electronics

Article

CEEMD-MultiRocket: Integrating CEEMD with Improved MultiRocket for Time Series Classification

Panjie Wang, Jiang Wu, Yuan Wei and Taiyong Li *

School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu 611130, China
* Correspondence: litaiyong@gmail.com

Abstract: Time series classification (TSC) is always a very important research topic in many real-world application domains. MultiRocket has been shown to be an efficient approach for TSC, by adding multiple pooling operators and a first-order difference transformation. To classify time series with higher accuracy, this study proposes a hybrid ensemble learning algorithm combining Complementary Ensemble Empirical Mode Decomposition (CEEMD) with improved MultiRocket, namely CEEMD-MultiRocket. Firstly, we utilize the decomposition method CEEMD to decompose raw time series into three sub-series: two Intrinsic Mode Functions (IMFs) and one residue. Then, the selection of these decomposed sub-series is executed on the known training set by comparing the classification accuracy of each IMF with that of raw time series using a given threshold. Finally, we optimize convolution kernels and pooling operators, and apply our improved MultiRocket to the raw time series, the selected decomposed sub-series and the first-order difference of the raw time series to generate the final classification results. Experiments were conducted on 109 datasets from the UCR time series repository to assess the classification performance of our CEEMD-MultiRocket. The extensive experimental results demonstrate that our CEEMD-MultiRocket has the second-best average rank on classification accuracy against a spread of the state-of-the-art (SOTA) TSC models. Specifically, CEEMD-MultiRocket is significantly more accurate than MultiRocket even though it requires a relatively long time, and is competitive with the currently most accurate model, HIVE-COTE 2.0, only with 1.4% of the computing load of the latter.

Keywords: time series classification; complementary ensemble empirical mode decomposition (CEEMD); MultiRocket; feature selection; hybrid model

check for updates

Citation: Wang, P.; Wu, J.; Wei, Y.; Li, T. CEEMD-MultiRocket: Integrating CEEMD with Improved MultiRocket for Time Series Classification CEEMD-MultiRocket: Integrating CEEMD with Improved MultiRocket for Time Series Classification. *Electronics* **2023**, *12*, 1188. https://doi.org/10.3390/electronics12051188

Academic Editor: Daniel Gutiérrez Reina

Received: 30 January 2023
Revised: 23 February 2023
Accepted: 28 February 2023
Published: 1 March 2023

1. Introduction

A time series is a set of data arranged in chronological order, which is widely applied in different domains in real life. With the fast advancement of information acquisition equipments and improvement of acquisition methods, time series have gotten more sophisticated, and their application involves a wide variety of fields, such as traffic [1], energy [2,3], finance [4], medical diagnosis [5–7] and social media [8]. By classifying time series into groups based on their underlying stochastic process, we can gain insights into the underlying phenomenon being measured and potentially make predictions. This involves identifying features in the time series data that are indicative of the underlying process, such as the autocorrelation structure, the distribution of values, or the frequency spectrum. Therefore, time series classification (TSC), as a task of characterizing a series of values observed at a continuous time as belonging to one of two or more categories, has always been the focus of research [9].

Several TSC algorithms have been presented over the years. These algorithms are generally separated into traditional approaches and deep learning approaches. The main groups of traditional TSC algorithms are introduced as follows: (1) Distance-based classifiers use distance metrics to determine class membership, and their representatives include a combination of K-Nearest Neighbors (KNN) and Dynamic Time Warping (DTW) [10] and Proximity Forest [11]. (2) Frequency-based classifiers are based on frequency data extracted from time series, and their representative is Random Interval Spectral Ensemble (RISE) [12], which is viewed as a popular Time Series Forest (TSF) [13] variation. (3) Interval-based classifiers rely their classification on information contained in distinct series intervals, and their representatives include TSF and Diverse representation Canonical Internal Forest (DrCIF) [14]. DrCIF builds on RISE and TSF, and uses the catch22 [15] to expand the original features. (4) Dictionary-based classifiers first convert discrete "words" from real-valued time series. The distribution of the retrieved symbolic terms is used as the basis for classification. Their representatives include Bag of Symbolic-Fourier-Approximation Symbols (BOSS) [16] and Temporal Dictionary Ensemble (TDE) [17]. (5) Shapelets are short subsequences of time series that are typical of their class. It is possible to utilize them to discover the similarity between two time series belonging to the same class [18]. Their representatives include Shapelet Transformation (ST) [19] and Shapelet Transform Classifier (STC) [20].

An ensemble classifier is a meta ensemble based on the previously described classifiers, and the typical representatives include HIVE-COTE [21], HIVE-COTE 2.0 [22], InceptionTime [23] and Time Series Combination of Heterogeneous and Integrated Embedding Forest (TS-CHIEF) [24]. HIVE-COTE 2.0 is a meta ensemble consisting of four components: STC, TDE, DrCIF and Arsenal [22]. InceptionTime is a collection of five TSC deep learning models generated by cascading numerous inception modules [23]. Each model has the same design but distinct random initialization weight values. TS-CHIEF builds on an ensemble tree-structured classifier that incorporates the most efficient time series embeddings created in the previous ten years of study [24].

On the other hand, deep learning methods for TSC are generally classified into two types: generative models and discriminative models [25].

The most common generative models include Stacked Denoising Auto-Encoders (SDAE) [26,27] and Echo State Networks (ESN) [28]. To model the time series, SDAE is preceded by an unsupervised pre-training stage [26,27]. As Recurrent Neural Networks (RNN) frequently experience the vanishing gradient problem as a result of training on lengthy time series [29], ESNs were created to ameliorate the difficulties of RNNs [30]. Discriminative models are classifiers that can quickly figure out how to transfer a time series' original input to a dataset's output, which is a probability distribution over the class variables. These models may be further classified into two types: (1) deep learning models using hand-engineered features and (2) end-to-end deep learning models [30]. The translation of series into images utilizing specialized imaging approaches is the most common feature extraction algorithm for hand-engineered approaches, such as recurrence plots [31,32] and Gramian fields [33]. In contrast, end-to-end deep learning tries to include the feature learning procedure while optimizing the discriminative classifier [34]. Convolutional Neural Networks (CNN) are the most extensively used for the TSC issue due to their robustness and training efficiency [30].

Overall, the state-of-the-art (SOTA) TSC models in terms of classification accuracy mainly include HIVE-COTE and its variants, TS-CHIEF, InceptionTime, Rocket, MiniRocket, MultiRocket, etc. [35]. Among them, Rocket, MiniRocket, and MultiRocket are not only accurate, but also ensure scalability. Rocket employs lots of randomly initialized convolution kernels for feature extraction, and uses a linear classifier for classification, without training the kernels [36]. MiniRocket is about 75 times faster than Rocket, and it employs a limited number of kernels and just one pooling operation [37]. MultiRocket is built on MiniRocket and uses the same set of convolution kernels that are used in MiniRocket [35]. MultiRocket differs in two ways from MiniRocket. On one hand, MultiRocket uses the first-order difference of raw time series, along with the raw time series, as the inputs to the classification model. On the other hand, MultiRocket includes three extra pooling operators in addition to PPV to derive more discriminative features.

Although Rocket and its improved versions MiniRocket and MultiRocket have achieved satisfactory classification performance, there is certainly room for improvement in series transform, the design of convolution kernels and feature extraction. To solve the existing defects and enhance classification performance, this study proposes a novel hybrid ensemble leaning model incorporating Complementary Ensemble Empirical Mode Decomposition (CEEMD) and improved MultiRocket, namely CEEMD-MultiRocket, to enhance the classification performance of time series. Raw time series is firstly divided into three sub-series utilizing CEEMD [38–40]. The sub-series refer to the individual Intrinsic Mode Functions (IMFs) that make up the decomposition of the raw time series into its oscillatory components. Since the decomposition is performed using a sifting process that extracts the highest frequency component first and continues with lower frequency components until the residual is obtained, these three sub-series represent high-, medium- and low-frequency portions of the original time series, respectively. Since not every decomposed sub-series as the input has a positive contribution to the performance of the classification model, the selection of the more crucial sub-series and pruning the redundant and less important ones are necessary to enhance the final classification performance and reduce computational complexity. The selection of these decomposed sub-series is executed on the known training set by comparing the classification accuracy of each sub-series with that of the raw time series using a given threshold. Finally, we improve the original MultiRocket and apply it to the raw time series and the selected decomposed sub-series to derive features and generate the final classification results. In improved MultiRocket, the convolution kernels are modified, and one additional pooling operator is applied to convolution outputs. CEEMD-MultiRocket has been empirically tested with 109 datasets from the UCR time series repository. Compared with some SOTA classification models, the experiments demonstrate that our proposed CEEMD-MultiRocket achieves promising classification performance. Specifically, our proposed CEEMD-MultiRocket is more accurate than MultiRocket even though it takes a relatively long time, and is competitive with the HIVE-COTE 2.0 which ranks the best at present in terms of classification accuracy, only with a small fraction of the training time of the latter. One of the main theoretical and technical implications of CEEMD-MultiRocket is that it is the first time that CEEMD has been integrated with convolution kernel transform for the feature extraction of time series, making it outperform almost all of the previous SOTA methods. Furthermore, CEEMD-MultiRocket improves convolution kernel and pooling operator design and is demonstrated to be a fast, effective and scalable method for time series classification tasks, showing that the optimization of convolution kernels and pooling operator is a promising field worth studying for improving classification performance. The main contributions of this research lie in five aspects:

(1) A novel hybrid TSC model that integrates CEEMD and improved MultiRocket is proposed. Raw time series is decomposed into high-, medium- and low-frequency portions, and convolution kernel transform is utilized to derive features from the raw time series, the decomposed sub-series and the first-order difference of raw time series. This kind of transformation is able to obtain more detailed and discriminative information of time series from various aspects.

(2) A sub-series selection method is proposed based on the whole known training data. This method selects the more crucial sub-series and prunes the redundant and less important ones, which helps to further enhance classification performance and also reduce computational complexity.

(3) The length and number of convolution kernels are modified, and one additional pooling operator is applied to convolutional outputs in our improved MultiRocket. These improvements contribute to the enhancement of classification accuracy.

(4) Extensive experiments demonstrate that the proposed classification algorithm is more accurate than most SOTA algorithms for TSC.

(5) We further analyze some characteristics of the proposed CEEMD-MultiRocket for TSC, including the CEEMD parameter settings, the selection of decomposed sub-series, the design of convolution kernel and pooling operators.

The rest of this paper is organized as follows. Section 2 briefly introduces CEEMD and MultiRocket. Section 3 gives the description of the proposed CEEMD-MultiRocket algorithm in detail, including CEEMD and sub-series selection, improved MultiRocket and feature extraction. Section 4 reports experimental results and assesses the proposed algorithm in terms of accuracy and training time. Section 5 discusses the impact of the CEEMD parameters, the threshold setting for sub-series selection, the convolution kernel length and an additional pooling operator on the classification performance of CEEMD-MultiRocket, followed by conclusions in Section 6.

2. Related Works

2.1. Complementary Ensemble Empirical Mode Decomposition

CEEMD [38] is an extension built on Ensemble Empirical Mode Decomposition (EEMD) [41] and Empirical Mode Decomposition (EMD) [42]. EMD is a time-frequency analysis approach which is created for nonlinear and nonstationary signals or time series [43]. EMD applies local extreme points of the raw time series to form the envelope step by step, separates fluctuations or trends at diverse scales and generates a group of relatively stable components, including IMFs and one residue. Specifically, the EMD algorithm involves iteratively extracting local oscillations from the signal by means of a sifting process. The extracted oscillations are called IMFs, and they represent the underlying oscillatory modes that make up the signal. The remaining signal after extracting the IMFs is called the residue, which contains the trends and other non-oscillatory components. The main disadvantage of mode mixing in EMD is that the significantly diverse scales may appear in the same IMF component [44]. To reduce mode mixing, EEMD was proposed [41]. In EEMD, IMFs are defined as a combination of time series and white noise with a limited amplitude, which can significantly reduce mode mixing. Despite the fact that EEMD has effectively handled the mode mixing problem, the residual noise in signal reconstruction has increased. Therefore, CEEMD was proposed, where a specific type of white noise was introduced at each stage of the decomposition [45]. It not only suppresses the mode mixing but also reduces the reconstruction signal errors caused by residue noise. The CEEMD is described as follows:

(1) Add two equal-amplitude, opposite-phase white noises to the signal $x(t)$, to obtain the following sequences.

$$\begin{cases} P_i(t) = x(t) + n_i(t) \\ N_i(t) = x(t) - n_i(t) \end{cases} \tag{1}$$

where $n_i(t)$ is the white noise superimposed in the ith stage, $P_i(t)$ and $N_i(t)$ denote the sequence after adding noise in the ith stage .

(2) CEEMD firstly breaks down the sequence's noise to generate the components IMF, C_{1_j} and the trend surplus r_1.

(3) In the same way, process the white noise with opposing symbols in step (1) to generate the components C_{-1_j} and r_{-1}.

(4) Repeat steps (1)∼(3) n times to obtain n sets.

(5) The ultimate result is chosen as the average of the components of two sets of residual positive and negative white noise acquired by repeated decomposition, i.e.,

$$\begin{cases} C_i(t) = \frac{1}{2n}\sum_{j=1}^{n}(C_{n_j} + C_{-n_j}) \\ r_n(t) = \frac{1}{2n}\sum_{j=1}^{n}(r_j + r_{-j}) \end{cases} \tag{2}$$

2.2. MultiRocket

Rocket employs lots of randomly initialized convolution kernels for transform, applies pooling operators to convolutional outputs and uses a linear classifier, without training the kernels [36]. For Rocket, a time series is convolved using 10 k random convolution kernels, whose weights are sampled from $N(-1,1)$; length is selected from $\{7,9,11\}$ with equal probability; padding is alternating; dilation is exponentially scaled; and bias is sampled from $U(-1,1)$. Additionally, the Proportion of Positive Values (PPV) and global max pooling (Max) pooling operators are applied to each convolutional output to generate two features, and to generate 20 K features in total for each input series. Finally, for a larger dataset, the derived features are employed to train a logistic regression classifier, while for a relatively small dataset, a ridge regression classifier is trained. Rocket has been proved to be a efficient, fast and novel algorithm for the feature extraction of time series [36].

MiniRocket is built on Rocket and becomes further deterministic by pre-defining a set of convolution kernels with fixed lengths and weights. MiniRocket retains the dilation and PPV, while it discards the max pooling which is of no benefit for enhancing the classification accuracy [37]. It performs a convolution operation on the input series using a fixed group of 84 kernels with each kernel generating multi-dilation (74 by default) and using different bias which are obtained by sampling on the convolutional output from a randomly selected instance in the training set. Since only PPV is used in MiniRocket, the number of features ($84 \times 119 = 9996$ by default) generated by MiniRocket is only about half of the number of features generated by Rocket.

The kernels used in MultiRocket are the same as MiniRocket. Unlike MiniRocket, MultiRocket injects the diversity of features by adding the first-order difference of raw time series and three additional pool operators to enhance the performance of MiniRocket. Inspired by DrCIF, MultiRocket uses the first-order difference of raw time series as the input to offer more diverse information related to the transformation of raw time series. MultiRocket has 84 fixed convolution kernels and each convolutional kernel will produce 74 kinds of dilation. Firstly, MultiRocket performs a convolution operation on the input series and the first-order difference of the input series using the kernels with dilations to obtain the convolutional outputs. Next, four features (PPV and an additional three pooling operators) are calculated for each convolutional output and then about 50 k (more accurately, $84 \times 74 \times 2 \times 4 = 49{,}728$) features are generated. Finally, a linear regression classifier is trained on the features. MultiRocket is faster than all TSC algorithms (except for MiniRocket) and more accurate than all TSC algorithms (except for HIVE-COTE 2.0) [35].

In summary, Rocket, MiniRocket and MultiRocket are representations of the scalable and most accurate algorithms on the UCR time series repository. As a series of algorithms, their differences can be seen in Table 1.

Table 1. Summary of changes from Rocket to MiniRocket and then to MultiRocket.

	Rocket	**MiniRocket**	**MultiRocket**
kernel length	$7, 9, 11$	9	9
weights	$N(0,1)$	$-1, 2$	$-1, 2$
bias	$U(-1,1)$	from convolutional output	from convolutional output
dilation	random	fixed (range $\lfloor 2^0 \rfloor, \cdots, \lfloor 2^{max} \rfloor$)	fixed (range $\lfloor 2^0 \rfloor, \cdots, \lfloor 2^{max} \rfloor$)
padding	random	fixed	fixed
pooling operators	PPV, MAX	PPV	PPV, MPV, MIPV, LSPV
num. features	20 K	10 K	50 K

3. The Proposed CEEMD-MultiRocket

This research proposes a hybrid ensemble model that combines CEEMD and improved MultiRocket, termed CEEMD-MultiRocket, for TSC. The proposed model includes three steps which are decomposition, sub-series selection and feature extraction and classification, as demonstrated in Figure 1.

Step 1: Decomposition. Each time series in a dataset is decomposed into three sub-series using CEEMD: IMF_i (i = 1, 2) and one residual.

Step 2: Sub-series selection. In order to enhance the final classification accuracy and decrease computational load, the selection of these decomposed sub-series is executed on the whole known training dataset by comparing the classification accuracy of each decomposed sub-series with that of the raw time series using a pre-set threshold.

Step 3: Feature extraction and classification.The convolution kernel transform is applied to the raw time series, the selected sub-series and the first-order difference of raw series. Then, five pooling operators are designed to extract features from the convolutional output. Finally, a ridge regression classifier is trained using these extracted features. In our improved MultiRocket, the length and number of convolution kernels are modified, and one additional pooling operator is applied to the convolutional output.

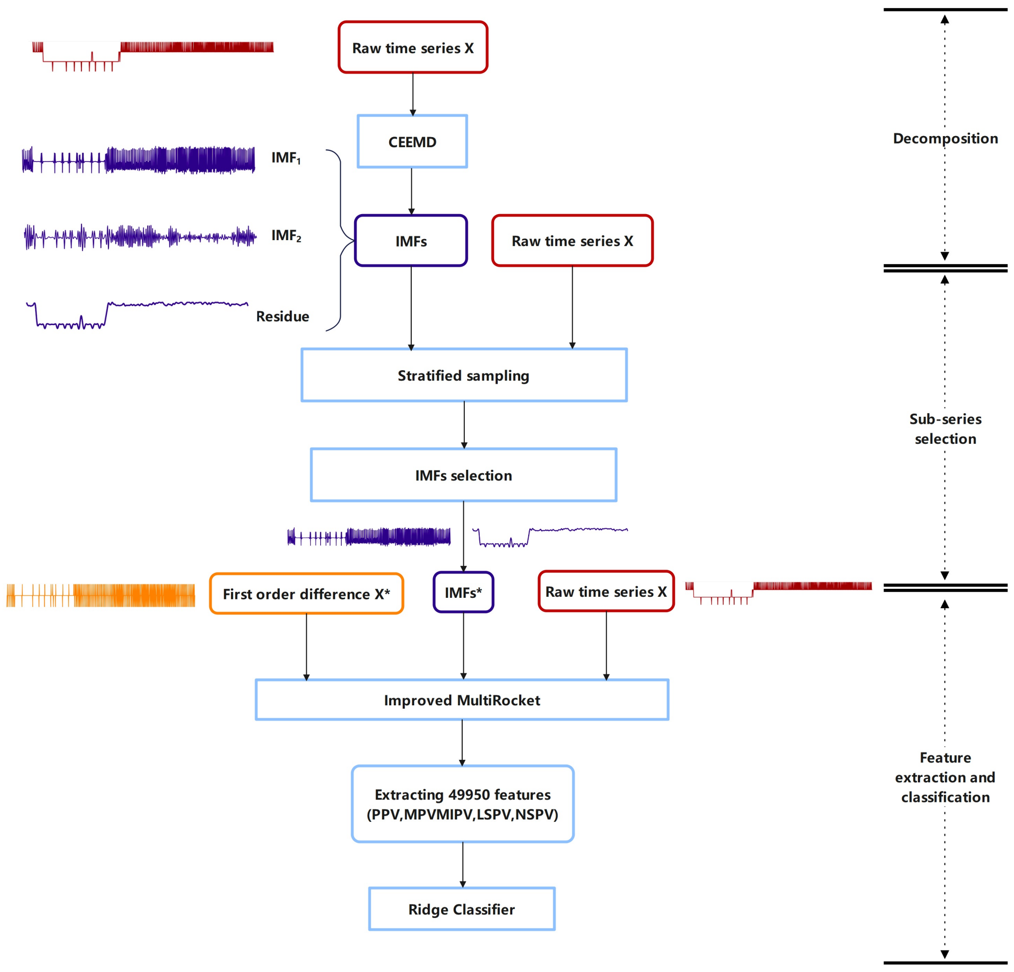

Figure 1. The flowchart of the proposed CEEMD-MultiRocket.

Firstly, the proposed CEEMD-MultiRocket applies CEEMD to decompose raw time series into three sub-series (two IMFs and one residue), each of which contains information about the different frequency of raw time series. In general, the first and second IMF represent the high- and medium-frequency portions and the residue represents the low-frequency portion of raw time series. Secondly, in order to enhance the final classification performance and decrease computational complexity, it is necessary to select the most crucial sub-series and discard less important ones. The selection of these sub-series is executed on the whole known training set which is further subdivided into training and testing sets using stratified sampling. Improved MultiRocket is used for the raw time series and each sub-series on the newly generated training and testing sets, and the appropriate sub-series is selected when its testing accuracy is higher than a given threshold, which is set to a percentage of the testing accuracy of raw time series. Finally, convolution operation is performed on the raw time series, the selected sub-series and the first-order difference of raw time series, respectively. It should be specially noted that the transform is only applied to the raw time series and its first-order difference when there is a dataset without any selected sub-series. Feature extraction is conducted on each convolutional output, and these extracted features are eventually applied to train a ridge regression classifier. In improved MultiRocket, the length and number of convolution kernels are modified, and five pooling operators are used in each convolutional output to derive features. The combination of these modifications has the potential to enhance the classification performance of MultiRocket. Overall, this hybrid ensemble learning paradigm, CEEMD-MultiRocket, can diversify the input series and comprehensively extract more extensive features from the raw series and the decomposed sub-series for classification, which makes it possible to enhance classification performance.

3.1. CEEMD and Sub-Series Selection

The CEEMD algorithm is usually applied in the field of signal processing, which decomposes raw time series into several components to obtain better classification performance [46]. The proposed CEEMD-MultiRocket firstly uses the CEEMD decomposition algorithm to decompose raw time series into two IMFs and a residue. Figure 2 illustrates a decomposition of a time series from the electricity consumption dataset ScreenType from the UCR repository [47] using CEEMD. The length of each series in the ScreenType dataset is 720 (24 h of readings taken every 2 min). The x-axis represents the time (every 2 min) and the y-axis represents the electricity consumption in Figure 2.

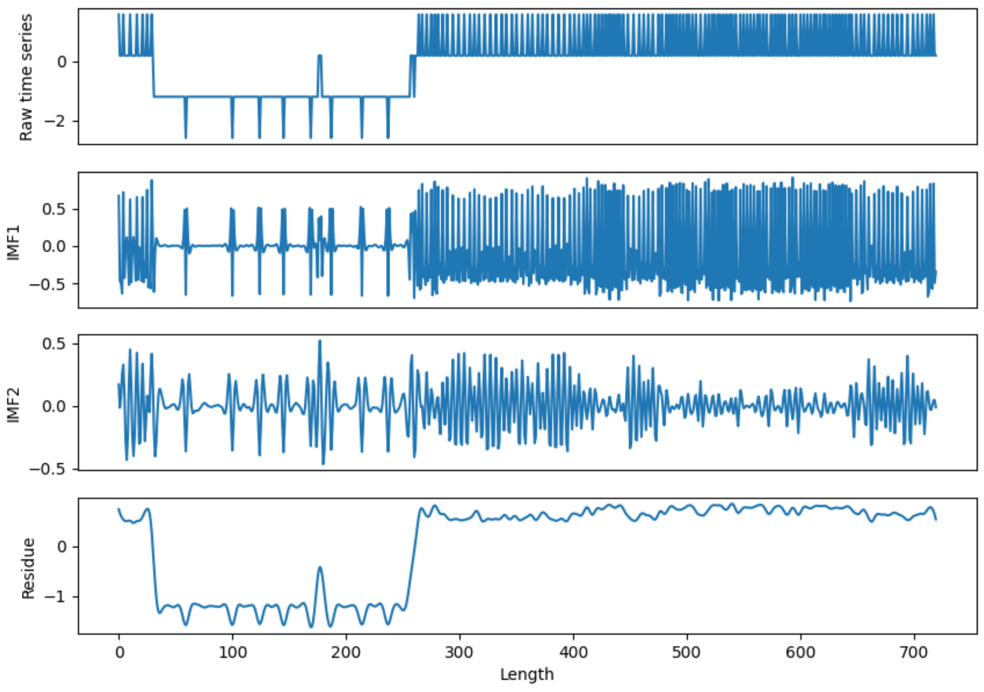

Figure 2. A raw time series and its corresponding sub-series decomposed by CEEMD in the ScreenType dataset.

It is a challenging issue to select the appropriate sub-series for extracting discriminative characteristics of the raw time series for time series analysis correctly [48]. To select the appropriate sub-series (two IMFs or one residue) as the inputs to our classification model, we propose a novel approach to select the appropriate sub-series generated by CEEMD using the known training data. The main idea is to subdivide the original training dataset into two parts including a new training dataset and a new testing dataset, then train a classification model using improved MultiRocket and obtain the testing accuracy, and finally select the sub-series with satisfactory testing accuracies. This kind of selection approach is based on the inference that the sub-series with relatively high testing accuracy may contain more potentially useful characteristics as the input to improve MultiRocket. The decomposition and sub-series selection are described as follows:

(1) Utilize CEEMD to decompose raw time series into two IMFs and a residue. For convenience, we refer to three of them as IMFs.
(2) Perform stratified sampling to subdivide the original training set and its corresponding three IMFs into new training sets and testing sets, respectively, and ensure that the new training and testing sets contain all labels (the split ratio is 1:1).
(3) Apply improved MultiRocket to the newly generated training set of original training time series, and then obtain the testing classification accuracy $acc_{original}$ on the newly generated testing set. Perform the corresponding operations for each IMF and obtain the testing classification accuracy $acc_i, i = 1, 2, 3$.
(4) Select the IMFs whose testing accuracies acc_i are more than $acc_{original} \times threshold$ as the inputs of improved MultiRocket. We refer to the selected IMFs as $IMFs^*$ throughout the paper, which may contain 0–3 sub-series generated by CEEMD. The threshold is set to 0.9 by default.

By comparing the testing accuracy of each sub-series with that of raw time series, we can select the most crucial sub-series and discard the redundant and less important ones as the inputs to classification model, thereby enhancing classification performance and reducing computational cost.

3.2. Improved MultiRocket

This section provides a comprehensive explanation of improved MultiRocket which retains the basic architecture as the original MultiRocket [35]. The main difference between the improved MultiRocket and the original MultiRocket lies in two aspects. The first is the modification of convolution kernel length, and the second involves an addition pooling operator for feature extraction. We expect that these modifications are able to significantly enhance classification ability. The comparison of original MultiRocket and improved MultiRocket is listed in Table 2.

Table 2. Comparison of MultiRocket and improved MultiRocket.

	MultiRocket	**Improved MultiRocket**
kernel length	9	6
num. kernels	84	15
weights	$-1, 2$	$-1, 2$
bias	from convolutional output	from convolutional output
dilation	fixed (range $\lfloor 2^0 \rfloor, \cdots, \lfloor 2^{max} \rfloor$)	fixed (range $\lfloor 2^0 \rfloor, \cdots, \lfloor 2^{max} \rfloor$)
padding	fixed	fixed
pooling operators	PPV, MPV, MIPV, LSPV	PPV, MPV, MIPV, LSPV, NSPV
num. features	50 K	50 K

3.2.1. Convolution Kernels

Improved MultiRocket employs 15 fixed convolution kernels with length 6 and has fixed weights. Except for the length of convolution kernel, the dilation, the padding and bias of the improved MultiRocket are the same as those of MultiRocket. The detailed kernel design, dilation, bias and padding are described as follows.

- Kernel length and weight setting: To simplify the computation complexity as much as possible, the number of convolution kernels ought to be as small as possible [37]. Therefore, our proposed CEEMD-MultiRocket tries to employ 15 convolution kernels with length 6 instead of the 84 kernels with length 9 in the original MultiRocket. The convolution kernel weights are restricted to two values, α and β, and there are $2^6 = 64$ possible dual-valued kernels with a length of 6. Improved MultiRocket employs the subset of convolution kernels that have two values of β, and this provides a total of $C_6^2 = 15$ fixed kernels, which strikes a good balance between computing efficiency and classification accuracy. In the improved MultiRocket, we set the weight $\alpha = -1$ and $\beta = 2$. As long as α and β increase by multiples, equivalently, that is $\beta = -2\alpha$, it has no effect on the results, because bias and features are extracted from the output of convolution [37]. Since the original MultiRocket uses 84 kernels with length 9, the number of kernels used in our improved MultiRocket is less than a fifth of the number of kernels in the original MultiRocket, effectively decreasing computing load.

- Dilation: The dilations used by each kernel are the same and fixed. The total number of dilations of each kernel n depends on the number of features, where $n = f/3/15$ and f represent the total number of features (50 k by default). Dilations are specified in range $\{\lfloor 2^0 \rfloor, \ldots, \lfloor 2^{\max} \rfloor\}$, where the exponent obeys a uniform distribution between 0 and max = min $(\frac{l_{input}-1}{l_{kernel}-1}, 64)$, where l_{kernel} is the length of the kernel and l_{input} is the length of input time series.
- Bias: Bias values are determined by the convolutional outputs for each kernel/dilation combination. For a kernel/dilation combination, we randomly select a training example and calculate its convolutional output, then sample from the convolutional output based on many quantiles to obtain bias, in which the quantiles are drawn from a low-discrepancy sequence.
- Padding: Each combination of kernel and dilation alternates between using and not using padding, with half of the combinations using padding.

We refer to the feature vector extracted by the convolution operation as Z and the length of the input time series as l. According to [36], the result of applying a kernel to a time series, X, from index i in X can be obtained using Equation (3):

$$Z = X_i \times w = b + (\sum_{j=0}^{l_{kernel}-1} X_{i+(j\times d)} \times w_j) \tag{3}$$

where ω is weights, d is dilation and b is bias of the kernel.

3.2.2. Pooling Operators

MultiRocket injects diversity through two main aspects: the first-order difference of raw time series and an additional three pooling operators. In order to enhance the diversity of derived features, we propose an additional pooling operator, Number of Stretch of Positive Values (NSPV), to extract more comprehensive features from the convolutional output. Thus, we employ five pooling operators together to derive features, including the four existing pooling operators used in the original MultiRocket [35]. Table 3 summarizes the pooling operators in the improved MultiRocket, including Proportion of Positive Values (PPV), Mean of Positive Values (MPV), Mean of Indices of Positive Values (MIPV), Longest Stretch of Positive Values (LSPV) and NSPV.

Table 3. The summary of pooling operators in improved MultiRocket uses a virtual example to illustrate that the four pooling operators in original MultiRocket cannot distinguish different scenarios with different convolutional outputs. Each convolutional output contains 6 zeros and 6 ones, MPV = 1, PPV = 0.5, MIPV = 5.5, LSPV = 2.

Convolutional Outputs	PPV	MPV	MIPV	LSPV	NSPV
A = [1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1]	0.5	1	5.5	2	3
B = [1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1]	0.5	1	5.5	2	1
C = [1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1]	0.5	1	5.5	2	2

PPV was first used in Rocket. It uses Equation (4) to compute the proportion of positive value of Z.

$$\mathrm{PPV(Z)} = \frac{1}{l}\sum_{i=1}^{l}[z_i > 0] \tag{4}$$

MPV, MIPV and LSPV were used in MultiRocket. The MPV value is calculated using Equation (5), where m is the number of positive values in Z.

$$\text{MPV}(\text{Z}) = \frac{\sum_{l}^{i=1} z_i [z_i > 0]}{m} \tag{5}$$

MIPV is calculated by Equation (6), where i^+ is the index of positive value and m is the number of positive values in Z.

$$\text{MIPV}(\text{Z}) = \begin{cases} -1 & \text{otherwise} \\ \frac{1}{m}\sum_{j=1}^{m} i_{\text{j}}^{+} & \text{if } m > 0 \end{cases} \tag{6}$$

LSPV is calculated using Equation (7) and represents the maximum length of any subsequence of positive values in Z.

$$\text{LSPV}(\text{Z}) = \max\left[j - i \mid \forall_{i \le k \le j}\, z_k > 0\right] \tag{7}$$

We propose NSPV to calculate the number of continuous subsequences with positive values in Z, as defined in Equation (8). It can offer a distinctive kind of information compared with the other four pooling operators provided in the original MultiRocket. As shown in Table 3, NSPV is the key to distinguishing between three time series, A to C.

$$\text{NSPV}(\text{Z}) = \sum_{k=1}^{n}\left[j - i > 1 \mid \forall_{i \le k \le j}\, z_k > 0\right] \tag{8}$$

3.3. Feature Extraction

Original MultiRocket produces 50 k features by default. For a fair comparison, the improved MultiRocket extracts five aggregate features from each convolutional output and also generates about 50 k (more accurately, $15 \times 222 \times 3 \times 5$ = 49,950) features for each time series by default. Specifically, it has 15 fixed convolution kernels and each convolution kernel produces 222 kinds of dilation, making the length of the convolution kernel from 6 to the input time series' length. The input data consists of three parts: (1) the raw time series; (2) the selected $IMFs^*$; and (3) the first-order difference of raw time series. Firstly, the three parts as the inputs are convolved by each combination of kernel and dilation in turn to obtain the convolutional output. Next, a total of 49,950 features are eventually derived by calculating five features for each convolutional output using five pooling operators. Finally, a ridge regression classifier is trained on the extracted features.

4. Experimental Results

4.1. Datasets

To better assess the performance of the proposed CEEMD-MultiRocket, the experiments were conducted on 109 univariate time series classification datasets from the UCR time series repository [47], which includes datasets from many different fields and has been used to evaluate various TSC models.

4.2. Experimental Settings

From the perspective of classification accuracy and runtime analysis, the proposed CEEMD-MultiRocket was contrasted with some SOTA algorithms for classifying time series, including MultiRocket, HIVE-COTE 2.0, InceptionTime, MiniRocket, Arsenal, STC, TS-CHIEF, DrCIF, TDE and ProximityForest. In order to directly compare the classification accuracy with other most accurate algorithms as mentioned above, we assessed the proposed CEEMD-MultiRocket on 30 resamples of 109 datasets from the UCR univariate time series repository used in [22,35] and adopted exactly the same resampling method as used in MultiRocket [35]. Thus, on the basis of the same data sets and stratified split, we examined the effectiveness of our proposed CEEMD-MultiRocket in enhancing classification accuracy, and compared the runtime of CEEMD-MultiRocket with that of the above time series classification algorithms.

In addition, the noise standard deviation was set to 0.4 and the number of realizations was set to 30 in CEEMD. The threshold of sub-series selection was set to 0.9 of the testing accuracy of raw time series. We performed CEEMD using MATLAB R2016a and improved MultiRocket using pycharm IDE on a cluster with an Intel Xeon Gold 5218 CPU @2.30 GHz using a single thread.

4.3. Results and Analysis

4.3.1. Classification Results

We compared CEEMD-MultiRocket with the 10 SOTA TSC algorithms mentioned above to examine the effectiveness of our proposed method. Figure 3 illustrates the mean rank of CEEMD-MultiRocket in comparison to the 10 TSC algorithms. According to a two-sided Wilcoxon signed-rank test with Holm correction (as a post hoc test to the Friedman test), algorithms connected with a black line have no pairwise statistical difference in their accuracy [49]. The Wilcoxon signed-rank test is a nonparametric statistical hypothesis test used to determine if two related samples have the same distribution, which is often used to compare the significance of the differences between two related samples. From Figure 3, we can see that CEEMD-MultiRocket is significantly more accurate than MultiRocket and other TSC algorithms, and only marginally less accurate than HIVE-COTE 2.0. Note that the accuracy difference between CEEMD-MultiRocket and HIVE-COTE 2.0 is statistically insignificant, showing that CEEMD-MultiRocket achieves almost the same level of classification performance as the latter.

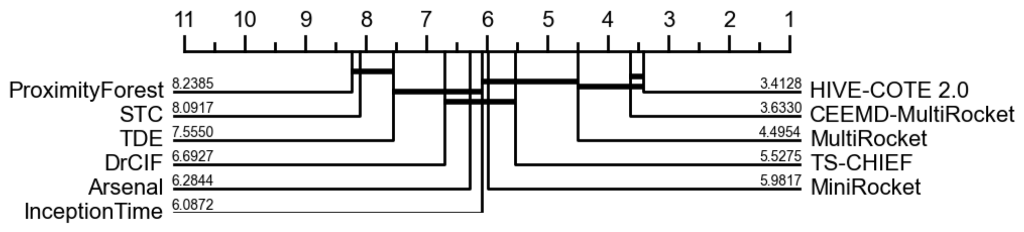

Figure 3. Mean rank of CEEMD-MultiRocket in terms of accuracy over 30 resamples of 109 datasets from the UCR time series repository, against 10 other SOTA algorithms.

Figure 4 shows the pairwise difference of the CEEMD-MultiRocket and 10 other SOTA TSC algorithms in terms of statistical significance. The first row of each cell in the matrix indicates the wins, draws and losses of the algorithm in the Y-axis versus the algorithm in the X-axis, and the second row shows the p-value of the Holm-corrected two-sided Wilcoxon signed-rank test between the pairwise algorithms. The bold numbers in the cells represent that significant differences do not exist in the classification accuracy of the pairwise algorithms after applying the Holm correction. As shown in Figure 4, our proposed CEEMD-MultiRocket is significantly more accurate than all SOTA classification algorithms except for HIVE-COTE 2.0, where the p-values for most of algorithms are close to 0. CEEMD-MultiRocket outperforms MultiRocket with 70 wins and only 31 losses out of 109 datasets. Compared with HIVE-COTE 2.0, CEEMD-MultiRocket achieves higher accuracy on 51 datasets, lower on 50 datasets and is the only algorithm with a p-value close to 1 after applying Holm correction.

As we can see, there are many different SOTA algorithms that can be used for time series classification, and the suitability of a particular algorithm depends on various factors such as the length of the time series, the sampling frequency, the number of classes and the complexity of underlying patterns. In general, if the time series are very short, with only a few data points, then simpler algorithms, such as nearest neighbor or decision trees, may be more appropriate. These algorithms can be effective for small datasets and can quickly classify time series based on their similarity to other time series in the training set.

On the other hand, if the time series are very long and have a high sampling frequency, then more complex algorithms, such as RNN or CNNs, may be more suitable. Through the experiments on 109 datasets, we find that our CEEMD-MultiRocket algorithm performs well in classification and outperforms the vast majority of existing classification algorithms. Among these 109 datasets, the shortest time series length is 15 (SmoothSubspace), and the longest is 2844 (Rock), indicating that our algorithm is effective for both short and long time series.

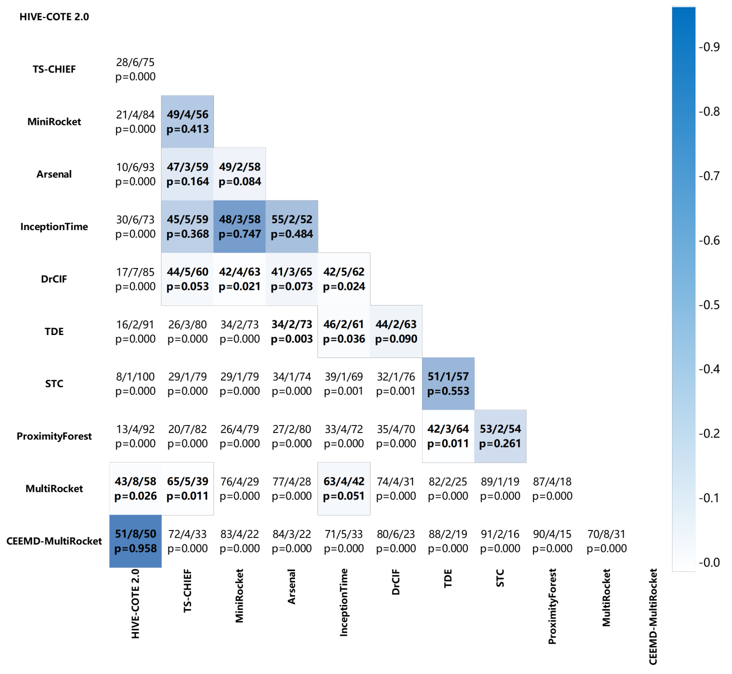

Figure 4. Pairwise difference between CEEMD-MultiRocket and 10 other SOTA algorithms in terms of statistical significance.

Figure 5 illustrates the pairwise accuracy of CEEMD-MultiRocket versus MultiRocket over 30 resamples of the 109 datasets from the UCR time series repository. Overall, we can find that most points are scattered above the dotted line, showing that CEEMD-MultiRocket achieves significantly better classification accuracy than MultiRocket.

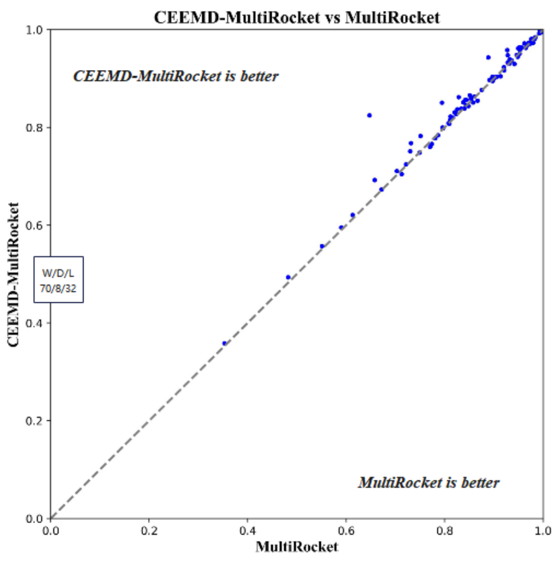

Figure 5. Pairwise accuracy of CEEMD-MultiRocket versus MultiRocket over 30 resamples of 109 datasets from the UCR time series repository.

4.3.2. Runtime Analysis

Although CEEMD-MultiRocket significantly outperforms MultiRocket in terms of accuracy, the additional decomposition operation, the sub-series selection and one extra pooling operator increase computational complexity. Fortunately, the reduction in the number of convolution kernels decreases the runtime of our proposed CEEMD-MultiRocket algorithm. We evaluated the overall training time of 10 other SOTA algorithms to train a single resample on 112 UCR datasets and compared the training time of these algorithms with our proposed CEEMD-MultiRocket, as shown in Table 4. From Table 4, we can find that CEEMD-MultiRocket is obviously faster than most TSC algorithms. All the SOTA algorithms, except the Rocket family, take lots of time to train, as reported by [22]. As for the Rocket family, MiniRocket, unsurprisingly, is the fastest, with a training time of under 4 min. After that is MultiRocket, which can be trained in under 24 min. Next is Rocket, with a training time of over 4 h, followed by our CEEMD-MultiRocket, taking under 5 h for training. Arsenal is an ensemble of Rocket, with a training time of about 28 h. The training time of six non-Rocket algorithms running on a high-performance computing (HPC) cluster with a single thread was reported in [22], which used higher-level hardware than ours. DrCIF takes about 2 days and the ensemble algorithms, including STC, HIVE-COTE 1.0/2.0 and TS-CHIEF, take at least 4 days. By comparison, the Rocket family algorithms use lots of randomly initialized convolution kernels for feature extraction, and only use one linear classifier for classification, without training the kernels, while non-Rocket algorithms, such as HIVE-COTE 2.0, TS-CHIEF, etc., integrate many classifiers with a large number of parameters and therefore require a lot of time for training. As a result, we can find that these non-Rocket algorithms are clearly more time-consuming than our algorithm. Specifically, the training time of CEEMD-MultiRocket consists of three parts: CEEMD (4.11 h), the IMFs selection (26.3 min) and model training using the improved MultiRocket (20.3 min). Due to the decomposition cost of raw time series, the proposed CEEMD-MultiRocket is slower than original MultiRocket, but it significantly outperforms MultiRocket in classification accuracy. Compared with HIVE-COTE 2.0, our proposed CEEMD-MultiRocket achieves almost the same level of classification accuracy but only costs 1.4% of the training time, showing that our proposed CEEMD-MultiRocket is an effective algorithm in TSC.

Table 4. Runtime to train single resample of 112 UCR datasets. The runtime of Rocket family and CEEMD-MultiRocket algorithm is calculated by running with a single thread on Intel Xeon Gold 5218 CPU. The runtime of the others is cited from [22].

TSC Algorithm	Total Training Time
MiniRocket (10 k features)	3.61 min
MultiRocket (50 k features)	23.65 min
Rocket (20 k features)	4.25 h
CEEMD-MultiRocket (50 k features)	4.88 h
Arsenal	27.91 h
DrCIF	45.40 h
TDE	75.41 h
STC	115.88 h
HIVE-COTE 2.0	340.21 h
HIVE-COTE 1.0	427.18 h
TS-CHIEF	1016.87 h

5. Discussion

For a more comprehensive evaluation of CEEMD-MultiRocket, we continue to discuss several characteristics of the proposed algorithm on 109 datasets from the UCR time series repository in detail, including the parameter setting of CEEMD, the sub-series selection, the convolution kernel design and pooling operators.

5.1. CEEMD Parameter Settings

During the procedure of decomposing raw time series, CEEMD adds a specific white noise to the time series. The addition of white noise is an important step in the CEEMD method that can eliminate the mode mixing problem and help to improve the accuracy and reliability of the decomposition results. The decomposition contains two main parameters: the number of realizations R and the noise signal intensity N. Different numbers of realizations and noise signal intensities may produce different IMFs. Experiments were conducted on 109 datasets from the UCR time series repository to assess the impact of these two parameters for classification performance. Figure 6 shows the mean rank of different noise intensities (N = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7) applied on CEEMD-MultiRocket with a fixed number of realizations R = 30. Figure 7 shows the mean rank of different numbers of realizations (R = 10, 20, 30, 40, 50, 60, 70) applied on CEEMD-MultiRocket with a fixed noise intensity N = 0.4.

As shown in Figure 6, CEEMD-MultiRocket obtains the best mean rank in classification accuracy on the 109 datasets when N = 0.4. This indicates that a too high or too low noise intensity leads to a reduction in classification accuracy, though there is not a statistically significant difference in classification accuracy. The findings of the experiment reveal that the intensity of noise has only a marginal impact on the classification performance and an optimum value for the intensity of added noise is somewhere around 0.4.

Figure 7 shows that CEEMD-MultiRocket achieves the best classification accuracy when the number of realizations R = 70, but the difference of classification performance between R = 70 and R = 30 is relatively small and statistically insignificant. As shown in Section 4.3.2, the decomposition of raw time series takes up most of the runtime (about 84%) of CEEMD-MultiRocket, it is necessary to reduce the decomposition time as much as possible. Since the number of realizations R = 30 takes less than half of the time of R = 70, we adopt 30 as the number of realizations in our CEEMD-MultiRocket to achieve a balance between the classification accuracy and time cost.

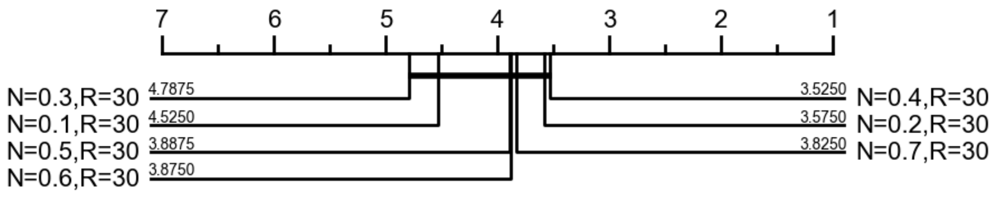

Figure 6. Mean rank of different noise intensities applied on CEEMD-MultiRocket with a fixed number of realizations.

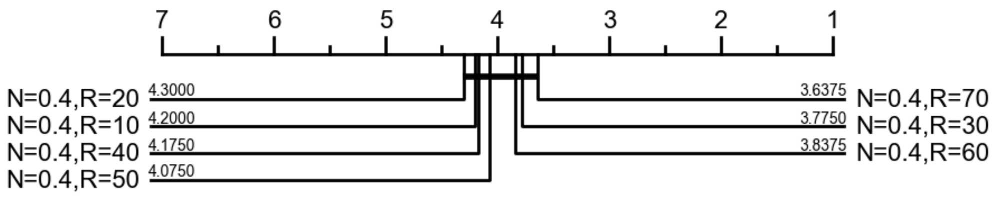

Figure 7. Mean rank of different numbers of realizations applied on CEEMD-MultiRocket with a fixed noise intensity.

5.2. Sub-Series Selection

Integrating all decomposed sub-series as the input to the classification model is not necessarily a guarantee of improvement of classification performance. Therefore, it is necessary to select the most important sub-series and discard less important ones to enhance the overall classification performance and decrease the computational complexity. The selection is executed on the whole known training set which is further divided into training and testing sets by stratified sampling, by comparing the testing accuracy (using improved MultiRocket) of each IMF with that of the raw time series, respectively, using a predetermined threshold. When the ratio of the testing accuracy of the IMF to that of the raw time series is more than the given threshold, this IMF is selected as the input to the classification model. We set different thresholds and the corresponding classification results are shown in Figure 8.

From Figure 8, we can find that CEEMD-MultiRocket achieves the best classification accuracy when the value of the threshold is 0.9, although the difference by setting different thresholds is negligible and statistically insignificant in terms of the classification accuracy. Table 5 shows the number of datasets with 0, 1, 2 or 3 IMFs which are selected using different thresholds on 109 datasets. When the threshold is set to 0, all three IMFs are unconditionally selected for each dataset, and the classification accuracy is the worst because some of these IMFs may produce negative impacts on the performance of the classification algorithm. When the threshold is set to 1, more than half of the datasets do not have any IMFs selected. Although this can decrease the computational complexity, it may also lose many crucial sub-series and reduce the classification performance. We find that when the threshold is set to 0.9, 93 datasets out of all 109 datasets select at least one decomposed IMF, and our model achieves the best classification accuracy due to the addition of appropriate $IMFs^*$.

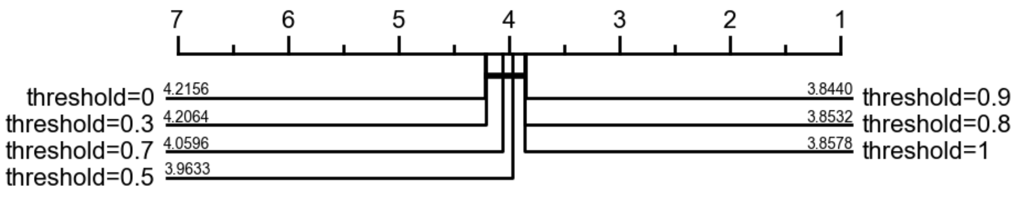

Figure 8. Mean rank of CEEMD-MultiRocket using different threshold.

Table 5. Number of datasets of selecting different number of IMFs using different thresholds on 109 datasets from the UCR time series repository.

Threshold	**Number of IMFs Selected**			
	0	1	2	3
0	0	0	0	109
0.3	0	3	3	103
0.5	0	14	7	88
0.7	0	31	13	65
0.8	1	40	22	46
0.9	16	48	17	28
1	60	31	11	7

5.3. Convolution Kernel Design

In CEEMD-MultiRocket, we decrease the length of the convolution kernel to 6. Figure 9 demonstrates the effectiveness of different kernel lengths on classification accuracy. In Figure 9, 6_2 means the kernel length is 6, in which two weights are one value and the remaining weights are another value. Thus, it gives $C_6^2 = 15$ fixed kernels in total. As can be seen from Figure 9, convolution kernels of length 5, 6 or 7 significantly outperform other convolution kernels with a length of 8, 9 or 11 in terms of classification accuracy.

It is also worth mentioning that the entire set of kernels of length 6 produces higher accuracy than the 6_2 kernel subset, but the classification accuracy is statistically insignificant. Since the 6_2 kernel subset only has about a quarter of the number of convolution kernels of the entire set of kernels with length 6, it is particularly suitable for the optimizations of avoiding multiplications, which can significantly shorten training time but achieve almost the same level of classification accuracy. Therefore, the convolution kernel of length 6_2 is applied in the proposed CEEMD-MultiRocket.

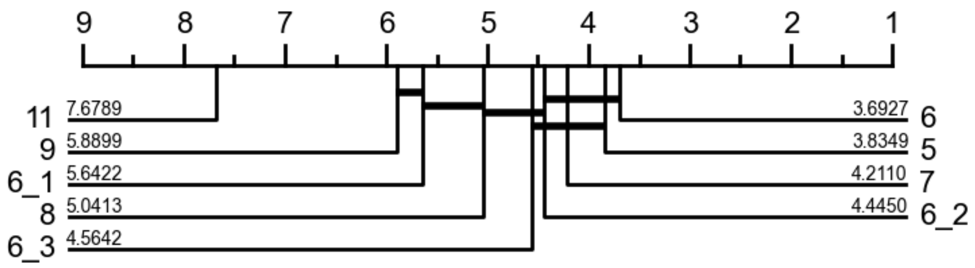

Figure 9. Mean rank of CEEMD-MultiRocket using different convolution kernel lengths.

5.4. Pooling Operators

In CEEMD-MultiRocket, we add an extra pooling operator NSPV (Number of Stretch of Positive Values) to enrich the discriminatory power of derived features. Figure 10 compares the effectiveness of using all five pooling operators (PPV, MPV, MIPV, LSPV, NSPV), four pooling operators (PPV, MPV, MIPV, LSPV), two pooling operators (PPV and NSPV) and only PPV in CEEMD-MultiRocket with 50k features. The experimental result shows that compared with four pooling operators, an additional NSPV pooling operator is able to significantly increase classification accuracy, indicating that NSPV contributes to the improvement of classification performance in CEEMD-MultiRocket. Furthermore, we also find that using four pooling operators (PPV + MPV + MIPV + LSPV) is not significantly better than only using two pooling operators (PPV + NSPV).

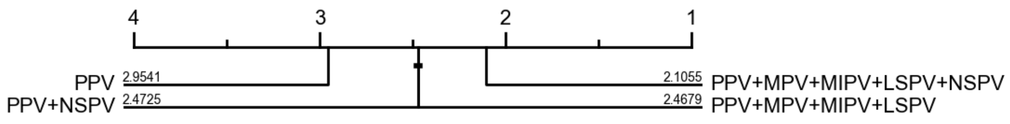

Figure 10. Mean rank of CEEMD-MultiRocket using different combinations of pooling operators.

5.5. Summary

From the above results and analysis, some findings can be summarized as follows:

(1) Decomposing raw time series into sub-series and extracting features from them can obtain more detailed and discriminative information from various aspects, which significantly contributes to the enhancement of classification accuracy.

(2) Selecting the more crucial sub-series and pruning the redundant and less important ones can both enhance classification performance and reduce computational complexity.

(3) The optimization in convolution kernel design can generate more efficient transform, which helps to improve the overall classification accuracy.

(4) The additional pooling operator NSPV enriches the discriminatory power of derived features.

6. Conclusions

To enhance the classification performance of the original MultiRocket, this study proposes a hybrid classification model CEEMD-MultiRocket which integrates CEEMD and improved MultiRocket. Firstly, the CEEMD algorithm is employed to decompose raw time series into two IMFs and one residue, which represent the high-, medium- and low-frequency portions of raw time series, respectively. Then, the selection of these decomposed sub-series is conducted on the whole known training set which is further divided into new training and testing sets using stratified sampling, by comparing the classification accuracy of each sub-series with that of the raw time series using a given threshold. Finally, we improve the convolutional kernel and pooling operators of the original MultiRocket, apply the improved MultiRocket to the raw time series, the selected decomposed sub-series and the first-order difference of raw time series to extract features, and build a ridge regression classifier. The experimental results demonstrate that: (1) in comparison to all SOTA classification algorithms except for HIVE-COTE 2.0, the proposed algorithm can significantly enhance the classification accuracy on 109 datasets from the UCR time series repository; (2) CEEMD-MultiRocket achieves almost the same level of classification accuracy as HIVE-COTE 2.0, with a fraction of the computing cost of the latter; (3) the CEEMD algorithm has the ability to generate a variety of representations of raw time series as the inputs of the algorithm, which contributes to the improvement of classification accuracy; (4) the improvement of convolution kernel length and the reduction in the number of convolution kernels can enhance classification performance while reducing computational load; and (5) the additional pooling operator contributes to enhancing the classification accuracy.

There are two main limitations in our work: (1) CEEMD is a relatively time-consuming decomposition method; (2) the values of weights in the convolution kernel are pre-defined and cannot be dynamically adjusted in line with increases in the dilation. The main directions for future research could be extended in two aspects: (1) continuing to improve MultiRocket to build the hybrid ensemble classification algorithm for time series; (2) considering faster decomposition algorithms and sub-series selection algorithms to improve the runtime and classification accuracy of the algorithm.

Author Contributions: Conceptualization, P.W., J.W. and Y.W.; Formal analysis, P.W. and J.W.; Investigation, P.W. and Y.W.; Methodology, P.W. and J.W.; Project administration, J.W.; Resources, J.W. and T.L.; Software, P.W. and J.W.; Supervision, T.L.; Validation, P.W. and Y.W.; Writing—original draft, P.W., J.W. and Y.W.; Writing—review and editing, J.W., Y.W. and T.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Education of Humanities and Social Science Project (grant no. 19YJAZH047) and the Social Practice Research for Teachers of Southwestern University of Finance and Economics (grant no. 2022JSSHSJ11).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the data in this paper are publicly available. They can be accessed at https://www.cs.ucr.edu/~eamonn/time_series_data/ (all accessed on 20 October 2022).

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

References

1. Rezaei, S.; Liu, X. Deep learning for encrypted traffic classification: An overview. *IEEE Commun. Mag.* **2019**, *57*, 76–81. [CrossRef]
2. Susto, G.A.; Cenedese, A.; Terzi, M. Time-series classification methods: Review and applications to power systems data. *Big Data Appl. Power Syst.* **2018**, 179–220. [CrossRef]
3. Li, T.; Qian, Z.; Deng, W.; Zhang, D.; Lu, H.; Wang, S. Forecasting crude oil prices based on variational mode decomposition and random sparse Bayesian learning. *Appl. Soft Comput.* **2021**, *113*, 108032. [CrossRef]
4. Chao, L.; Zhipeng, J.; Yuanjie, Z. A novel reconstructed training-set SVM with roulette cooperative coevolution for financial time series classification. *Expert Syst. Appl.* **2019**, *123*, 283–298. [CrossRef]
5. Ebrahimi, Z.; Loni, M.; Daneshtalab, M.; Gharehbaghi, A. A review on deep learning methods for ECG arrhythmia classification. *Expert Syst. Appl. X* **2020**, *7*, 100033. [CrossRef]
6. Wu, J.; Zhou, T.; Li, T. Detecting epileptic seizures in EEG signals with complementary ensemble empirical mode decomposition and extreme gradient boosting. *Entropy* **2020**, *22*, 140. [CrossRef]
7. Craik, A.; He, Y.; Contreras-Vidal, J.L. Deep learning for electroencephalogram (EEG) classification tasks: A review. *J. Neural Eng.* **2019**, *16*, 031001. [CrossRef]
8. Liu, Y.; Wu, Y.F. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
9. Pantiskas, L.; Verstoep, K.; Hoogendoorn, M.; Bal, H. Taking ROCKET on an efficiency mission: Multivariate time series classification with LightWaves. *arXiv* **2022**, arXiv:2204.01379.
10. Nishikawa, Y.; Sannomiya, N.; Itakura, H. A method for suboptimal design of nonlinear feedback systems. *Automatica* **1971**, *7*, 703–712.
11. Lucas, B.; Shifaz, A.; Pelletier, C.; O'Neill, L.; Zaidi, N.; Goethals, B.; Petitjean, F.; Webb, G.I. Proximity forest: An effective and scalable distance-based classifier for time series. *Data Min. Knowl. Discov.* **2019**, *33*, 607–635. [CrossRef]

12. Flynn, M.; Large, J.; Bagnall, T. The contract random interval spectral ensemble (c-RISE): The effect of contracting a classifier on accuracy. In *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 381–392.
13. Deng, H.; Runger, G.; Tuv, E.; Vladimir, M. A time series forest for classification and feature extraction. *Inf. Sci.* **2013**, *239*, 142–153. [CrossRef]
14. Middlehurst, M.; Large, J.; Bagnall, A. The canonical interval forest (CIF) classifier for time series classification. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 188–195.
15. Lubba, C.H.; Sethi, S.S.; Knaute, P.; Schultz, S.R.; Fulcher, B.D.; Jones, N.S. catch22: CAnonical Time-series CHaracteristics: Selected through highly comparative time-series analysis. *Data Min. Knowl. Discov.* **2019**, *33*, 1821–1852. [CrossRef]
16. Schäfer, P. The BOSS is concerned with time series classification in the presence of noise. *Data Min. Knowl. Discov.* **2015**, *29*, 1505–1530. [CrossRef]
17. Middlehurst, M.; Large, J.; Cawley, G.; Bagnall, A. The temporal dictionary ensemble (TDE) classifier for time series classification. In Proceedings of the Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, 14–18 September 2020; Part I; Springer: Berlin/Heidelberg, Germany, 2021; pp. 660–676.
18. 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]
19. 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.
20. Bostrom, A.; Bagnall, A. Binary shapelet transform for multiclass time series classification. In *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXII: Special Issue on Big Data Analytics and Knowledge Discovery*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 24–46.
21. Bagnall, A.; Flynn, M.; Large, J.; Lines, J.; Middlehurst, M. On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (hive-cote v1.0). In Proceedings of the Advanced Analytics and Learning on Temporal Data: 5th ECML PKDD Workshop, AALTD 2020, Ghent, Belgium, 18 September 2020; Revised Selected Papers 6; Springer: Berlin/Heidelberg, Germany, 2020; pp. 3–18.
22. 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]
23. 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]
24. Shifaz, A.; Pelletier, C.; Petitjean, F.; Webb, G.I. TS-CHIEF: A scalable and accurate forest algorithm for time series classification. *Data Min. Knowl. Discov.* **2020**, *34*, 742–775. [CrossRef]
25. Längkvist, M.; Karlsson, L.; Loutfi, A. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognit. Lett.* **2014**, *42*, 11–24. [CrossRef]
26. Bengio, Y.; Yao, L.; Alain, G.; Vincent, P. Generalized denoising auto-encoders as generative models. *Adv. Neural Inf. Process. Syst.* **2013**, *26*.
27. Hu, Q.; Zhang, R.; Zhou, Y. Transfer learning for short-term wind speed prediction with deep neural networks. *Renew. Energy* **2016**, *85*, 83–95. [CrossRef]
28. Gallicchio, C.; Micheli, A. Deep echo state network (deepesn): A brief survey. *arXiv* **2017**, arXiv:1712.04323.
29. Pascanu, R.; Mikolov, T.; Bengio, Y. Understanding the exploding gradient problem. *arXiv* **2012**, arXiv:1211.5063.
30. 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]
31. 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 2017; SPIE: Bellingham, WA, USA, 2018; Volume 10696, pp. 242–249.
32. Tripathy, R.; Acharya, U.R. Use of features from RR-time series and EEG signals for automated classification of sleep stages in deep neural network framework. *Biocybern. Biomed. Eng.* **2018**, *38*, 890–902. [CrossRef]
33. Wang, Z.; Oates, T. Imaging time-series to improve classification and imputation. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, 25–31 July 2015.
34. Nweke, H.F.; Teh, Y.W.; Al-Garadi, M.A.; Alo, U.R. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Syst. Appl.* **2018**, *105*, 233–261. [CrossRef]
35. Tan, C.W.; Dempster, A.; Bergmeir, C.; Webb, G.I. MultiRocket: Multiple pooling operators and transformations for fast and effective time series classification. *Data Min. Knowl. Discov.* **2022**, *36*, 1623–1646. [CrossRef]
36. Dempster, A.; Petitjean, F.; Webb, G.I. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Min. Knowl. Discov.* **2020**, *34*, 1454–1495. [CrossRef]
37. Dempster, A.; Schmidt, D.F.; Webb, G.I. Minirocket: A very fast (almost) deterministic transform for time series classification. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 248–257.

38. Torres, M.E.; Colominas, M.A.; Schlotthauer, G.; Flandrin, P. A complete ensemble empirical mode decomposition with adaptive noise. In Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic, 22–27 May 2011; pp. 4144–4147.
39. Zhou, Y.; Li, T.; Shi, J.; Qian, Z. A CEEMDAN and XGBOOST-based approach to forecast crude oil prices. *Complexity* **2019**, *2019*, 1–15. [CrossRef]
40. Li, T.; Qian, Z.; He, T. Short-term load forecasting with improved CEEMDAN and GWO-based multiple kernel ELM. *Complexity* **2020**, *2020*, 1–20. [CrossRef]
41. Wu, Z.; Huang, N.E. Ensemble empirical mode decomposition: A noise-assisted data analysis method. *Adv. Adapt. Data Anal.* **2009**, *1*, 1–41. [CrossRef]
42. Huang, N.E.; Shen, Z.; Long, S.R.; Wu, M.C.; Shih, H.H.; Zheng, Q.; Yen, N.C.; Tung, C.C.; Liu, H.H. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proc. R. Soc. London. Ser. A Math. Phys. Eng. Sci.* **1998**, *454*, 903–995. [CrossRef]
43. Li, S.; Zhou, W.; Yuan, Q.; Geng, S.; Cai, D. Feature extraction and recognition of ictal EEG using EMD and SVM. *Comput. Biol. Med.* **2013**, *43*, 807–816. [CrossRef] [PubMed]
44. Tang, B.; Dong, S.; Song, T. Method for eliminating mode mixing of empirical mode decomposition based on the revised blind source separation. *Signal Process.* **2012**, *92*, 248–258. [CrossRef]
45. Wu, J.; Chen, Y.; Zhou, T.; Li, T. An adaptive hybrid learning paradigm integrating CEEMD, ARIMA and SBL for crude oil price forecasting. *Energies* **2019**, *12*, 1239. [CrossRef]
46. Li, T.; Zhou, M. ECG classification using wavelet packet entropy and random forests. *Entropy* **2016**, *18*, 285. [CrossRef]
47. 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]
48. Chai, J.; Wang, Y.; Wang, S.; Wang, Y. A decomposition–integration model with dynamic fuzzy reconstruction for crude oil price prediction and the implications for sustainable development. *J. Clean. Prod.* **2019**, *229*, 775–786. [CrossRef]
49. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.