EdgeMTSC: A Lightweight Large-Kernel ConvNet for Multivariate Time Series Classification

Xueyi Zhou[1], Zhenyu Li[2,3], Dong-Kyu Chae[1*]

[1]Department of Computer Science, Hanyang University, Seoul, South Korea
[2]Hangzhou Mingshi Technologies Co.,Ltd., Hangzhou, China
[3]Laboratory of Data & Network Security, Advanced Institute of Information Technology (AIIT), Hangzhou, China
{hokyeejau,dongkyu}@hanyang.ac.kr, zyli@aiit.org.cn

Abstract

In large-scale sensor networks, *Multivariate Time Series Classification* (MTSC) is a pivotal task for identifying events dependent on longitudinal data at the edge. However, existing methods focus on neither the inherent ability of convolutional networks to perceive subsequence features, nor the prolonged processing pipeline and the model deployment overhead brought by the highly parameterized models. To resolve these difficulties, we present **EdgeMTSC**, a lightweight large-kernel ConvNet for MTSC, which naturally extracts and learns features of diverse subsequences. Specifically, a novel module named **Inter-channel Message Passing-driven Kernel Block (IMP-KB)** is proposed, which maintains a learnable correlation matrix to propagate and merge inter-channel messages, and fuses miscellaneous patterns learned by parallel conv kernels of different sizes. EdgeMTSC sequences two modules of different receptive fields to aggregate local features using small kernels and study long-term representation provided by large kernels, respectively. For inference parameter reduction and accelerating inference without performance loss, the conv blocks in IMP-KBs are structurally reparameterizable. The performance of our model (76.2%) is benchmarked on 26 UEA MTSC datasets and is superior to the SOTA model (MPTSNet, 75%). At the same time, EdgeMTSC uses the fewest parameters and achieves the minimum inference time, applicable on any machine (8 devices ranging from large-scale distributed AI computing servers to resource-constrained edge devices) and in any application scenario.

Code & Appendix —
https://github.com/HokyeeJau/EdgeMTSC

1 Introduction

Multivariate time-series classification (MTSC) refers to the task of classifying longitudinal data consisting of a set of synchronous variables observed over time (Ismail Fawaz et al. 2019). Due to the temporal and multi-source nature of input data, MTSC is widely prevalent in diverse real-world applications, especially in Internet of Things (IoT) (Bouchabou et al. 2021; Tahaei et al. 2020; Aldahiri, Alrashed, and Hussain 2021; Zhou, Lu, and Chae 2025a,b) and Edge Computing (Murshed et al. 2021; Liu et al. 2019; Abdellatif et al. 2019). IoT devices continuously provision multivariate time-series (MTS) data (*e.g.*, temperature, humidity, vibration, just to name a few) in large-scale sensor networks. MTS data in the context of edge computing undergoes local processing to detect anomalies, predict failures, or enable the recognition of other events *on devices* or *nearby edge nodes*. These devices and nodes are often resource-constrained but must demonstrate rapid responsiveness to stakeholders or the clouds. Using the precise and concise MTSC method at the edge has increasingly captured public awareness. However, myriad studies on open-source MTSC benchmarks are proposed to handle MTSC though, few of them (Pantiskas et al. 2022; Mukhopadhyay et al. 2024) discuss the practicality of model deployment at the edge.

Over the past decade, MTSC methods have evolved into deep learning (DL)-based models (Mohammadi Foumani et al. 2024). We witness that 1D convolution (conv)-based methods (Karim et al. 2019; Zhang et al. 2020; Tang et al. 2020; Hao and Cao 2020; Li et al. 2021; Wang et al. 2023; Liu et al. 2024; Eldele et al. 2024) were studied and provably effective at the early phase. Their superiority comes from many factors. Models of such kind are particularly functional at detecting local patterns (*e.g.*, small fluctuations and repeating motifs) in small shifting windows by employing multi-branch residual convolution networks to learn multi-scale patterns; ultimate feature maps are plausibly representative after processing several convolution stages. Due to the surge of Transformer (Vaswani 2017) and its growing extension in various applications (Lin et al. 2022), highly parameterized MTSC models (Zerveas et al. 2021; Foumani et al. 2024; Liu et al. 2021; Zuo et al. 2023; Le et al. 2024) later were advocated, showing superior proficiency compared with conv-based methods. The general time series analysis frameworks designed for addressing classification, imputation, forecasting, and anomaly detection, on the other hand, show similar technical evolution (Wu et al. 2022; Zhang and Yan 2023; Luo and Wang 2024). Their experimental analysis further claims that the core of success is significant subsequence extraction and local attention mechanism; its capability of extracting and matching significant subsequences (*e.g.*, wavelets and shapelets, we use '*timelet*' hereinafter) allows the model to focus on intensive and significant parts of the time-series data (D'Urso and Maharaj

*Corresponding author

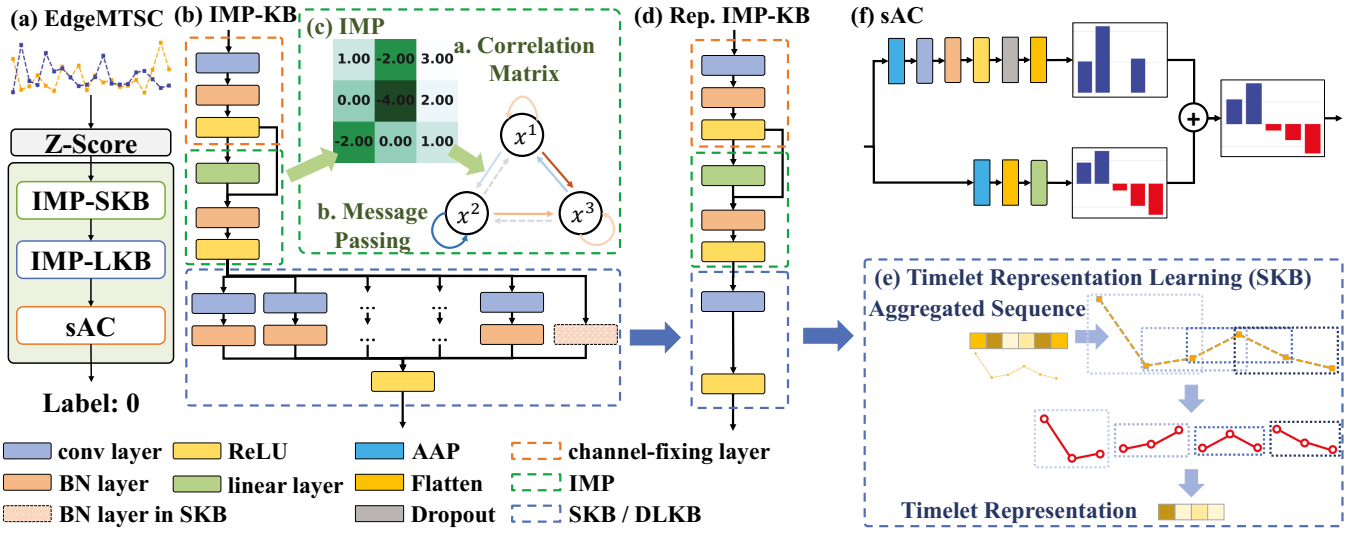

Figure 1: Overview of the EdgeMTSC model. The unexplained elements in this figure: positive impact (solid orange arrows), negative impact (solid blue arrows), and no impact/balanced (dashed grey arrow) in propagation, sliding windows (dashed boxes), and data points (red circles and yellow blocks) in timelet representation learning. The 'conv layer' indicates a 1D convolutional layer, and 'AAP' denotes an Adaptive Average Pooling layer.

2012; Grabocka, Wistuba, and Schmidt-Thieme 2016).

Despite these advancements, numerous challenges persist in terms of computing *at the edge*. Specifically, (C1) The receptive field (RF) of convolution-based methods at each phase remains small and insufficient to capture long-term dependencies within the sliding window. In other words, when treating the sliding window as a timelet extractor, the perceived timelet is too short to be discriminative. Even using large-kernel convolutions, small-scale or stride-sampled patterns would be lost (Ding et al. 2022b, 2024). (C2) The design of multi-branch topology addresses this challenge but exacerbates the usage of runtime memory and model complexity, resulting in increased inference time and memory consumption. Introducing the structural re-parameterization (SRP) mechanism (Ding et al. 2019, 2021, 2024), which merges the multi-branch topology into a single conv layer, from Computer Vision models, may resolve this issue; however, the SRP mechanism overlooks the inter-channel correlation. (C3) On the other hand, highly parameterized models are severely affected by excessive time and space complexity during deployment, which is due to the quadratic growth of the attention mechanism *w.r.t.* sequence length (Lin et al. 2022). (C4) Recent SOTA MTSC methods share a common but redundant (pre-)processing issue: shapelet extraction (Li et al. 2021; Zuo et al. 2023; Le et al. 2024), resulting in an overly long processing pipeline and reduced deployability at the edge. (C5) Several representative lightweight works (Eldele et al. 2024; Campos et al. 2023) either fail to achieve adequate model compactness, suffer from low accuracy, or are primarily limited to univariate tasks.

Considering the prerequisites of MTSC at the edge and the facing challenges, we present a lightweight large-kernel ConvNet named **EdgeMTSC**. It extracts and learns correlated patterns of timelets owing to the nature of conv layers and shows superior performance to that of the SOTA models but with far fewer parameters at the edge. Specifically, we propose a novel module **Inter-channel Message Passing-driven Kernel Block (IMP-KB)**. The IMP layer is designed to propagate and aggregate the message from positively / negatively correlated channels, measured by a learnable inter-variable correlation matrix. The adjusted variables are then taken as input for KBs to discover the intra-channel latent representations of timelet candidates. We extend the RepVGG block (Small Kernel Block, SKB) (Ding et al. 2021) and the Dilated Reparam Block (Large Kernel Block, LKB) (Ding et al. 2024) from 2D vision tasks to this end. Both well-designed conv blocks are reparameterizable Grouped Convolutions, capable of identifying both long-term and short-term dependencies and merging branches into a single conv layer during inference. EdgeMTSC ends with a *Self-adjusted Classifier*, which we designed to employ two different classifiers to cooperatively estimate and adjust the likelihood adaptively.

Through our extensive evaluation, we observe that the performance of our EdgeMTSC is superior to the SOTA model MPTSNet (Mu, Shahzad, and Zhu 2025) on 26 UEA MTSC datasets; EdgeMTSC shows outperformance (ACC: 76.2%) compared with the MPTSNet (ACC: 75%), and is competitive with or unequivocally superior to the other general time series frameworks. At the same time, EdgeMTSC uses the fewest parameters and achieves the minimum inference time, applicable on any machine (including those with and without GPU acceleration) and in any application scenario. This bears out that the IMP-KBs are simple-yet-effective, thereby facilitating parameter reduction and inference acceleration, and thus indicates that our EdgeMTSC is the most lightweight and efficient MTSC model to date.

2 Preliminaries

Multivariate Time Series Classification

We follow (Zuo et al. 2023) and (Le et al. 2024) to formulate MTSC. We denote the MTS data as $\mathbf{X} \in \mathbb{R}^{V \times T}$, where V stands for the number of variables (channels) and T is the series length of each variable. To be more specific, $X^v \in \mathbf{X}$, where X^v is the time series of the v-th variable; $X^v = (x_1^v, x_2^x, ..., x_T^v)$, where x_t^v is the value of the t-th timestamp in the v-th variable. An MTS dataset consists of N instances, and each instance has one corresponding class label y. We further formulate MTSC as $\tilde{y} = f(\mathbf{X})$, where $\tilde{y}$ is the prediction (likelihood) and $f(\cdot)$ is an MTSC model. Given a time series X of length T, a consecutive time series *subsequence* of length $T' \leq T$ is denoted by $X[s : s + T'] = x_s, ..., x_{s+T'}$, where s is a start index.

Structural Reparameterization

The Structural Reparameterization (**SRP**) (Ding et al. 2019, 2021, 2022a,b, 2024) has been introduced in the Computer Vision field to efficiently learn visual patterns as well as to reduce heavy GPU usage. The models following SRP generally sequence designed 2D ConvNets, each in a multi-branch topology during training, while using a sequential FCN without branching during inference. This structural transition benefits from the equivalent transformation of parallel kernels, which hold different kernel sizes but share the same RF. For instance, a 1×1 kernel after with 1 layer of 0s can be equivalently added with a 3×3 kernel, under the same RF.

The early models following the SRP paradigm (*e.g.*, ACNet (Ding et al. 2019), RepVGG (Ding et al. 2021), and RepMLPNet (Ding et al. 2022a)) used diverse convolutional networks with small kernels ($k_i < 10$). Since large spatial convolution (Ding et al. 2022b; Liu et al. 2022) is of importance to enable ConvNets with the capability to compete with ViT (Dosovitskiy 2020) (a Transformer-based image recognition model), researchers later devised large kernel-driven ConvNets, *i.e.*, RepLKNet (Ding et al. 2022b) and UniRepLKNet (Ding et al. 2024) to obtain features from ConvNets with large Effective Receptive Fields (ERFs) without inference loss. Inspired by the superiority of SRP in vision tasks, we extend its concept to 1D convolution (for MTSC), targeted at squeezing the multi-branch convolution blocks without inference loss.

SRP in 1D

Assume that a set of parallel convolutional layers in a conv block is given, and each layer can be represented by a triplet, in i-th of which, k_i is the kernel size, r_i is the dilation (non-dilated if $r_i = 1$) and $p_i = \left\lfloor \frac{k_i}{2} \right\rfloor$. The layer with the largest kernel size $\hat{k}$ is assigned with the triplet $(\hat{k}, \hat{r}, \hat{p})$. To shrink the conv layers, the batch normalization (BN) layer is first fused with each kernel $W_i \in \mathbb{R}^{k_i}$. In the evaluation (eval) phase, BN does not compute the mean and variance from the current batch. Instead, it uses the running averages of these statistics that were collected during training. For each BN layer BN$(\cdot)$ following the i-th conv layer, we denote its accumulated mean, standard deviation, learned scaling factor, and bias by μ_i, σ_i, γ_i, and β_i, respectively. Given the hidden input $\mathbf{h}_i$ and padded kernel W_i of the i-th branch, the original operations are formulated:

$$\text{BN}(\mathbf{h}_i, \mu_i, \sigma_i, \gamma_i, \beta_i) = (\mathbf{h}_i \circ W_i - \mu_i)\gamma_i/\sigma_i + \beta_i, \quad (1)$$

where $\circ$ is the convolution operator. Then we can easily derive the fused kernel W_i^* and bias b_i^* as follows:

$$W_i^* = \gamma_i W_i/\sigma_i, \quad b_i^* = \mu_i\gamma_i/\sigma_i + \beta_i, \quad (2)$$

such that, the i-th convolutional layer and the following BN layer can be presented by:

$$\text{BN}(\mathbf{h}_i, \mu_i, \sigma_i, \gamma_i, \beta_i) = \mathbf{h}_i \circ W_i^* + b_i^*. \quad (3)$$

Next W_i^* is transformed into a non-dilated convolution conducted by the following PyTorch-style pseudo code:

$$W_i' = \text{conv\_transpose1d}(W_i^*, \mathbf{I}, \text{stride} = r_i), \quad (4)$$

where $\mathbf{I}$ is a kernel tensor filled with 1s. To match the exact RF of the layer with $\hat{k}$, W_i' should be padded with $\bar{p}_i$ 0s:

$$\bar{p}_i = \left\lfloor \hat{k}/2 \right\rfloor - \lfloor k_i'/2 \rfloor . \quad (5)$$

By accumulating $\mathbf{W}^* = \sum \tilde{W}_i$ and $\mathbf{B}^* = \sum b_i^*$ where $\tilde{W}_i$ is the i-th padded kernel, all parallel sequences of conv layers and BN layers are merged into one. Since the branches share the same input, the conv block can be presented by:

$$\text{ConvBlock}(\mathbf{h}, \mu, \sigma, \gamma, \beta) = \mathbf{h} \circ \mathbf{W}^* + \mathbf{B}^*. \quad (6)$$

A further intuitive illustration is depicted in Figure 2.

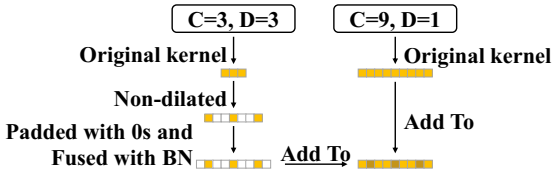

Figure 2: An intuitive progress of SRP. Given the largest kernel size $\hat{k} = 9$ with $\hat{p} = 0$, the smaller kernel should be first non-dilated, then padded with 0s, such that it could merge with the largest one.

3 Edge Multivariate Time Series Classifier

We present **EdgeMTSC**, a lightweight large-kernel ConvNet that naturally extracts and learns representations of diverse timelet candidates using reparameterizable convolution blocks, which are specifically designed and named with the Inter-channel Message Passing-driven Kernel Block (**IMP-KB**). The IMP focuses on propagating and aggregating the message passed from positively / negatively correlated variables. The KBs are used for extracting and learning representations of diverse timelet candidates. We extend the Small Kernel Block (SKB) from the RepVGG block (Ding et al. 2021) to dense local features and the Large Kernel Block (LKB) from the Dilated Reparam Block (DRB) (Ding et al. 2024) to study long-term representations. EdgeMTSC sequences both IMP-SKB and IMP-LKB and ends with the Self-adjusted Classifier (sAC) to estimate and adjust the likelihood. The overall pipeline of EdgeMTSC is shown in Figure 1(a).

IMP-KB

This section introduces the overview of the Inter-channel Message Passing-driven Kernel Block (IMP-KB) illustrated in Figure 1(b). IMP-KB initially adjusts the channel numbers to an assigned number via a channel-fixing layer to handle input with a variable number of channels, as follows:

$$h^{cf} = \text{Conv}(\mathbf{X}), \tag{7}$$

where $\text{Conv}(\cdot)$ indicates a conventional conv block comprised of a conv layer ($K = 1$, K is the size of the kernel), a BN layer, and a rectified linear unit (ReLU); h^{cf} is the rectified features. Inspired by the dynamic attention scores in the attention mechanism in Transformer (Vaswani 2017), we consider that the correlation between variables does exist and matter; however, the used conventional conv layers of the SRP mechanism are Grouped Convolutions and can only focus on intra-channel features. Two-step representation learning of time series is accordingly presented: inter-channel message propagation and intra-channel timelet representation learning. IMP, consequently, is designed to propagate and aggregate the messages from correlated variables and update the local variables, imitating the core mechanism of Graph Neural Networks (Xu et al. 2018). We maintain a learnable correlation matrix $W^c \in \mathbb{R}^{V\times V}$ to this end, as shown in Figure 1(c). The following formula implements this process:

$$h^{na} = h^{cf} + W^c \cdot h^{cf}, \tag{8}$$

where $W^c \cdot h^{cf}$ stands for message aggregation; h^{na} is the updated features by merging aggregated messages and own features. The KB is subsequently employed to extract timelet candidates and conduct intra-channel timelet representation learning; it effectively fuses the features of exclusive timelets in each sliding window as outlined in Figure 1(b). We apply SKB and LKB to learn short-term and long-term representation, respectively. To be noticed, both SKB and LKB follow the SRP mechanism; that is, the parallel conv layers together with their BN layers can be merged into one as described in Figure 1(d). The KBs thus can reduce model complexity and accelerate inference. Through IMP and KBs, we equip the model with the ability to learn correlation-aware time series representation.

Small Kernel Block. The Small Kernel Block (SKB) is extended from a 2D vision model RepVGG (Ding et al. 2021). It primarily contains two branches: (3, 1, 1), (1, 1, 0) *w.r.t.* (k_i, r_i, p_i) and a branch with a pure BN layer. After being structurally reparameterizable, SKB is allowed to learn sequential short-term dependencies as illustrated in Figure 1(e). SKB is targeted at aggregating the local features within a window of size 3 such that the successive large-kernel block can have a broader RF.

Large Kernel Block. LKB is delicately extended for studying long-term dependencies using large kernels. Since we prioritize the intra-channel time series representation learning of different timelets from a single window, we extended the Dilated Reparam Block (DRB) from the 2D tasks (Ding et al. 2024) to the 1D time-series representation learning. We generalize its kernel design and add a residual connection to the original block design to improve network convergence. In other words, a 1D conv layer with the kernel of size $k_i = 1$ filled with 1s is appended. The conv layers in LKBs are thus characterized as follows: (1) the non-dilated conv layer covers the whole window to learn the full channel-wise representation; (2) the dilated conv layers capture features from those short-term stride-sampled subsequences to augment the local features in long-term dependencies; (3) the residual layer avoids vanishing and exploding gradients, facilitating the convergence of the network. All the conv layers are Grouped Convolutions, learning channel-independent representations.

We adapt the original kernel design from the 7 LKB variants ($\hat{k} \in \{5, 7, 9, 11, 13, 15, 17\}$) from UniRepLKNet (Ding et al. 2024) into our LKB. Specifically, the 2D DRB in Ding et al. 2024 is extended to the 1D LKB and adds a residual connection (*i.e.*, a conv layer with a kernel ($K = 1$) filled with 1s) to avoid vanishing and exploding gradients. We observe that the short-term conv layers in each LKB can be categorized into three kinds: a short-term dense conv layer ($k_i = 5$, $r_i = 1$), a stride-sampled conv layer with stride 2 (Given k, s.t. $2k - 1 \leq K$, maximize k), and three conv layers targeted at combining peripheral and central observations with the kernel size $k_i = 3$ and three extra-large r_is. We thus accordingly provide an extra-large LKB ($K = 47$) including the conv layers with $\mathbf{k} = \{5, 23, 3, 3, 3\}$ and $\mathbf{r} = \{1, 2, 21, 19, 17\}$. Since LKB follows the SRP mechanism, all the conv layers fused with BN layers can be equivalently transformed into a single large-kernel conv layer. Figures 1(b) and (d) illustrate the IMP-KBs before and after reparameterization.

Self-adjusted Classifier

Our *Self-adjusted Classifier* (sAC) is designed with two branches; one dominates the *likelihood* and the other supplements the *class-specific information*, which is illustrated in Figure 1 (f) with a test sample. The upper branch (conv branch) is formulated by $h^{am} = \text{Amplifier}(h^{LKB})$, where $h^{LKB} \in \mathbb{R}^{64\times T}, h^{SC} \in \mathbb{R}^{C}_{\geq 0, \leq 1}$ and C represents the latent generated by IMP-LKB, the non-negative likelihood and the class number, respectively. The lower branch (linear branch) is formulated by $h^{est} = \text{Estimator}(h^{LKB})$, where $h^{LKB} \in \mathbb{R}^{C}$ represents the estimated likelihood. Thus, the sAC could be further formulated by $\tilde{y} = h^{am} + h^{est}$. This accumulation can amplify the likelihood of the vague estimation of the ground-truth label, providing stable classification. This is empirically shown in Section 4.

EdgeMTSC

In summary, the MTS data is normalized using the z-score normalizer for one thing, then goes through our IMP-SKB, IMP-LKB, and sAC. In the inference phase, both SKB and LKB are individually transformed into a single conv layer. The output is activated by the Softmax function. The Cross-Entropy loss is used to optimize the model parameters.

	EDI	DTWD	W.+ M.	Mini-Rocket	LCEM	MLSTM-FCNs	Tapnet	Shape-net	OS-CNN	MOS-CNN	Tody-Net	TSLA-Net	DKN	MPTS-Net	Ours
AWR	0.970	0.987	0.990	0.992	0.993	0.973	0.987	0.987	0.988	0.991	0.987	0.990	0.993	0.977	0.983
CT	0.964	0.989	0.990	0.993	0.979	0.985	0.997	0.980	N/A	N/A	N/A	N/A	0.986	N/A	0.991
CR	0.944	1.000	1.000	0.986	0.986	0.917	0.958	0.986	0.993	0.990	1.000	0.986	0.951	0.944	1.000
ER	0.133	0.929	0.133	0.981	0.200	0.133	0.133	0.133	0.881	0.915	0.915	N/A	0.933	0.944	0.893
EW	0.549	0.618	0.890	0.962	0.527	0.504	0.489	0.878	0.414	0.508	0.840	N/A	0.628	N/A	0.542
EP	0.666	0.964	1.000	1.000	0.986	0.761	0.971	0.987	0.980	0.996	0.971	0.986	0.979	0.971	0.993
EC	0.293	0.323	0.430	0.468	0.372	0.373	0.323	0.312	0.240	0.415	0.350	0.304	0.372	0.433	0.354
FD	0.519	0.529	0.545	0.620	0.614	0.545	0.556	0.602	0.575	0.597	0.627	0.668	0.631	0.698	0.679
FM	0.550	0.530	0.490	0.550	0.590	0.580	0.530	0.580	0.568	0.568	0.676	0.610	0.600	0.640	0.680
HMD	0.278	0.231	0.365	0.392	0.649	0.365	0.378	0.338	0.443	0.361	0.649	0.527	0.662	0.635	0.608
HW	0.200	0.286	0.605	0.507	0.287	0.286	0.357	0.452	0.668	0.677	0.436	0.579	0.231	0.344	0.386
HB	0.619	0.717	0.727	0.771	0.761	0.663	0.751	0.756	0.489	0.604	0.756	0.776	0.765	0.756	0.771
IW	0.128	N/A	N/A	0.595	0.228	0.167	0.208	0.250	N/A	N/A	N/A	0.100	0.362	N/A	0.588
JV	0.924	0.949	0.973	0.989	0.978	0.976	0.965	0.984	N/A	N/A	N/A	0.992	0.930	N/A	0.995
LSST	0.456	0.551	0.590	0.643	0.652	0.373	0.568	0.590	0.413	0.521	0.615	0.663	0.347	0.604	0.412
LIB	0.833	0.870	0.878	0.922	0.772	0.856	0.850	0.856	0.950	0.965	0.850	0.928	0.900	0.872	0.928
MI	0.510	0.500	0.500	0.550	0.600	0.510	0.590	0.610	0.535	0.515	0.640	0.620	0.620	0.650	0.620
NATO	0.850	0.883	0.870	0.928	0.916	0.889	0.939	0.883	0.968	0.951	0.972	0.956	0.872	0.944	0.989
PEMS	0.973	0.711	N/A	0.522	0.942	0.699	0.751	0.751	0.760	0.764	0.780	0.838	0.930	0.942	0.884
PD	0.705	0.977	0.948	N/A	0.977	0.978	0.980	0.977	0.985	0.983	0.987	0.989	0.948	0.989	0.989
PS	0.104	0.151	0.190	0.292	0.288	0.110	0.175	0.298	0.299	0.295	0.309	0.178	0.525	0.144	0.256
RS	0.868	0.803	0.934	0.868	0.941	0.803	0.868	0.882	0.877	0.929	0.803	0.908	0.879	0.875	0.914
SRS1	0.771	0.775	0.710	0.925	0.839	0.874	0.652	0.782	0.835	0.829	0.898	0.918	0.913	0.928	0.898
SRS2	0.483	0.539	0.460	0.522	0.550	0.472	0.550	0.578	0.532	0.510	0.550	0.617	0.600	0.572	0.594
SAD	0.967	0.963	0.982	0.620	0.973	0.990	0.983	0.975	N/A	N/A	N/A	0.999	0.963	N/A	0.998
UW	0.881	0.903	0.916	0.938	0.897	0.891	0.894	0.906	0.927	0.926	0.850	0.913	0.893	0.881	0.866
Avg. ACC	0.621	0.707	0.713	0.741	0.711	0.641	0.669	0.704	0.696	0.719	0.748	0.741	0.747	0.75	0.762
Avg. Rank	10.269	8.731	7.154	4.885	5.962	9.115	7.923	6.577	7.385	6.731	6.192	4.962	5.808	6.5	4.385

Table 1: Accuracies of EdgeMTSC and 14 baseline methods on 26 UEA MTSC datasets.

4 Experiments

We evaluate EdgeMTSC on 26 UEA MTSC datasets from the UEA archive (Bagnall et al. 2018).

Baselines

We select 14 baseline methods, comprising two distance-based methods: *EDI*, *DTW_D* (Bagnall et al. 2018); a pattern-based algorithm: *WEASEL+MUSE* (Schäfer and Leser 2017); a feature-based algorithm: *MiniRocket* (Dempster, Schmidt, and Webb 2021); an ensemble method: *LCEM* (Fauvel et al. 2020); a recurrent model: *MLSTM-FCN* (Karim et al. 2019); four deep learning methods: *Tapnet* (Zhang et al. 2020), *Shapenet* (Li et al. 2021), *TSLANet* (Eldele et al. 2024), *TodyNet* (Liu et al. 2024); and three methods with large RFs: *OS-CNN*, *MOS-CNN* (Tang et al. 2020), *DKN* (Xiao et al. 2024).

Performance Evaluation

Table 1 compares the results of each baseline derived from (Mu, Shahzad, and Zhu 2025; Eldele et al. 2024; Xiao et al. 2024; Liu et al. 2024) on 26 UEA datasets. The result 'N/A' indicates that the performance of the corresponding methods is not reported. EdgeMTSC, observed from the table, gains superior average accuracy (76.2%) compared to the SOTA model MPTSNet (75%), TodyNet (74.8%) and DKN (74.7%). The results of the top-1/top-2/top-3 are 5/9/14 compared with 6/10/12 of the MiniRocket and 4/7/9 of MPTSNet. The average rank (4.385) experimentally supports the excellence of the well-crafted design of EdgeMTSC. Notably, compared with TSLANet, which is the most recent lightweight MTSC model, EdgeMTSC achieves more than 2.4% average accuracy. We therefore advocate that our EdgeMTSC can be considered as a SOTA lightweight MTSC model. According to the p-value, the EdgeMTSC remains in similar ranks with MiniRocket (0.764), MOS-CNN (0.276), TodyNet (0.396), TSLANet (0.735), DKN (0.127), and MPTSNet (0.378).

Computational Complexity

Here, we use 8 devices in total ($\mathcal{M}_1 \sim \mathcal{M}_8$). The other devices involving a mini-computer with an NVIDIA GeForce RTX 3070 Laptop GPU (VRAM 16GB, $\mathcal{M}_1$), an AI server with an NVIDIA GeForce RTX 3090 GPU ($\mathcal{M}_2$), an NVIDIA A100 GPU installed in an HPC (DELL DSS8440, $\mathcal{M}_3$), a high-end server equipped with 4 NVIDIA A100 80G GPU ($\mathcal{M}_4$) a Raspberry Pi 4 Model B (RAM 8GB, $\mathcal{M}_5$), a Raspberry Pi 5 (RAM 8G, $\mathcal{M}_6$), an NVIDIA Jetson Nano (RAM 4G, $\mathcal{M}_7$), and a virtual machine (8 cores, RAM 32GB) on a scalable business server (PowerEdge T640, $\mathcal{M}_8$).

We select 6 models for comparison: ShapeFormer (Le

$\mathcal{D}$	Model	Params. (M)	MADD (M)	Params. Size (MB)	Total Size (MB)	FLOPs (MMac)	Latency (ms) $\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	$\mathcal{M}_6$	$\mathcal{M}_7$	$\mathcal{M}_8$
LIB	TodyNet	0.467	28.170	1.957	3.727	14.21	2.499	1.031	2.692	2.97	33.008	11.798	18.383	226.31
	ShapeFormer	0.192	**0.44**	0.771	1.231	**3.17**	5.445	2.433	6.44	7.522	57.649	16.52	86.167	140.91
	MPTSNet	19.297	2475.91	70.268	99.386	2516.31	73.339	32.155	89.432	87.835	1987	701.94	736.37	5558
	TSLANet	0.479	4.970	1.998	2.213	4.99	1.1	0.646	1.78	1.569	13.881	**3.694**	13.937	9.154
	EdgeMTSC	0.092	4.940	0.385	1.584	4.340	1.261	0.694	1.92	1.71	20.435	6.67	12.753	47.065
	RepEdgeMTSC	**0.088**	4.530	**0.369**	**0.946**	4.180	**0.82**	**0.445**	**1.198**	**1.013**	**10.27**	4.071	**8.64**	**5.101**
PD	TodyNet	0.465	4.700	1.952	2.247	2.37	2.28	1.007	2.669	2.394	20.167	10.774	21.235	55.274
	ShapeFormer	0.177	**0.22**	0.716	0.916	1.53	4.333	2.254	6.05	5.445	36.807	11.618	239.561	27.201
	MPTSNet	11.549	166.460	44.091	46.290	171.87	19.314	8.671	23.665	21.248	319.04	177.93	182.26	493.14
	TSLANet	0.477	0.500	1.995	2.017	**0.5**	1.08	0.583	1.572	1.387	10.57	8.292	11.084	48.947
	EdgeMTSC	0.091	2.240	0.382	0.595	0.77	1.54	0.681	2.117	1.663	16.315	7.088	12.455	65.08
	RepEdgeMTSC	**0.087**	1.840	**0.366**	**0.468**	0.740	**0.821**	**0.431**	**1.187**	**1.007**	**8.632**	**3.942**	**8.769**	**30.616**
EW	TodyNet	0.464	36318	1.948	1954.87	18275	34.231	15.843	11.753	11.702	10953	4974	N/A	890.65
	ShapeFormer	1.032	**296.27**	0.800	183.230	33249	1329	59.318	43.65	44.626	N/A	N/A	N/A	2996
	MPTSNet	2466.55	881904	10335	20315	1061387	18240	N/A	302.33	311.27	N/A	N/A	N/A	9592
	TSLANet	1.029	2250.23	2.009	**99.101**	2261.74	5.391	0.913	2.265	2.183	**792.69**	**226.03**	109.67	32.109
	EdgeMTSC	0.160	1304.89	0.671	479.909	1750.64	3.042	1.492	2.376	2.051	2890	1442	180.22	54.686
	RepEdgeMTSC	**0.156**	1299.89	**0.655**	231.282	**1686.18**	**1.698**	**0.504**	**1.446**	**1.174**	1249	631.54	**76.983**	**26.706**
DDG	TodyNet	18.281	1046501	76.675	6580.94	537870	10400	165.52	121.73	116.41	N/A	N/A	N/A	8261
	ShapeFormer	3.212	993.78	13.141	295.122	1080.68	100.57	5.78	10.197	11.757	957.87	410.94	3190	28.609
	MPTSNet	301.821	51932	1221.78	1405.38	53195	1357	56.17	155.82	113.14	11730	5811	N/A	804.64
	TSLANet	1.796	123.54	7.494	10.366	123.7	2.6	0.673	2.216	1.538	**31.694**	14.517	13.116	**7.285**
	EdgeMTSC	0.419	109.23	1.759	10.400	118.84	2.98	0.701	2.372	1.654	51.56	17.262	14.302	13.705
	RepEdgeMTSC	**0.415**	**108.76**	**1.742**	**6.651**	**117.87**	**1.732**	**0.449**	**1.445**	**1.006**	32.363	**12.296**	**7.751**	7.985
FK	TodyNet	10.559	22034434	44.288	17697	1110287	14303	N/A	171.133	173.68	N/A	N/A	N/A	16838
	ShapeFormer	2.763	2736.41	10.666	788.321	3062.19	208.80	6.4	11.453	10.503	2498	1126	N/A	173.83
	MPTSNet	543.734	48072	2269.91	2811.55	48721	603.32	20.838	42.215	41.673	10037	4986	N/A	381.4
	TSLANet	1.604	378.23	6.594	**15.957**	378.86	2.64	0.697	2.296	1.836	**68.955**	**27.709**	26.704	7.624
	EdgeMTSC	0.465	269.51	1.951	32.599	351.94	2.822	0.704	2.341	1.945	129.23	47.195	15.73	8.847
	RepEdgeMTSC	**0.461**	**268.85**	**1.934**	18.758	**348.36**	**1.67**	**0.446**	**1.435**	**1.177**	77.979	29.928	**8.401**	**5.25**

Table 2: Comparison of model complexity on 4 dataset selected from (Bagnall et al. 2018) and a sythetic dataset FK. This table compares the metrics focusing on efficiency on the 8 devices ($\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_8$), which are designed for comparing feasibility and deployability on various computational devices. Especially, the resource-limited devices are highlighted in blue.

et al. 2024), TSLANet (Eldele et al. 2024), MPTSNet (Mu, Shahzad, and Zhu 2025), TodyNet (Liu et al. 2024), EdgeMTSC, and the reparameterized EdgeMTSC (RepEdge MTSC). We collect the number of parameters (Params., unit: M), Multiply-Add operations (MADD, unit: M), parameters in memory (Params. Size, unit: MB), estimated total size in memory (Total Size, unit: MB), Floating Point Operations (FLOPs, unit: MMac) and machine-specific inference time (Latency; here, the pre-processing time is not included, unit: ms) on 8 devices ($\mathcal{M}_1 \sim \mathcal{M}_8$). All the models are assessed on 5 representative datasets: Libras (with the lowest number of channels (=2)), PenDigits (with the minimum sequence length (=8)), EigenWorms (with the maximum sequence length (=17,984)), DuckDuckGeese (with highest number of channels (=1,345)), and Fake (synthetically generated with $V = 1,000$ and $T = 1,000$).

Table 2 presents the overall results, with the optimal values are bold. RepEdgeMTSC consistently maintains the lowest number of parameters and memory as well as the latency on AI servers across all the datasets. ShapeFormer yields the lowest MADD when the channel size is small, whereas our model consistently remains the minimum in opposition. TSLANet occupies the least estimated total size in memory when the length of the time series increases. RepEdgeMTSC leverages the competitive time with that of TSLANet to classify the data while scaling up the MTS data on edge devices. On a virtual machine of a scalable business server, our model achieves slightly better runtime. Overall, our model is comparatively superior and feasible to be deployed and applied on any device. This comprehensive comparison and the carefully designed model structure help us confirm that the proposed model can address the following challenges: high complexity of multi-branch conv blocks during inference (C2), large overhead of high-performance models (C3) and insufficient model lightweighting (C5).

Ablation Study

Effectiveness of Large Kernels. EdgeMTSC leverages parallel filters to enumerate and merge the latent features of various long-term and short-term timelets. Figure 3 illustrates that EdgeMTSC variants ($K \in \{5,7,9,11,13,15,17,47\}$) exhibit inconsistent overall performance, but the variants with larger kernel sizes tend to achieve better performance on more datasets, bearing out that large RFs can facilitate the model performance. The accuracy re-ordered by the numbers of variables in Table 5 further evidence this point. This is reasoned by various factors, such as volatility and periodicity of time series and the

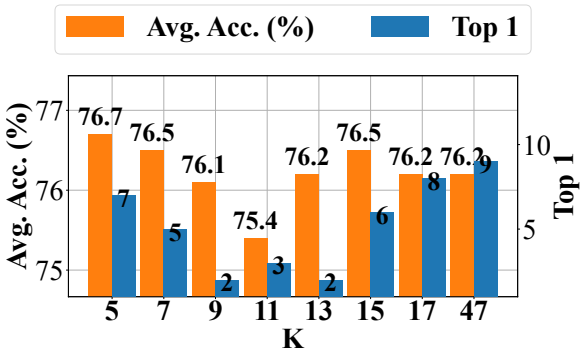

Figure 3: Accuracies and the numbers of top-1 of multiple EdgeMTSC variants with different kernel sizes.

discriminability of the stride-sampled subsequences. Larger sliding windows, put another way, can identify significant subsequences and longer dependencies. These points substantiate that we resolved the challenges of limited long-term dependency learning in ConvNets (C1) and low accuracy of lightweight models (C5).

Effectiveness of Multi-Branch Blocks. To evaluate the efficiency and robustness of the SRP blocks, we substitute either or both of LKB and SKB with their merged one(s) (*i.e.*, EdgeMTSC-w/o SKB, EdgeMTSC-w/o LKB, and EdgeMTSC-w/o KB) and rank their performance by a critical difference (CD) diagram in Figure 4 . It can be observed that removing the multi-branch topology with small RF (*i.e.*, EdgeMTSC-w/o SKB) does not lead to a discriminative difference. However, when the multi-branch topology with large RF is removed, the EdgeMTSC variants perform significantly worse. When the dilated conv layers in KBs are all removed, the EdgeMTSC-w/o KB model performs worse. This highlights the significance of applying diverse large-kernel multi-branch blocks in MTSC models to extract both long-term and short-term dependencies simultaneously. We therefore conclude that our EdgeMTSC can address the challenges of learning both short-term and long-term dependencies (C1) and intricate timelet extraction (C4).

Effectiveness of sAC. EdgeMTSC shows inferior performance if either of branches in sAC is dropped. The CD diagram Figure 5 ranks the EdgeMTSC, EdgeMTSC w/o the linear branch (EdgeMTSC-w/o linear), and EdgeMTSC w/o the conv branch in sAC (EdgeMTSC-w/o conv). The CD score is 1.482. If the conv branch is dropped, the performance of EdgeMTSC exhibits a severe decline, achieving only 72% average accuracy. This experimental result evidences that the conv layer can compensate for the weak discriminability of the linear layer. On the other hand, if the linear layer is disabled, the conv layer provides relatively strong cues on the target label. However, it may produce ambiguous decision boundaries when the ground-truth category is similar to others. As a result, the design of the sAC experimentally resolves the low accuracy challenge (C5).

Effectiveness of IMP. The correlation between channels does matter in EdgeMTSC. We compare EdgeMTSc-w/o

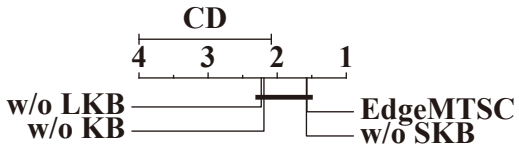

Figure 4: Average accuracy ranks of EdgeMTSC, EdgeMTSC-w/o SKB, EdgeMTSC-w/o LKB and EdgeMTSC-w/o KB. The CD score is 1.915.

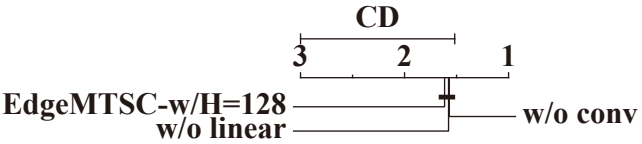

Figure 5: Average accuracy ranks of EdgeMTSC, EdgeMTSC-w/o linear and EdgeMTSC-w/o conv. The CD score is 1.482.

imp with EdgeMTSC using a comprehensive set of classification metrics. Our experimental results show that without the IMP block, the average accuracy drops by 0.7%. On certain datasets, the drops are prominent (*e.g.*, ACC of `FingerMovements` drops 8%). These significant drops demonstrate the necessity and efficiency of the IMP block, or, further, the inter-channel correlation. We thus confirm that the inter-channel correlation in learning multivariate time series representations is valid, indicating the IMP resolves the loss of the inter-channel correlation (C2) using a simple learnable correlation matrix.

Robustness. The previous experimental results are all derived with only one random seed. To demonstrate the overall performance and robustness of EdgeMTSC, we conduct experiments by tuning the random seed ranging from 1 to 10 and collect the following scores: average accuracies (ACC), balanced accuracies (BACC), macro/micro f1-scores (F1), macro/micro recalls (REC), and macro/micro precisions (PRE). The standard deviation is minimal and can be considered negligible, indicating that the performance of EdgeMTSC is robust.

5 Conclusion

This study proposes a novel lightweight large-kernel ConvNet, named EdgeMTSC, for multivariate time series classification at the edge. It agrees with an extremely austere pipeline without additional pre-processing, but z-score normalization. The novel conv block named IMP-KB is introduced to decouple and learn correlation-aware time series representations using multi-branch conv layers, which follow the SRP mechanism so as to reduce parameters and speed up inference. The experimental results show that the performance of EdgeMTSC is superior to the SOTA model and address the existing challenges of MTSC at the edge. The comprehensive case studies show that our EdgeMTSC (especially, its re-parameterized version, RepEdgeMTSC) is feasible to be deployed and used on various kinds of devices in different application scenarios.

Acknowledgements

This work was supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT)(**RS-2025-25422680**, Metacognitive AGI Framework and its Applications, and **RS-2020-II201373**, Artificial Intelligence Graduate School Program (Hanyang University)).

References

Abdellatif, A. A.; Mohamed, A.; Chiasserini, C. F.; Tlili, M.; and Erbad, A. 2019. Edge computing for smart health: Context-aware approaches, opportunities, and challenges. *IEEE Network*, 33(3): 196–203.

Aldahiri, A.; Alrashed, B.; and Hussain, W. 2021. Trends in using IoT with machine learning in health prediction system. *Forecasting*, 3(1): 181–206.

Bagnall, A.; Dau, H. A.; Lines, J.; Flynn, M.; Large, J.; Bostrom, A.; Southam, P.; and Keogh, E. 2018. The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*.

Bouchabou, D.; Nguyen, S. M.; Lohr, C.; LeDuc, B.; and Kanellos, I. 2021. A survey of human activity recognition in smart homes based on IoT sensors algorithms: Taxonomies, challenges, and opportunities with deep learning. *Sensors*, 21(18): 6037.

Campos, D.; Zhang, M.; Yang, B.; Kieu, T.; Guo, C.; and Jensen, C. S. 2023. LightTS: Lightweight time series classification with adaptive ensemble distillation. *Proceedings of the ACM on Management of Data*, 1(2): 1–27.

Dempster, A.; Schmidt, D. F.; and Webb, G. I. 2021. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 248–257.

Ding, X.; Chen, H.; Zhang, X.; Han, J.; and Ding, G. 2022a. Repmlpnet: Hierarchical vision mlp with re-parameterized locality. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 578–587.

Ding, X.; Guo, Y.; Ding, G.; and Han, J. 2019. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1911–1920.

Ding, X.; Zhang, X.; Han, J.; and Ding, G. 2022b. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11963–11975.

Ding, X.; Zhang, X.; Ma, N.; Han, J.; Ding, G.; and Sun, J. 2021. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13733–13742.

Ding, X.; Zhang, Y.; Ge, Y.; Zhao, S.; Song, L.; Yue, X.; and Shan, Y. 2024. UniRepLKNet: A Universal Perception Large-Kernel ConvNet for Audio Video Point Cloud Time-Series and Image Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5513–5524.

Dosovitskiy, A. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

D'Urso, P.; and Maharaj, E. A. 2012. Wavelets-based clustering of multivariate time series. *Fuzzy Sets and Systems*, 193: 33–61.

Eldele, E.; Ragab, M.; Chen, Z.; Wu, M.; and Li, X. 2024. Tslanet: Rethinking transformers for time series representation learning. *arXiv preprint arXiv:2404.08472*.

Fauvel, K.; Fromont, É.; Masson, V.; Faverdin, P.; and Termier, A. 2020. Local cascade ensemble for multivariate data classification. *arXiv preprint arXiv:2005.03645*, 43.

Foumani, N. M.; Tan, C. W.; Webb, G. I.; and Salehi, M. 2024. Improving position encoding of transformers for multivariate time series classification. *Data Mining and Knowledge Discovery*, 38(1): 22–48.

Grabocka, J.; Wistuba, M.; and Schmidt-Thieme, L. 2016. Fast classification of univariate and multivariate time series through shapelet discovery. *Knowledge and information systems*, 49: 429–454.

Hao, Y.; and Cao, H. 2020. A new attention mechanism to classify multivariate time series. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*.

Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4): 917–963.

Karim, F.; Majumdar, S.; Darabi, H.; and Harford, S. 2019. Multivariate LSTM-FCNs for time series classification. *Neural networks*, 116: 237–245.

Le, X.-M.; Luo, L.; Aickelin, U.; and Tran, M.-T. 2024. Shapeformer: Shapelet transformer for multivariate time series classification. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1484–1494.

Li, G.; Choi, B.; Xu, J.; Bhowmick, S. S.; Chun, K.-P.; and Wong, G. L.-H. 2021. Shapenet: A shapelet-neural network approach for multivariate time series classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 8375–8383.

Lin, T.; Wang, Y.; Liu, X.; and Qiu, X. 2022. A survey of transformers. *AI open*, 3: 111–132.

Liu, F.; Tang, G.; Li, Y.; Cai, Z.; Zhang, X.; and Zhou, T. 2019. A survey on edge computing systems and tools. *Proceedings of the IEEE*, 107(8): 1537–1562.

Liu, H.; Yang, D.; Liu, X.; Chen, X.; Liang, Z.; Wang, H.; Cui, Y.; and Gu, J. 2024. Todynet: temporal dynamic graph neural network for multivariate time series classification. *Information Sciences*, 677: 120914.

Liu, M.; Ren, S.; Ma, S.; Jiao, J.; Chen, Y.; Wang, Z.; and Song, W. 2021. Gated transformer networks for multivariate time series classification. *arXiv preprint arXiv:2103.14438*.

Liu, S.; Chen, T.; Chen, X.; Chen, X.; Xiao, Q.; Wu, B.; Kärkkäinen, T.; Pechenizkiy, M.; Mocanu, D.; and Wang, Z. 2022. More convnets in the 2020s: Scaling

up kernels beyond 51x51 using sparsity. *arXiv preprint arXiv:2207.03620*.

Luo, D.; and Wang, X. 2024. Moderntcn: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*.

Mohammadi Foumani, N.; Miller, L.; Tan, C. W.; Webb, G. I.; Forestier, G.; and Salehi, M. 2024. Deep learning for time series classification and extrinsic regression: A current survey. *ACM Computing Surveys*, 56(9): 1–45.

Mu, Y.; Shahzad, M.; and Zhu, X. X. 2025. MPTSNet: Integrating Multiscale Periodic Local Patterns and Global Dependencies for Multivariate Time Series Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 19572–19580.

Mukhopadhyay, S.; Dey, S.; Mukherjee, A.; Pal, A.; and Ashwin, S. 2024. Time Series Classification on Edge with Lightweight Attention Networks. In *2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 487–492. IEEE.

Murshed, M. S.; Murphy, C.; Hou, D.; Khan, N.; Ananthanarayanan, G.; and Hussain, F. 2021. Machine learning at the network edge: A survey. *ACM Computing Surveys (CSUR)*, 54(8): 1–37.

Pantiskas, L.; Verstoep, K.; Hoogendoorn, M.; and Bal, H. 2022. Taking ROCKET on an efficiency mission: Multivariate time series classification with LightWaveS. In *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 149–152. IEEE.

Schäfer, P.; and Leser, U. 2017. Multivariate time series classification with WEASEL+ MUSE. *arXiv preprint arXiv:1711.11343*.

Tahaei, H.; Afifi, F.; Asemi, A.; Zaki, F.; and Anuar, N. B. 2020. The rise of traffic classification in IoT networks: A survey. *Journal of Network and Computer Applications*, 154: 102538.

Tang, W.; Long, G.; Liu, L.; Zhou, T.; Blumenstein, M.; and Jiang, J. 2020. Omni-scale cnns: a simple and effective kernel size configuration for time series classification. *arXiv preprint arXiv:2002.10061*.

Vaswani, A. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Wang, J.; Yang, C.; Jiang, X.; and Wu, J. 2023. WHEN: A Wavelet-DTW hybrid attention network for heterogeneous time series analysis. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2361–2373.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2022. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*.

Xiao, Z.; Xing, H.; Qu, R.; Feng, L.; Luo, S.; Dai, P.; Zhao, B.; and Dai, Y. 2024. Densely knowledge-aware network for multivariate time series classification. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(4): 2192–2204.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Zerveas, G.; Jayaraman, S.; Patel, D.; Bhamidipaty, A.; and Eickhoff, C. 2021. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2114–2124.

Zhang, X.; Gao, Y.; Lin, J.; and Lu, C.-T. 2020. Tapnet: Multivariate time series classification with attentional prototypical network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 6845–6852.

Zhang, Y.; and Yan, J. 2023. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*.

Zhou, X.; Lu, Q.; and Chae, D.-K. 2025a. What If LLMs Can Smell: A Prototype. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-25)*, 11141–11144.

Zhou, X.; Lu, Q.; and Chae, D.-K. 2025b. "Where Does This Strange Smell Come from?": Enabling Conversational Interfaces for Artificial Olfaction. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 10719–10745.

Zuo, R.; Li, G.; Choi, B.; Bhowmick, S. S.; Mah, D. N.-y.; and Wong, G. L. 2023. SVP-T: a shape-level variable-position transformer for multivariate time series classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 11497–11505.