, 720}	(18317, 2633, 5261)	Electricity (hourly)
	Traffic	862	{96, 192, 336, 720}	(12185, 1757, 3509)	Transportation (hourly)
	Weather	21	{96, 192, 336, 720}	(36792, 5271, 10540)	Weather (10 mins)
	Exchange	8	{96, 192, 336, 720}	(5120, 665, 1422)	Exchange rate (daily)
	ILI	7	{24, 36, 48, 60}	(617, 74, 170)	Illness (weekly)
Short-term forecasting original	M4-Yearly	1	6	(23000, 0, 23000)	Demographic
	M4-Quarterly	1	8	(24000, 0, 24000)	Finance
	M4-Monthly	1	18	(48000, 0, 48000)	Industry
	M4-Weekly	1	13	(359, 0, 359)	Macro
	M4-Daily	1	14	(4227, 0, 4227)	Micro
	M4-Hourly	1	48	(414, 0, 414)	Other
Extended	ETTm1	7	{6, 12, 24}	(34465, 11521, 11521)	Electricity (15 mins)
	ETTh1	7	{6, 12, 24}	(8545, 2881, 2881)	Electricity (hourly)
Imputation	ETTm1, ETTm2	7	96	(34465, 11521, 11521)	Electricity (15 mins)
	ETTh1, ETTh2	7	96	(8545, 2881, 2881)	Electricity (15 mins)
	Electricity	321	96	(18317, 2633, 5261)	Electricity (15 mins)
	Weather	21	96	(36792, 5271, 10540)	Weather (10 mins)
Classification original	EthanolConcentration	3	1751	(261, 0, 263)	Alcohol industry
	FaceDetection	144	62	(5890, 0, 3524)	Face (250Hz)
	Handwriting	3	152	(150, 0, 850)	Handwriting
	Heartbeat	61	405	(204, 0, 205)	Heart beat
	JapaneseVowels	12	29	(270, 0, 370)	Voice
	PEMS-SF	963	144	(267, 0, 173)	Transportation (daily)
	SelfRegulationSCP1	6	896	(268, 0, 293)	Healthcare (256Hz)
	SelfRegulationSCP2	7	1152	(200, 0, 180)	Healthcare (256Hz)
	SpokenArabicDigits	13	93	(6599, 0, 2199)	Voice (11025Hz)
	UWaveGestureLibrary	3	315	(120, 0, 320)	Gesture
Extended	Speech Commands MFCC	20	161	(24483, 5246, 5246)	Voice(16000Hz)
	Speech Commands Raw	1	16000	(24483, 5246, 5246)	Voice(16000Hz)
	PhysioNet	75	72	(28235, 6050, 6050	Healthcare
Anomaly detection	SMD	38	100	(566724, 141681, 708420)	Server machine
	MSL	55	100	(44653, 11664, 73729)	Spacecraft
	SMAP	25	100	(108146, 27037, 427617)	Spacecraft
	SWaT	51	100	(396000, 99000, 449919)	Infrastructure
	PSM	25	100	(105984, 26497, 87841)	Server machine

and ILI (CDC), covering real-world applications from different domains. In these multivariate forecasting tasks, the model predicts future values for all variables in the time series simultaneously.

Short-term forecasting focuses on predicting values over a shorter time frame, typically requiring the model to capture immediate trends and patterns. ModernTCN uses the M4 dataset (Makridakis, 2018), which contains univariate time series with different frequencies from different domains. We extend the experiments to ETT (Zhou et al., 2021) by adapting it for short term multivariate forecasting.

Imputation aims to fill in missing values in a time series, which is crucial for handling incomplete data and ensuring data quality. ModernTCN (Luo & Wang, 2024) uses ETT (Zhou et al., 2021), Electricity (UCI), and Weather (Wet) datasets for this task.

Classification is sequence-level that verifies the model's capacity in high-level representation learning. ModernTCN (Luo & Wang, 2024) uses 10 multivariate datasets from the UEA Time Series Classification Archive (Bagnall et al., 2018). We extend to Speech Commands (Warden, 2018) and PhysioNet (Reyna et al., 2019), following the experimental setup from CKConv (Romero et al., 2021b). Speech Commands enables us to compare ModernTCN with other convolution-based models that has global receptive fields.

PhysioNet's irregular sampling and high proportion of missing values present a challenging classification task.

Anomaly detection seeks to identify unusual patterns or outliers in a time series, which is essential for detecting abnormal events and potential issues. ModernTCN (Luo & Wang, 2024) evaluates anomaly detection performance on SMD (Su et al., 2019a), MSL (Hundman et al., 2018), SMAP (Hundman et al., 2018), SWaT (Mathur & Tippenhauer, 2016), and PSM (Abdulaal et al., 2021), covering service monitoring, space & earth exploration, and water treatment applications.

3.3 Experimental Setup

For the reproduction of the experiments from ModernTCN, the optimal settings specified in the paper (Luo & Wang, 2024) and the source code[3] are used.

Drop Last Trick. As pointed out by Qiu et al. (2024), many implementations of existing methods often employ a "Drop Last Trick" during the testing phase (Nie et al., 2022; Wang et al., 2022; Zhou et al., 2022; 2021), which involves discarding the last batch if it contains fewer instances than the batch size. This approach may lead to unfair comparisons by excluding part of the test data. Upon reviewing the source code, we found this trick is employed for long-term forecasting and imputation tasks. Given that most datasets in the long-term forecasting experiments use a batch size of 512, this could result in discarding up to 511 data points from the test set. This is a significant number relative to the size of the test sets for long-term forecasting, as shown in Table 1. Therefore, we additionally experiment without employing this trick to assess its impact on the results and ensure a fair comparison.

Data Leakage from Test to Validation. As detailed in Table 1, the M4 dataset (Makridakis, 2018) for short-term forecasting and the UEA dataset (Bagnall et al., 2018) for classification are provided with only train and test splits. Upon reviewing the source code, we identify that the test set is used for validation. This practice introduces data leakage as the model gains indirect exposure to the test data, leading to an overestimation of its generalization capabilities. Consequently, the reproduction results for these datasets are excluded from the main results section and are instead presented in Appendix C.1 and Appendix C.2. Notably, this issue is not isolated to the ModernTCN implementation but also appears in the source code for TimesNet (Wu et al., 2023), from which ModernTCN's implementation is partially derived.

Point Adjustment in Anomaly Detection. Reconstruction-based anomaly detection identifies anomalies when reconstruction errors between input sequences and their model-generated reconstructions exceed a threshold. ModernTCN uses sliding windows of 100 timestamps as input sequences, but calculates the threshold using both training and test sets, introducing information leakage. We reproduce the experiments using thresholds calculated from training data only. Additionally, the evaluation employs "point adjustment" (PA), which considers an entire anomaly segment correctly detected if just one point within it is flagged - a protocol that Kim et al. (2022) demonstrated can severely overestimate performance. To illustrate this limitation, we implemented a naive baseline that periodically flags every 100th timepoint as anomalous, which should perform well under PA despite having no actual anomaly detection capability.

Extending short-term forecasting to ETT. Inspired by the appendices in Luo & Wang (2024), we evaluate short-term multi-variate forecasting on the ETT dataset (Zhou et al., 2021). While ModernTCN (Luo & Wang, 2024) keeps the input sequence length constant at 2x the prediction length, we explored the input sequence length as a hyperparameter, experimenting with 2x, 3x, and 4x the prediction length. We consider three prediction lengths: 6, 12, and 18 time points, selecting the best-performing input length for reporting. We employ the optimal hyperparameters from the long-term forecasting experiments for these datasets, adjusting only the input and prediction lengths. For comparison, based on the long-term forecasting results without the "Drop Last Trick" from Qiu et al. (2024), we decided to include three top-performing models: TimesNet (convolution-based) (Wu et al., 2023), PatchTST (transformer-based) (Nie et al., 2023), and DLinear (MLP-based) (Zeng et al., 2023). Each model is run with its respective optimal long-term forecasting parameters, with adjustments made to the input and prediction lengths.

[3]The official source code for ModernTCN can be found at `https://github.com/luodhhh/ModernTCN`

Extending classification to Speech Commands. For the classification task, we employ the Speech Commands dataset (Warden, 2018). This dataset has two versions: Speech Commands Raw and Speech Commands MFCC. The Speech Commands Raw version consists of 105,809 one-second audio recordings of 35 spoken words sampled at 16kHz. Following the preprocessing steps of Kidger et al. (Kidger et al., 2020), we extract 34,975 recordings from ten spoken words to construct a balanced classification problem. The Speech Commands MFCC utilizes mel-frequency cepstrum coefficients extracted from the raw data, resulting in time series of length 161 and 20 channels. This dataset is frequently used by convolution-based models with global receptive fields (Romero et al., 2021b;a; Knigge et al., 2023; Gu et al., 2021).

Extending to irregularly sampled data. The PhysioNet 2019 challenge on sepsis prediction provides an irregularly sampled, partially observed dataset with 40,335 time series of variable lengths, capturing ICU patient data (Reyna et al., 2019). Each series includes 5 static features, like age, and 34 dynamic features, such as respiration rate, with only 10.3% of values observed. Following Kidger et al. (2020); Romero et al. (2021b), we analyze the first 72 hours of patient data to predict sepsis development over their entire stay, which can last up to a month. This setup tests the model's ability to handle sparse and irregular data. ModernTCN (Luo & Wang, 2024) conducts an ablation study on the cross-variable component for imputation tasks, demonstrating significant performance degradation when this component is removed. To explore this component's impact on handling missing values, we perform the same ablation study on the PhysioNet dataset for classification.

Optimal parameters for extended datasets. For the optimal parameters on Speech Commands and PhysioNet, we adhere to the classification settings recommended in the appendix of ModernTCN (Luo & Wang, 2024). The models are trained using Cross Entropy Loss with the ADAM optimizer (Kingma & Ba, 2014), starting with an initial learning rate of 10^{-3}. The training process is conducted over 100 epochs with early stopping to prevent overfitting. While ModernTCN typically employs 2 blocks, we additionally experimented with 3 blocks and reported the configuration yielding the best performance. The channel number D is calculated as $\min\{\max\{2^{\lceil \log M \rceil}, d_{\min}\}, d_{\max}\}$, where $d_{\min}$ is 32 and $d_{\max}$ is 512. The FFN ratio is set to $r = 1$, with both the patch size and stride configured to $P = 1$ and $S = 1$ in the patchify embedding process. The ideal depthwise convolution kernel size is not specified for classification, so we explored kernel sizes of 13, 31, 51, and 71. These are the commonly utilized kernel sizes in ModernTCN (Luo & Wang, 2024).

All experiments are repeated 5 times with different seeds, and the mean of the corresponding metric is reported as the final result.

4 Experimental Results

4.1 Results Reproducing the Original Paper

4.1.1 Long-Term Forecasting

The results are presented in Table 2. Our rerun results generally align with those reported in the original paper, with minor differences likely attributable to unspecified library versions in the source code. However, when eliminating the "Drop Last Trick" during testing, performance notably degrades. For fair comparison, we included three top-performing models from Qiu et al. (2024): convolution-based TimesNet (Wu et al., 2023), transformer-based PatchTST (Nie et al., 2023), and MLP-based DLinear (Zeng et al., 2023). The results demonstrate that while ModernTCN shows competitive performance, PatchTST outperforms other models in most scenarios, achieving the best results in 11 out of 18 metrics across the datasets.

4.1.2 Imputation

ModernTCN (Luo & Wang, 2024) highlights the model's strong performance in imputation tasks, achieving significant reductions in MSE and MAE compared to previous baselines. Our rerun of the experiments yielded results consistent with the original findings (Table 3). Eliminating the "Drop Last Trick" produced results similar to both the rerun and reported outcomes, unlike in long-term forecasting. This consistency is

Table 2: Long-term forecasting task. All results averaged from 4 different prediction lengths: {24, 36, 48, 60} for ILI and {96, 192, 336, 720} for others. Lower MSE/MAE indicates better performance. Reported: scores from Luo & Wang (2024). Rerun: naively reproduced results. No Drop Last: results without the "Drop Last Trick". Bold indicates best performance and underline indicates second best performance among models without the "Drop Last Trick".

		ETTh1		ETTh2		ETTm1		ETTm2		Electricity		Weather		Traffic		Exchange		ILI	
Model	**1st/2nd**	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Reported	-	0.404	0.420	0.322	0.379	0.351	0.380	0.253	0.314	0.156	0.253	0.224	0.264	0.396	0.270	0.302	0.366	1.440	0.786
Rerun	-	0.404	0.421	0.322	0.379	0.355	0.383	0.255	0.319	0.159	0.256	0.224	0.267	0.401	0.274	0.305	0.368	1.490	0.797
No Drop Last	7/8	0.419	0.429	**0.347**	0.394	0.356	0.383	**0.254**	0.318	**0.163**	**0.259**	0.226	0.267	0.410	**0.280**	**0.343**	**0.392**	2.000	0.892
PatchTST	11/6	**0.411**	**0.428**	**0.347**	**0.389**	**0.349**	0.381	0.255	**0.313**	**0.163**	0.261	0.225	**0.262**	**0.405**	0.283	0.352	0.397	**1.770**	**0.859**
TimesNet	0/1	0.459	0.455	0.394	0.416	0.430	0.428	0.294	0.332	0.186	0.287	0.261	0.287	0.626	0.328	0.421	0.442	2.174	0.951
DLinear	2/3	0.420	0.432	0.492	0.478	0.354	**0.377**	0.259	0.324	0.167	0.264	**0.239**	0.290	0.434	0.295	0.349	0.411	2.185	1.040

likely due to the smaller test batch size in imputation tasks, which avoids dropping too many points in the case of an incomplete last batch.

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 better performance. Reported: scores from Luo & Wang (2024). Rerun: naively reproduced results. No Drop Last: results without the "Drop Last Trick".

	ETTh1		ETTh2		ETTm1		ETTm2		Electricity		Weather	
Model	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Reported	0.050	0.150	0.042	0.131	0.020	0.093	0.019	0.082	0.073	0.187	0.027	0.044
Rerun	0.050	0.150	0.042	0.131	0.021	0.094	0.020	0.083	0.073	0.186	0.027	0.044
No Drop Last	0.050	0.150	0.042	0.131	0.021	0.094	0.020	0.083	0.073	0.186	0.027	0.044

4.1.3 Anomaly Detection

As shown in Table 4, the rerun results generally align with those reported in the original paper. When eliminating data leakage by calculating thresholds using only training data, we observe a slight performance degradation. Most notably, our naive baseline—which simply flags every 100th timepoint as anomalous—outperforms ModernTCN across most datasets, achieving the highest average F1 score. This finding raises significant questions about the effectiveness of sophisticated models for anomaly detection when evaluated using point adjustment protocols, supporting the critique by Kim et al. (2022) that such evaluation methods can severely overestimate model performance.

Table 4: Anomaly Detection task. P: precision, R: recall, F1: F1-score (all values in %). Reported: scores from Luo & Wang (2024). Rerun: naively reproduced results. No Leakage: results calculating the thresholds using only training data. Naive: baseline approach. Bold: best score between valid models (No Leakage and Naive). Avg.: average F1 score across all 5 datasets.

Model	**SMD**			**MSL**			**SMAP**			**SWaT**			**PSM**			**Avg.**
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	
Reported	87.86	83.85	85.81	83.94	85.93	84.92	93.17	57.69	71.26	91.83	95.98	93.86	98.09	96.38	97.23	86.62
Rerun	87.43	81.64	84.44	89.59	74.94	81.61	90.82	55.93	69.23	95.77	90.28	92.94	98.65	94.57	96.57	84.96
No Leakage	78.42	83.76	81.00	86.07	77.12	81.35	92.31	55.42	69.26	**93.45**	93.16	93.30	**98.62**	**94.58**	**96.56**	84.29
Naive	**79.86**	**91.40**	**85.24**	**90.14**	**80.76**	**85.19**	**93.27**	**94.48**	**93.87**	93.05	**96.93**	**94.95**	97.32	94.55	95.91	**91.03**

4.2 Results Beyond the Original Paper

4.2.1 Short-Term Forecasting on ETT

Referring to the results presented in Table 5, ModernTCN exhibits competitive performance, particularly within the ETTm1 dataset, where it achieves the best results for very short-term forecasting (6x15m and 12x15m). This aligns with findings from Yang et al. (2024), which demonstrated ModernTCN's effectiveness

for ultra-short-term multi-step prediction tasks. However, when considering both ETTh1 and ETTm1 datasets across all horizons, PatchTST generally demonstrates superior performance for longer forecasting horizons.

Table 5: Short-Term Forecasting Results on ETT (ETTh1 and ETTm1). 6x1h, 12x1h, and 18x1h refer to steps ahead forecasting for ETTh1, while 6x15m, 12x15m, and 18x15m refer to steps ahead forecasting for ETTm1.

	6x15m		12x15m		18x15m		6x1h		12x1h		18x1h	
Model	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ModernTCN	**0.127**	**0.216**	**0.204**	**0.269**	0.220	0.296	0.300	0.350	0.300	0.346	0.308	0.351
PatchTST	0.240	0.312	0.244	0.295	**0.139**	**0.222**	**0.253**	**0.321**	**0.278**	**0.333**	**0.288**	**0.344**
TimesNet	0.129	0.218	0.231	0.295	0.256	0.326	0.316	0.367	0.337	0.383	0.348	0.393
DLinear	0.146	0.233	0.287	0.323	0.303	0.347	0.284	0.339	0.292	0.341	0.300	0.347

4.2.2 Classification on Speech Commands

ModernTCN was evaluated on the Speech Commands dataset using the recommended parameters from Luo & Wang (2024). Since the optimal DWConv kernel size for classification tasks wasn't specified, we explored sizes of 13, 31, 51, and 71. As shown in Table 6, performance doesn't show a clear trend across the selected kernel sizes. The best results (kernel size 13 for Speech Commands and 31 for Speech Commands Raw) are used in Table 7 for comparison with other models.

Table 6: Performance of Different Kernel Sizes on Speech Commands - Accuracy

	13	**31**	**51**	**71**
Speech Commands MFCC	**0.837**	0.825	0.794	0.818
Speech Commands Raw	0.384	**0.400**	**0.399**	0.388

The classification results, as shown in Table 7, indicate that ModernTCN's performance consistently lags behind continuous kernel convolution methods (Romero et al., 2021b;a; Knigge et al., 2023) and the S4 model (Gu et al., 2021) on the Speech Commands dataset. ModernTCN also exhibits less parameter efficiency, as evidenced by the higher parameter counts compared to the other methods in Table 7. However, in terms of training time, ModernTCN is faster overall, taking roughly 12 seconds per epoch and finishing around 35 epochs with early stopping, compared to CCNN{4,140} Knigge et al. (2023), which takes 13 seconds per epoch but requires around 110 epochs.

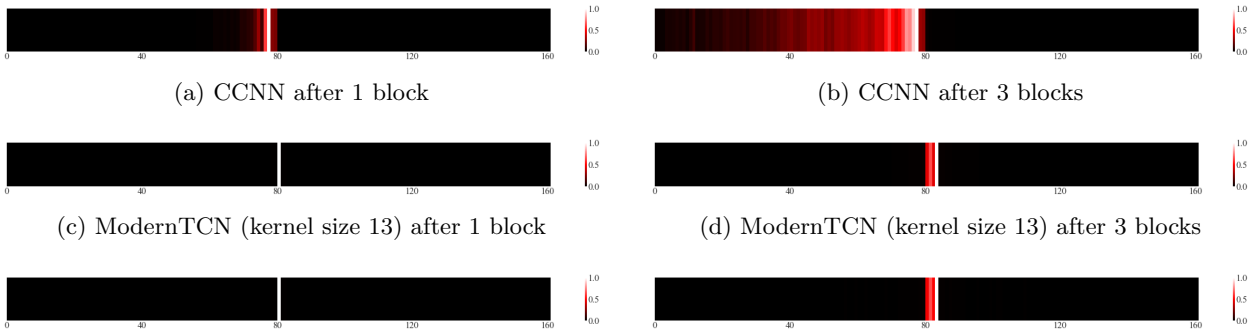
Following the ERF visualization methodology from Kim et al. (2023) and Ding et al. (2022a) as adapted in Luo & Wang (2024), we sample 50 sequences from the Speech Commands MFCC validation set to visualize the relative gradient contribution of each input timepoint to the center of the feature map after applying a certain number of model blocks. The visualization in Figure 2 shows these contributions for input sequence length 161, with lighter areas indicating stronger influence. CCNN{4,140} (Knigge et al., 2023) achieves a significantly larger ERF with fewer parameters (200K vs. 5M). This advantage is reflected in the classification performance in Table 7. See Appendix A for detailed methodology.

Table 7: Classification Results on Speech Commands - Accuracy

Model	**Size**	**Speech Commands MFCC**	**Speech Commands Raw**
S4 (Gu et al., 2021)	300K	0.940	0.983
CKCNN-Seq (Romero et al., 2021b)	98K	0.953	0.717
FlexTCN-6 (Romero et al., 2021a)	375K	0.977	0.917
CCNN{4,140} (Knigge et al., 2023)	200K	0.950	0.983
CCNN{6,380} (Knigge et al., 2023)	2M	**0.980**	**0.984**
ModernTCN	5M/1M	0.837	0.400

4.2.3 Irregularly Sampled Time Series Classification

Since the optimal DWConv kernel size is not specified for the time series classification task, we experimented with different kernel sizes (13, 31, 51, and 71) and found very similar results across configurations (Table 8). The ModernTCN model achieved a peak AUC score of 0.866 on the PhysioNet sepsis prediction dataset, as



(a) CCNN after 1 block

(b) CCNN after 3 blocks

(c) ModernTCN (kernel size 13) after 1 block

(d) ModernTCN (kernel size 13) after 3 blocks

(e) ModernTCN (kernel size 71) after 1 block

(f) ModernTCN (kernel size 71) after 3 blocks

Figure 2: Comparison of effective receptive fields (ERF) for different models. Each row shows a different model configuration, while columns show the ERF after 1 block (left) and 3 blocks (right).

detailed in Table 8. While this result is below the 0.895 AUC attained by CKConv (Romero et al., 2021b), it still demonstrates competitive performance on irregularly sampled time series data.

To further investigate the contribution of the cross-variable component (ConvFFN2), we conducted an ablation study (Table 8). In ModernTCN (Luo & Wang, 2024), this component is posited to improve imputation performance by capturing cross-variable dependencies. Therefore, we hypothesized it should also be important for irregularly sampled data, which is sparse and has missing values. The removal of the ConvFFN2 component resulted in a performance decrease across all tested kernel sizes. This suggests its importance in effectively handling missing data.

Table 8: Ablation Study of ConvFFN2 on PhysioNet Dataset (AUC)

	Kernel Size 13	**Kernel Size 31**	**Kernel Size 51**	**Kernel Size 71**
ModernTCN	0.865	**0.866**	0.859	0.864
Ablated	0.851	0.842	0.853	0.849

5 Discussion

This reproducibility study evaluated the claims of the ModernTCN paper (Luo & Wang, 2024), revealing a nuanced performance profile. While ModernTCN demonstrates competitive results across multiple time series tasks, its claim of consistent state-of-the-art performance is tempered by sensitivity to experimental setup and data handling. When eliminating methodological issues like the "Drop Last Trick" in long-term forecasting, ModernTCN is outperformed by PatchTST in most scenarios. Our anomaly detection experiments revealed that a naive baseline outperforms ModernTCN. On the other hand, ModernTCN's strong performance on imputation tasks was successfully reproduced, with results remaining consistent even after addressing methodological issues.

The paper's assertion regarding enhanced effective receptive fields (ERFs) is not fully supported by our findings. Our ERF visualizations and Speech Commands experiments demonstrate that while ModernTCN claims to effectively enhance receptive fields, it achieves significantly smaller ERFs and lower performance on Speech Commands compared to continuous kernel approaches like CCNN (Knigge et al., 2023). Regarding efficiency, ModernTCN shows faster training times compared to CCNN but requires significantly more parameters (5M vs 200K), indicating a trade-off between computational speed and parameter efficiency.

Lastly, the cross-variable component (ConvFFN2) proves valuable for handling missing data, as confirmed by our ablation study on PhysioNet.

5.1 What was easy?

The extensive experiments and detailed reporting in the ModernTCN paper made reproduction straightforward. Most hyperparameters were clearly specified for each task, and the great majority of the code was well-written, clear, and well-documented, facilitating a smooth execution of the experiments.

5.2 What was difficult?

The missing ERF visualization code presented a challenge. Additionally, the experimental setup for anomaly detection was not thoroughly detailed in the original paper. The work by Kim et al. (2022) was instrumental in helping us understand and address the issues in that task. The extensive experimental scope, coupled with the additional experiments conducted in this study, also presented a significant time investment, requiring substantial computational resources and meticulous attention to detail.

5.3 Contact with the authors

The authors were contacted via email to request the code for ERF visualization and to discuss the flaws in the experimental setup. Unfortunately, no response was received.

References

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

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

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

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

Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. Practical approach to asynchronous multivariate time series anomaly detection and localization. *KDD*, 2021.

Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The uea multivariate time series classification archive. *arXiv preprint arXiv:1811.00075*, 2018.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023.

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, 2022a.

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, 2022b.

Daniel Y. Fu, H. Kumbong, Eric Nguyen, and Christopher Ré. Flashfftconv: Efficient convolutions for long sequences with tensor cores. *arXiv preprint arXiv:2311.05908*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderström. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. *KDD*, 2018.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *arXiv preprint arXiv:2005.08926*, 2020.

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.

Siwon Kim, Kukjin Choi, Hyun-Soo Choi, Byunghan Lee, and Sungroh Yoon. Towards a rigorous evaluation of time-series anomaly detection. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 7194–7201, 2022.

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.

Dani Kiyasseh, Tingting Zhu, and David A. Clifton. Clocs: Contrastive learning of cardiac signals across space, time, and patients. In *International Conference on Machine Learning*, pp. 5606–5615. PMLR, 2021.

D. M. Knigge, D. W. Romero, A. Gu, E. Gavves, E. J. Bekkers, J. M. Tomczak, and J. J. Sonke. Modelling long range dependencies in nd: From task-specific to a general purpose cnn. *arXiv preprint arXiv:2301.10540*, 2023.

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

Yuhong Li, Tianle Cai, Yi Zhang, Deming Chen, and Debadeepta Dey. What makes convolutional models great on long sequence modeling? In *International Conference on Learning Representations (ICLR)*, 2023.

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*, 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.

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, 2022c.

Donghao Luo and Xue Wang. Moderntcn: A modern pure convolution structure for general time series analysis. *International Conference on Learning Representations (ICLR)*, 2024.

Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, and Yuan Xiaojie. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, 2018.

Spyros Makridakis. M4 dataset. `https://github.com/M4Competition/M4-methods/tree/master/Dataset`, 2018.

Aditya P. Mathur and Nils Ole Tippenhauer. Swat: a water treatment testbed for research and training on ics security. In *CySWATER*, 2016.

Eric Nguyen, Michael Poli, M. Faizi, A. Thomas, C. Birch-Sykes, M. Wornow, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *arXiv preprint arXiv:2306.15794*, 2023.

Y. Nie, N. Nguyen, P. Sinthong, and J. Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *International Conference on Learning Representations (ICLR)*, 2023.

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*, 2022.

Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Christopher Ré, and Albert Gu. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866*, 2023.

Zhen Qin, Xiaodong Han, Weixuan Sun, Bowen He, Dong Li, Dongxu Li, Yuchao Dai, Lingpeng Kong, and Yiran Zhong. Toeplitz neural network for sequence modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. 2024.

Matthew A Reyna, Chris Josef, Salman Seyedi, Russell Jeter, Supreeth P Shashikumar, M Brandon Westover, Ashish Sharma, Shamim Nemati, and Gari D Clifford. Early prediction of sepsis from clinical data: the physionet/computing in cardiology challenge 2019. In *2019 Computing in Cardiology (CinC)*, 2019.

David W. Romero, R. J. Bruintjes, J. M. Tomczak, E. J. Bekkers, M. Hoogendoorn, and J. C. van Gemert. Flexconv: Continuous kernel convolutions with differentiable kernel sizes. *arXiv preprint arXiv:2110.08059*, 2021a.

David W Romero, Anna Kuzina, Elias J Bekkers, J.M. Tomczak, and M. Hoogendoorn. Ckconv: Continuous kernel convolution for sequential data. *arXiv preprint arXiv:2102.02611*, 2021b.

Jiaxin Shi, Ke Alexander Wang, and Emily Fox. Sequence modeling with multiresolution convolutional memory. In *International Conference on Machine Learning (ICML)*, pp. 31312–31327. PMLR, 2023.

Ya Su, Y. Zhao, Chenhao Niu, Rong Liu, W. Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. *KDD*, 2019a.

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 (KDD)*, pp. 2828–2837, 2019b. doi: 10.1145/3292500.3330672.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 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 *ICLR*, 2022.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

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.

Mao Yang, Xiangyu Li, Zifen Han, Yong Sun, and Xiqiang Chang. Ultra-short-term multi-step prediction of wind power based on moderntcn and multi-task learning. In *The 9th International Conference on Power and Renewable Energy*, 2024.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *AAAI*, 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. *AAAI*, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. *International Conference on Machine Learning (ICML)*, 2022.

Appendix

Contents

A Visualizing the ERF

Formally, let $I(n \times m \times l)$ be the input time series, where n is the number of samples, m is the number of variables, and l is the input sequence length. Let $F(n \times m \times d \times l')$ be the final output feature map, we desire to measure the contributions of every time point on I to the central points of every channel on F, *i.e.*, $\mathrm{F}_{:,:,:,l'/2}$, which can be simply implemented via taking the derivatives of $\mathrm{F}_{:,:,:,l'/2}$ to I with the auto-grad mechanism. Concretely, we sum up the central points, take the derivatives to the input as the time-wise contribution scores and remove the negative parts (denoted by P). Then we aggregate the entries across all the examples and the input variables, and take the logarithm for better visualization. Formally, the aggregated contribution score matrix $A(l)$ is given by

$$P = \max(\frac{\partial(\sum_i^n \sum_j^m \sum_k^d \mathrm{F}_{i,j,k,l'/2})}{\partial \mathrm{I}}, 0), \tag{1}$$

$$A = \log_{10}(\sum_i^n \sum_j^m P_{i,j,:} + 1). \tag{2}$$

Then we respectively rescale A of each model to $[0, 1]$ via dividing the maximum entry for the comparability across models.

B Full Results

B.1 Long-Term Forecasting

Table 9: Long-term forecasting results. "Reported": scores from ModernTCN paper. "Rerun": our reproduced results. "N.D.L.": No Drop Last, results with test loader not dropping incomplete last batches. "Len" indicates prediction length in time steps (96, 192, 336, and 720 for most datasets; 24, 36, 48, and 60 for ILI). Lower MSE/MAE indicates better performance.

Model		ETTh1		ETTh2		ETTm1		ETTm2		Electricity		Weather		Traffic		Exchange		ILI	
	Len	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Reported	96	0.368	0.394	0.263	0.332	0.292	0.346	0.166	0.256	0.129	0.226	0.149	0.200	0.368	0.253	0.080	0.196	1.347	0.717
	192	0.405	0.413	0.320	0.374	0.332	0.368	0.222	0.293	0.143	0.239	0.196	0.245	0.379	0.261	0.166	0.288	1.250	0.778
	336	0.391	0.412	0.313	0.376	0.365	0.391	0.272	0.324	0.161	0.259	0.238	0.277	0.397	0.270	0.307	0.398	1.388	0.781
	720	0.450	0.461	0.393	0.433	0.416	0.417	0.351	0.381	0.191	0.286	0.314	0.334	0.440	0.296	0.656	0.582	1.774	0.868
Rerun	96	0.369	0.394	0.264	0.333	0.297	0.348	0.169	0.256	0.131	0.227	0.150	0.204	0.371	0.256	0.081	0.197	1.348	0.718
	192	0.406	0.414	0.318	0.373	0.334	0.370	0.227	0.299	0.146	0.243	0.195	0.247	0.384	0.267	0.167	0.290	1.448	0.820
	336	0.392	0.412	0.314	0.376	0.371	0.395	0.276	0.329	0.166	0.264	0.237	0.283	0.404	0.273	0.314	0.402	1.389	0.781
	720	0.450	0.461	0.394	0.432	0.419	0.419	0.349	0.390	0.194	0.288	0.315	0.335	0.446	0.298	0.659	0.583	1.775	0.868
N.D.L.	96	0.376	0.397	0.274	0.340	0.295	0.346	0.168	0.255	0.133	0.228	0.148	0.203	0.382	0.265	0.081	0.198	1.935	0.818
	192	0.410	0.417	0.335	0.384	0.334	0.370	0.226	0.299	0.152	0.248	0.194	0.246	0.395	0.270	0.168	0.290	2.109	0.951
	336	0.435	0.433	0.368	0.412	0.374	0.395	0.273	0.327	0.174	0.268	0.244	0.284	0.408	0.277	0.307	0.399	2.008	0.895
	720	0.456	0.466	0.410	0.442	0.420	0.419	0.351	0.391	0.198	0.290	0.318	0.336	0.452	0.309	0.814	0.679	1.949	0.904

B.2 Imputation

Table 10: Imputation task results with varying percentages of missing data (12.5%, 25%, 37.5%, 50%). "Original": reproduced results. "N.D.L.": No Drop Last, results with test loader not dropping incomplete last batches. "Miss %" indicates percentage of missing data. Lower MSE/MAE indicates better performance. All settings from original paper's repository.

Model		ECL		Weather		ETTh1		ETTh2		ETTm1		ETTm2	
	Miss %	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Original	12.5	0.0588	0.1702	0.0241	0.0394	0.0338	0.1263	0.0378	0.1220	0.0158	0.0831	0.0175	0.0765
	25.0	0.0694	0.1812	0.0250	0.0410	0.0436	0.1427	0.0398	0.1271	0.0184	0.0887	0.0186	0.0800
	37.5	0.0805	0.1956	0.0272	0.0443	0.0540	0.1571	0.0434	0.1342	0.0219	0.0961	0.0207	0.0851
	50.0	0.0842	0.1986	0.0314	0.0523	0.0675	0.1747	0.0481	0.1424	0.0270	0.1064	0.0231	0.0914
N.D.L.	12.5	0.0588	0.1702	0.0239	0.0396	0.0338	0.1263	0.0378	0.1220	0.0158	0.0831	0.0175	0.0761
	25.0	0.0694	0.1812	0.0252	0.0412	0.0436	0.1427	0.0398	0.1271	0.0184	0.0888	0.0187	0.0803
	37.5	0.0805	0.1956	0.0273	0.0442	0.0540	0.1571	0.0434	0.1342	0.0219	0.0961	0.0205	0.0848
	50.0	0.0842	0.1986	0.0313	0.0521	0.0675	0.1747	0.0481	0.1424	0.0270	0.1064	0.0230	0.0910

C Excluded Results

The results presented in this section were excluded from our main analysis due to methodological concerns regarding validation procedures. As detailed in the Experimental Setup section, both the M4 dataset for short-term forecasting and the UEA dataset for classification are provided with only train and test splits, with no designated validation set. Upon reviewing the source code, we identified that the test set was used for validation in both cases, introducing data leakage as the model gains indirect exposure to the test data during training through early stopping and hyperparameter selection.

A methodologically sound approach would involve creating a validation split from the training data. For instance, N-BEATS (Oreshkin et al., 2020) addresses this issue with M4 by first creating a validation split from the training set, finding optimal training settings on that validation set, and then training on the full training set using these optimal settings. However, we chose not to implement this approach for several reasons:

First, many of these datasets are already small for deep learning approaches. For example, M4-Weekly (359 samples), M4-Hourly (414 samples), and several UEA classification datasets like EthanolConcentration (261 samples) and Heartbeat (204 samples) have limited training samples to create a validation split from.

Second, implementing a proper validation approach like N-BEATS (Oreshkin et al., 2020) for ModernTCN would yield worse results (at best the same results) than the reported, as the training is on full training set in both cases. On top of that, those reported results are already worse than the reported results in N-BEATS Oreshkin et al. (2020).

Rather than allocating resources to rerun these experiments with proper validation procedures, we extended our study to new datasets such as multivariate short-term forecasting on ETT, time series classification on Speech Commands, and the PhysioNet 2019 sepsis challenge. These extensions allowed for more methodologically sound comparisons and a more comprehensive evaluation of ModernTCN's capabilities across diverse time series tasks.

The tables below present the reproduced results for completeness, but we emphasize that these should be interpreted with caution due to the methodological concerns outlined above.

C.1 M4 Dataset Results

Table 11: Short-term forecasting task. Results are weighted averaged from several datasets under different sample intervals. Lower metrics indicate better performance. Reported refers to the scores reported in the ModernTCN paper. Rerun refers to the scores obtained by rerunning the model with the settings specified in the source code.

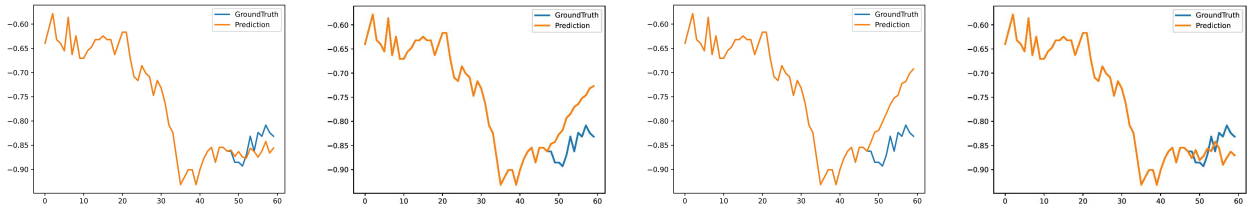
	Yearly			Quarterly			Monthly			Others			Weighted Average		
Model	SMAPE	MASE	OWA	SMAPE	MASE	OWA	SMAPE	MASE	OWA	SMAPE	MASE	OWA	SMAPE	MASE	OWA
Reported	13.226	2.957	0.777	9.971	1.167	0.878	12.556	0.917	0.866	4.715	3.107	0.986	11.698	1.556	0.838
Rerun	13.231	2.957	0.777	10.001	1.170	0.881	12.598	0.920	0.868	4.835	3.175	1.009	11.732	1.561	0.841

C.2 UEA Dataset Results

Table 12: Classification task. Accuracy metric is used. Reported refers to the scores reported in the ModernTCN paper. Rerun refers to the scores obtained by rerunning the model with the settings specified in the source code. Datasets: EthanolConcentration (EC), FaceDetection (FD), Handwriting (HW), Heartbeat (HB), JapaneseVowels (JV), PEMS-SF (PS), SelfRegulationSCP1 (SR1), SelfRegulationSCP2 (SR2), SpokenArabicDigits (SAD), UWaveGestureLibrary (UWL).

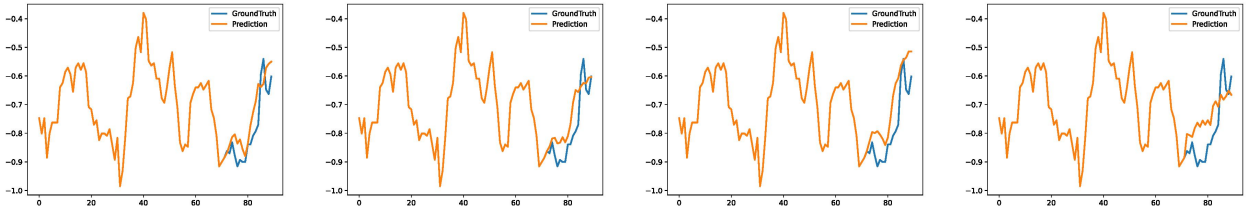
	EC	FD	HW	HB	JV	PS	SR1	SR2	SAD	UWL	Averaged
Reported	0.363	0.708	0.306	0.772	0.988	0.891	0.934	0.603	0.987	0.867	0.742
Rerun	0.319	0.687	0.284	0.771	0.981	0.832	0.928	0.617	0.981	0.859	0.726

D Short-Term Forecasting on ETT



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

Figure 3: Short-term forecasting on ETTm1 with prediction length 12 and input length 48.



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

Figure 4: Short-term forecasting on ETTh1 with prediction length 18 and input length 72.