sensors

MDPI

Article

Beyond Information Distortion: Imaging Variable-Length Time Series Data for Classification

Hyeonsu Lee and Dongmin Shin *

Department of Industrial and Management Engineering, Hanyang University, Ansan 15588, Republic of Korea; hyeonsulee@hanyang.ac.kr

* Correspondence: dmshin@hanyang.ac.kr

Abstract: Time series data are prevalent in diverse fields such as manufacturing and sensor-based human activity recognition. In real-world applications, these data are often collected with variable sample lengths, which can pose challenges for classification models that typically require fixed-length inputs. Existing approaches either employ models designed to handle variable input sizes or standardize sample lengths before applying models; however, we contend that these approaches may compromise data integrity and ultimately reduce model performance. To address this issue, we propose Time series Into Pixels (TIP), an intuitive yet strong method that maps each time series data point into a pixel in 2D representation, where the vertical axis represents time steps and the horizontal axis captures the value at each timestamp. To evaluate our representation without relying on a powerful vision model as a backbone, we employ a straightforward LeNet-like 2D CNN model. Through extensive evaluations against 10 baseline models across 11 real-world benchmarks, TIP achieves 2–5% higher accuracy and 10–25% higher macro average precision. We also demonstrate that TIP performs comparably on complex multivariate data, with ablation studies underscoring the potential hazard of length normalization techniques in variable-length scenarios. We believe this method provides a significant advancement for handling variable-length time series data in real-world applications. The code is publicly available.

Keywords: time series classification (TSC); variable-length time series

check for updates

Academic Editors: Jan Cornelis, Qiong Wang, Teng Huang and Yan Pang

Received: 14 November 2024
Revised: 20 January 2025
Accepted: 21 January 2025
Published: 21 January 2025

Citation: Lee, H.; Shin, D. Beyond Information Distortion: Imaging Variable-Length Time Series Data for Classification. *Sensors* **2025**, *25*, 621. https://doi.org/10.3390/s25030621

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Time series data are ubiquitous and appear across various domains, from trajectories [1] and sensor-based human action recognition [2] to monitoring data from facilities and machines in manufacturing systems [3]. Time series data collected from the real-world often have heterogeneous traits, such as *variable length*—the lengths of the series sample are diverse [4]. Given the prevalence of variable-length time series data in real-world scenarios [5], finding effective ways to handle such data is crucial. Previous studies have preprocessed variable-length time series data to make their lengths uniform by sliding a fixed-size window [6], resampling, or introducing artificial values [7]. However, the best method for managing variable-length time series data and their effectiveness are still unclear, with many recent studies having highlighted it as a future research question in TSC [4,8–10].

This issue is even more pronounced for time series data collected by sensors, one of the most widely used tools to gather real-world time series data [11]. Such data often contain consistent noise [12], thus making it critical to capture *key local patterns*—the patterns or subsequences that best explain inter-class variability [13] (e.g., time series data from walking

versus sitting classes, or normal versus anomalous data in machine monitoring, may each have distinct key local patterns). To capture key local patterns for TSC, 1D-CNN methods are traditionally employed due to their success in detecting local patterns [14]. One-dimensional CNNs apply filters along the time dimension, enabling the network to identify local patterns that distinguish between classes. Additionally, CNNs can detect features regardless of their position within the time series, which is particularly advantageous in TSC, where the location of key local patterns can vary across instances [15–17].

However, capturing key local patterns in variable-length time series presents a significant challenge for 1D CNNs. These models require a fixed input size for batch processing, necessitating length uniformization techniques such as truncation, interpolation, or padding. While these methods enable the use of a 1D CNN architecture, they can introduce noise and distort critical information, including key local patterns, in the original data. This may negatively affect model performance by interfering with the capture of key local patterns, eventually leading to suboptimal results. Figure 1 illustrates this phenomenon.

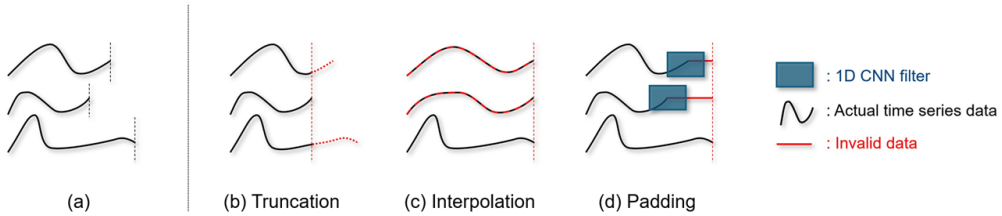


Figure 1. (**a**) Toy example illustrations of variable-length time series data. (**b**–**d**) Common techniques to uniformize the variable-length time series data. (**b**) Truncation cuts off parts of the series to fit a predetermined length, potentially removing crucial information. (**c**) Interpolation inserts data points to adjust the series to a standard length, distorting true temporal dynamics. (**d**) Padding adds repeating values at the end, creating misleading patterns and forcing 1D convolutional filters to process both actual and invalid data indiscriminately.

Although alternatives to 1D CNNs are viable for handling data heterogeneity and are frequently used in various domains [4], their ability to capture features for TSC is questionable. Specifically, for non-deep learning-based methods, the separation of feature extraction and downstream tasks can constrain their performance. While RNN-based and transformer-based approaches can preserve data integrity by using mask layers to avoid padding effects, their processing architecture may struggle to capture the often position-variant key local patterns in TSC. This creates an unresolved trade-off between capturing key local patterns and preserving the data integrity of variable-length time series.

Motivated by these limitations, we focus on one primary question: "***How can we effectively capture the key local pattern for TSC while preserving the original representation of variable-length time series?***"

To address this question, we introduce **TIP** (Time series Into Pixels), which directly maps each value at each time step into a pixel in a 2D image, where the vertical axis represents time steps and the horizontal axis represents the value at each time step. By converting 1D time series data into a 2D format, TIP enables the use of CNNs to capture key local patterns while preserving data integrity for variable-length TSC.

To evaluate our method, we conduct extensive experiments, comparing TIP to 10 baseline methods across 11 real-world benchmarks. Without relying on a powerful vision model as a backbone, we use a straightforward LeNet-like 2D CNN model. Through these experiments, TIP consistently outperforms existing methods, showing 2–5% higher accuracy and 10–25% higher macro average precision, demonstrating the effectiveness of our method for variable-length time series classification. Furthermore, we show that TIP performs comparably in complex multivariate scenarios. Finally, through an ablation study,

we confirm that length uniformization techniques for handling variable-length data can lead to performance degradation.

Our contributions are twofold:

1. We introduce TIP, a simple yet robust method that converts variable-length time series into a 2D pixel-based format, mapping time steps to the vertical axis and values to the horizontal. This representation enables vision models, including CNNs, to capture key local patterns effectively while preserving data integrity.
2. We validate TIP by employing a straightforward LeNet-like CNN, demonstrating that its performance gains arise from the data representation itself rather than the complexity or power of the vision model.

2. Existing Solutions for Variable-Length TSC

In this section, we review existing methods and potential solutions not yet explored in the literature for variable-length TSC. We categorize these solutions into three groups: non-deep learning-based solutions, masking-based solutions, and 1D-CNN-based solutions with preprocessing. We exclude single-batch-sized solutions from deep learning-based methods; although technically feasible, they are impractical due to high variance in batch loss, resulting in unstable and prolonged training. Additionally, this approach underutilizes GPU resources. We organized the main differences between previous methods and our method in Table 1.

Table 1. Comparison of existing solutions and our proposed method for variable-length TSC.

Aspect	Non-Deep Learning-Based (Section 2.1)	Masking-Based (Section 2.2)	1D CNN-Based (Section 2.3)	Ours
End-to-end learning	×	✓	✓	✓
Effective feature extraction for TSC	✓	×	✓	✓
Preserving original data integrity	✓	✓	×	✓

We include early classification methods within the scope of non-deep learning-based solutions. Although they could fit other categories depending on the classifier, recent studies employ SVMs, so we include them in this category.

2.1. Non-Deep Learning-Based Solutions

Before the emergence of deep learning in TSC, non-deep learning or machine learning models were primarily used. These models configured the feature space by measuring distances between samples or identifying key local patterns or shapelets essential for classification. With flexible feature extraction approaches, many of these models do not require fixed-size inputs, making them suitable for variable-length TSC problems. For example, DTW-1NN [18], a go-to method in TSC [19], handles variable-length time series data using dynamic time warping (DTW) distance, capturing time series characteristics like variations in speed and duration without modifying the series. Similarly, shapelet-based methods extract key local patterns by sampling subsequences or shapelets and selecting the ones that best characterize each class, supporting variable-length TSC without uniform length conversion. Early classification methods further address variable-length TSC by making confident predictions on partial data through adaptive thresholding or window-based segmentation, thus enabling efficient classification without full sequence input.

While these methods are effective, with some performing comparably to state-of-the-art models [8], they deviate from deep learning's end-to-end approach. The separation between feature extraction and classification can limit their ability to optimize the feature space for identifying key local patterns.

2.2. Masking-Based Solutions

RNN-based models were initially the primary choice in deep learning for time series analysis due to their ability to capture temporal dependencies, making them widely used in TSC across various domains. More recently, transformers have gained popularity in time series analysis, particularly for handling variable-length sequences. Having proven their success in NLP, where sentence lengths vary, they manage sequence variability with padding, masking, and selective computation. RNNs handle padded sequences by excluding artificial values from loss calculations through element-wise summation, while transformers use attention mechanisms, assigning large negative values to padding elements, which reduces their softmax attention scores to zero.

While RNNs and transformers perform well in time series tasks like long-term forecasting and imputation, they have limited application in TSC. Apart from other issues detailed in [14], one key limitation is their tendency to capture global dependencies rather than local ones. This poses challenges in identifying key local patterns essential for classification, especially given common variations in these patterns (e.g., phase, warping, offset) in time series data [20,21]. Some approaches utilize RNNs to address variable-length TSC by embedding series into fixed-size latent representations and applying a classifier, similar to a seq2seq architecture [22]. However, this approach underperforms due to ineffective feature extraction, especially when compared with non-deep learning methods and 1D CNN-based deep learning methods, which will be discussed next.

2.3. 1D CNN-Based with Preprocessing Solutions

One-dimensional CNN-based methods have become a gold standard in TSC thanks to their robust feature extraction capabilities [19]. By employing 1D convolutional filters, these models effectively capture various local patterns—including phase shifts, warping, and offsets—regardless of where they appear along the temporal axis. Despite these strengths, 1D CNNs typically require fixed-length inputs, posing challenges for TSC tasks with variable-length sequences.

A common workaround is to uniformly extend each time series in a dataset to the maximum sequence length via padding. Unfortunately, this practice can degrade data integrity by mixing valid observations with synthetic zeros, thereby obscuring the true signal. In 1D CNNs, convolutional filters then process both real and padded values together, generating noisy feature maps that struggle to differentiate between authentic and invalid data.

This issue is compounded by z-normalization, a widely recognized preprocessing step in TSC that scales each time series to zero mean and unit variance [19,23,24]. Used extensively across the TSC domain—including many state-of-the-art methods such as InceptionTime [25]—z-normalization promotes stable training and consistent input scaling. However, once normalized, zero-padding can become statistically indistinguishable from genuinely low-value data points, making it even harder for CNN filters to separate signals from padding. This often leads to uniform, non-informative activations in the padded regions, ultimately diluting or biasing the model's learned features.

While z-normalization helps stabilize training by standardizing each series' distribution, it can inadvertently cause padded zeros to appear as legitimate (low-valued) data. Consequently, in variable-length TSC settings—where padding is unavoidable—the model may treat spurious regions as meaningful, obscuring true temporal patterns and impairing overall classification performance.

In certain controlled scenarios—such as when zero is a valid and meaningful measurement that naturally appears in the data—padding with zeros may still convey some structural information about sequence length. However, if zero values are frequent or hold

no meaningful semantics in a given domain, padding risks merging authentic signals with artificially introduced noise, hindering the network's ability to differentiate between real observations and placeholders.

These findings highlight the need for careful consideration when applying padding and z-normalization together, especially for sequences of differing lengths. Despite the recognized benefits of z-normalization in accelerating convergence and promoting stable training, practitioners should remain mindful that zero-padding can confound learned feature representations—ultimately hindering the performance of 1D CNN-based TSC models such as InceptionTime [25]. See Figure 2.

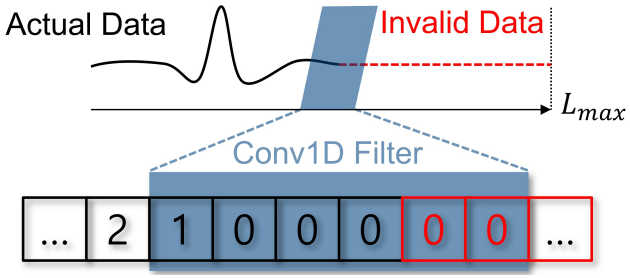


Figure 2. Illustration of the convolution operation with padding in a 1D CNN. Padding extends the series to the maximum length $L_{\max}$, introducing artificial values that can affect the feature maps and the performance of the CNN.

In addition to padding, other length uniformization techniques, such as interpolation and truncation, distort the data's inherent properties (Figure 1a,b). Interpolation can alter event timing within the series, and truncation may remove essential information. Although these techniques enable deep learning models to handle variable-length TSC, they compromise the integrity of the original time series representation, limiting the model's ability to capture critical characteristics.

While some modified 1D CNN architectures attempt to handle variable-length inputs through masking layers [26,27], these approaches remain limited in scope and are typically evaluated only against older baselines like DTW. Such masking-based strategies assume a certain level of temporal correlation between short and long sequences, which can lead to misrepresentations. In contrast, our method transforms time series data into a 2D representation and applies a two-dimensional CNN (2D-CNN), allowing convolutional models to capture key local patterns with inherent variability while preserving data integrity. Additionally, we compare our method to 10 baselines across 11 real-world benchmark datasets, positioning our approach as an accurate and effective solution for variable-length TSC.

3. Proposed Method

3.1. Preliminaries: Variable-Length TSC Problem Formulation

Let $\mathcal{D} = \{(\mathcal{X}, \mathcal{Y})\}$ be our dataset, where $\mathcal{X} = \{\mathbf{x}_n^{(1:L_n)}\}_{n=1}^{N}$ consists of variable-length time series data, with each series $\mathbf{x}_n^{(1:L_n)} := (\mathbf{x}_n^t)_{t=1}^{L_n} = (\mathbf{x}_n^1, ..., \mathbf{x}_n^{L_n})$. Each element $\mathbf{x}_n^t$ within a series lies in $\mathbb{R}^d$, where L_n denotes the length of the n-th series and d represents the number of variables or variates measured at each time point. We characterize $\mathcal{X}$ as a set of variable-length time series if there exist distinct indices $i, j \in \{1, \ldots, N\} \subset \mathbb{N}^+$ such that $L_i \neq L_j$. For simplicity in notation, we will initially focus on the univariate case ($d = 1$), where $\mathbf{x}_n^t$ is simplified to $x_n^t \in \mathbb{R}$. The primary objective of variable-length TSC is to develop a model that assigns each variable-length times series $\mathbf{x}_n^{(1:L_n)}$ to one of the possible classes $y_n \in \mathcal{Y}$.

3.2. TIP: Time Series Into Pixels

To preserve original data from information distortion in variable-length time series classification (TSC), we introduce a transformation method called *Time series Into Pixels* (*TIP*). In this method, each time series value is mapped to a corresponding pixel using a binary mask: pixels representing the original data values are set to 1, while all other pixels are set to 0. Each time step in the series is mapped to its respective pixel by finding the appropriate horizontal axis index, with the vertical axis representing the time axis.

By transforming the time series into pixels, it eliminates the need to handle invalid values for variable-length time series, thereby preserving the original data representation while possessing the effective feature extraction capabilities of CNNs. The procedure is detailed in Algorithm 1.

Algorithm 1 TIP: Time series Into Pixels

Input: Variable-length time series $x_n^{(1:L_n)}$
Input: Maximum length L_{max}
Output: Transformed variable-length time series I_n
Parameter: Boundary constants β^-, β^+

1: Set the boundary constants to fit the time series data $x_n^{(1:L_n)}$ in $[\beta^-, \beta^+]$
2: $\mathrm{I}_n \leftarrow \mathbf{0}_{L_{max} \times L_{max}}$ ▷ Initialize image matrix
3: **for** each time step column index $i = 0, ..., L_{\max} - 1$ **do**
4: Find the corresponding row j using Equation (2)
5: Set $\mathrm{I}_n[i, j] = 1$ ▷ Mark the corresponding pixel
6: **end for**
7: **(optional):** For very long sequences, resize I_n with certain ratio
8: **(optional):** For multivariate case, repeat from step 1 to 6 by each variate and stack
9: **return** I_n

First, we scale the time series values to fit in the image; all time series data x_n^t for each time step $t \in [1, L_n]$ fit to the range $[\beta^-, \beta^+]$, where β^-, β^+ are boundary constants. This approach ensures that every time series value is effectively mapped to a corresponding position within the 2D representation, preserving the full range of the original data [28–30]. Then, we construct an image matrix I_n with dimensions $L_{max} \times L_{max}$. Each column of I_n corresponds to a time step in the time series, and each row corresponds to a value of x_n^t. To map the time series value to a corresponding row index $j \in \{0, \ldots, L_{max} - 1\}$, the range $[\beta^-, \beta^+]$ is divided into $L_{\max}$ bins, each covering an interval $[\beta^- + j\Delta, \beta^- + (j+1)\Delta]$. The incremental amount is calculated as $\Delta = \frac{\beta^+ - \beta^-}{L_{\max}}$. This discretization can be represented as:

$$[\beta^-, \beta^+] \to \{[\beta^-, \beta^- + \Delta], \ldots, [\beta^- + j\Delta, \beta^- + (j+1)\Delta], \ldots, [\beta^+ - \Delta, \beta^+]\} \quad (1)$$

Thus, the value at each time step x_n^t is mapped to a specific pixel in I_n according to:

$$x_n^t \mapsto I_n[i, j] \text{ where } x_n^t \in [\beta^- + j\Delta, \beta^- + (j+1)\Delta],\ i = t - 1, \text{ and } j \in [0, L_{max} - 1] \subset \mathbb{N}_0 \quad (2)$$

One of the notable strengths of our method is its dynamic adjustment of the representation of the original time series data through the hyperparameters β^-, β^+. Since the incremental value $\Delta = \frac{\beta^+ - \beta^-}{L_{\max}}$ between each discretized interval $[\beta^- + j\Delta, \beta^- + (j+1)\Delta]$ $= \left[\beta^- + j\left(\frac{\beta^+ - \beta^-}{L_{\max}}\right), \beta^- + (j+1)\left(\frac{\beta^+ - \beta^-}{L_{max}}\right)\right]$ is derived from β^-, β^+, a higher value of the range $[\beta^-, \beta^+]$ results in a wider range for each discretized interval, whereas a lower value means a narrower range. This enables dynamic adjustments to the representation of time series data, allowing models to consider the appropriate scale or shape of it.

Unlike traditional techniques that rely on padding, interpolation, or truncation, our Time series Into Pixels (TIP) method transforms a variable-length time series into

a two-dimensional binary pixel representation without distorting the underlying signal. Unlike existing variable-length TSC solutions that typically rely on one-dimensional architectures with masking layers or manual temporal alignment, TIP directly maps each original time series value to a unique pixel, effectively preserving the original timing, scale, and structure of the data. This representation simultaneously enables the use of vision models such as 2D-CNNs, which are known to be effective in capturing local patterns across spatial dimensions. Thus, TIP departs from prior approaches that often either compromise data integrity or require specialized modules to handle variable-length inputs.

3.3. 2D-CNN Backbone

To verify the effectiveness of our 2D representation without relying on the performance of a strong backbone model, we employed a very straightforward 2D CNN model similar to LeNet [31], the primitive CNN model. Like LeNet, our model consists of standard convolutional layers, pooling layers, and a fully connected layer. The primary differences between our model and LeNet are the inclusion of dropout layers and a global pooling layer, which are now commonly used to improve model robustness with lower computational cost. See Figure 3.

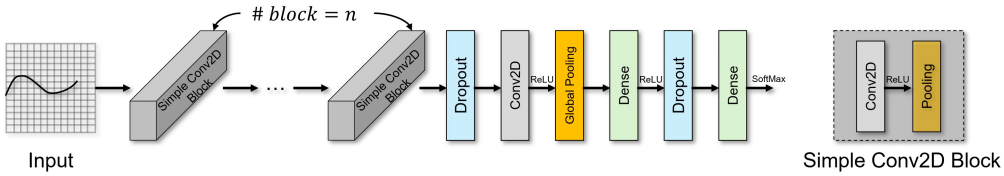


Figure 3. Illustration of our simple 2D CNN backbone. Detailed settings (e.g., kernel size, dense units) are provided in the Appendix A.

We used two baselines for TIP: ***Not-Refined TIP*** and ***Refined TIP***. Not-Refined TIP uses the same architecture and hyperparameters (i.e., batch size, learning rate, $\beta = 4$) across all datasets, hence the name. Refined TIP, on the other hand, involves tuning β and all hyperparameters of the simple backbone model, such as the number of convolutional layers and the units within specific ranges. Despite these refinements, it retains only the basic features of LeNet, including convolutional layers, pooling layers, and dense layers. Detailed explanations of our backbone are provided in Appendix A.

4. Experiment

4.1. Datasets and Metrics

To evaluate the effectiveness of our method on real-world datasets, we conducted experiments on 11 variable-length time series datasets from the UCR Time Series Classification Archive (https://www.cs.ucr.edu/~eamonn/time_series_data_2018/, accessed on 10 July 2024), one of the most widely used benchmarks in the TSC domain. For evaluation metrics, we used accuracy and macro average precision (MAP) since all datasets are class-balanced. Detailed descriptions of the datasets are available in Appendix A.

4.2. Baselines for Comparison

We compared our method with the leading existing solutions for variable-length TSC, as introduced in Section 2. We selected the top-performing methods from each category. Below is a brief description of the methods used in our comparison and how each method handles variable-length inputs:

4.2.1. Non-DL Methods

- ED-1NN: A simple distance-based baseline using Euclidean distance for classification. Often used as a reference due to its simplicity. To handle variable-length sequences, ED-1NN typically requires padding or truncation of time series to a fixed length. We chose zero-padding here.
- DTW-1NN [18]: Long considered the gold standard for TSC, DTW-1NN excels at handling misaligned time series and has remained difficult to surpass [32]. DTW inherently handles variable-length sequences by allowing elastic shifts in the time axis, effectively aligning sequences of different lengths without the need for padding.
- TEASER [33]: An early classifier that efficiently handles variable-length inputs without extensive preprocessing. TEASER efficiently processes variable-length time series data by identifying critical patterns early in the sequence, without being influenced by the overall sequence length.
- RDST [34]: A shapelet-based method known for extracting key local patterns, providing one of the state-of-the-art performances on TSC [8]. RDST manages variable-length sequences by identifying and utilizing shapelets of different lengths, allowing it to flexibly match and classify time series without requiring uniform sequence lengths.

4.2.2. Masking Methods

- Vanilla RNN [35]: A basic recurrent model using masking to handle variable-length sequences, though surpassed by more advanced architectures. Masking involves padding shorter sequences and using a mask to indicate the actual length, ensuring that the RNN processes only the valid time steps during training and inference.
- BiLSTM [36]: Bidirectional LSTM, leveraging masking and bidirectional processing, and offering more robust results than RNN. Similar to vanilla RNN, BiLSTM uses masking to manage variable-length inputs, allowing the model to capture dependencies in both forward and backward directions without being affected by the padded values.
- TST [37]: A transformer-based model using masking, with competitive performance across diverse time series datasets. In TST, masking is employed to handle variable-length sequences by padding shorter sequences and applying attention masks to prevent the model from attending to padded positions, thereby maintaining the integrity of the actual data.

4.2.3. 1D-CNN Methods

- FCN [38]: A fully convolutional model that efficiently extracts spatial patterns, widely used for TSC tasks. To handle variable-length sequences, FCN typically applies preprocessing steps such as padding, truncation, or interpolation to convert all input time series to a fixed length.
- ResNet [38]: A deep architecture adapted from image classification, known for its strong results in TSC [39]. Like FCN, ResNet also manages variable-length time series by incorporating preprocessing steps like padding, truncation, or interpolation to standardize input lengths.
- InceptionTime [25]: A top-performing deep learning model for TSC, consistently demonstrating superior performance with its multi-scale architecture, as highlighted in recent reviews [8]. We selected this baseline for its robustness, though variants like LITETime [40] offer reduced computation time while maintaining the same performance as the original. InceptionTime handles variable-length inputs by utilizing preprocessing techniques such as padding, truncation, or interpolation to achieve a fixed input size.

As evaluated in a recent review [8], other methods such as [41–43] have also demonstrated strong performance in TSC. However, we excluded these methods because, when compared to the baselines we selected, the performance differences were negligible, and they were either computationally too expensive or not applicable to variable-length TSC due to fixed size input constraint.

4.3. Implementation Details

Following prior publications [18,25] and the recommendations from the UCR Archive [19,24], all experiments used the default training and test splits provided by the UCR Archive, and each series was z-normalized. We implemented the best architectures reported in the literature, such as using five classifiers in the ensemble for InceptionTime [25] and 10,000 shapelets for RDST [34].

Since few studies adopt RNN-based models for TSC, we developed the vanilla RNN and BiLSTM in-house. We provide the detailed implementations of the two models in Appendix A. In addition to these models, we used implementations available in aeon (https://www.aeon-toolkit.org, accessed on 10 July 2024) and tsai (https://timeseriesai.github.io/tsai, accessed on 10 July 2024), the popular third-party libraries in time series analysis.

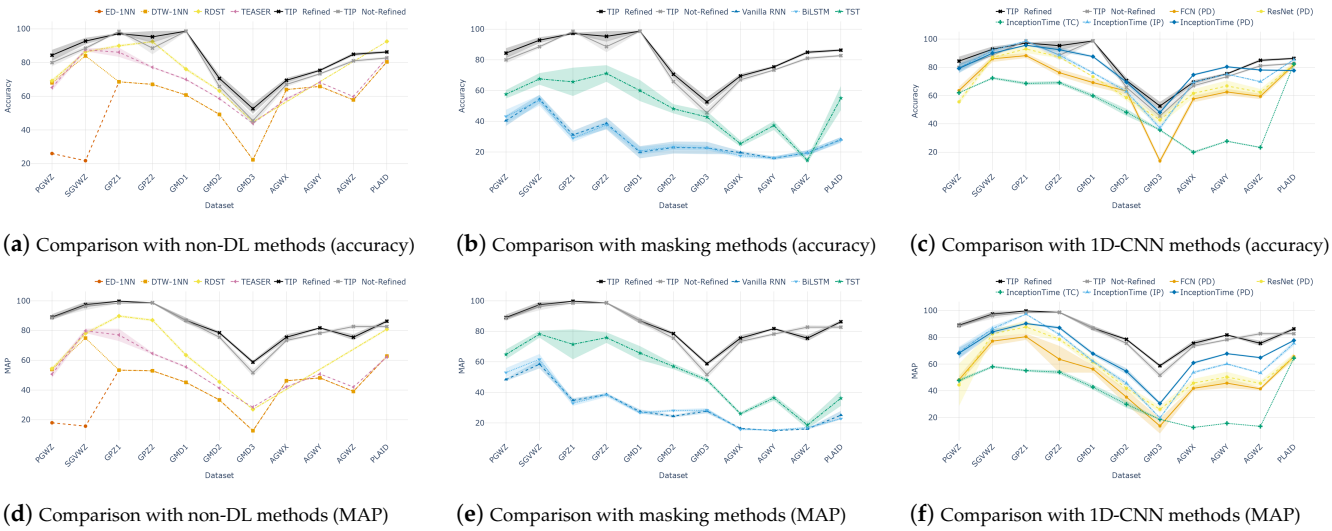
For 1D-CNN methods, we applied various length uniformization techniques, including truncation (TC), interpolation (IP), and padding (PD). For the most powerful 1D-CNN-based method, InceptionTime, we applied all three length uniformization techniques, while for other 1D-CNN baselines, we used only padding, as it yielded the best performance.

For all deep learning-based baselines, we added an early-stopping strategy and a plateau learning rate scheduler to avoid overfitting and getting stuck in local optima. These strategies are often omitted in the related literature despite their importance. By implementing these strategies and fine-tuning the batch size and learning rate over sufficient trials, we achieved improved performance for the baselines compared to their reported results. Additional implementation details, including the search space for fine-tuning and training strategy, are provided in Appendix A. All models were trained using the TensorFlow framework on a single NVIDIA RTX 2080 Ti GPU (NVIDIA Corporation, Santa Clara, CA, USA). We have made the complete source code available.

4.4. Results

As demonstrated in Figure 4, our approach consistently achieves the best or second-best results in both accuracy and MAP. Specifically, non-DL methods exhibited inferior performance due to the separation between configuring the feature space and the downstream classification task. In particular, ED-1NN recorded significantly lower performance than DTW-1NN because it failed to capture key aspects of variable-length sequences, such as speed variation. Although TEASER and RDST showed relatively decent performance, the aforementioned two-step learning process hindered the optimal configuration of the feature space for classification.

Similarly, recurrent models and transformers also demonstrated subpar performance. While these models maintain data integrity through the use of masking, their classification performance remains low. Beyond the traditional reasons for their low performance in TSC [14], their forward structures are inadequate for classification tasks where key local patterns are often position-variant, making it difficult to effectively capture the underlying dynamics. Although TST outperformed both recurrent models and transformers, it still lagged behind 1D-CNN-based methods and our convolutional model-based method, TIP, which excel at learning local patterns.



(**a**) Comparison with non-DL methods (accuracy) (**b**) Comparison with masking methods (accuracy) (**c**) Comparison with 1D-CNN methods (accuracy)

(**d**) Comparison with non-DL methods (MAP) (**e**) Comparison with masking methods (MAP) (**f**) Comparison with 1D-CNN methods (MAP)

Figure 4. Comparison to existing variable-length TSC solutions on 11 UCR real-world benchmarks, evaluated by (**a**–**c**) Accuracy (%) and (**d**–**f**) Macro Average Precision (MAP), with standard deviations computed across five seeds.

Lastly, 1D-CNN-based models generally performed well; however, our method, TIP, outperformed them across most datasets. Notably, InceptionTime, which employs truncation as a preprocessing step, achieved the lowest performance despite being the latest and ostensibly the best-performing algorithm in its category. This result indicates that truncation as a preprocessing method significantly distorts the original data information. When padding, instead of truncation, was applied, InceptionTime, ResNet, and FCN showed progressively higher performance. This improvement can be attributed to their complex structures: InceptionTime utilizes ensemble techniques and inception blocks, which are modern feature extraction methods; ResNet incorporates residual connections; and FCN consists solely of convolutional layers. Additionally, padding generally outperformed interpolation, likely because interpolation excessively alters the temporal dynamics of the original data. However, overall, these imputation methods using artificial values resulted in lower performance compared to our method, which preserves data integrity by employing a straightforward vision model with a 2D-CNN backbone.

5. Additional Analysis

5.1. What About in Multivariate Setting?

To evaluate the effectiveness of our method in multivariate setting, we conducted experiments on variable-length and multivariate real-world benchmark datasets from the UCI Archive [44]: JapaneseVowels and CharacterTrajectories. For this analysis, each variate time series data was converted into an image and then stacked (i.e., three variates = three-channel images). Note that we continued to utilize the simple 2D CNN as a backbone for this analysis. Unlike other baselines, this model does not capture the relationships between variables, which is crucial in multivariate TSC.

As shown in Table 2, our method achieves performance comparable to other methods, despite lacking features specifically designed to capture spatial patterns between variables. While InceptionTime, the current state-of-the-art TSC integrated with length uniformization techniques, achieved the highest accuracy, our method delivered a comparable accuracy score and outperformed it in MAP. Nevertheless, our approach may face computational challenges when scaling to extremely large datasets or handling high-dimensional multivariate settings. This is primarily because each variate is treated as a separate channel and stacked, which could increase memory and processing demands. However, our results

suggest that even under these constraints, the method remains effective in maintaining high performance and robustness for complex multivariate TSC problem.

Table 2. A performance comparison with existing solutions for variable-length TSC in multivariate settings. **Best results** are highlighted in **bold**. Note that our backbone lacks the ability to capture spatial patterns and the relationships between variables.

Dataset	**JapaneseVowels**		**CharacterTrajectories**	
# Variates	12		3	
Lengths	[7,29]		[60,182]	
Train/Test	270/370		1422/1436	
Metrics	Accuracy	MAP	Accuracy	MAP
DTW-1NN	96.93	94.01	66.22	48.46
Vanilla RNN	25.52 ± 0.50	22.44 ± 0.24	22.76 ± 3.48	20.99 ± 0.58
FCN (PD)	95.10 ± 1.36	90.43 ± 2.47	76.43 ± 1.87	61.53 ± 2.11
ResNet (PD)	98.75 ± 0.16	97.44 ± 0.33	78.38 ± 2.03	65.20 ± 1.81
InceptionTime (IP)	99.21 ± 0.03	98.40 ± 0.06	**82.81** ± 0.40	69.28 ± 0.45
InceptionTime (PD)	**99.39** ± 0.05	98.73 ± 0.10	82.43 ± 0.34	68.67 ± 0.55
Not Refined TIP	98.43 ± 0.07	99.66 ± 0.03	55.30 ± 2.99	58.74 ± 2.60
Refined TIP	98.64 ± 0.23	**99.69** ± 0.05	78.11 ± 1.18	**83.35** ± 0.83

5.2. Is Preserving the Original Representation Really a Matter in Variable-Length TSC?

To investigate the impact of information distortion caused by length uniformization techniques in variable-length TSC, we conducted an ablation study of our method. We assessed performance on the PGWZ dataset using Not-Refined TIP with four different strategies for length uniformization: zero-padding, interpolation, truncation, and no adjustment (Default), which is our approach.

As shown in Table 3, the performance in terms of accuracy and MAP significantly dropped when using truncation, interpolation, and padding compared to no length uniformization techniques. This demonstrates that distorting the representation of time series data has a critical impact on model performance.

Table 3. An ablation study comparing three length uniformization techniques—truncation (TC), interpolation (IP), and padding (PD)—and their impact on information distortion in the original representation, as opposed to our technique, which introduces no information distortion.

Strategy	**Accuracy**	**MAP**
TIP (TC)	46.80 ± 1.60	58.13 ± 0.91
TIP (IP)	72.80 ± 2.04	82.81 ± 3.40
TIP (PD)	78.00 ± 2.83	85.78 ± 1.96
TIP (Default)	**80.00 ± 2.19**	**88.57 ± 0.36**

6. Conclusions and Discussion

This work highlights the challenges inherent in applying current deep learning methods to the variable-length TSC problem. To address these challenges, we propose TIP, a novel approach that directly maps time series data into 2D image representations. We believe our findings contribute significantly to advancing the handling of variable-length time series, which represent the most natural form of time series data encountered in real-world applications.

While our results in variable-length TSC are promising, several issues remain unresolved. One notable concern is the potential for information loss when resizing series, particularly for long sequences—a challenge that is pervasive and yet unresolved in current

TSC methods. Furthermore, our method treats each variate as a separate channel, which could pose challenges when dealing with high-dimensional data or sparse datasets. Such scenarios may result in computational inefficiencies or difficulty capturing the complex interdependencies between variates. To address these challenges, future research could explore methods to better preserve interdependencies in multivariate time series. For example, instead of treating each variate as a distinct channel, randomly positioning multivariate images into a shared channel stack could allow the backbone vision model to capture intricate relationships between variables more effectively. Alternatively, incorporating feature extraction or attention mechanisms tailored to multivariate data may enhance both computational efficiency and performance. Resolving these limitations remains an open area for further investigation, which we aim to address in future work.

Author Contributions: Conceptualization, H.L.; methodology, H.L.; software, H.L.; validation, H.L. and D.S.; formal analysis, H.L.; investigation, H.L.; resources, D.S.; data curation, H.L.; writing—original draft preparation, H.L.; writing—review and editing, H.L. and D.S.; visualization, H.L.; supervision, D.S.; project administration, H.L.; funding acquisition, D.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the reported results in this study are publicly available. All datasets used in this research are accessible from the UCR Time Series Classification Archive at https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (accessed on 10 July 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Experimental Details

Appendix A.1. Datasets Description

The datasets used in this study are from the UCR Time Series Archive [15], a widely utilized resource in time series research. Originally introduced in 2002 with 16 datasets, the archive has grown to encompass 128 datasets, making it a critical benchmark in the field. Among these, 11 datasets are specifically designed for variable-length time series, presenting unique challenges for classification methods. A detailed description of these datasets can be found in Table A1.

One example of such a dataset is GesturePebble, which showcases the variability in time series lengths. The data is derived from the z-axis readings of a 3-axis accelerometer in Pebble smartwatches, worn on participants' wrists as they perform six distinct gestures. The dataset consists of 304 gestures, with lengths varying naturally due to differences in individual movements. Two versions of the dataset were created:

- GesturePebbleZ1: The training set contains data from all participants in the first session, while the test set contains data from the second session.
- GesturePebbleZ2: Participants are split between the training and testing sets, making it more challenging due to the distinctiveness of individual gaits and movements.

For evaluation, we adhered to preprocessing methods, including z-normalization, as described in [25]. To ensure consistency with previous research, we used the default train/test splits provided by the UCR archive, following standard practices in the field. Additionally, each split was resampled using five random seeds to evaluate robustness. Table A1 provides an overview of the datasets used in this study.

Table A1. Descriptions of 11 UCR Benchmark Univariate Datasets with Variable Lengths.

Dataset	Train/Test	Range of Series Length	Class
PGWZ (PickupGestureWiimoteZ)	50/50	[29, 361]	10
SGWZ (ShakeGestureWiimoteZ)	50/50	[40, 385]	10
GPZ1 (GesturePebbleZ1)	132/172	[100, 455]	6
GPZ2 (GesturePebbleZ2)	146/158	[100, 455]	6
GMAD1 (GestureMidAirD1)	208/130	[80, 360]	26
GMAD2 (GestureMidAirD2)	208/130	[80, 360]	26
GMAD3 (GestureMidAirD3)	208/130	[80, 360]	26
ADWX (AllGestureWiimoteX)	300/700	[2, 385]	10
ADWY (AllGestureWiimoteY)	300/700	[2, 385]	10
ADWZ (AllGestureWiimoteZ)	300/700	[2, 385]	10
PLAID	537/537	[100, 1344]	11

Appendix A.2. Implementing Baselines

Although most baselines were primarily adapted from aeon and tsai, well-known third-party libraries for time series analysis, we developed the vanilla RNN and BiLSTM baselines from scratch. These RNN-based solutions are rarely explored in TSC due to their suboptimal feature extraction capabilities.

To address the padding issue, we utilized a mask layer to differentiate the padded parts. We padded with a high random integer and set it as the mask value, allowing us to exclude padded elements from computation. For the model architecture, we stacked RNN layers for the vanilla RNN and bidirectional LSTM layers for the BiLSTM, followed by two dense layers. We used the Adam optimizer [45] and treated the number of hidden layers and units in dense layers as hyperparameters, allowing the model to adjust its complexity based on the dataset.

Appendix A.3. Hyperparameter Settings

To ensure a fair comparison between baselines, we fine-tuned each deep learning baseline (except Not-Refined TIP) with 30 trials. Given our limited computational resources, we used the best architectures reported in the related literature (e.g., using five classifiers for the ensemble in InceptionTime [25], and 10,000 shapelets for RDST [34]). For all deep learning baselines, we fine-tuned their batch size and learning rate for each dataset, with the batch size increasing in powers of two starting from 4, and setting the maximum batch size to 32. For the learning rate, we used float numbers between 1×10^{-5} and 1×10^{-3} with logarithmic scaling.

For RNN-based methods, we additionally fine-tuned the number of units in dense layers, choosing from 64, 128, and 256, and the number of hidden layers, ranging from 2 to 5.

For the Not-Refined TIP model, no hyperparameter fine-tuning was performed. The specific hyperparameters used are as follows: $\beta = 4$, batch size = 32, learning rate = 0.0001, dense units = 256, dropout rate = 0.5, number of Conv2D layers = 5, number of Conv2D filters = 64, kernel size = 9×9, pooling type = MaxPooling2D, and global pooling type = GlobalMaxPooling2D.

In contrast, for the Refined TIP model, we fine-tuned all hyperparameters. The range for β was set between 3 and 6. The dense units were fine-tuned with values of 64, 128, and 256. The number of Conv2D layers was adjusted between 3 and 7, while the number of filters was chosen from 16, 32, and 64. The filter size was varied from 3×3 to 11×11 (odd numbers). The pooling type was selected between MaxPooling and AveragePooling. The batch size and learning rate were fine-tuned within the same ranges as the other compared baselines. We include all hyperparameters for every dataset in the code.

Using a simple 2D CNN model as the backbone, these refinements were found to be acceptable. As mentioned in the main text, apart from the inclusion of global pooling layers and dropout to enhance model robustness with reduced computational cost, no advanced techniques were incorporated into our 2D CNN backbone model beyond those derived from LeNet.

Appendix A.4. Training Strategy

Since the UCR archive provides default splits for the training and test sets, and prior publications follow this convention, we ensured consistency by using the same default splits. We used the test sets, instead of splitting the training set into a validation set, to refine each deep learning baseline, applying this method identically to all baselines. This approach is acceptable for comparative purposes.

For the deep learning baselines, we implemented an early-stopping strategy and a plateau learning rate scheduler across all methods, a detail often omitted in the related literature. We set the patience for early stopping at 50 epochs and for the plateau learning rate scheduler at 15 epochs, with a reduction factor of 0.8 and a minimum learning rate of 1×10^{-8}. We set the maximum number of epochs for both fine-tuning and evaluation at 1000, and confirmed that all baselines converged in fewer than 1000 epochs. Notably, our implementation of these methods and extensive trials resulted in higher performance scores than those reported in the related literature. The entire training procedures for each baseline are provided in our code.

Table A2. Comparison to non-DL solutions on 11 UCR real-world benchmarks, evaluated by accuracy, with standard deviations computed across five seeds. The **highest-performing results** are emphasized in **bold**, while the second-highest are underlined. "NA" indicates that the corresponding method is not applicable on this dataset.

Dataset	ED-1NN	DTW-1NN	RDST	TEASER	Not Refined TIP	Refined TIP
PGWZ	26.00	68.00	69.20 ± 0.98	65.20 ± 1.79	80.00 ± 2.22	**84.40 ± 3.20**
SGWZ	21.75	84.00	86.40 ± 0.80	87.60 ± 0.89	88.57 ± 1.13	**92.80 ± 1.94**
GPZ1	68.60	68.60	90.00 ± 0.77	86.16 ± 2.71	**98.60 ± 0.28**	97.21 ± 0.44
GPZ2	67.09	67.09	92.41 ± 0.57	77.22 ± 0.45	88.61 ± 3.61	**95.32 ± 3.42**
GMAD1	60.77	60.77	76.15 ± 0.96	70.00 ± 0.54	**98.66 ± 0.45**	**98.66 ± 0.41**
GMAD2	49.23	49.23	63.23 ± 0.58	58.62 ± 0.34	65.85 ± 2.34	**70.62 ± 2.25**
GMAD3	22.31	22.31	44.77 ± 0.58	43.85	45.54 ± 2.94	**52.62 ± 2.94**
ADWX	64.00	64.00	NA	58.43 ± 0.74	67.03 ± 1.42	**69.49 ± 1.42**
ADWY	65.86	65.86	NA	68.23 ± 0.13	73.36 ± 0.61	**75.40 ± 0.72**
ADWZ	57.86	57.86	NA	59.83	81.04 ± 0.98	**84.94 ± 0.94**
PLAID	80.45	80.45	**92.59 ± 0.22**	82.79 ± 0.47	82.75 ± 0.67	86.32 ± 0.69

Appendix A.5. Comparison to Existing Variable-Length TSC Solutions on 11 UCR Real-World Benchmarks

Here, we included the quantitative details for the contents in Figure 4. for better comparison.

Table A3. Comparison to non-DL solution on 11 UCR real-world benchmarks, evaluated by mean average precision (MAP), with standard deviations computed across five seeds. The **highest-performing results** are emphasized in **bold**, while the second-highest are underlined. "NA" indicates that the corresponding method is not applicable on this dataset.

Dataset	ED-1NN	DTW-1NN	RDST	TEASER	Not Refined TIP	Refined TIP
PGWZ	18.00	53.88	54.39 ± 1.16	50.71 ± 2.42	88.57 ± 0.36	**89.10 ± 1.94**
SGWZ	15.75	74.96	78.09 ± 1.16	79.83 ± 1.47	96.28 ± 2.44	**97.48 ± 1.81**
GPZ1	53.41	53.41	89.76 ± 0.80	77.07 ± 4.14	98.55 ± 0.41	**99.77 ± 0.14**

Table A3. *Cont.*

Dataset	ED-1NN	DTW-1NN	RDST	TEASER	Not Refined TIP	Refined TIP
GPZ2	52.96	52.96	87.01 ± 0.85	64.59 ± 0.67	98.66 ± 0.45	**98.68 ± 0.41**
GMAD1	45.19	45.19	63.50 ± 1.08	55.55 ± 0.68	**86.87 ± 1.82**	**86.87 ± 1.82**
GMAD2	33.36	33.36	45.52 ± 0.51	41.30 ± 0.13	75.57 ± 1.30	**78.51 ± 0.93**
GMAD3	12.68	12.68	27.00 ± 0.53	28.50 ± 0.02	51.64 ± 2.40	**58.76 ± 1.20**
ADWX	46.26	46.26	NA	42.18 ± 0.51	73.36 ± 1.07	**75.55 ± 1.95**
ADWY	48.20	48.20	NA	50.82 ± 0.23	78.24 ± 0.90	**81.79 ± 0.43**
ADWZ	39.08	39.08	NA	42.05	**82.75 ± 1.30**	75.55 ± 1.95
PLAID	62.84	62.84	80.94 ± 0.80	62.28 ± 0.91	82.75 ± 1.00	**86.32 ± 0.99**

Table A4. Comparison to existing masking-based solutions on 11 UCR real-world benchmarks, evaluated by accuracy, with standard deviations computed across five seeds. The **highest-performing results** are emphasized in **bold**, while the second-highest are underlined. "NA" indicates that the corresponding method is not applicable on this dataset.

Dataset	VanillaRNN	BiLSTM	TST	Not Refined TIP	Refined TIP
PGWZ	40.40 ± 3.44	43.20 ± 4.31	57.60 ± 2.76	80.00 ± 2.22	**84.40 ± 3.20**
SGWZ	54.40 ± 2.81	53.60 ± 3.45	67.60 ± 3.78	88.57 ± 1.13	**92.80 ± 1.94**
GPZ1	31.16 ± 2.69	29.07 ± 2.57	65.70 ± 9.20	**98.60 ± 0.28**	97.21 ± 0.44
GPZ2	38.61 ± 3.89	37.59 ± 2.21	71.14 ± 5.40	88.61 ± 3.61	**95.32 ± 3.42**
GMAD1	19.85 ± 3.98	20.77 ± 2.56	60.00 ± 6.83	**98.66 ± 0.45**	**98.66 ± 0.41**
GMAD2	22.92 ± 3.89	23.38 ± 1.98	48.15 ± 3.00	65.85 ± 2.34	**70.62 ± 2.25**
GMAD3	22.62 ± 3.98	22.62 ± 1.43	42.77 ± 3.75	45.54 ± 2.94	**52.62 ± 2.94**
ADWX	19.63 ± 1.04	17.60 ± 1.37	25.43 ± 1.95	67.03 ± 1.42	**69.49 ± 1.42**
ADWY	16.03 ± 1.14	16.11 ± 1.14	37.29 ± 2.93	73.36 ± 0.61	**75.40 ± 0.72**
ADWZ	19.46 ± 1.65	19.46 ± 1.65	14.54 ± 1.67	81.04 ± 0.98	**84.94 ± 0.94**
PLAID	28.08 ± 1.65	27.78 ± 1.83	55.12 ± 7.84	82.75 ± 0.67	**86.32 ± 0.69**

Table A5. Comparison to existing masking-based solutions on 11 UCR real-world benchmarks, evaluated by mean average precision (MAP), with standard deviations computed across five seeds. The **highest-performing results** are emphasized in **bold**, while the second-highest are underlined. "NA" indicates that the corresponding method is not applicable on this dataset.

Dataset	VanillaRNN	BiLSTM	TST	Not Refined TIP	Refined TIP
PGWZ	48.43 ± 0.90	53.04 ± 2.91	64.88 ± 3.26	88.57 ± 0.36	**89.10 ± 1.94**
SGWZ	58.48 ± 2.73	61.68 ± 3.45	78.21 ± 2.31	96.28 ± 2.44	**97.48 ± 1.81**
GPZ1	34.82 ± 1.40	32.88 ± 1.67	71.51 ± 9.80	98.55 ± 0.41	**99.77 ± 0.14**
GPZ2	38.68 ± 0.80	38.41 ± 1.44	75.90 ± 3.44	98.66 ± 0.45	**98.68 ± 0.41**
GMAD1	27.47 ± 1.37	26.64 ± 0.98	65.78 ± 4.56	**86.87 ± 1.82**	**86.87 ± 1.82**
GMAD2	24.41 ± 0.82	28.11 ± 0.79	57.05 ± 1.83	75.57 ± 1.30	**78.51 ± 0.93**
GMAD3	27.73 ± 1.41	28.38 ± 1.28	48.14 ± 1.35	51.64 ± 2.40	**58.76 ± 1.20**
ADWX	16.45 ± 0.48	15.72 ± 0.52	26.00 ± 1.06	73.36 ± 1.07	**75.55 ± 1.95**
ADWY	14.90 ± 0.29	15.35 ± 0.34	36.39 ± 1.84	78.24 ± 0.90	**81.79 ± 0.43**
ADWZ	16.03 ± 0.37	16.93 ± 0.48	18.60 ± 2.34	**82.75 ± 1.30**	75.55 ± 1.95
PLAID	25.07 ± 2.78	22.77 ± 0.32	36.18 ± 5.01	82.75 ± 1.00	**86.32 ± 0.99**

Table A6. Comparison to 1D-CNN-based with preprocssing solutions on 11 UCR real-world benchmarks, evaluated by accuracy, with standard deviations computed across five seeds. The **highest-performing results** are emphasized in **bold**, while the second-highest are underlined. "NA" indicates that the corresponding method is not applicable on this dataset.

Dataset	FCN (PD)	ResNet (PD)	InceptionTime (TC)	InceptionTime (PD)	InceptionTime (IP)	Not Refined TIP	Refined TIP
PGWZ	63.60 ± 2.65	55.60 ± 2.05	62.00 ± 0.23	79.20 ± 3.25	79.60 ± 3.25	80.00 ± 2.22	**84.40 ± 3.20**
SGWZ	86.00 ± 2.19	87.00 ± 1.79	72.40 ± 0.80	90.00 ± 1.13	92.00 ± 1.79	88.57 ± 1.13	**92.80 ± 1.94**
GPZ1	88.26 ± 1.70	93.02 ± 2.96	68.60 ± 1.10	95.60 ± 0.44	**98.60 ± 0.28**	**98.60 ± 0.28**	97.21 ± 0.44
GPZ2	76.20 ± 2.56	87.22 ± 0.74	69.11 ± 1.23	92.41 ± 1.69	88.61 ± 1.80	88.61 ± 3.61	**95.32 ± 3.42**
GMAD1	69.23 ± 2.68	73.38 ± 0.45	59.85 ± 1.49	87.61 ± 0.65	76.15 ± 0.58	**98.66 ± 0.45**	**98.66 ± 0.41**
GMAD2	63.23 ± 2.48	58.57 ± 2.09	48.15 ± 2.51	69.54 ± 1.84	62.92 ± 1.79	65.85 ± 2.34	**70.62 ± 2.25**
GMAD3	13.72 ± 1.72	42.77 ± 3.43	35.69 ± 0.62	48.15 ± 1.34	37.08 ± 1.75	45.54 ± 2.94	**52.62 ± 2.94**
ADWX	57.63 ± 2.34	61.57 ± 3.05	19.94 ± 0.87	**74.69 ± 0.67**	69.46 ± 0.67	67.03 ± 1.42	69.49 ± 1.42
ADWY	62.57 ± 2.21	66.83 ± 3.79	27.74 ± 0.44	**80.37 ± 0.72**	75.20 ± 0.50	73.36 ± 0.61	75.40 ± 0.72
ADWZ	59.48 ± 1.94	62.37 ± 2.35	23.34 ± 0.47	78.11 ± 0.56	69.86 ± 0.49	81.04 ± 0.98	**84.94 ± 0.94**
PLAID	81.75 ± 1.64	82.16 ± 1.85	82.35 ± 0.25	77.71 ± 0.67	85.52 ± 0.55	82.75 ± 0.67	**86.32 ± 0.69**

Table A7. Comparison to 1D-CNN-based with preprocssing solutions on 11 UCR real-world benchmarks, evaluated by mean average precision (MAP), with standard deviations computed across five seeds. The **highest-performing results** are emphasized in **bold**, while the second-highest are underlined. "NA" indicates that the corresponding method is not applicable on this dataset.

Dataset	FCN (PD)	ResNet (PD)	InceptionTime (TC)	InceptionTime (PD)	InceptionTime (IP)	Not Refined TIP	Refined TIP
PGWZ	48.32 ± 2.67	44.43 ± 15.77	47.75 ± 1.36	68.15 ± 4.13	68.80 ± 4.30	88.57 ± 0.36	**89.10 ± 1.94**
SGWZ	77.21 ± 3.27	82.59 ± 3.04	58.04 ± 0.85	84.09 ± 1.93	86.57 ± 2.03	96.28 ± 2.44	**97.48 ± 1.81**
GPZ1	80.50 ± 2.37	87.81 ± 4.79	55.16 ± 0.85	90.34 ± 0.73	97.42 ± 0.51	98.55 ± 0.41	**99.77 ± 0.14**
GPZ2	63.61 ± 9.93	78.49 ± 1.45	53.97 ± 1.54	87.16 ± 1.09	82.16 ± 1.09	98.66 ± 0.45	**98.68 ± 0.41**
GMAD1	56.29 ± 2.68	61.62 ± 3.50	42.80 ± 1.85	67.82 ± 1.24	62.14 ± 1.00	**86.87 ± 1.82**	**86.87 ± 1.82**
GMAD2	35.29 ± 5.54	41.89 ± 2.31	29.92 ± 2.69	54.60 ± 2.26	45.62 ± 1.79	75.57 ± 1.30	**78.51 ± 0.93**
GMAD3	13.72 ± 5.61	26.18 ± 2.49	18.68 ± 0.32	30.56 ± 1.12	19.69 ± 0.76	51.64 ± 2.40	**58.76 ± 1.20**
ADWX	41.89 ± 1.82	45.78 ± 3.22	12.62 ± 0.33	60.92 ± 0.94	53.80 ± 0.98	73.36 ± 1.07	**75.55 ± 1.95**
ADWY	45.77 ± 3.67	50.33 ± 3.79	15.70 ± 0.17	67.81 ± 0.58	60.15 ± 0.68	78.24 ± 0.90	**81.79 ± 0.43**
ADWZ	41.51 ± 1.16	45.64 ± 1.98	13.42 ± 0.19	64.81 ± 0.68	53.35 ± 0.66	**82.75 ± 1.30**	75.55 ± 1.95
PLAID	65.64 ± 1.93	64.99 ± 2.97	64.47 ± 0.64	77.71 ± 0.67	75.55 ± 1.95	82.75 ± 1.00	**86.32 ± 0.99**

References

1. Bian, J.; Tian, D.; Tang, Y.; Tao, D. Trajectory data classification: A review. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–34. [CrossRef]
2. Norgaard, S.; Saeedi, R.; Gebremedhin, A.H. Multi-sensor time-series classification for activity tracking under variable length. *IEEE Sens. J.* **2019**, *20*, 2701–2709. [CrossRef]
3. Wang, Y.; Perry, M.; Whitlock, D.; Sutherland, J.W. Detecting anomalies in time series data from a manufacturing system using recurrent neural networks. *J. Manuf. Syst.* **2022**, *62*, 823–834. [CrossRef]
4. Farahani, M.A.; McCormick, M.; Harik, R.; Wuest, T. Time-series classification in smart manufacturing systems: An experimental evaluation of state-of-the-art machine learning algorithms. *Robot. Comput. Integr. Manuf.* **2025**, *91*, 102839. [CrossRef]
5. Kahveci, T.; Singh, A. Variable length queries for time series data. In Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, 2–6 April 2001; IEEE: Piscataway, NJ, USA, 2001; pp. 273–282.
6. Schäfer, P. Towards time series classification without human preprocessing. In Proceedings of the International Workshop on Machine Learning and Data Mining in Pattern Recognition, St. Petersburg, Russia, 21–24 July 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 228–242.
7. Tan, C.W.; Petitjean, F.; Keogh, E.; Webb, G.I. Time series classification for varying length series. *arXiv* **2019**, arXiv:1910.04341.
8. Middlehurst, M.; Schäfer, P.; Bagnall, A. Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Min. Knowl. Discov.* **2024**, *38*, 1958–2031. [CrossRef]
9. Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv* **2022**, arXiv:2210.02186.
10. Wang, Y.; Wu, H.; Dong, J.; Liu, Y.; Long, M.; Wang, J. Deep time series models: A comprehensive survey and benchmark. *arXiv* **2024**, arXiv:2407.13278.
11. Shi, Y.; Ying, X.; Yang, J. Deep unsupervised domain adaptation with time series sensor data: A survey. *Sensors* **2022**, *22*, 5507. [CrossRef]

12. Spiegel, S.; Gaebler, J.; Lommatzsch, A.; De Luca, E.; Albayrak, S. Pattern recognition and classification for multivariate time series. In Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data, San Diego, CA, USA, 21 August 2011; pp. 34–42.
13. Tang, W.; Long, G.; Liu, L.; Zhou, T.; Blumenstein, M.; Jiang, J. Omni-scale cnns: A simple and effective kernel size configuration for time series classification. *arXiv* **2020**, arXiv:2002.10061.
14. Mohammadi Foumani, N.; Miller, L.; Tan, C.W.; Webb, G.I.; Forestier, G.; Salehi, M. Deep learning for time series classification and extrinsic regression: A current survey. *ACM Comput. Surv.* **2024**, *56*, 1–45. [CrossRef]
15. Dau, H.A.; Bagnall, A.; Kamgar, K.; Yeh, C.C.M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C.A.; Keogh, E. The UCR time series archive. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 1293–1305. [CrossRef]
16. Lines, J.; Davis, L.M.; Hills, J.; Bagnall, A. A shapelet transform for time series classification. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing China, 12–16 August 2012; pp. 289–297.
17. Schäfer, P. The BOSS is concerned with time series classification in the presence of noise. *Data Min. Knowl. Discov.* **2015**, *29*, 1505–1530. [CrossRef]
18. Keogh, E.; Ratanamahatana, C.A. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.* **2005**, *7*, 358–386. [CrossRef]
19. Bagnall, A.; Lines, J.; Bostrom, A.; Large, J.; Keogh, E. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Min. Knowl. Discov.* **2017**, *31*, 606–660. [CrossRef] [PubMed]
20. Cui, Z.; Chen, W.; Chen, Y. Multi-scale convolutional neural networks for time series classification. *arXiv* **2016**, arXiv:1603.06995.
21. Xiao, Q.; Wu, B.; Zhang, Y.; Liu, S.; Pechenizkiy, M.; Mocanu, E.; Mocanu, D.C. Dynamic sparse network for time series classification: Learning what to "see". *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 16849–16862.
22. Tang, Y.; Xu, J.; Matsumoto, K.; Ono, C. Sequence-to-sequence model with attention for time series classification. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, Spain, 12–15 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 503–510.
23. Ratanamahatana, C.A.; Keogh, E. Three myths about dynamic time warping data mining. In Proceedings of the 2005 SIAM International Conference on Data Mining, Newport Beach, CA, USA, 21–23 April 2005; SIAM: Philadelphia, PA, USA, 2005; pp. 506–510.
24. Rakthanmanon, T.; Campana, B.; Mueen, A.; Batista, G.; Westover, B.; Zhu, Q.; Zakaria, J.; Keogh, E. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Trans. Knowl. Discov. Data* **2013**, *7*, 1–31. [CrossRef]
25. Ismail Fawaz, H.; Lucas, B.; Forestier, G.; Pelletier, C.; Schmidt, D.F.; Weber, J.; Webb, G.I.; Idoumghar, L.; Muller, P.A.; Petitjean, F. Inceptiontime: Finding alexnet for time series classification. *Data Min. Knowl. Discov.* **2020**, *34*, 1936–1962. [CrossRef]
26. Schneider, M.; Greifzu, N.; Wang, L.; Walther, C.; Wenzel, A.; Li, P. An end-to-end machine learning approach with explanation for time series with varying lengths. *Neural Comput. Appl.* **2024**, *36*, 7491–7508. [CrossRef]
27. Sawada, A.; Miyagawa, T.; Ebihara, A.; Yachida, S.; Hosoi, T. Convolutional neural networks for time-dependent classification of variable-length time series. In Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–8.
28. Hatami, N.; Gavet, Y.; Debayle, J. Classification of time-series images using deep convolutional neural networks. In Proceedings of the Tenth International Conference on Machine Vision (ICMV 2017), Vienna, Austria, 13–15 November 2018; SPIE: Bellingham, WA, USA, 2017; Volume 10696, pp. 242–249.
29. Xu, G.; Liu, M.; Jiang, Z.; Shen, W.; Huang, C. Online fault diagnosis method based on transfer convolutional neural networks. *IEEE Trans. Instrum. Meas.* **2019**, *69*, 509–520. [CrossRef]
30. Wang, Z.; Oates, T. Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In Proceedings of the Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
31. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
32. Wang, X.; Mueen, A.; Ding, H.; Trajcevski, G.; Scheuermann, P.; Keogh, E. Experimental comparison of representation methods and distance measures for time series data. *Data Min. Knowl. Discov.* **2013**, *26*, 275–309. [CrossRef]
33. Schäfer, P.; Leser, U. TEASER: early and accurate time series classification. *Data Min. Knowl. Discov.* **2020**, *34*, 1336–1362. [CrossRef]
34. Guillaume, A.; Vrain, C.; Elloumi, W. Random dilated shapelet transform: A new approach for time series shapelets. In Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence, Xiamen, China, 23–25 September 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 653–664.
35. Elman, J.L. Finding structure in time. *Cogn. Sci.* **1990**, *14*, 179–211. [CrossRef]
36. Graves, A.; Schmidhuber, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Netw.* **2005**, *18*, 602–610. [CrossRef]

37. Zerveas, G.; Jayaraman, S.; Patel, D.; Bhamidipaty, A.; Eickhoff, C. A transformer-based framework for multivariate time series representation learning. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 2114–2124.
38. Wang, Z.; Yan, W.; Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1578–1585.
39. Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep learning for time series classification: a review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. [CrossRef]
40. Ismail-Fawaz, A.; Devanne, M.; Berretti, S.; Weber, J.; Forestier, G. Lite: Light inception with boosting techniques for time series classification. In Proceedings of the 2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA), Thessaloniki, Greece, 9–13 October 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–10.
41. Dempster, A.; Schmidt, D.F.; Webb, G.I. Hydra: Competing convolutional kernels for fast and accurate time series classification. *Data Min. Knowl. Discov.* **2023**, *37*, 1779–1805. [CrossRef]
42. Ismail-Fawaz, A.; Devanne, M.; Weber, J.; Forestier, G. Deep learning for time series classification using new hand-crafted convolution filters. In Proceedings of the 2022 IEEE International Conference on Big Data, Osaka, Japan, 17–20 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 972–981.
43. Middlehurst, M.; Large, J.; Flynn, M.; Lines, J.; Bostrom, A.; Bagnall, A. HIVE-COTE 2.0: a new meta ensemble for time series classification. *Mach. Learn.* **2021**, *110*, 3211–3243. [CrossRef]
44. Frank, A. UCI Machine Learning Repository. 2010. Available online: http://archive.ics.uci.edu/ml (accessed on 10 July 2024).
45. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.