Information Fusion 126 (2026) 103584

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full length article

Variable-length time series classification: Benchmarking, analysis and effective spectral pooling strategy

Shiling Wu [a,b,1], Siyang Song [e,1], Songhe Deng [a,b], Weicheng Xie [a,b,c], Linlin Shen [a,b,c,d,*]

[a] Computer Vision Institute, School of Artificial Intelligence, Shenzhen University, Shenzhen, Guangdong, 518060, China
[b] National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen, Guangdong, 518060, China
[c] Guangdong Provincial Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen, Guangdong, 518172, China
[d] Department of Computer Science, University of Nottingham Ningbo China, Ningbo, Zhejiang, 315100, China
[e] University of Exeter, Stocker road, Exeter, Devon, EX4 4PY, United Kingdom

ARTICLE INFO

Keywords:
Variable-length time series
Time series classification
Spectral pooling
Benchmark

ABSTRACT

Real-world time series classification (TSC) is challenging as time series collected in real-world conditions usually exhibit variations in their lengths, which makes standard deep learning (DL) models being difficult to directly process them (i.e., multiple variable-length time series (VTS)). Despite the existence of many pre-processing and pooling-based methods for achieving length normalization for VTS, there lacks a comprehensive and fair comparison across these methods through a uniform benchmark (e.g., standard backbones, datasets and evaluation strategies). To address this gap, we conduct the first comprehensive benchmark for variable-length time series classification tasks, evaluating the effectiveness of 22 previously widely-used length normalization methods across 14 publicly available VTS datasets and 8 backbones. Since these existing methods lead to varying degrees of information loss and distortion of the input VTS, we also propose a novel spectral pooling (SP) for variable-length time series classification (VTS classification) tasks, which is a plugin layer that can be inserted at any location within various DL models. Our SP allows DL models to process VTS or their variable-length representations in an end-to-end manner within mini-batches, without distortion or significant information loss. Experimental results demonstrate that the end-to-end length normalization methods generally outperformed pre-processing-based methods for VTS classification, where our SP achieved state-of-the-art performance across eight backbones over all existing 22 methods. Our code is publicly available at https://github.com/CVI-SZU/VTS_benchmark.

1. Introduction

Time series is a series of consecutive data points ordered in the temporal dimension, whose classification (time series classification (TSC)) involves assigning a categorical label based on its shape, trends, and other relevant temporal patterns. In the past decades, TSC solutions have been widely explored across different real-world applications, including financial investment [1], healthcare [2,3] and load device identification [4]. However, a common challenge of real-world time series is that they often exhibit variations in lengths caused by different sampling frequencies, acquisition duration or other factors. While the length variation issues caused by irregular sampling or missing data have been well studied [5–7], the problem of length variation due to varying recording duration remains under-explored.

Traditional hand-crafted time-series feature extraction methods frequently address such variable-length time series classification (VTS classification) tasks by employing specific measurements to compute distances between each pair of variable-length time series (VTS) (e.g., dynamic time warping [8] and longest common subsequence [9]). Based on such distances, each time series can be then predicted using various distance-related classifiers (e.g., nearest neighbor classification [10] and support vector machines [11]). Alternatively, other hand-crafted approaches also extract equal-length representations (i.e., using symbolic aggregation approximation [12], statistics [13] or identity-vector [14]) to represent VTS, which subsequently can be classified by standard machine learning techniques [12,14]. However, these hand-crafted feature-based methods usually heavily rely on the domain knowledge, making them less generalizable across different TSC tasks and datasets.

Recent deep learning (DL) models are more flexible and effective for learning task-specific features from various types of time series

* Corresponding author.
E-mail address: llshen@szu.edu.cn (L. Shen).
[1] Equal contribution.

https://doi.org/10.1016/j.inffus.2025.103584
Received 24 August 2024; Received in revised form 17 May 2025; Accepted 26 July 2025
Available online 8 August 2025

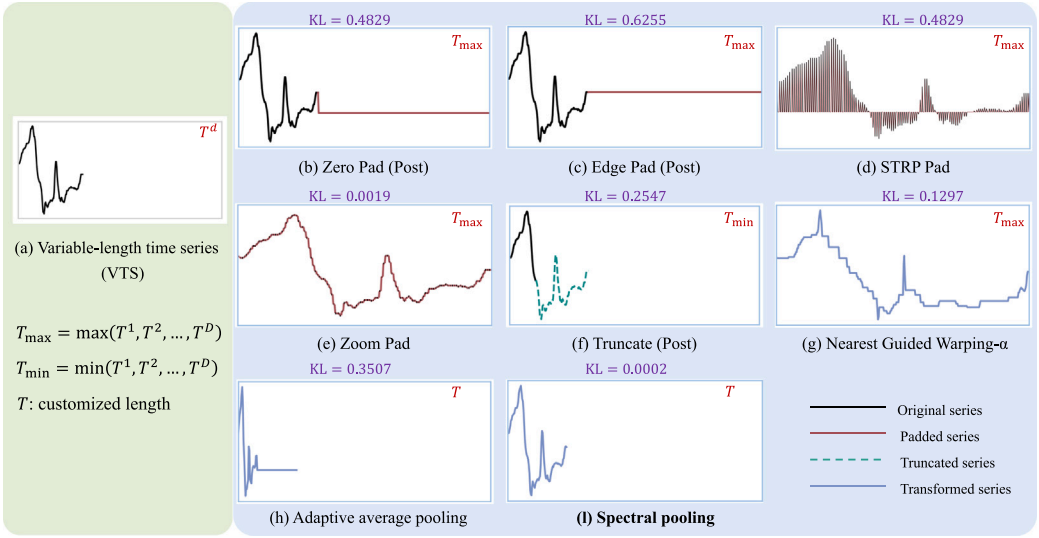

Fig. 1. Comparison between different length normalization strategies on a VTS dataset containing D examples $\mathcal{X} = \{(X^1, T^1), (X^2, T^2), \ldots, (X^D, T^D)\}$, where X^d $(d = 1, 2, \ldots, D)$ has the length T^d. (a) The original time series X^d; **(b) Zero Pad (Post):** padding zeros at the end of the X^d to make it having the length T_{max}, where T_{max} denotes the length of the longest example in $\mathcal{X}$; **(c) Edge Pad (Post):** padding the end element of the X^d to make it have the length T_{max}; **(d) STRP Pad:** resampling T^d time steps to T_{max} length, where the sampled $T_{max} - T^d$ time steps are defined as zeros; **(e) Zoom Pad:** resampling T^d time steps to T_{max} length, where each of the sampled $T_{max} - T^d$ time steps are defined by the nearest neighboring time step; **(f) Truncate (Post):** truncating the last components of X^d to make it have the length T_{min}, where T_{min} denotes the length of the shortest example in $\mathcal{X}$; **(g) Nearest Guided Warping-α:** applying dynamic time warping to warp X^d according to its most similar prototypical series of the length T_{max}; **(h) Adaptive average pooling:** first adaptively calculating the window length based on a pre-defined length T and the length of input time series, and then calculating the mean value of elements within each window to define the corresponding component of the output time series. However, this strategy still requires data padding for batch training, and cannot exclude the noise caused by padded data. **(i) Spectral pooling:** Our SP truncates X^d in its frequency domain according to the pre-defined length T, which can exclude the padded data used for batch training. It avoids information distortion ((b)–(e) and (h))/significant information loss ((f)). The Kullback–Leibler (KL) divergence measures the difference between the probability distribution of original time series X^d and their normalized ones, where the smaller the KL divergence, the more similar between them.

for different TSC tasks [15–18]. However, while typical DL layers (e.g., convolution, pooling, and activation layers) can handle multiple variable-length inputs, the Multi-layer Perceptron (i.e., fully connected (FC) layers)-based classifier/regressor used in standard DL models is limited to process equal-length representations, making standard DL models difficult to directly process multiple VTS. A common solution to address this problem is pre-processing (e.g., truncation, padding, resampling, or warping) multiple VTS to normalize them [19,20] to the target length. However, each of these length normalization strategies has been primarily evaluated on specific datasets/tasks based on task-specific model architectures, making it unclear about not only their performances for standard VTS classification, but also how well they generalize to different backbones. In other words, there lacks a comprehensive and fair comparison/investigation among existing length normalization strategies (**Research gap 1**). Although some methods—such as adaptive average or max pooling [21–23] are model-agnostic and can already effectively normalize the lengths of input VTS within DL models end-to-end, they usually lead to varying degrees of information loss or distortion (**Research gap 2**). For example, adaptive pooling methods, (e.g., adaptive average or max pooling) usually require padding all input VTS within a batch to a uniform length when batch training is necessary [23,24]. Consequently, the padded data points (e.g., padded zeros) are typically included in the pooling operation, leading to the information distortion of the obtained representations. In addition, these pooling operations only capture local (short-term) statistical information within the window without considering long-term shape/trend cues of the input VTS.

To comprehensively and fairly investigate the effectiveness of existing strategies for DL-based VTS classification, this paper presents the first benchmark evaluation and analysis of existing length normalization strategies based on eight widely-used DL architectures and 14 publicly available VTS classification datasets [25,26]. Targeting the limitations of existing length normalization strategies for variable-length time series (VTS) analysis, we propose a generic, flexible, and effective spectral pooling (SP) layer that can be seamlessly integrated into deep learning models. Our SP consists of two main components: (1) a Length Tracking and Undistorted Spectral Representation Generation (LSG) strategy, which encodes variable-length VTS into undistorted spectral representations in a batch manner; and (2) an Adaptive Length Normalization (ALN) mechanism, which transforms these spectral representations back into equal-length time-domain representations by selectively truncating less informative high-frequency components. Our SP reduces information distortion by excluding padded data points while reducing information loss by truncating less informative high-frequency components in the frequency domain. Fig. 1 illustrates the performance of various length normalization methods, clearly demonstrating SP's superior ability in preserving the original data distribution. The main novelties and contributions of this paper are summarized as:

- To the best of our knowledge, this is the first VTS classification benchmark which comprehensively evaluates the effectiveness of our SP layer and 22 existing widely-used length normalization strategies, across a total of eight backbones including Multi-layer Perceptron (MLP) [15], Long-short-term-memory Network (LSTM) [27], Convolutional Neural Networks (CNNs) [15,28,29], and Transformers [30,31].
- We propose an effective SP layer for DL-based VTS classification tasks, allowing multiple VTS to be jointly processed by DL models in a mini-batch manner reducing information loss/distortion.
- Our benchmark reveals that end-to-end pooling-based length normalization strategies outperformed the pre-processing-based strategies, which facilitate the DL model to obtain a more aggregated feature distribution, where our SP achieved the state-of-the-art performances across all eight backbones among all length normalization strategies.

2. Related work

This section first reviews previous time series classification (TSC) methods (including equal-length TSC) in Section 2.1, and then specifically discuss widely-used length normalization strategies for variable-length TSC analysis (Section 2.2).

2.1. Time series classification

Over the last two decades, TSC algorithms have flourished as TSC has been considered as one of the most challenging problems in data mining [32]. Early traditional approaches have widely employed distance-based methods [33], with dynamic time warping (DTW) frequently used to measure the distance between series. Zhao et al. [34] introduced shapeDTW, which considers point-wise local structural information during series warping to avoid matching points with distinct neighborhood structures. Liu et al. [35] proposed a DTW-based time distortion coefficient where the time distortion coefficients are computed based on the compression, stretching, or matching alignment of each point in the series and comprehensively considers both the DTW distance and the time distortion coefficient for nearest-neighbor classification. Alternatively, feature-based approaches for time series classification utilize shapelets to identify distinctive parts of the series [36], or employ auto-correlation to measure dependence between neighboring data points [37]. These extracted features are then fed to various types of classifiers, including support vector machines [38], decision trees [39], and naive bayes [40], for TSC.

In recent years, various deep learning network architectures have been explored for TSC tasks. Wang et al. [15] introduced three simple architectures—multilayer perceptron (MLP), fully convolutional network (FCN), and ResNet—as baselines for TSC. Kao et al. [27] employed LSTM for acoustic event classification, while Karim et al. [41] combined LSTM and FCN to construct a two-branch network for multivariate TSC. To achieve reliable TSC performance, Fawaz et al. [28] incorporated the ensemble idea based on the Inception network, where the proposed InceptionTime consists of five Inception ensembles with the same architecture but different initialization parameters. By combining these networks equally, InceptionTime can effectively reduce the bias frequently occurred in single Inception networks. TimesNet [29] achieved excellent TSC performances by transforming the original 1D time series into a set of 2D tensors based on multiple periods, which are then processed using 2D convolutional kernels. The Transformer introduced by Vaswani et al. [30] has led to further improvements in time series classification (TSC) compared to CNNs. Specifically, recently proposed TSC Transformers have introduced novel attention mechanisms. For example, Informer [31] proposed ProbSparse attention, which focuses on computing attention for a selected subset of important positions to improve efficiency. On the other hand, Pyraformer [42] proposed pyramidal attention, which constructs a multi-resolution C-ary tree to form a pyramidal graph and applies the attention mechanism within this graph. These approaches improve the computational efficiency of self-attention, aiming to optimize temporal and spatial complexity while enhancing the performance of time series analysis tasks.

2.2. Length normalization strategies

Standard DL models usually cannot directly process VTS, as their FC layer-based classifiers/regressors require inputs to have the same size [43]. Existing approaches frequently address this problem by introducing various pre-processing strategies to ensure that all time series have the same length before feeding them to DL models [19]. These pre-processing strategies include various padding methods (e.g., zero padding [20], noise padding [44], stratified (STRF) padding that distributes zeros evenly [20], random padding that randomly places zeros [20], and zoom padding that duplicates neighboring values [20]), truncation [45], interpolation [44], and nearest guided warping which warps time series to the length of the most similar time series via DTW [19]. Fourier transform-based methods have gained increasing attention in time series analysis. In FEDformer [46], randomly chosen frequency components are used to reduce computations for long time series. Wu et al. [29] and Cai et al. [47] use the Fourier transform to detect main periodicities and reshape time series accordingly. Similarly, Song et al. [3,13] applied the Fourier transform to select consistent frequency components across different series lengths to obtain equal-length representations, which has been further followed by several time-series analysis studies [48–51].

Alternatively, other studies normalize the length of time series within DL models, where pooling layers have been commonly used [24, 52]. For example, Fawazd et al. utilizes global average pooling to handle variable-length transfer learning [53], and Yu et al. [23] proposes a temporal pyramid pooling which uses multiple pooling layers with different output sizes for multi-scale feature extraction. Xception-Time [21] applies adaptive average pooling to retain more information through larger output dimensions. Malekzadeh et al. [22] proposes a DANA network to process variable-length multivariate sensor data based on adaptive pooling. As discussed before, models equipped with these pooling layers still require pre-processing-based length normalization to achieve batch training and cannot exclude the noise caused by pre-processing, which distorts the pooled feature maps to be distorted.

3. The proposed benchmarking framework

In this paper, we propose the first benchmarking framework that aims to provide a rigorous and reproducible evaluation of the existing length normalization strategies for variable-length time series classification (VTS classification) tasks on eight widely-used deep learning (DL) backbones. Specifically, this section introduces our benchmarking framework by presenting: (i) its coding infrastructure including employed DL backbones and their settings (Section 3.1); (ii) benchmarked length normalization strategies (Section 3.2); as well as (iii) the datasets used for evaluation (Section 3.3). The pipeline of the proposed VTS benchmarking framework is also illustrated in Fig. 2

3.1. Coding infrastructure

The goal of our benchmark is to fairly compare the capabilities of existing widely-used length normalization strategies for VTS classification as previous studies failed to provide fair comparison among them as: (i) previous studies evaluated their length normalization strategies on different datasets; (ii) their evaluation was only carried out on task/dataset-specific backbone architectures, lacking assessment of generalization performance across different standard DL backbones; and (iii) different training strategies and model hyperparameter settings have been employed, which also lead to effectiveness differences.

To facilitate a fair comparison among existing length normalization strategies for VTS classification, our benchmark emphasizes a unified framework. Specifically, it is built on PyTorch [54] with unified implementation of the data pipeline, model initialization, training, validation, evaluation, and coding platform/libraries. The only differences in experiments lies in the length normalization strategies and model architectures of the employed DL models.

3.1.1. Data pipeline

Our benchmark utilizes three different data pipelines at the training phase depending on the evaluated length normalization strategies: **(i) Pre-processing strategies:** each original time series is first normalized in length using the target pre-processing strategy, and then fed into DL models in a mini-batch manner; and **(ii) Other pooling-based strategies:** all original time series are first zero padded to have the same length T_{pad}, which are then fed to DL models in a mini-batch manner. **(iii) Our spectral pooling:** all original time series are first zero padded to have the same length T_{pad} and the original lengths of the input time series also be saved as a vector $\mathbf{T} = (T^1, T^2, \dots, T^B)$, which are fed to DL models in a mini-batch manner together.

3.1.2. Employed deep learning backbones

The goal of this paper is to explore length normalization strategies for deep learning-based VTS classification. To fairly evaluate the

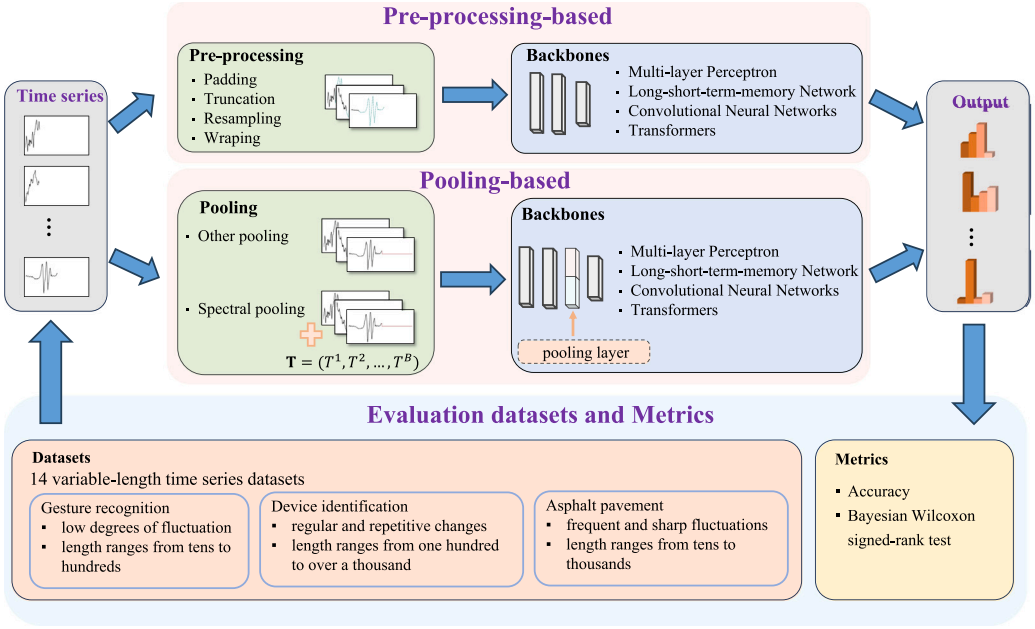

Fig. 2. Illustration of our variable-length time series classification benchmark framework.

generalization capability of these strategies, we utilized eight typical neural networks commonly used for TSC tasks [15,27,28,30], including Multi-layer Perceptron (MLP), Long-short-term-memory Network (LSTM), Convolution Neural Networks (CNNs) and Transformers, which are detailed as follows:

- **MLP** [15] consists of three fully-connected (FC) layers with 256, 512 and 256 neurons, respectively, where each FC layer is followed by a Gaussian Error Linear Unit (GELU) activation. Finally, a classification layer, i.e., a FC layer where the number of neurons is equal to the number of categories in the dataset and the activation function is Softmax, is attached at the end of this MLP. This is a typical classifier that has been widely used for TSC tasks [16,55].
- **LSTM** [27] is made up of two LSTM layers with each containing 128 hidden units. A feature-wise global max pooling is then applied across the sequence output, which summarizes patterns from the entire input time series. The pooled representation is finally fed into a final FC layer with the number of neurons equal to the number of categories in the dataset to output the predicted probabilities for each category. As a natural sequential data analysis model, LSTM also has been widely-used for tasks [56,57].
- **FCN** [15] consists of three convolutional layers (convs) with 128, 256 and 128 kernels of kernel sizes equaling to 7, 5, and 3 respectively. A GELU activation is attached after each convolutional layer without batch normalization (BN), as the padded zeros would cause incorrect batch normalization. A global average pooling is then deployed after the final convolutional layer to summarize features before classification. This standard CNN has been employed as baselines in TSC task [58,59].
- **ResNet** [15] is made up of 3 repeated residual blocks. Each block contains 3 convolutional layers with the kernel sizes 7, 5, and 3, and 64, 128, and 128 kernels, respectively. A global average pooling layer is employed after the last residual block to summarize features before classification. Residual connection was introduced on the basis of FCN, which aims to avoid gradient vanishing. This has been presented to be superior to many other DL models in TSC task [60,61].
- **Inception** [28] consists of 2 residual blocks, where each contains 3 inception modules that pass the input through parallel convolutions with 32 kernels and kernel sizes of 9, 19, and 39, respectively, which is followed by a max pooling with a fixed window, aiming to concatenate the outputs before sending to the next module. A global average pooling and a FC-based classification layer are attached at the end of the network. Inception also uses to residual connections and introduces parallel convolution on that basis and increases the convolution kernel size. This is widely used to extract multi-scale features in TSC task [29,62].
- **Transformer** [30] encodes the input and incorporates positional encoding to represent it as a 128-dimensional embedding. This embedding is then processed through 3 encoder layers, each containing 8-headed attention. Subsequently, global max pooling is applied to the encoder output, summarizing sequence features prior to being fed into the classification output layer. Transformer is attention-based network architectures, which have been widely influential in deep learning in recent years [18,63].
- **Informer** [31] encodes the input and incorporates a positional encoding to represent it as a 128-dimensional embedding. This embedding is then processed through 3 encoder layers, each containing 8-headed ProbSparse self-attention. A global average pooling and a FC-based classification layer are attached at the end of the network. This is the Top-3 model on TSC task according to the Time-Series-Library repository.[2]
- **TimesNet** [29] encodes the input and incorporates positional encoding to represent it as a 128-dimensional embedding. This embedding is then processed through 3 TimesBlock with $k = 3$ and each TimesBlock consists of two inception convolution blocks, each containing six 2D convolutional layers with increasing kernel sizes. A global average pooling and a FC-based classification layer are attached at the end of the network. TimesNet transforms the original 1D time series into a set of 2D tensors based on multiple periods, which are then processed using 2D convolutional kernels. This is the Top-1 model on TSC task as show in the Time-Series-Library repository.

3.1.3. Training and testing protocol

We employ the same training and testing protocol for all benchmarked length normalization strategies. Specifically, each model is

[2] https://github.com/thuml/Time-Series-Library.

Table 1
Training hyper-parameter settings for all experiments.

Learning rate	Optimizer	Beta1 (Adam)	Beta2 (Adam)	Batch size	Epochs	Loss function
0.001	Adam	0.9	0.999	64	100	Cross-entropy

trained using the training set, and the reported results denote the best variant achieved on the test set. This setting is consistent with previous studies [19,29], where no validation has been conducted. Training hyper-parameters (detailed provided in Table 1) were kept consistent across all experiments, despite that the possibility that different length normalization methods and their associated models might benefit from varied optimal settings. This decision was made due to the vast range of possible training parameters and the impracticality of conducting individual parameter experiments for each of the numerous length normalization methods. Instead, we opted for a widely-used training configuration [21,29] combined with a learning rate decay strategy. In this strategy, the learning rate is reduced to 10% of its current value if the training loss does not decrease for 10 consecutive epochs, with a minimum learning rate of 1e–5. This allows for reducing the learning rate when training becomes challenging, aiming to ensure fair comparison. All experiments were conducted on a single V100 GPU.

3.1.4. Evaluation metrics

Our benchmark follows previous TSC studies [19,29] to employ classification accuracy as the metrics to evaluate the performance of benchmarked approaches as:

$$\mathrm{ACC} = \frac{1}{N}\sum_{i=1}^{N}[y_i = \hat{y}_i] \tag{1}$$

where N denotes the total number of evaluated time-series samples; y_i denotes the label of the i_{th} time-series sample; $\hat{y}_i$ denotes the prediction for i_{th} time-series sample; and $[y_i = \hat{y}_i]$ represents an indicator function that equals 1 if $y_i = \hat{y}_i$, else 0. In addition, we also provide statistical analysis using the Bayesian Wilcoxon signed-rank test [64] compare two length normalization strategies.

3.2. Benchmarked length normalization strategies

In this paper, we benchmark 20 pre-processing strategies and two pooling-based strategies, as well as our proposed spectral pooling strategy based on 6 widely-used deep learning (DL) backbones and the state-of-the-art Informer [31] and TimesNet [29].

3.2.1. Inclusion and exclusion criteria

Inclusion criteria: To fairly benchmark the most widely-used and representative length normalization methods, this paper chooses the benchmarked length normalization methods based on the following criteria:

- (1) The main criterion for choosing length normalization methods for our benchmark is that they have been evaluated and compared on at least one variable-length time series dataset in the UCR archive [25], which is a well-known and widely used publicly available time series classification archive.
- (2) The second criterion is that the chosen methods must have been proposed for addressing the variable-length time series analysis problem within the last four years (from 2020 to 2024) [3,19,20].
- (3) The third criterion is that we choose the typical length normalization methods that do not have learnable parameters, and have been defined and widely discussed in previous studies [16,20,44,45,65], including zero-padding, truncation, as well as adaptive maximum pooling and adaptive average pooling that have been frequently used as baselines when comparing other pooling methods.

Exclusion criteria: In this benchmark, we excluded length normalization methods based on the following criteria:

- (1) We exclude pre-processing-based length normalization methods that require domain-specific post-processing or require domain knowledge to set specific parameters (e.g., window-based methods [26,66] and shapelet-based methods [67,68]), as these methods may not generalize for TSC on different datasets.
- (2) We exclude pooling-based length normalization methods that have trainable parameters, such as dynamic temporal pooling [58], which introduces hidden vectors in the pooling layer, thus creating an unfair comparison.

3.2.2. Benchmarked length normalization strategies

Existing length normalization methods can be divided into two main categories: pre-processing-based length normalization and pooling-based length normalization. Among the pre-processing methods, we present 20 pre-processing strategies that can be classified into four categories padding, truncation, resampling, and warping. Padding introduces series information that did not exist originally. Truncation loses a substantial amount of series information. Resampling based on linear interpolation slows the degree of change in the series, although it maintains the original shape as much as possible. Similarly, resampling in the frequency domain identifies the same frequency components across different series, solving the frequency misalignment problem but incurring a loss of low-frequency information crucial for classification tasks. The warping method utilizes DTW to distort according to most similar series to match the length of a reference series. The judgement and selection of the most similar series is related to the accuracy of classification. Additionally, we include two pooling-based strategies for normalizing the length inside the models and does not add additional training parameters.

Padding: Padding strategies extend all time series to match the length of the longest series by applying various padding techniques.

- **Zero Pad** [20]: The most commonly used padding approach, where zeros are filled at the start (Pre), middle (Mid), end (Post), or both ends (Outer) of the time series. This padding technique has been applied in various domains, including functional protein prediction [20], segmenting series into fixed periods and padding the remaining series length to the same length [29], and ensuring consistency between input and output lengths in convolutional operations through the use of zero padding [69].
- **Noise Pad** [44]: Random noise sampled from a low-amplitude distribution is used for padding at the start (Pre), end (Post), or both ends (Outer) of the time series. Noise pad is also a common use in the classification of variable series [44,70,71].
- **Edge Pad** [19]: Original values from the time series are duplicated for padding. The first (Pre) or last (Post) element is duplicated, or both (Outer) ends are duplicated. Edge pad is employed in emotion recognition for variable speech series [72] and prediction of battery capacity [73].
- **STRP Pad and Random Pad** [20]: These methods aim to enhance padding diversity. STRP Pad pads the series with zeros in an interpolated order so that the zeros are evenly distributed in the series, while Random Pad introduces zeros at random positions within the time series. STRP Pad and Random Pad were proposed for protein series padding by Lopez-del Rio et al. [20], but they have also been used as an attack to test the robustness of models [74].

- **Zoom Pad** [20]: The time steps of the original series are resampled, and then the added time steps are padded using neighboring time step elements. This is also the benchmarked padding method in [20].

Truncate: This is a method for addressing variable-length time series, where longer series in the training set are shortened to match the length of the shortest series. Truncation can be performed based on the position, such as Truncation (Pre) removing values from the start, Truncation (Post) removing values from the end, and Truncation (Outer) removing values from both ends. Cerqueira et al. utilized truncation to unify series lengths and speed up computation [75], a technique also employed in unifying the lengths of human activity series [76].

Resampling: This method matches a time series to a specified length by interpolating or extracting values at intervals from the original series.

- **Linear Interpolation** [19]: New points are interpolated along the linear slope between existing points in the time series. Linear interpolation is also a frequently used method for filling in missing data [77,78].
- **Select Common Frequency** [3]: Time series are converted to the frequency domain via Fourier transform, and an equal number of common frequency components are selected across all series, resulting in transformed representations with a consistent length. This method was proposed by Song et al. for length normalization of video series in depression detection.

Wrapping: Based on the most similar sequence, the original sequence is warped to match the length of the most similar series.

- **Nearest Guided Warping** [19]: This method utilizes dynamic time warping (DTW) to warp time series to a consistent length. It selects prototype series using the hyper-parameter α and then resamples them to equal lengths set by the hyper-parameter β. The remaining series are then compared to the prototypes via DTW and warped based on the most similar prototype. DTW is widely used in variable-length time series for alignment and similarity calculation between two series [79]

Pooling-based Length normalization can also be achieved within the model architecture through pooling implementations.

- **Adaptive max pooling** [80]: Adjusts its window size and stride based on the length of the input time series to produce a fixed output length. It finds the maximum value in each window and discards the other elements to generate the output. Adaptive max pooling is widely used in the downsampling phase of deep learning, leveraging its ability to receive inputs of arbitrary length and transform them to a fixed length without parameter training [81–83].
- **Adaptive average pooling**: This method adjusts its window size and stride based on the length of the input time series to produce a fixed output length. It calculates the average value in each window and discards all elements in the window to generate the output. Adaptive average pooling is similar to adaptive max pooling, except that it uses an averaging operation on the data within the window [21,84].

3.3. Datasets

All benchmarked systems are evaluated on 14 multi-class variable-length time series datasets recorded for three types of tasks: (1) **10 gesture recognition datasets** provided by the UCR archive [25], which mainly collect acceleration and hand trajectory data during hand movements. These time series exhibit low degrees of fluctuation. The length of most dataset's series ranges from tens to hundreds, and all are multi-categorical datasets, with the GestureMidAir dataset containing the most categories (26 kinds of hand movements); (2) **1 device identification datasets** provided by the UCR archive [25], which contain both steady-state operation and startup transient current and voltage measurements labeled by eleven appliance categories. These time series are characterized by regular and repetitive changes, whose lengths range from one hundred to over a thousand frames; and (3) **3 asphalt pavement datasets** provided by [26], which collect the acceleration of a car when it runs in different situations, i.e., the collected time series involve frequent and sharp fluctuations. Specifically, these three datasets were provided for pavement condition classification, pavement type classification, and pavement obstacle classification, with categories ranging from 2 to 4. The dataset with the largest length range (i.e., AsphaltRegularity) spans from tens to thousands of data points. For all datasets, the officially defined training and test splits are used for models' training and evaluation and Table 2 lists detailed statistics of all benchmarked datasets.

4. Spectral pooling strategy

This section introduces our proposed spectral pooling (SP), including its overview (Section 4.1) and key modules (Section 4.2 and Section 4.3).

Finding and motivation: We conducted a frequency analysis on 117 equal-length datasets provided by the UCR archive. As shown in Figs. 11, 12, and 13, only 8 out of 117 datasets (three of them are simulated datasets) exhibited high amplitude values on high frequency components, accounting for only 7% of all datasets. Furthermore, Fig. 3(a) and (b) show that adding random Gaussian noise to original time series significantly increases their high frequency amplitude values. **These indicate that low and middle frequency components usually represent the main patterns while high frequency components usually represent noises for most natural time series.** Inspired by these, we propose a novel spectral pooling strategy for VTS classification, which removes high frequency components to achieve length normalization (detailed in Section 4.1). It can prevent significant information loss while denoising the given time series for their length normalization. Different from truncation-based strategies [44,45] that directly remove frames, our spectral pooling truncate time series in the frequency domain, which largely reduces the information loss. On the other hand, compared to padding methods [20,45] and adaptive max/average pooling [81,83] which introduce data points that do not exist, ours can exclude the negative influences of padded data points, and thus avoid information distortion.

4.1. The overview of spectral pooling

Spectral pooling (SP) layer is an effective and plugin layer that facilitates DL models to end-to-end process a batch of VTS reducing information loss/distortion. Our SP layer can be inserted between any layers within standard DL models and jointly optimized with the rest layers in an end-to-end manner. As illustrated in Fig. 4, given a batch of (B) VTS $\mathcal{X}_B = \{X^1, X^2, \dots, X^B\}$, the proposed SP layer processes them using on a two-step strategy.

- **Step 1: Length Tracking and Undistorted Spectral Representation Generation (LSG)** first represents the original lengths (frame numbers) of all VTS in $\mathcal{X}_B$ as $\mathbf{T} = (T^1, T^2, \dots, T^B)$, and then pads zeros to each original time series X^b accordingly, which is done in pre-processing. This results in all padded time series $\mathbf{X}_{\text{pad}} = \{X^1_{\text{pad}}, X^2_{\text{pad}}, \dots, X^B_{\text{pad}}\}$ having the same length T_{pad} to be input to the model. Based on the obtained $\mathbf{T}$, the LSG then computes an undistorted spectral representation X^b_{freq} from each padded time series X^b_{pad} by excluding the zero padded data points in the transform process, which is done in model.

Table 2
Statistics of 14 employed variable-length time series classification (TSC) training datasets from provided by UCR and Souza et al. . Class represents the number of label classes provided by the dataset. Min. and Max. denote the shortest and the lengths of the longest time series in the dataset. Ave. and Med. are the average and median lengths computed from all time series in the dataset. Cv. represents the coefficient of variation [85], calculated as the standard deviation divided by the mean, quantifies variability in lengths within the dataset.

Application	Dataset name	Class	Min.	Max.	Ave.	Med.	Cv. (%)
Gesture	AllGestureWiimoteX	10	11	385	124	118	52
Gesture	AllGestureWiimoteY	10	8	369	128	116	54
Gesture	AllGestureWiimoteZ	10	33	326	125	116	52
Gesture	GestureMidAirD1	26	80	360	166	171	38
Gesture	GestureMidAirD2	26	80	360	166	171	38
Gesture	GestureMidAirD3	26	80	360	166	171	38
Gesture	GesturePebbleZ1	6	115	455	233	197	35
Gesture	GesturePebbleZ2	6	100	455	223	200	39
Gesture	PickupGestureWiimoteZ	10	29	361	145	136	53
Gesture	ShakeGestureWiimoteZ	10	41	385	171	158	51
Device	PLAID	11	100	1344	323	300	44
Pavement	AsphaltObstacles	4	111	736	297	227	38
Pavement	AsphaltPavementType	3	96	1543	396	413	40
Pavement	AsphaltRegularity	2	95	4201	387	335	65

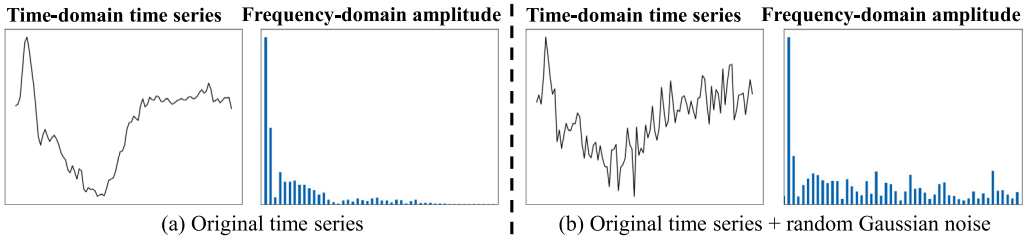

Fig. 3. (a) shows the original time series in the time and frequency domains; and (b) shows the original time series with added random Gaussian noise, in the time and frequency domains. The left part of each sub-graph displays the original time series in the time domain, while the right part shows the corresponding amplitude map in the frequency domain, with only half shown due to conjugate symmetry.

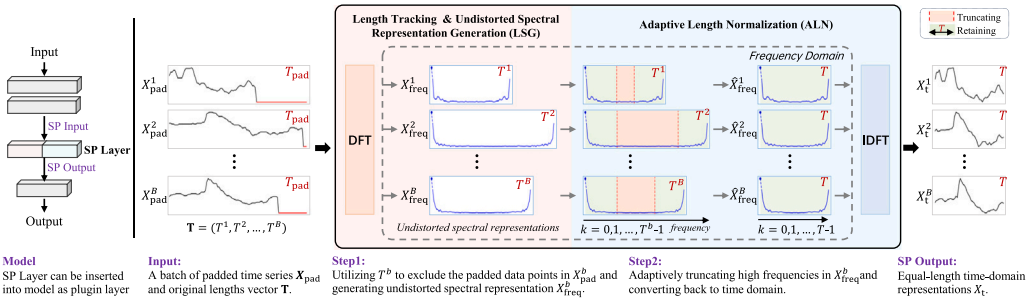

Fig. 4. Illustration of the proposed spectral pooling (SP). The LSG module (Section 4.2) first encodes the padded VTS as a set of undistorted spectral representations. Then, the ALN module (Section 4.3) further transforms them as a set of equal-length time domain representations.

- **Step 2: Adaptive Length Normalization (ALN)** is then carried out to remove several high frequency components from each undistorted spectral representation X^b_{freq} to ensure that all processed spectral representations have the same length. Then, it converts these spectral representations back to the time domain, avoiding frequency misalignment (explained in Section 4.3), i.e., the output of the SP layer is a set of equal-length time domain representations $\mathbf{X}_t = \{X^1_t, X^2_t, \ldots, X^B_t\}$, which retain the major of undistorted information of these input VTS.

In summary, our SP layer converts a set of VTS $\mathcal{X}_B$ to a set of equal-length time series $\mathbf{X}_t$ based on time-frequency transformation, which allows DL models to directly end-to-end batch process VTS reducing information loss/distortion. Importantly, applying our SP does not introduce additional model weights optimization burden, with its time complexity of $O(T\log(T))$.

4.2. Length tracking and undistorted spectral representation generation

The LSG enables DL models to batch process a set of VTS while preventing information distortion in an end-to-end manner. As illustrated in Fig. 5, LSG first records the length of all given time series (Step (i)) and applies zero-padding (Step (ii)) to pad all input variable-length time series as equal-length time-series, facilitating the network layers prior to our SP to batch process them. Then, the LSG leverages the recorded series lengths to exclude padded data points (Step (iii)) and performs the Discrete Fourier Transform (DFT) to generate spectral representations (Step (iv)). Since the padded zeros are not involved in the final DFT, the obtained spectral representations are undistorted. For a given batch of VTS $\mathcal{X}_B = \{X^1, X^2, \ldots, X^B\}$, the LSG first records their lengths in a vector $\mathbf{T} = (T^1, T^2, \ldots, T^B)$. Then, each time series $X^b \in \mathcal{X}_B$ is being padded with zeros at the end, making all padded time series X^b_{pad} having the same length T_{pad}. This zero-padding and

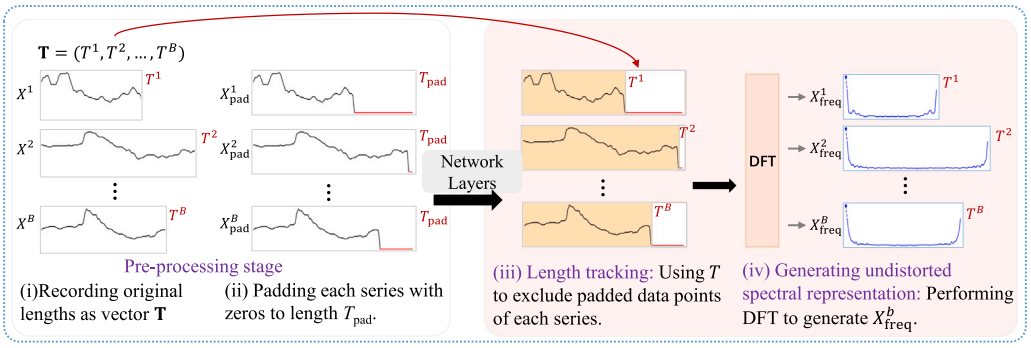

Fig. 5. Illustration of the entire LSG step of our SP, from the pre-processing stage to the generation of undistorted spectral representations.

length vector recording process is carried out in the pre-processing stage, enabling batch training of the network layers prior to the SP layer as well. Theoretically, T_{pad} can be an arbitrary length, where in subsequent experiments we follow a previous study [41] to set T_{pad} as the longest length in the training set as:

$$X^b_{\text{pad}} = (\underbrace{x^b_0, x^b_1, \dots, x^b_{T^b-1}}_{X^b}, \underbrace{x^0_{T^b}, \dots, x^0_{T_{\text{pad}}-1}}_{\text{padded zero data points}}) \tag{2}$$

where $x^0_{T^b}, \dots, x^0_{T_{\text{pad}}-1}$ denote the padded zero data points. Note that larger T_{pad} values just introduce more invalid padded data points, which do not enhance the performance. This way, all padded time series can be combined as a batch of equal-length time series $\mathbf{X}_{\text{pad}} = \{X^1_{\text{pad}}, X^2_{\text{pad}}, \dots, X^B_{\text{pad}}\}$ to be batch processed by DL models along with $\mathbf{T}$. At the inference stage, the test sample of an arbitrary length can be individually processed without padding or removing frames.

We then apply the DFT to encode each padded time series X^b_{pad} as a spectral representation $X^b_{\text{freq}} = \left(x^b_{\text{freq}}(0), x^b_{\text{freq}}(1), \dots, x^b_{\text{freq}}(T^b-1)\right)$ that contains T^b rather than T_{pad} frequency components from the padded time series X^b_{pad}, where the recorded length $T^b \in \mathbf{T}$ is utilized. Specifically, each frequency component $x^b_{\text{freq}}(k) \in X^b_{\text{freq}}$ is only computed based on the data points of the original time series X^b (i.e., the data points from x^b_0 to $x^b_{T^b-1}$ in X^b_{pad}), which avoids information distortion caused by the padded data points. This can be formulated as:

$$\begin{aligned} x^b_{\text{freq}}(k) &= \sum_{j=0}^{T^b-1} x^b_{\text{pad}}(j) e^{-i\frac{2\pi}{T^b}jk} \\ &= \text{Re}(x^b_{\text{freq}}(k)) + i\text{Im}(x^b_{\text{freq}}(k)) \end{aligned} \tag{3}$$

where $k = 0, 1, \dots, T^b - 1$; $\text{Re}(x^b_{\text{freq}}(k))$ and $\text{Im}(x^b_{\text{freq}}(k))$ represent the real and imaginary parts of the kth frequency component $x^b_{\text{freq}}(k)$, respectively. Here, each frequency component $x^b_{\text{freq}}(k)$ is computed from the entire time series X^b, capturing global temporal patterns at a specific frequency scale—where lower frequencies correspond to gradual, long-term trends, and higher frequencies capture sharp, short-term fluctuations. Consequently, the resulting spectral representation X^b_{freq} encodes multi-scale temporal dynamics of the original series, making the frequency domain a natural space for performing temporal information fusion.

4.3. Adaptive length normalization

The ALN step normalizes all obtained variable-length spectral representations X^b_{freq} $(b = 1, 2, \dots, B)$ in order to make them have the same length T while avoiding significant information loss/distortion. Here, T is set as the frame number corresponding to the shortest training time series. Specifically, for each spectral representation X^b_{freq} that has no less than T frequency components (i.e., $T^b \geq T$), our SP layer truncates its high frequency components to only retain T lowest frequency components that consist of: (i) the DC component $x^b_{\text{freq}}(0)$; (ii) the first $T/2$ lowest frequency components corresponding to $x^b_{\text{freq}}(k), k = 1, 2, \dots, T/2$; and (iii) the conjugate symmetric frequency components corresponding to $x^b_{\text{freq}}(k), k = T^b - T/2 + 1, \dots, T^b - 1$. This can be formulated as:

$$\hat{x}^b_{\text{freq}}(k) = \begin{cases} x^b_{\text{freq}}(k), & 0 \leq k \leq T/2 \\ x^b_{\text{freq}}(T^b - (T-k)), & T/2 < k < T \end{cases} \tag{4}$$

As a result, a batch of equal-length spectral representations $\hat{\mathbf{X}}_{\text{freq}} = \{\hat{X}^1_{\text{freq}}, \hat{X}^2_{\text{freq}}, \dots, \hat{X}^B_{\text{freq}}\}$ are obtained.

However, the spectral representations truncated from different VTS do not describe the same set of frequency components (i.e., if $T^1 \neq T^2$, the kth frequency components of the spectral representation X^1_{freq} and X^2_{freq} correspond to $2\pi k/T^1$ and $2\pi k/T^2$, respectively, and thus $2\pi k/T^1 \neq 2\pi k/T^2$.). As demonstrated in Eq. (3), this would result in frequency misalignment for the truncated spectral representations, and further lead to potential confusion for DL models' training and reasoning. To address this issue, we apply the Inverse Discrete Fourier Transform (IDFT) to decode the truncated equal-length spectral representations $\hat{\mathbf{X}}_{\text{freq}}$ to equal-length time domain representations $\mathbf{X}_t = \{X^1_t, X^2_t, \dots, X^B_t\}$ as:

$$x^b_t(n) = \frac{1}{T}\sum_{k=0}^{T-1} \hat{x}^b_{\text{freq}}(k) e^{i\frac{2\pi}{T}nk} \tag{5}$$

where $n = 0, 1, \dots, T-1$, and we set $\text{Im}(\hat{x}^b_{\text{freq}}(T/2)) = 0$ to ensure the conjugate symmetry of each spectral representation. This way, the twiddle factor $e^{i\frac{2\pi}{T}nk}$ [86] in Eq. (5) are consistent for all equal-length spectral representations during the IDFT. To better illustrate the implementation of ALN, we visualize the complete process in Fig. 6. First, the T lowest frequency components are selected from the original spectral representation X^b_{freq} based on Eq. (4). These retained components include the DC component, the lowest $T/2$ frequency components, and their conjugate symmetric frequency components. Next, conjugate symmetry is enforced by setting $\text{Im}(\hat{x}^b_{\text{freq}}(T/2)) = 0$, ensuring that the reconstructed time-domain representation obtained via IDFT remains real-valued. This process enables the transformation of spectral representations into equal-length and comparable time-domain representations that preserve the dominant information of the original variable-length inputs. In summary, the obtained time domain representations not only summarize the majority information of their corresponding VTS (i.e., from their low and middle frequency components), but also are encoded to the same length without distortion (i.e., the encoding process does not involve any re-shape/interpolation nor affected by padded information). More importantly, this IDFT operation avoid the frequency misalignment issue in directly processing these equal-length spectral representations.

However, directly defining T based on the shortest time series in the training set can cause excessive information loss (i.e., removed frequency components). Therefore, we introduce a trade-off factor α

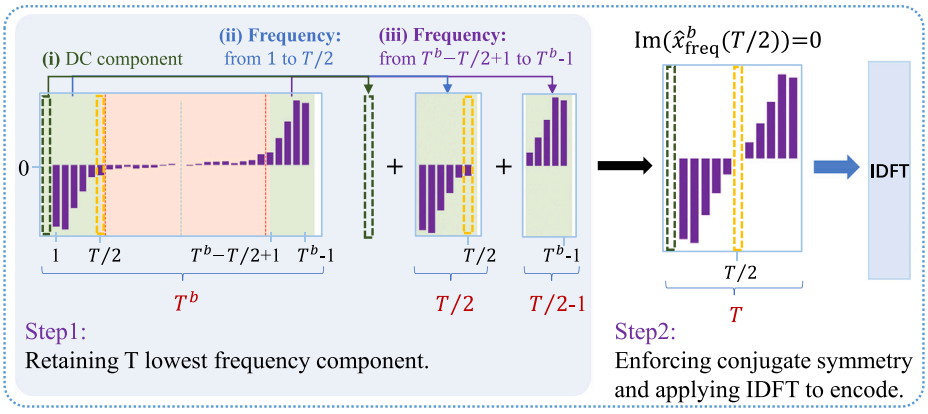

Fig. 6. Illustration of the entire ALN step in our SP, from retaining the lowest T frequency components to enforcing conjugate symmetry and applying IDFT to generate equal-length time domain representations. For clarity, only the imaginary part of the spectral representation is shown.

Table 3
Average test ACC results (%) over 14 variable-length datasets achieved by the benchmarked length normalization strategies for eight DL models. Spectral pooling **(p)** and Spectral pooling **(m)** denote that applying SP as the pre-processing and pooling layer, respectively. Bold values indicate the best results while the underlined values indicate the second best systems.

	Methods	MLP	LSTM	FCN	Resnet	Inception	Transformer	Informer	TimesNet
Pre-processing	***Padding***								
	Zero pad (Pre) [45]	62.82	62.98	68.80	72.25	77.27	**70.96**	63.08	69.66
	Zero pad (Post) [45]	62.50	62.74	68.80	70.47	77.57	70.28	61.93	70.53
	Zero pad (Outer) [20]	65.66	63.48	68.76	72.96	75.91	70.57	64.69	70.49
	Zero pad (Mid) [20]	63.39	62.12	67.38	71.42	73.70	69.07	64.07	70.00
	Zero pad Avg.	*63.59*	*62.83*	*68.43*	*71.77*	*76.11*	*70.22*	*63.44*	*70.17*
	Noise pad (Pre) [20]	62.88	65.32	68.79	73.14	77.10	70.81	63.98	69.64
	Noise pad (Post) [44]	62.57	61.19	68.77	72.23	77.80	69.75	63.05	69.43
	Noise pad (Outer) [44]	65.51	62.94	68.85	72.07	76.17	70.07	64.87	70.97
	Noise pad Avg.	*63.65*	*63.15*	*68.80*	*72.48*	*77.02*	*70.21*	*63.96*	*70.01*
	Edge pad (Pre)	60.59	61.65	58.37	64.32	70.91	61.53	62.16	68.00
	Edge pad (Post)	60.04	63.00	60.10	64.57	71.72	61.05	59.65	66.81
	Edge pad (Outer)	64.46	58.22	58.75	61.78	69.93	60.92	59.50	67.75
	Edge pad Avg.	*61.69*	*60.95*	*59.07*	*63.55*	*70.85*	*61.16*	*60.43*	*67.52*
	STRF pad [20]	55.07	60.93	62.49	65.92	71.37	70.92	69.73	69.13
	Random pad [20]	64.72	58.71	59.00	64.74	71.55	68.94	66.39	69.84
	Zoom pad [20]	64.32	58.91	63.62	65.02	71.10	63.57	69.17	66.82
	Avg.	*62.65*	*61.70*	*64.80*	*68.53*	*74.00*	*67.57*	*64.02*	*69.15*
	Truncation								
	Truncate (Pre) [45]	47.91	48.37	52.52	53.60	53.90	50.07	48.85	49.94
	Truncate (Post) [45]	43.65	45.77	48.16	50.32	52.14	46.37	47.67	48.11
	Truncate (Outer) [44]	53.12	51.47	55.41	55.48	57.95	52.70	54.23	55.80
	Avg.	*48.22*	*48.53*	*52.03*	*53.13*	*54.66*	*49.71*	*50.25*	*51.28*
	Resampling								
	Linear interpolate	63.83	59.34	63.71	65.95	70.45	64.50	68.88	65.92
	Frequency selection [3]	54.58	53.88	59.41	60.72	59.72	51.81	48.82	54.97
	Avg.	*59.20*	*56.61*	*61.56*	*63.33*	*65.08*	*58.15*	*58.55*	*60.44*
	Warping								
	Nearest guided warping- α [19]	58.96	55.57	59.03	61.47	62.49	60.39	58.47	61.82
	Nearest guided warping- $\alpha\beta$ [19]	66.67	61.95	65.42	67.14	68.45	63.77	63.33	66.39
	Avg.	*62.81*	*58.76*	*62.22*	*64.30*	*65.47*	*62.08*	*60.90*	*64.10*
	Spectral pooling **(p)** (Ours)	**67.08**	**67.65**	**72.22**	**74.90**	**78.74**	69.55	**70.98**	**71.23**
Pooling	***Pooling***								
	Adaptive max pooling	–	70.85	73.50	77.11	79.45	70.37	57.95	69.07
	Adaptive average pooling	–	63.64	71.67	74.48	78.66	70.33	65.67	69.29
	Spectral pooling **(m)** (Ours)	–	**75.50**	**77.10**	**78.44**	**80.10**	**73.18**	**74.47**	**71.80**
	Avg.	–	*69.99*	*74.09*	*76.67*	*79.40*	*71.29*	*66.03*	*70.05*

to define T, i.e., it sets T as the length of the α-percentile shortest time series in the training set. In this case, each time series X^b that has less than T frames (i.e. $T^b < T$), its time domain representation X^b_t is simply equaling the first T frames of its padded time series X^b_{pad} (i.e., $X^b_t = \{x^b_{\text{pad}}(0), x^b_{\text{pad}}(1), \ldots, x^b_{\text{pad}}(T-1)\}$. Each time series X^b that has more than T frames is still processed based on previously introduced LSG and ALN steps. While this strategy would introduce distortion for time series whose length is less than T, it reduces information loss for longer ones, i.e., α decides the trade-off between distortion/information loss during training (evaluated in Section 5.2.2).

5. Experiment

In this section, we first present the performances of the 22 benchmarked length normalization methods achieved for 14 widely-used datasets based on 8 benchmarked network architectures in Section 5.1. We then demonstrate the effectiveness and ablation analysis of our proposed spectral pooling (SP) method in Section 5.2.

5.1. Benchmarking results

Table 3 reports the results achieved by all benchmarked length normalization strategies, based on which we provide the following

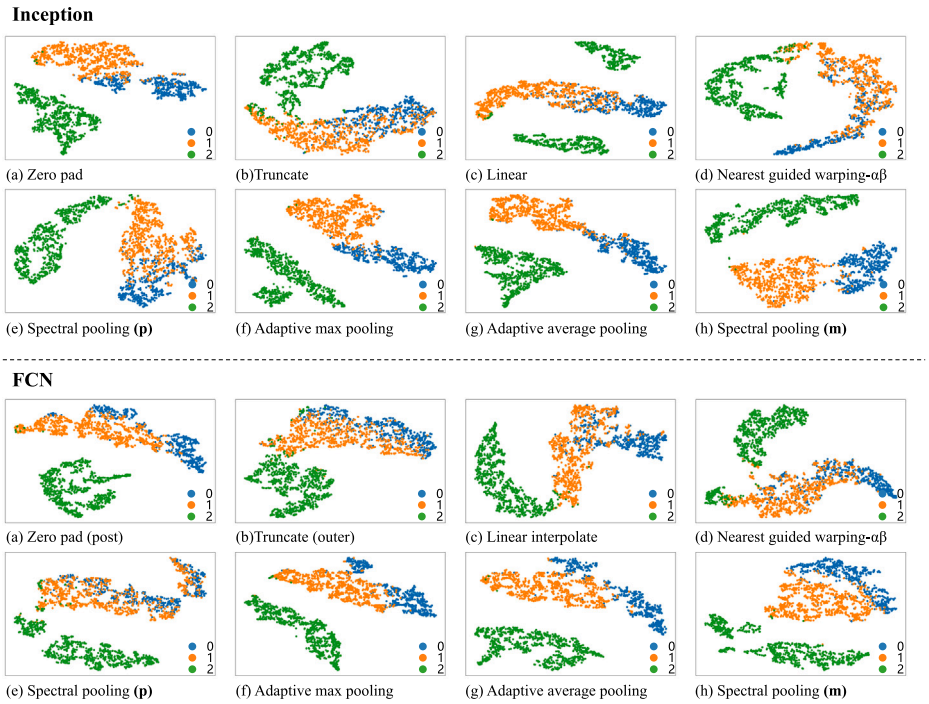

Fig. 7. t-SNE visualizations of features output from different length normalization strategies on the variable-length AsphaltPavementType dataset, where two deep learning models are individually employed. Top row: Inception-based models. Bottom row: FCN-based models. Each point represents a sample in the 2D projected feature space, with colors denoting class labels.

discussions. Based on the discussion below, we concluded that end-to-end pooling-based length normalization strategies are superior to pre-processing-based strategies, as pooling-based length normalization strategies facilitate DL models to obtain a more aggregated feature distribution, where our proposed SP achieves state-of-the-art results across various model architectures and datasets. Meanwhile, choosing different padding locations according to model architectures can improve the performance of padding methods. It is also interesting to note that models performing well on equal-length series classification are unlikely to perform well on VTS classification.

Comparison between pre-processing and pooling-based length normalization strategies: As shown in Fig. 7, pooling-based length normalization strategies (integrated within the deep learning models) consistently yield clearer and more compact feature representations compared to various pre-processing-based strategies. The t-SNE visualizations clearly illustrate that features learned by end-to-end pooling strategies form tighter clusters within each class and exhibit clearer separation between classes, which indicates more effective preservation of task-specific information. In contrast, features learned via pre-processing methods (e.g., zero-padding or truncation) are generally less compact and show more overlap between classes, reflecting potential loss or distortion of discriminative information. This advantage arises because model training inherently and jointly optimizes these pooling operations in coordination with other model layers, i.e., end-to-end optimization.

Comparison among different pooling strategies: Our SP can effectively encode global information across all time points and consistently outperformed adaptive max/avg pooling that only summarize local time domain patterns, which cannot avoid the influence of the padded data points, while the average and max pooling beat each other depending on the model. In particular, our SP achieved significantly higher accuracy compared to max/avg pooling when integrating them into LSTM, FCN, and Informer models.

Comparison among different pre-processing strategies: Except the SP, the average performances of padding strategies are generally superior to the other three types of solutions. Among all pre-processing strategies, truncation-based methods achieved the worst average performance, as directly removing a large part of data points lead to substantial information loss, while resampling strategies either introduce non-existent time points or discard some low and middle frequency components. Although warping-based strategies can partially address the distortion problem and achieved better performances, they have a quadratic time complexity. Also, padding time series with either zeros or low-amplitude noise at their beginning or end, achieved superior results than other padding strategies, where noise padding frequently exhibits greater robustness with marginally improved results over the simple zero padding. In contrast, random and STRF pad performed less effectively, as distributing padded zeros into time series would distort their shape. Furthermore, zoom pad that duplicates neighboring values, outperformed the edge pad which relies on edge values of time series. Importantly, the optimal padding locations are different across various backbones. For example, MLP performed best with padding at both ends (denoted as 'Outer'), while LSTM is less affected by pre-padding due to its sequential processing. Similarly, the padding location has limited impacts on CNNs (e.g., FCN, ResNet, Inception and TimesNet) and Transformer's performances. Results also reveal that information described by the middle segments (Truncate (Outer)) in the VTS is more valuable than the information represented by the front/end (Truncate (Pre)/(Truncate (Post))). Consequently, separately truncating both ends of VTS is more effective.

Comparison of the benchmarked results achieved on datasets whose time series are mainly represented by low-frequency components and high-frequency components: Fig. 8 shows that the time series in 11 out of the 14 employed VTS datasets have clearly less information at high frequency components compared to low/middle frequency components, while the high frequency components represent more information for VTS in three pavement classification datasets. We compare the average accuracy results of eight benchmarked backbones achieved for each of these two types of datasets. As we can see from the left part of Table 4, the results of all benchmarked length normalization methods (except our SP) achieved for each sub-frequency type are similar to their corresponding results achieved on all 14 datasets, where the

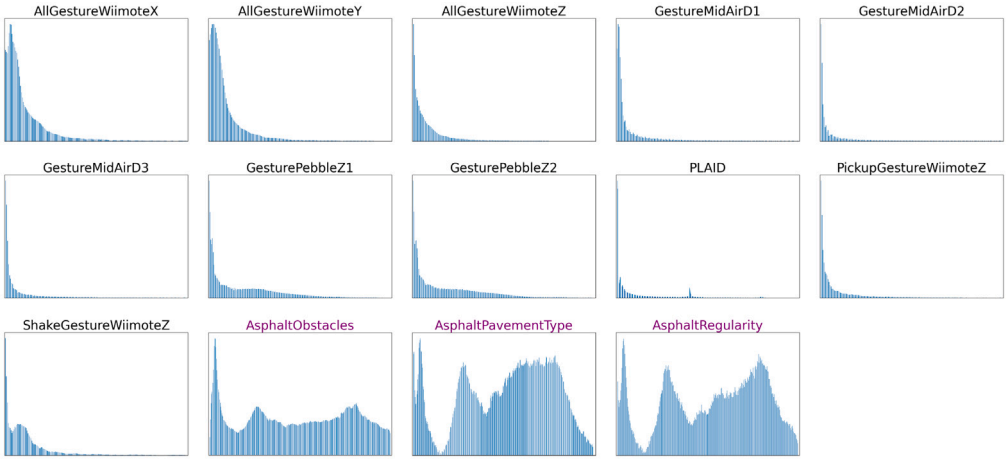

Fig. 8. The amplitude maps of 14 variable-length series datasets, only half of which are shown due to conjugate symmetry, with the names of the datasets where the high-frequency components have information labeled in purple.

Table 4
Average test accuracy (%) across eight backbone models on variable-length datasets categorized by frequency content and length variation: the left part summarizes results on 11 datasets **with few high-frequency information (Few-hf)** and 3 datasets with **more high-frequency information (More-hf)**, while the right part presents results on 4 datasets with different levels of length variation, where AsphaltRegularity (Asphalt-R) and AllGestureWiimoteY (Gesture-Y) exhibit larger variation in series length, and GesturePebbleZ1 (Gesture-Z1) and AsphaltObstacles (Asphalt-O) show smaller variation. SP (p) and SP (m) refer to applying spectral pooling as a pre-processing method and as a model-integrated pooling layer, respectively. Bold values indicate the best results, and underlined values indicate the second-best.

	Methods	Few-hf	More-hf	Gesture-Z1	Gesture-Y	Asphalt-O	Asphalt-R
Pre-processing	***Padding***						
	Zero pad (Pre) [45]	62.51	90.37	80.97	63.65	84.27	96.71
	Zero pad (Post) [45]	62.12	90.02	81.63	63.43	84.62	95.71
	Zero pad (Outer) [20]	63.22	90.51	82.53	63.81	84.55	96.91
	Zero pad (Mid) [20]	61.70	89.44	79.88	63.77	81.97	96.43
	Noise pad (Pre) [20]	63.13	90.34	81.56	63.56	83.96	96.86
	Noise pad (Post) [44]	62.07	90.22	81.87	63.48	83.75	97.06
	Noise Pad (Outer) [44]	63.04	**90.54**	81.85	64.07	**84.96**	96.98
	Edge pad (Pre)	56.31	89.58	64.33	64.67	83.51	96.58
	Edge pad (Post)	56.25	89.45	71.91	64.23	82.90	96.90
	Edge pad (Outer)	55.26	89.80	60.63	64.98	83.71	97.26
	STRF pad [20]	59.54	88.26	87.75	57.91	83.12	94.43
	Random pad [20]	58.79	90.03	85.85	61.18	84.55	97.32
	Zoom pad [20]	59.00	88.48	83.43	60.31	82.45	95.68
	Truncation						
	Truncate (Pre) [45]	42.32	81.19	69.86	30.28	69.78	92.47
	Truncate (Post) [45]	39.43	78.36	46.47	30.07	62.03	95.98
	Truncate (Outer) [44]	46.54	83.80	81.62	28.65	78.27	91.97
	Resampling						
	Linear interpolate	58.94	88.60	84.70	60.15	83.33	95.78
	Frequency selection[3]	48.30	81.87	52.27	52.65	64.87	98.02
	Warping						
	Nearest guided warping- α [19]	54.43	79.37	79.67	62.66	73.55	90.10
	Nearest guided warping- $\alpha\beta$ [19]	60.04	85.00	82.42	**68.23**	81.06	95.37
	Spectral pooling **(p)** (Ours)	**66.71**	89.28	**91.44**	65.36	84.81	**98.81**
Pooling	***Pooling***						
	Adaptive max pooling	65.27	92.87	85.10	64.14	88.57	98.32
	Adaptive average pooling	64.21	93.72	64.21	58.15	88.38	99.01
	Spectral pooling **(m)** (Ours)	**70.81**	**94.10**	**86.82**	**71.60**	**89.10**	**99.34**

two pre-processing methods, zero pad and noise pad, achieved decent performances, while the truncation-based methods are least effective. While the above benchmarked result are not sensitive to the dominant frequency components of the processed time series, our proposed SP strategy demonstrates superiority on datasets whose time series have clearly less information at high frequency components, i.e., both SP **(p)** for pre-processing and SP **(m)** for pooling achieved the highest average accuracy, with pre-processing-based SP **(p)** even outperforming the end-to-end adaptive max/average pooling. Importantly, our end-to-end SP **(m)** attains the highest accuracy on both types of datasets.

Comparison among datasets with different degrees of length variation: The right part of Table 4 shows that the average accuracy results achieved by eight backbones for different length normalization strategies on datasets with different length variations. The results demonstrate that almost all benchmarked length normalization strategies are robust to this factor. However, for truncation methods, truncating only the beginning or the end of the time series performs better than truncating both ends on datasets with large distributional differences. The frequency selection method is suitable for datasets spanning a wide range of lengths, e.g., it achieves sub-optimal

Table 5
Average test ACC (%) over 14 datasets achieved by different poolings equipped with (w) and without (w/o) our length tracking.

		LSTM	FCN	Resnet	Inception	Transformer	Informer	TimesNet
Spectral pooling **(m)**	w/o length tracking	73.53	74.57	77.35	78.37	70.74	63.07	69.28
	w/ length tracking	**75.50**	**77.10**	**78.44**	**80.10**	**73.18**	**74.47**	**71.80**
Adaptive max pooling	w/o length tracking	**70.85**	73.50	**77.11**	**79.45**	70.37	57.95	69.07
	w/ length tracking	70.80	**75.51**	77.00	78.85	**72.81**	**70.19**	**70.89**
Adaptive average pooling	w/o length tracking	63.64	71.67	74.48	78.66	70.33	65.67	69.29
	w/ length tracking	**68.57**	**73.50**	**75.30**	**79.65**	**71.63**	**73.92**	**70.24**

performance among pre-processing methods on Asphalt-R, but performs poorly on other datasets with narrower length ranges. SP **(p)** is consistently top/second-performing, even exceeding based-pooling on the GesturePebbleZ1 dataset. SP **(m)** is consistently top-performing compared to the pooling-based methods across all four datasets.

Performances of existing TSC DL models on VTS classification: It can be observed that the Inception achieved the best performances on VTS classification when employing different benchmarked pre-processing and pooling-based length normalization strategies. However, the state-of-the-art TimesNet for equal-length time series classification only achieved middle-ranking VTS classification results among all benchmarked competitors. We explain this as the TimesNet attempts to treat each time series as a periodical signal, where padding data points would strongly affect its performance in defining the period division. In contrast, the attention mechanism of the Transformer effectively manages interference from the padded data points, and thus achieved promising results. While the ProbSparse self-attention in Informer degrades the attention to reduce time complexity, it fails to demonstrate this advantage.

5.2. Results achieve by our spectral pooling

5.2.1. Comparison with benchmarked strategies

The proposed SP layer can be used either as a pooling layer inside DL models or attached at the top of them as a pre-processing layer. It is clear that our SP-based pre-processing strategies outperformed all other pre-processing strategies on 7/8 backbones, which achieved the fourth best performance on the remaining backbone. Meanwhile, our SP-based pooling is also superior to other pooling strategies across all backbones, and it performs well on datasets with different frequency types and varying degrees of length variation, suggesting that **our SP layer is a robust and state-of-the-art length normalization strategy for variable time series classification (TSC) tasks**. This can be explained by the facts: (i) when employing our SP as a pre-processing layer, it causes much less information loss/distortion compared to existing pre-processing truncation/wrapping/resampling strategies; (ii) when SP is employed as a pooling layer, it retains global information and thus effectively captures the trend of the input time series. This is different from adaptive pooling layers focusing on modeling local information; and (iii) our length tracking ensures the SP to process time series independent of the padded data points, which avoids the introduction of non-existent information. Table 5 shows that this strategy significantly enhanced performances.

To further evaluate the robustness and superiority of our SP, we conducted a comprehensive statistical analysis using the Bayesian Wilcoxon signed-rank test [64]. As shown in Figs. 14 and 15, our SP consistently outperformed most baseline strategies with high confidence. When using our SP as a pre-processing method (SP **(p)**), it achieved near-certain superiority (i.e., posterior probability close to 100%) over truncation, resampling, and warping methods across all backbones (except for Nearest Guided Warping-$\alpha\beta$ on MLP). Among padding-based strategies, only the noise pad (pre) in LSTM slightly surpasses SP **(p)**. Additionally, due to the inherent robustness of the Transformer's self-attention mechanism, half of the evaluated padding methods exhibit marginally higher probability than SP **(p)** in that model (in 7 out of 13 padding variants). When applying our SP as a pooling layer (SP **(m)**), it achieved clearly advantages over both adaptive average pooling and adaptive max pooling with high confidence—exceeding 70% across all backbones and reaching over 90% in most cases. The only exception is on Inception, where SP **(m)** performs comparably to adaptive max pooling, with a 53% probability of practical equivalence.

5.2.2. Ablation studies

This section first analyzes the impact of length tracking and adaptive length normalization (ALN) of the proposed SP method on VTS classification, and then demonstrates how the performance varies with different values of the hyperparameter α and with the SP layer placed at various positions within the model.

Influence of the length tracking: Table 5 demonstrates that the proposed length tracking clearly improved the average performances on most cases (18/21). Particularly, it largely impacted on our SP layer. This is because the length tracking prevent all frequency components in DFT from distorted by the padded data points. Besides, excluding padded data points in generating outputs also lead more benefits to average pooling than max pooling, as padded data points would certainly impact the average value but may not change the local max value. When without utilizing our length tracking, the SP still consistently outperformed the other two pooling strategies.

Influence of the ALN: Our ALN consists of two main stages: (1) truncating high frequency components from the variable-length spectral representations; and (2) inversely converting the truncated spectral representations back to time domain representations. Table 6 shows that the first stage of truncating the signal in frequency domain and using the truncated spectral representation for classification, significantly improves the average performance compared to direct truncating in the time domain, except for Transformer and Informer. The second stage which converts the truncated spectral representation back to the time domain results in further performance gain, suggesting that our strategy can effectively address the frequency misalignment issue.

Influence of different α: Table 7 suggests that DL models' performances are also influenced by α values, where the best result of some models are achieved when $\alpha \neq 0$. This suggests that retaining more information at the cost of certain degrees of distortion could improve the SP's performance. However, if a large part of VTS are distorted ($\alpha \geq 0.4$), the additionally retained information cannot compensate their negative impacts, i.e., the best results are achieved when $\alpha = 0$ or $\alpha = 0.2$ for most cases.

Influence of different α from different task: We further investigated the selection of α in terms of different classification tasks. For **gesture recognition datasets**, Table 8 suggests that most models (except Transformer) achieved their best performances when $\alpha = 0.2$. The amplitude maps (Fig. 8) indicate that the energy distribution for gesture datasets is predominantly concentrated in the low-frequency region, with relatively little information represented by middle and high-frequency bands. Therefore, during the length normalization, a smaller α value is more appropriate, as truncating middle and high-frequency components would not only retain almost all crucial information but also remove irrelevant noises. For **device identification dataset PLAID**,

Table 6
Average test ACC (%) over 14 datasets achieved with different ALN settings. Trunc-t. represents directly truncating VTS in time domain; Trunc-f. represents truncating high frequencies (Eq. (4)); and Conv. represents converting to time domain (Eq. (5)).

Trunc-t.	Trunc-f.	Conv.	MLP		LSTM		FCN		ResNet		Inception		Transformer		Informer		TimesNet	
			p	m	p	m	p	m	p	m	p	m	p	m	p	m	p	m
✓			43.65	–	59.02	46.42	48.16	48.91	50.32	50.82	73.57	70.92	64.98	72.44	59.68	71.82	58.12	54.97
	✓		61.09	–	56.06	75.30	65.35	77.00	67.25	78.16	67.01	77.28	59.68	71.87	52.68	70.08	62.13	65.20
	✓	✓	**67.08**	–	**67.65**	**75.50**	**72.22**	**77.10**	**74.90**	**78.44**	**78.74**	**80.10**	**69.55**	**73.18**	**70.98**	**74.47**	**71.23**	**71.80**

Table 7
Average test ACC (%) achieved for different trade-off factor α value with our SP as pre-processing **(p)** and pooling layer **(m)**.

	MLP		LSTM		FCN		ResNet		Inception		Transformer		Informer		TimesNet	
	p	m	p	m	p	m	p	m	p	m	p	m	p	m	p	m
$\alpha = 0$	**67.08**	–	67.57	**75.50**	**72.22**	**77.10**	**74.90**	**78.44**	76.07	78.83	63.97	72.87	69.15	74.17	69.26	**71.80**
$\alpha = 0.2$	66.98	–	**67.65**	71.89	71.25	75.52	74.00	77.40	77.05	**80.10**	66.83	**73.18**	**70.98**	**74.47**	**71.23**	71.07
$\alpha = 0.4$	66.88	–	65.87	70.84	71.19	73.67	72.77	76.00	**78.74**	78.51	69.55	73.12	70.65	73.62	69.89	67.57
$\alpha = 0.6$	64.39	–	65.21	69.60	71.52	72.98	73.07	75.44	78.37	77.36	69.19	73.35	68.00	73.66	69.62	67.00
$\alpha = 0.8$	64.15	–	65.59	66.86	70.53	72.99	74.02	74.87	77.77	77.27	69.87	73.55	67.94	73.25	69.41	65.32
$\alpha = 1$	62.50	–	62.74	–	68.80		70.47	–	77.57	–	**70.28**	–	61.93	–	70.53	–

Table 8
Average test ACC (%) achieved for different trade-off factor α value with our SP as pre-processing **(p)** and pooling layer **(m)** on gesture datasets.

	MLP		LSTM		FCN		ResNet		Inception		Transformer		Informer		TimesNet	
	p	m	p	m	p	m	p	m	p	m	p	m	p	m	p	m
$\alpha = 0$	63.27	–	61.50	71.28	67.38	72.28	70.38	74.24	70.58	73.56	56.32	67.06	63.98	69.67	64.20	**66.23**
$\alpha = 0.2$	**66.98**	–	**67.65**	**71.90**	**71.25**	**75.52**	**74.00**	**77.40**	**77.05**	**80.10**	59.74	67.45	**66.44**	**69.88**	**65.05**	65.21
$\alpha = 0.4$	62.50	–	60.20	66.31	65.42	68.35	66.85	70.75	73.45	72.74	63.32	67.39	65.85	68.71	63.63	60.28
$\alpha = 0.6$	59.59	–	58.99	64.44	65.59	67.50	67.25	70.12	73.35	71.26	62.53	67.72	62.08	68.75	63.26	59.99
$\alpha = 0.8$	59.37	–	59.60	61.14	64.26	67.72	68.27	68.83	72.37	71.34	63.24	**68.03**	61.72	68.38	63.06	57.26
$\alpha = 1$	58.57	–	56.08	–	62.96	–	66.01	–	71.85	–	**64.00**	–	55.10	–	63.58	–

Table 9
Average test ACC (%) achieved for different trade-off factor α value with our SP as pre-processing **(p)** and pooling layer **(m)** on device datasets.

	MLP		LSTM		FCN		ResNet		Inception		Transformer		Informer		TimesNet	
	p	m	p	m	p	m	p	m	p	m	p	m	p	m	p	m
$\alpha = 0$	65.50	–	**59.70**	**63.10**	65.30	**69.20**	73.10	70.20	82.60	78.30	69.00	72.90	64.20	64.20	62.90	64.40
$\alpha = 0.2$	66.20	–	54.30	50.00	61.80	66.10	72.20	73.10	83.90	**85.80**	71.80	72.80	**65.70**	**65.70**	**74.80**	66.60
$\alpha = 0.4$	**66.80**	–	46.50	51.20	**65.70**	62.50	**73.50**	74.40	**87.80**	84.90	**72.40**	73.70	65.10	65.50	72.90	67.50
$\alpha = 0.6$	66.10	–	48.00	52.80	64.60	63.50	72.40	72.00	80.20	84.30	**72.40**	**75.00**	57.30	65.10	68.70	63.80
$\alpha = 0.8$	65.90	–	48.40	52.10	64.80	61.00	72.80	**74.80**	80.80	84.10	71.80	73.50	57.70	64.80	68.70	**70.00**
$\alpha = 1$	65.10	–	47.60	–	52.50	–	45.90	–	82.80	–	69.40	–	50.40	–	72.00	–

where the energy of time series in this dataset is distributed relatively even across their low, middle and high frequency bands (illustrated in Fig. 8), the optimal α values for different backbones generally range from 0 to 0.4 (demonstrated in Table 9). These results suggest that the optimal choice of α depend on both the model architecture and the specific location where our spectral pooling is applied within the model. For **pavement dataset**, which contains richer high-frequency information, Table 10 shows that the optimal α values are consistently greater than 0.5 when our spectral pooling is applied as a pre-processing step. This suggests that, for datasets with substantial high-frequency content, padding is preferable to truncation, as it effectively preserves critical high-frequency information in these time-series. However, when our spectral pooling is integrated as a pooling layer within the model, the optimal α values tend toward zero but the results achieved by α values ranging from 0 to 0.4 are stable. This occurs because the models have already captured the essential features predominantly residing in the lower frequency range at early feature extraction layers. Further detailed analysis of this phenomenon is provided in the subsequent sections.

Impacts of the SP layer's location in the model: Table 11 shows that integrating our SP layer into models is more beneficial than employing it as a pre-processing step, which again suggests the effectiveness of end-to-end normalizing over pre-normalizing. This can be explained by the fact that deep learning models often prioritize learning lower frequency components first, regardless of whether higher frequency components or lower frequency components have larger amplitudes in the original time series (as illustrated in Fig. 9), a phenomenon known as spectral bias [87]. This explains why pooling-based SP outperformed pre-processing-based SP, i.e., since the crucial cues of the extracted feature already concentrated in low and middle frequency components compared to the original series which contain more high frequency contents, placing SP in the model means less information loss when truncating high frequency components. Fig. 10 further employs t-SNE to visualize the intermediate feature representations extracted from each block, both with and without the use of our SP. It shows a clear transition from a compact feature distribution (Fig. 10(b)) to a scattered distribution (Fig. 10(d)), suggesting that deeper layers without SP may generate features with larger intra-class distances. In contrast, applying SP at earlier stages helps refine feature representations, enhancing both intra-class compactness and inter-class separability of the extracted features. Moreover, Fig. 10(e), (f), and (g) highlight that the optimal position for placing SP varies depends on the model architecture. Specifically, SP achieved optimal performances when placing it immediately after the block that initially generates the most compact intra-class features (i.e., after the first block for FCN and after the second block for Informer).

6. Conclusion

This paper presents the first comprehensive benchmark for VTS classification, evaluating 22 existing length normalization strategies

Table 10
Average test ACC (%) achieved for different trade-off factor α value with our SP as pre-processing **(p)** and pooling layer **(m)** on pavement datasets.

	MLP		LSTM		FCN		ResNet		Inception		Transformer		Informer		TimesNet	
	p	m	p	m	p	m	p	m	p	m	p	m	p	m	p	m
$\alpha = 0$	80.33	–	90.43	**93.66**	90.66	**95.80**	90.60	**95.23**	92.20	**96.60**	87.83	92.26	88.06	92.50	88.26	**92.93**
$\alpha = 0.2$	**82.10**	–	91.33	92.56	92.00	95.00	91.60	94.13	93.06	96.06	88.83	**92.43**	87.90	92.70	90.66	92.10
$\alpha = 0.4$	81.53	–	91.20	92.50	92.26	95.13	92.30	94.06	93.36	95.63	89.40	92.06	88.50	92.70	89.76	91.90
$\alpha = 0.6$	79.83	–	**91.70**	92.40	93.63	94.43	92.70	94.33	94.53	95.40	90.30	91.60	91.33	**92.90**	91.16	91.46
$\alpha = 0.8$	79.50	–	91.30	90.86	93.36	94.56	**93.60**	95.03	94.80	94.76	91.36	92.00	**92.10**	92.33	90.83	90.66
$\alpha = 1$	74.73	–	90.00	–	**93.70**	–	93.53	–	**94.90**	–	**91.53**	–	88.56	–	**93.23**	–

Table 11
Average test ACC (%) achieved by placing SP at different locations (after different layers) in DL models.

	LSTM	FCN	Resnet	Inception	Transformer	Informer	TimesNet
Pre	67.65	72.22	74.90	78.74	69.55	70.98	71.23
1	**75.50**	**77.10**	**78.44**	**80.10**	72.20	73.30	**71.80**
2	71.85	73.96	77.67	78.22	**73.18**	**74.47**	71.29
3	–	68.38	71.82	–	72.12	72.36	67.50

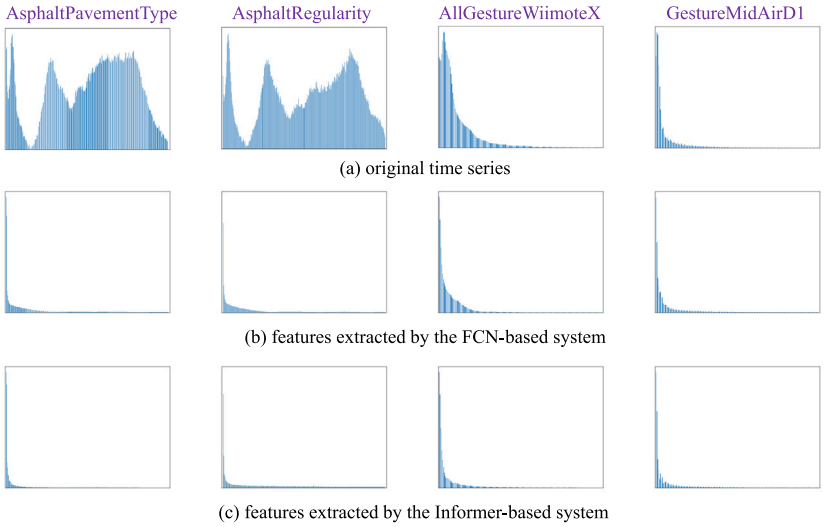

Fig. 9. Comparison between the **Fourier domain amplitude maps** representing: (a) original time series; (b) features extracted by the FCN-based system; and (c) features extracted by the Informer-based system.

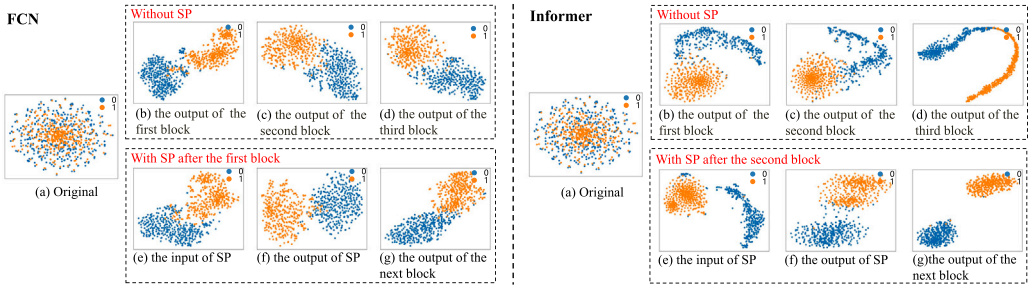

Fig. 10. t-SNE visualizations of intermediate feature representations learned from FCN (left) and Informer (right) models, illustrating the effect of spectral pooling (SP) on improving features' intra-class compactness and inter-class separability. In the FCN, each block contains a convolutional layer followed by a GELU activation. In the Informer, each block consists of 8-headed ProbSparse self-attention and layer normalization.

across 14 VTS datasets using 8 representative DL models. In addition, we propose a novel and effective SP-based normalization method, which can be flexibly used either as a pre-processing step or integrated as a pooling layer within end-to-end DL architectures. The experimental results lead to several key findings: (i) End-to-end pooling-based strategies generally outperform pre-processing-based ones, facilitating the DL model to obtain a more aggregated feature distribution; (ii) Zero padding and noise padding are relatively robust pre-processing strategies, though the optimal padding position is model-dependent; (iii) Truncation methods yield the poorest results due to significant information loss, with symmetric truncation typically performing better than one-sided truncation; (iv) Most normalization methods show consistent performance across different dataset types and degrees of length variation; (v) DL models tend to focus on low-frequency components for classification, even when the original data contains rich high-frequency content; (vi) Models performing well on equal-length series classification are unlikely to perform well on VTS classification; and (vii) our proposed SP effectively reduces information loss and distortion during normalization, achieving state-of-the-art performance across multiple models and datasets.

Fig. 11. The amplitude maps of 40 equal-length series datasets in UCR, only half of which are shown due to conjugate symmetry, with the names of the datasets where the high-frequency components have information labeled in purple.

The proposed spectral pooling (SP) method have several advantages over existing strategies in processing VTS; (i) It can be flexibly used as either a pre-processing operation or a pooling layer integrated into standard deep learning models; (ii) **as a pre-processing step (SP (p))**, it fundamentally differs from widely used truncation-based methods that directly discard data points in the time domain. Instead, our SP performs truncation in the frequency domain, preserving main information by maintaining dominant spectral components. On datasets where information is concentrated in low-frequency components, our SP **(p)** even outperformed traditional adaptive pooling approaches; (iii) **When used as a pooling layer (SP(m))** within deep learning models, SP effectively captures global temporal patterns, surpassing conventional adaptive pooling methods that focus on local information. Moreover, the selective truncation of high-frequency components aligns well with the spectral bias observed in deep networks, thereby mitigating information loss during length normalization; and (iv) our length-tracking mechanism ensures accurate processing of valid (unpadded) regions, thereby avoiding misleading information introduced through padding.

Despite these advantages, our SP also has several limitations. When used as a pre-processing strategy (SP**(p)**), its performance may decline on datasets dominated by high-frequency information due to the truncation of potentially information. In such cases, we recommend using SP**(m)**, which incorporates spectral pooling into the model for end-to-end optimization to align well with the spectral bias. Moreover, since SP inherently suppresses high-frequency components, it may be fundamentally less suitable for tasks that rely heavily on such information—such as short-term time-series forecasting or anomaly detection.

In terms of scope, the current benchmark focuses exclusively on univariate raw time-series data, aiming to establish a clear and standardized foundation for variable-length time-series classification. However, our approach is naturally extensible to multivariate and multi-modal time series. For multivariate time series, our SP can be applied independently to each variable or jointly across variables to capture

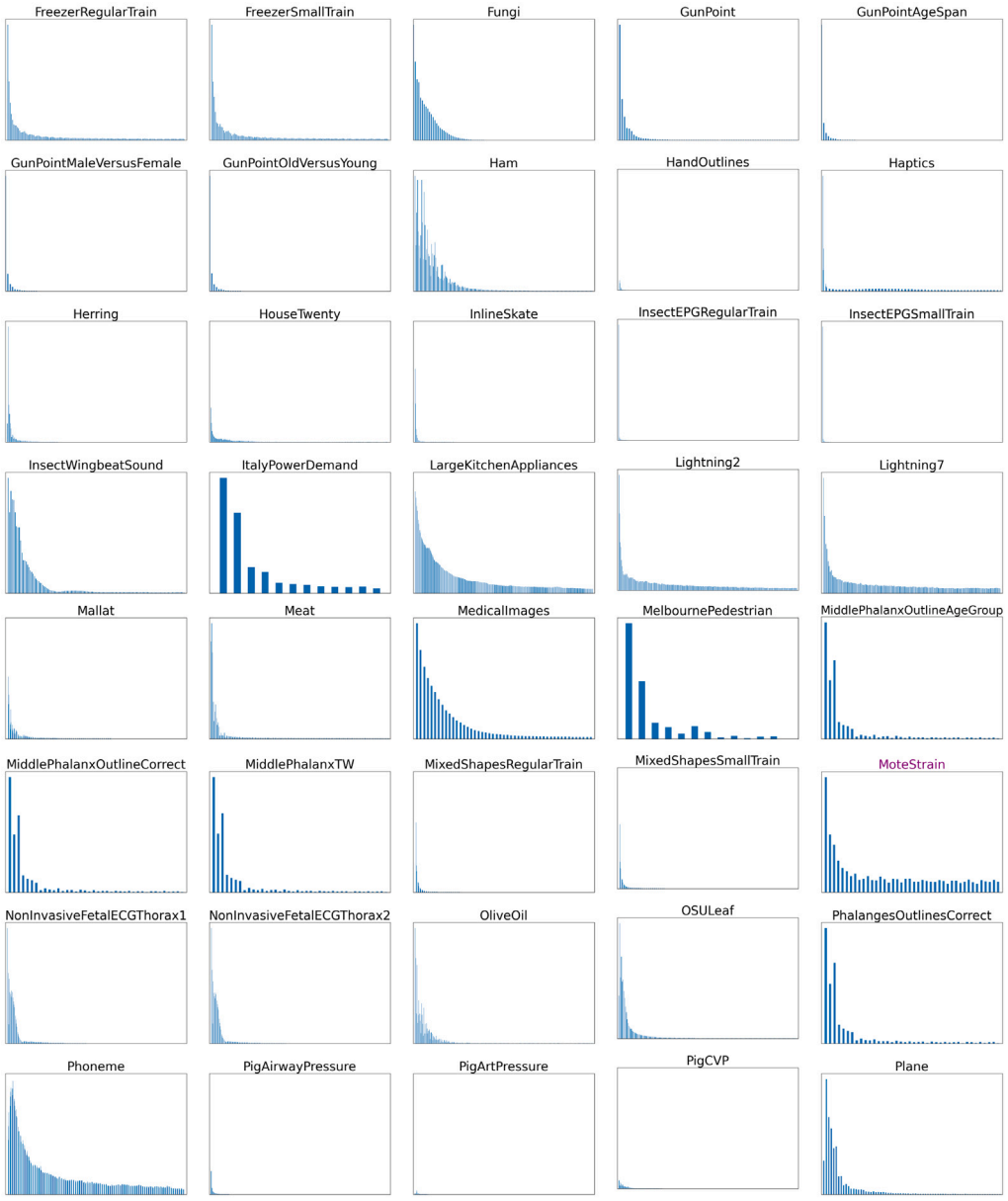

Fig. 12. The amplitude maps of 40 equal-length series datasets in UCR, only half of which are shown due to conjugate symmetry, with the names of the datasets where the high-frequency components have information labeled in purple.

inter-dimensional dependencies. For multi-modal data, our SP can be used to temporally normalize variable-length time series (e.g., different numbers of audio frames and visual frames) from different modalities to the same length, enabling straightforward fusion. In addition, our framework supports benchmarking both early fusion (e.g., integrating features from different modalities at the input level) and late fusion (e.g., aggregating predictions from modality-specific branches) strategies for multi-modal time series, which we plan to explore in future work.

CRediT authorship contribution statement

Shiling Wu: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Formal analysis, Data curation, Conceptualization. **Siyang Song:** Writing – review & editing, Formal analysis, Conceptualization. **Songhe Deng:** Visualization, Data curation. **Weicheng Xie:** Writing – review & editing. **Linlin Shen:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 82261138629 and 12326610; Guangdong Basic and Applied Basic Research Foundation, China under Grant 2023A1515010688; Guangdong Provincial Key Laboratory, China under Grant 2023B1212060076.

Data availability

I have shared the link to my code at the manuscript.

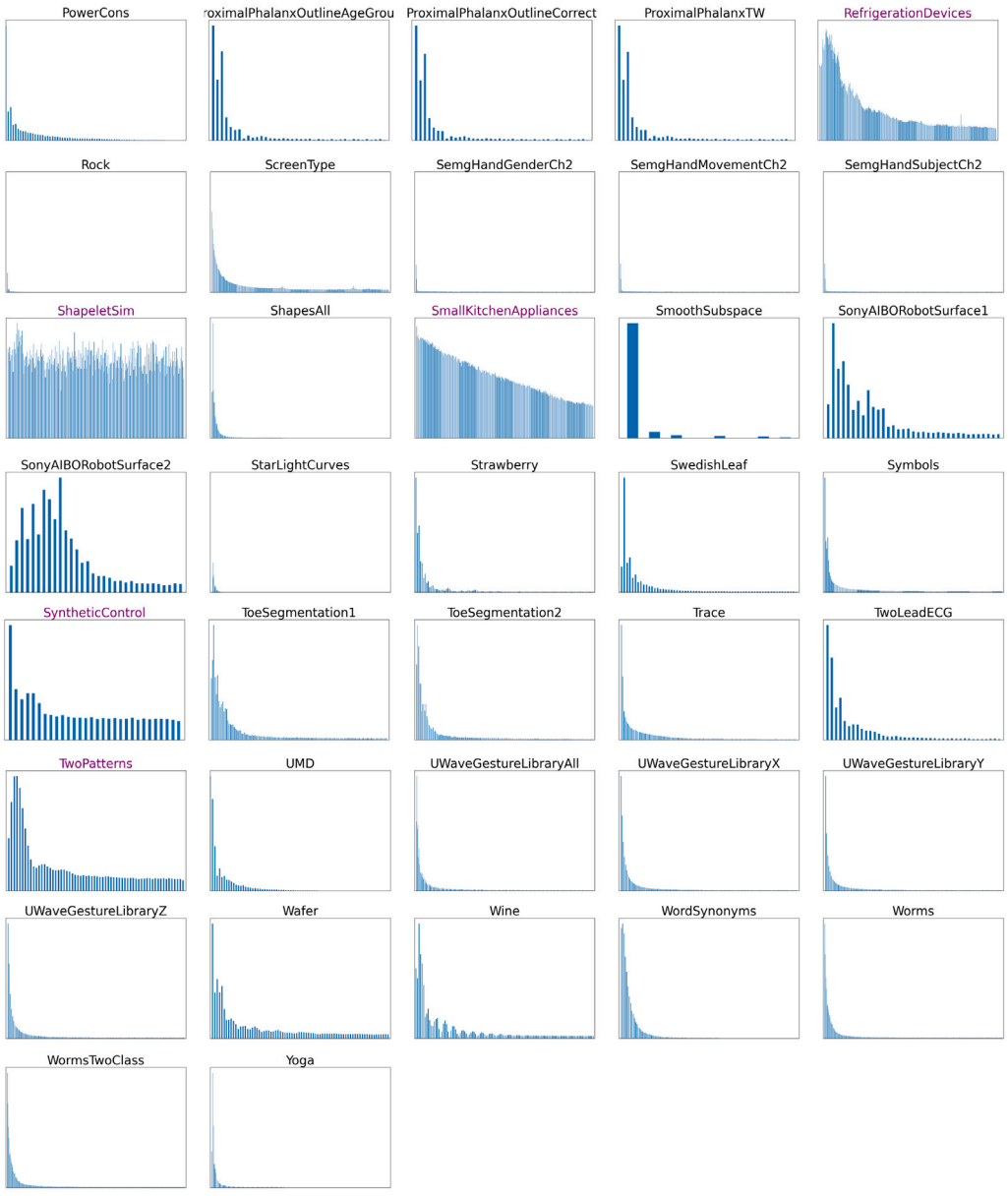

Fig. 13. The amplitude maps of 37 equal-length series datasets in UCR, only half of which are shown due to conjugate symmetry, with the names of the datasets where the high-frequency components have information labeled in purple.

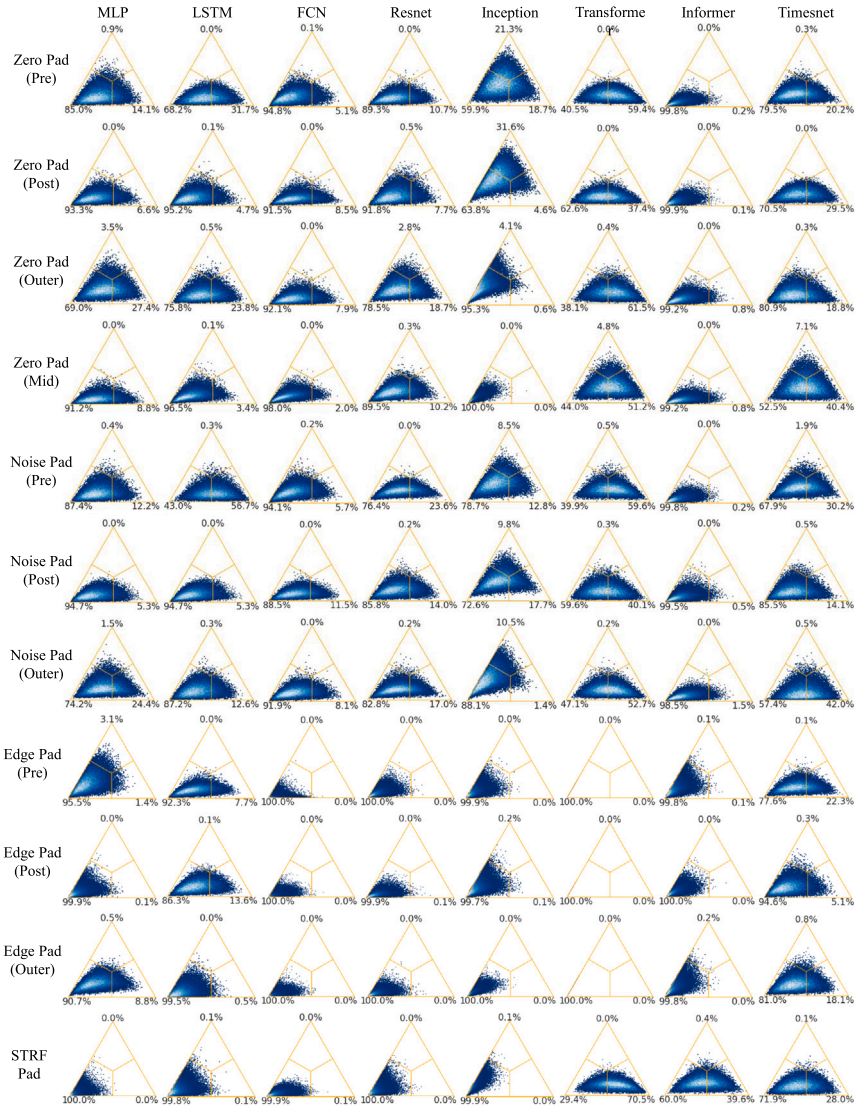

Fig. 14. Statistical analysis (based on Bayesian Wilcoxon signed-rank test) between our spectral pooling (used as a pre-processing step (SP **(p)**)) and various padding-based length normalization methods across 14 VTS classification datasets and eight backbone models (columns). Each triangle illustrates the probability that SP **(p)** outperforms the compared method (left corner), the compared method outperforms SP **(p)** (right corner), or their performance is practically equivalent (top corner, accuracy difference < 1%). Higher density toward the left corner indicates larger advantages achieved by our SP **(p)**.

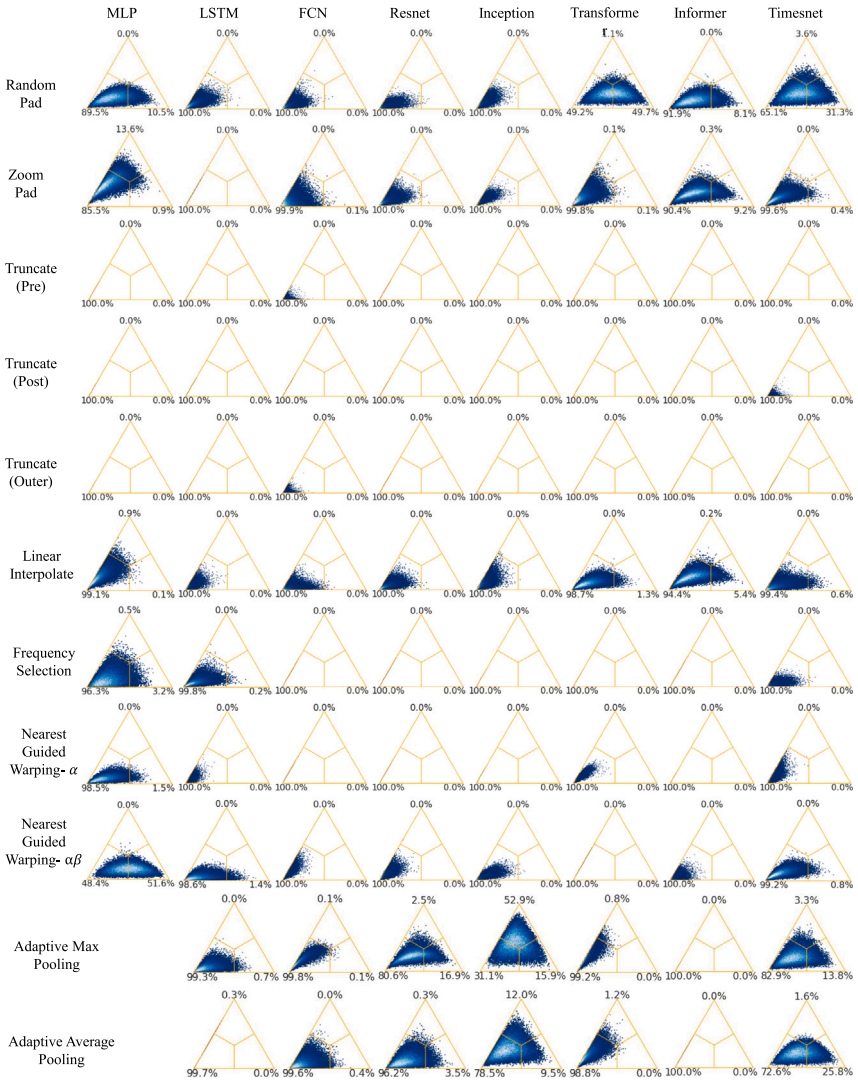

Fig. 15. Bayesian statistical comparisons of spectral pooling (SP) as a pre-processing strategy (SP **(p)**) and as a model-integrated pooling layer (SP **(m)**) against a broader set of benchmarks. SP **(p)** is compared with random padding, zoom padding, truncation, resampling, and warping strategies; SP **(m)** is compared with adaptive average and max pooling. Each triangle illustrates the probability that SP outperforms the compared method (left corner), the compared method outperforms SP (right corner), or their performance is practically equivalent (top corner, accuracy difference < 1%). Higher density toward the left corner indicates larger advantages achieved by our SP.

References

[1] J. Wu, K. Xu, X. Chen, S. Li, J. Zhao, Price graphs: Utilizing the structural information of financial time series for stock prediction, Inform. Sci. 588 (2022) 405–424.

[2] H.F. Nweke, Y.W. Teh, M.A. Al-Garadi, U.R. Alo, Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges, Expert Syst. Appl. 105 (2018) 233–261.

[3] S. Song, S. Jaiswal, L. Shen, M. Valstar, Spectral representation of behaviour primitives for depression analysis, IEEE Trans. Affect. Comput. 13 (2) (2020) 829–844.

[4] L. Yin, C. Ma, Interpretable incremental voltage-current representation attention convolution neural network for non-intrusive load monitoring, IEEE Trans. Ind. Inform. (2023).

[5] P.B. Weerakody, K.W. Wong, G. Wang, W. Ela, A review of irregular time series data handling with gated recurrent neural networks, Neurocomputing 441 (2021) 161–178.

[6] C. Sun, S. Hong, M. Song, H. Li, A review of deep learning methods for irregularly sampled medical time series data, 2020, arXiv preprint arXiv:2010.12493.

[7] D.M. Kreindler, C.J. Lumsden, The effects of the irregular sample and missing data in time series analysis, in: Nonlinear Dynamical Systems Analysis for the Behavioral Sciences using Real Data, CRC Press, 2016, pp. 149–172.

[8] Y.-S. Jeong, M.K. Jeong, O.A. Omitaomu, Weighted dynamic time warping for time series classification, Pattern Recognit. 44 (9) (2011) 2231–2240.

[9] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E. Keogh, Querying and mining of time series data: experimental comparison of representations and distance measures, Proc. VLDB Endow. 1 (2) (2008) 1542–1552.

[10] Z. Geler, V. Kurbalija, M. Ivanović, M. Radovanović, Weighted kNN and constrained elastic distances for time-series classification, Expert Syst. Appl. 162 (2020) 113829.

[11] A. Jalalian, S.K. Chalup, GDTW-P-SVMs: Variable-length time series analysis using support vector machines, Neurocomputing 99 (2013) 270–282.

[12] A. Mezari, I. Maglogiannis, Gesture recognition using symbolic aggregate approximation and dynamic time warping on motion data, in: Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare, 2017, pp. 342–347.

[13] S. Song, L. Shen, M. Valstar, Human behaviour-based automatic depression analysis using hand-crafted statistics and deep learned spectral features, in: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition, FG 2018, IEEE, 2018, pp. 158–165.

[14] N. Dehak, P.J. Kenny, R. Dehak, P. Dumouchel, P. Ouellet, Front-end factor analysis for speaker verification, IEEE Trans. Audio Speech Lang. Process. 19 (4) (2010) 788–798.

[15] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: 2017 International Joint Conference on Neural Networks, IJCNN, IEEE, 2017, pp. 1578–1585.

[16] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Discov. 33 (4) (2019) 917–963.

[17] J. Faouzi, Time series classification: A review of algorithms and implementations, Mach. Learn. (Emerg. Trends Appl.) (2022).

[18] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, L. Sun, Transformers in time series: A survey, 2022, arXiv preprint arXiv:2202.07125.

[19] B.K. Iwana, On mini-batch training with varying length time series, in: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2022, pp. 4483–4487.

[20] A. Lopez-del Rio, M. Martin, A. Perera-Lluna, R. Saidi, Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction, Sci. Rep. 10 (1) (2020) 1–14.

[21] E. Rahimian, S. Zabihi, S.F. Atashzar, A. Asif, A. Mohammadi, Xceptiontime: A novel deep architecture based on depthwise separable convolutions for hand gesture classification, 2019, arXiv preprint arXiv:1911.03803.

[22] M. Malekzadeh, R. Clegg, A. Cavallaro, H. Haddadi, Dana: Dimension-adaptive neural architecture for multivariate sensor data, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 5 (3) (2021) 1–27.

[23] Z. Yu, X. Xu, X. Chen, D. Yang, Temporal pyramid pooling convolutional neural network for cover song identification, in: IJCAI, 2019, pp. 4846–4852.

[24] A. Sawada, T. Miyagawa, A. Ebihara, S. Yachida, T. Hosoi, Convolutional neural networks for time-dependent classification of variable-length time series, in: 2022 International Joint Conference on Neural Networks, IJCNN, IEEE, 2022, pp. 1–8.

[25] H.A. Dau, A. Bagnall, K. Kamgar, C.-C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, E. Keogh, The UCR time series archive, IEEE/CAA J. Autom. Sin. 6 (6) (2019) 1293–1305.

[26] V.M. Souza, Asphalt pavement classification using smartphone accelerometer and complexity invariant distance, Eng. Appl. Artif. Intell. 74 (2018) 198–211.

[27] C.-C. Kao, M. Sun, W. Wang, C. Wang, A comparison of pooling methods on LSTM models for rare acoustic event classification, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2020, pp. 316–320.

[28] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D.F. Schmidt, J. Weber, G.I. Webb, L. Idoumghar, P.-A. Muller, F. Petitjean, Inceptiontime: Finding alexnet for time series classification, Data Min. Knowl. Discov. 34 (6) (2020) 1936–1962.

[29] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, M. Long, TimesNet: Temporal 2D-variation modeling for general time series analysis, 2022, arXiv preprint arXiv:2210.02186.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).

[31] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 11106–11115, 12.

[32] H.I. Fawaz, Deep learning for time series classification, 2020, arXiv preprint arXiv:2010.00567.

[33] A. Shifaz, C. Pelletier, F. Petitjean, G.I. Webb, Elastic similarity and distance measures for multivariate time series, Knowl. Inf. Syst. 65 (6) (2023) 2665–2698.

[34] J. Zhao, L. Itti, Shapedtw: Shape dynamic time warping, Pattern Recognit. 74 (2018) 171–184.

[35] Y. Liu, Y.-A. Zhang, M. Zeng, J. Zhao, A novel distance measure based on dynamic time warping to improve time series classification, Inform. Sci. 656 (2024) 119921.

[36] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 947–956.

[37] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with COTE: the collective of transformation-based ensembles, IEEE Trans. Knowl. Data Eng. 27 (9) (2015) 2522–2535.

[38] C. Avcı, M. Budak, N. Yağmur, F. Balçık, Comparison between random forest and support vector machine algorithms for LULC classification, Int. J. Eng. Geosci. 8 (1) (2023) 1–10.

[39] A. Shifaz, C. Pelletier, F. Petitjean, G.I. Webb, TS-CHIEF: a scalable and accurate forest algorithm for time series classification, Data Min. Knowl. Discov. 34 (3) (2020) 742–775.

[40] Y. Narayan, Comparative analysis of SVM and naive Bayes classifier for the SEMG signal classification, Mater. Today: Proc. 37 (2021) 3241–3245.

[41] F. Karim, S. Majumdar, H. Darabi, S. Harford, Multivariate LSTM-FCNs for time series classification, Neural Netw. 116 (2019) 237–245.

[42] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A.X. Liu, S. Dustdar, Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting, in: International Conference on Learning Representations, 2021.

[43] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, H. Hao, Semantic clustering and convolutional neural network for short text categorization, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2015, pp. 352–357.

[44] C.W. Tan, F. Petitjean, E. Keogh, G.I. Webb, Time series classification for varying length series, 2019, arXiv preprint arXiv:1910.04341.

[45] M. Dwarampudi, N. Reddy, Effects of padding on LSTMs and CNNs, 2019, arXiv preprint arXiv:1903.07288.

[46] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, in: International Conference on Machine Learning, PMLR, 2022, pp. 27268–27286.

[47] W. Cai, Y. Liang, X. Liu, J. Feng, Y. Wu, MSGNet: Learning multi-scale inter-series correlations for multivariate time series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, 2024, pp. 11141–11149, 10.

[48] J. Xu, H. Gunes, K. Kusumam, M. Valstar, S. Song, Two-stage temporal modelling framework for video-based depression recognition using graph representation, IEEE Trans. Affect. Comput. (2024).

[49] M. Chen, X. Xiao, B. Zhang, X. Liu, R. Lu, Neural architecture searching for facial attributes-based depression recognition, in: 2022 26th International Conference on Pattern Recognition, ICPR, IEEE, 2022, pp. 877–884.

[50] H. Shu, W. Song, Z. Song, H. Guo, C. Li, Y. Wang, Multistep short-term wind speed prediction with rank pooling and fast Fourier transformation, Wind. Energy 27 (7) (2024) 667–694.

[51] R. Liao, S. Song, H. Gunes, An open-source benchmark of deep learning models for audio-visual apparent and self-reported personality recognition, IEEE Trans. Affect. Comput. (2024).

[52] R. Ju, P. Zhou, S. Wen, W. Wei, Y. Xue, X. Huang, X. Yang, 3D-CNN-SPP: A patient risk prediction system from electronic health records via 3D CNN and spatial pyramid pooling, IEEE Trans. Emerg. Top. Comput. Intell. 5 (2) (2020) 247–261.

[53] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Transfer learning for time series classification, in: 2018 IEEE International Conference on Big Data, Big Data, IEEE, 2018, pp. 1367–1376.

[54] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019).

[55] F.A. Del Campo, M.C.G. Neri, O.O.V. Villegas, V.G.C. Sánchez, H.d.O. Domínguez, V.G. Jiménez, Auto-adaptive multilayer perceptron for univariate time series classification, Expert Syst. Appl. 181 (2021) 115147.

[56] S. Belagoune, N. Bali, A. Bakdi, B. Baadji, K. Atif, Deep learning through LSTM classification and regression for transmission line fault detection, diagnosis and location in large-scale multi-machine power systems, Measurement 177 (2021) 109330.

[57] H. Sheng, M. Liu, J. Hu, P. Li, Y. Peng, Y. Yi, LA-ESN: a novel method for time series classification, Information 14 (2) (2023) 67.

[58] D. Lee, S. Lee, H. Yu, Learnable dynamic temporal pooling for time series classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 8288–8296, 9.

[59] M. Khan, H. Wang, A. Ngueilbaye, A. Elfatyany, End-to-end multivariate time series classification via hybrid deep learning architectures, Pers. Ubiquitous Comput. 27 (2) (2023) 177–191.

[60] Y. Lei, Z. Wu, Time series classification based on statistical features, EURASIP J. Wirel. Commun. Netw. 2020 (1) (2020) 46.

[61] M. Middlehurst, P. Schäfer, A. Bagnall, Bake off redux: a review and experimental evaluation of recent time series classification algorithms, Data Min. Knowl. Discov. (2024) 1–74.

[62] Y. Zhang, Y. Hou, K. OuYang, S. Zhou, Multi-scale signed recurrence plot based time series classification using inception architectural networks, Pattern Recognit. 123 (2022) 108385.

[63] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, Adv. Neural Inf. Process. Syst. 34 (2021) 22419–22430.

[64] A. Benavoli, G. Corani, F. Mangili, M. Zaffalon, F. Ruggeri, A Bayesian wilcoxon signed-rank test based on the Dirichlet process, in: International Conference on Machine Learning, PMLR, 2014, pp. 1026–1034.

[65] B. McFee, J. Salamon, J.P. Bello, Adaptive pooling operators for weakly labeled sound event detection, IEEE/ACM Trans. Audio Speech Lang. Process. 26 (11) (2018) 2180–2193.

[66] X. Wu, C. Huang, P. Robles-Granda, N.V. Chawla, Representation learning on variable length and incomplete wearable-sensory time series, ACM Trans. Intell. Syst. Technol. (TIST) 13 (6) (2022) 1–21.

[67] C. Ji, Y. Hu, S. Liu, L. Pan, B. Li, X. Zheng, Fully convolutional networks with shapelet features for time series classification, Inform. Sci. 612 (2022) 835–847.

[68] G. Li, B. Choi, J. Xu, S.S. Bhowmick, K.-P. Chun, G.L.-H. Wong, Shapenet: A shapelet-neural network approach for multivariate time series classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 8375–8383, 9.

[69] Y. He, J. Zhao, Temporal convolutional networks for anomaly detection in time series, in: Journal of Physics: Conference Series, vol. 1213, IOP Publishing, 2019, 042050, 4.

[70] M. Imamura, T. Nakamura, Parameter-free spikelet: Discovering different length and warped time series motifs using an adaptive time series representation, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 857–866.

[71] R. Rai, T. Basikolo, S.B. TSB, Enhancing user experience in home networks with machine learning-based classification, 2024.

[72] Z. Aldeneh, E.M. Provost, Using regional saliency for speech emotion recognition, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2017, pp. 2741–2745.

[73] Y. Wang, W. Yao, M. Dong, Y. Li, L. Zhu, S. Bi, Prediction of battery capacity based on deep residual network, in: 2022 12th International Conference on CYBER Technology in Automation, Control, and Intelligent Systems, CYBER, IEEE, 2022, pp. 462–467.

[74] A.M. Sadeghzadeh, S. Shiravi, R. Jalili, Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification, IEEE Trans. Netw. Serv. Manag. 18 (2) (2021) 1962–1976.
[75] V. Cerqueira, L. Torgo, I. Mozetič, Evaluating time series forecasting models: An empirical study on performance estimation methods, Mach. Learn. 109 (11) (2020) 1997–2028.
[76] A.P. Ruiz, M. Flynn, J. Large, M. Middlehurst, A. Bagnall, The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. 35 (2) (2021) 401–449.
[77] Z. Che, S. Purushotham, G. Li, B. Jiang, Y. Liu, Hierarchical deep generative models for multi-rate multivariate time series, in: International Conference on Machine Learning, PMLR, 2018, pp. 784–793.
[78] G. Huang, Missing data filling method based on linear interpolation and lightgbm, in: Journal of Physics: Conference Series, vol. 1754, IOP Publishing, 2021, 012187, 1.
[79] C. Zhang, H. Fanaee-T, M. Thoresen, Feature extraction from unequal length heterogeneous EHR time series via dynamic time warping and tensor decomposition, Data Min. Knowl. Discov. 35 (4) (2021) 1760–1784.
[80] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.
[81] I. Martín-Morató, M. Cobos, F.J. Ferri, Adaptive distance-based pooling in convolutional neural networks for audio event classification, IEEE/ACM Trans. Audio Speech Lang. Process. 28 (2020) 1925–1935.
[82] Y. Zhu, H. Luo, R. Chen, F. Zhao, L. Su, DenseNetX and GRU for the sussex-huawei locomotion-transportation recognition challenge, in: Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers, 2020, pp. 373–377.
[83] K. Huang, F. Wang, Y. Wang, TS-TWC: A time series representation learning framework based on time-wavelet contrasting, Biomed. Signal Process. Control. 88 (2024) 105678.
[84] X. Wang, P. Wang, Y. Song, Q. Xiang, J. Li, Recognition of high-resolution range profile sequence based on TCN with sequence length-adaptive algorithm and elastic net regularization, Expert Syst. Appl. (2024) 123417.
[85] C.E. Brown, Coefficient of variation, in: Applied Multivariate Statistics in Geohydrology and Related Sciences, Springer, 1998, pp. 155–157.
[86] W.M. Gentleman, G. Sande, Fast Fourier transforms, in: Proceedings of the November 7-10, 1966, Fall Joint Computer Conference on XX - AFIPS '66, Fall, 1966, http://dx.doi.org/10.1145/1464291.1464352.
[87] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, A. Courville, On the spectral bias of neural networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 5301–5310.