SuperRocket: On Efficient Selection of Kernel-based Transformations for Fast and Accurate Time Series Classification

Zhiyu Liang[1], Chen Li[1], Xiongfei Li[1], Shourong Li[2], Zheng Liang[1]
Chen Liang[1], Shi Qiu[3], Jin Wang[3], Hongzhi Wang[1]✉
[1]Harbin Institute of Technology, China [2]BYD Company Limited, China [3]Central South University, China
{zyliang, lz20, wangzh}@hit.edu.cn, {lichen2022, 24s136042, 23B903050}@stu.hit.edu.cn
1278724770@qq.com, {sheldon.qiu, jerryW}@csu.edu.cn

***Abstract*—The kernel-based approach is one of the most representative solutions for time series classification (TSC) due to its remarkable training efficiency and fairly high accuracy. However, existing kernel-based methods use abundant (e.g., 50K) randomly generated transformations directly for feature extraction, ignoring that a portion of them can produce redundant or irrelevant features, causing unnecessary computational costs and bottlenecking their accuracy. Based on our observations, it is feasible to generate many more transformations (e.g., 400K) using existing kernel-based methods and select a subset that can produce high-quality features, so as to improve the TSC accuracy with little computational overhead to maintain the superior efficiency. Unfortunately, it is difficult for existing solutions to address the transformation selection problem for such a large number of candidates due to their high complexity and insufficient consideration of the correlation between transformations and classifiers. To tackle this issue, we propose SuperRocket, a novel approach that can efficiently select high-quality kernel-based transformations to achieve fast and accurate TSC. We customize a learnable indicating vector with two constraints to select all transformations simultaneously without iteratively building or evaluating the classifier as the existing solutions. On this basis, we model transformation selection as an unconstrained bi-level optimization problem of learning the optimal indicating vector that is regularized based on the constraints, and propose an end-to-end optimization algorithm to efficiently solve it. Theoretical analysis validates the effectiveness of SuperRocket. Extensive experiments on a total of 135 datasets show that our proposal not only outperforms existing selection solutions in terms of both accuracy and efficiency, but also achieves competitive accuracy and two orders of magnitude faster training speed compared to the state-of-the-art TSC method.**

***Index Terms*—efficiency and scalability in data analytics, time series, classification**

I. Introduction

Time series classification (TSC) addresses the problem of creating a function (a.k.a. model) that maps from the space of input time series to the space of possible class labels [1]. It is one of the most fundamental techniques in data engineering [2], with applications in various scenarios [3], [4]. However, although tens of TSC algorithms [5], [6] have been proposed in the last decade, it remains *difficult for existing solutions to achieve fast and accurate TSC*.

✉ Corresponding author.

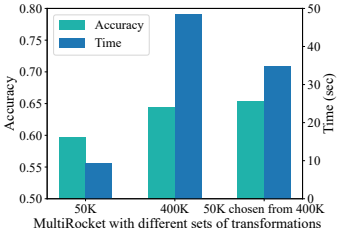


Fig. 1: Accuracy and training time of MultiRocket on PigAirwayPressure dataset using different sets of transformations.

Challenge I: How to ensure state-of-the-art (SOTA) classification accuracy while improving training efficiency? Despite the increasing accuracy of existing TSC approaches, most of them are inefficient due to the high computational overhead incurred in the training phase for building the TSC model. For example, as we evaluated in § VI-C, the SOTA TSC method HIVE-COTE 2.0 (HC2) [7] requires *more than two weeks* to train for the commonly used 109 UCR datasets [3]. The low efficiency can limit their practical applications. For this reason, instead of focusing on improving accuracy, some studies are dedicated to efficient TSC solutions without much sacrificing accuracy. Typically, a recent line of work based on kernels [8]–[10] has attracted much attention, owing to its simple structure, remarkable training efficiency and passable accuracy. In specific, the SOTA kernel-based method, namely MultiRocket [10], takes only *17.77 minutes* to train on the same 109 datasets. It is *almost the fastest* one of the existing TSC methods that are *less accurate than but approach* the SOTAs on average (see § VI-C).

However, the existing kernel-based TSC methods are still limited in achieving competitive accuracy compared to the SOTA method HC2 (see § VI-C), because they directly adopt the time series features extracted using multiple transformations[1] randomly generated given the total number to build the classifier, ignoring that a portion of transformations may produce redundant or even irrelevant features to predict the class labels for different datasets. This issue also causes unnecessary computational costs that limit the training efficiency.

[1]Each transformation is an operation sequence of {*preprocessing*, *convolution*, *pooling*} that produces one dimension of the feature [10].

Example 1. For example, we show in Fig. 1 the accuracy and training time of three MultiRocket variants using different sets of transformations on a UCR dataset PigAirwayPressure [3], from which we have two observations. First, compared to using the default setting of 50K transformations (50K), the variant that uses more transformations (400K) can improve the accuracy by 4.8%, but has a cost of $5.23\times$ slower training speed. Next, if we use only a subset of 50K transformations chosen[2] from the 400K, the resulted accuracy is 0.1% higher than that of using all the 400K. At the same time, using the selected transformations also saves 28.5% of training time.

The above observations imply that it is possible to *generate a large number of transformations* (e.g. 400K) following the existing kernel-based methods, and *select a subset of transformations that can produce high-quality features* to transform the raw time series and build the classifier. In this way, it is expected to *further improve the accuracy to achieve the SOTA while incurring acceptable computational costs to maintain the superiority in training efficiency*. However, a major challenge exists to achieve this goal.

Challenge II: How to efficiently select a subset of transformations from such a large number of candidates for optimizing the classifier? The problem is known to be NP-hard [11]. Although there are heuristic solutions, such as popular variable (a.k.a. feature) selection techniques in the traditional machine learning area [12], including filter [13], wrapper [14] and embedded methods [12], [13], these solutions are mainly designed for scenarios where the variable size is relatively small [15], [16] (e.g., tens to thousands). When addressing the *hundreds of thousands* of transformations, they can be either *less accurate* due to insufficient consideration of the correlation between variables and classifiers (for filter methods) or *unable to scale* due to the high time complexity (for wrapper and embedded methods) [12].

Although there are also selection techniques tailored for the kernel-based TSC methods, including S-Rocket [17] that adopts the evolutionary algorithm [18] to search for the optimal subset of transformations, POCKET [19] that explores group elastic net, a variant of the embedded method LASSO [20] for the selection, and Detach-ROCKET [21] (D-Rocket) that designs an improved wrapper method based on a backward-stepwise selection technique [22] to iteratively prune the transformations with a greedy strategy, the primary goal of these studies is to prune the model parameters of the existing kernel-based approaches (e.g., the standard MultiRocket with 50K transformations [10]), rather than exploring more high-quality transformations to improve the accuracy, and they *fail to address the challenge of efficiently selecting optimal transformations from orders of magnitude more candidates* (see § VI-B). For example, S-Rocket and D-Rocket are *computationally intensive* because they need to repeatedly train or evaluate the classifier multiple times with different selections, while POCKET has *poor scalability* due to the quadratic time and space complexities with respect to the number of transformations to be selected (see Table I).

To address the above issues, we propose **SuperRocket**, a novel kernel-based transformation selection approach that can *efficiently select high-quality transformations to achieve fast and accurate TSC*. To ensure efficiency, the basic idea is to specifically design a learnable continuous-valued vector with customized constraints to indicate whether each candidate is selected or not, so as to *determine all transformations simultaneously by efficiently learning the indicating vector, avoiding the time-consuming repeated training or evaluation of the classifier* as existing solutions. However, such a solution still faces a challenge.

Challenge III: How to efficiently select high-quality transformations via indicating vector learning? First, as detailed in § IV-C, with the designed indicating vector, the transformation selection problem is converted into a complex constrained optimization problem that is difficult to solve efficiently. To address this issue, SuperRocket *models the transformation selection problem as an unconstrained bi-level optimization problem* [23], where the inner optimization aims to build an optimized classifier using the transformations selected by the indicating vector, while the outer optimization searches for the indicating vector that is *regularized following the constraints* to encourage it to select the transformations by which the optimized classifier achieves the highest accuracy.

Second, solving the bi-level problem straightforwardly or using existing solutions is computationally intensive, because it still needs to iteratively build the classifier. Thus, SuperRocket derives *an end-to-end optimization algorithm to efficiently learn the indicating vector without retraining the classifier*, by combining the indicating vector and the classifier parameters into one learnable model and alternately updating them using a simple approximation strategy.

To sum up, we make the following notable contributions.

- **Objective.** This study aims to tackle the challenge of efficiently selecting a subset of kernel-based transformations that can produce high-quality features for TSC from hundreds of thousands of candidates. In this way, our goal is to further improve the accuracy of the existing kernel-based TSC methods to achieve the SOTA accuracy while incurring acceptable computational costs to maintain the superior training efficiency. Unlike existing work, this is the first study that pays special attention to the efficiency issue of transformation selection.
- **Methodology.** We propose SuperRocket, a novel efficient transformation selection approach that determines all transformations simultaneously by learning a customized continuous-valued and constrained indicating vector. To effectively learn the indicating vector, we model the transformation selection problem as an unconstrained bi-level optimization problem of finding the optimal indicating vector that is regularized based on the constraints, and design an end-to-end optimization algorithm to efficiently solve it.
- **Theoretical analysis.** We show theoretically that the proposed SuperRocket is more efficient in time (and also

[2]It is conducted by repeating 1000 times of transformation selection at random and reporting the highest test accuracy, which can be seen as the oracle of a sub-optimal solution of the transformation selection.

in space) than existing transformation selection methods. Meanwhile, we derive an upper bound for the difference in classification error on unseen time series (a.k.a. generalization error) between SuperRocket and the oracle solution that can ideally select the transformations yielding to the minimum generalization error. The analysis indicates that both the training efficiency and the classification accuracy of the proposed method are theoretically guaranteed.
- **Evaluation.** Extensive experiments are conducted on the popular 109 UCR and 26 UEA datasets. The results show that SuperRocket surpasses existing transformation selection solutions in terms of both efficiency and accuracy, and achieves the *SOTA accuracy* with a total training time *two orders of magnitude faster* than the SOTA competitor HC2.

II. Related Work

A. Time Series Classification (TSC).

TSC has been widely studied in the last decade [5], [24]. In general, existing approaches can be categorized into four types, including the *distance-based* method, the *feature-based* method, the *deep learning* method, and the *hybrid* method.

The *distance-based methods* [5] adopt a one-nearest-neighbor model in conjunction with a metric that quantifies the distance between two time series (e.g., dynamic time warping, DTW [25]) as the classifier, which is easy to implement and can be a basic baseline for evaluating a new TSC algorithm. However, this kind of method cannot well capture complex time series patterns and is sensitive to noise [5], which limits its performance in terms of accuracy.

The *feature-based approaches* [26] extract features from the raw time series to build a traditional classifier, such as logistic regression or random forest. There are four types of representative features in the literature, including the **interval-based** features [7], [27], [28] that extract the statistics (e.g., mean and median) of the time series in specific time intervals (e.g., from timestamp 10 to 20), the **shapelet-based** features [29]–[35] that measure the similarity between the time series and the salient subsequences (i.e., shapelets), the **dictionary-based** features [36]–[39] that count the frequency of the specific patterns occurring in the time series, and the **kernel-based** features [8]–[10] that extract the features using abundant convolutional kernel-based transformations. With the carefully designed feature space, the feature-based methods can achieve competitive accuracy. Different types of features can also complement each other through ensemble learning to further improve the performance [7], [26], [40], [41].

The *deep learning methods* [24], [42], [43] adopt deep neural networks (DNNs) for end-to-end TSC, where a DNN can be seen as a complex layer-by-layer feature extractor that automatically embeds the time series into low-dimensional features, appended by a linear classifier that makes predictions based on the features. As evaluated in [24], [42], the deep learning methods can achieve remarkable accuracy with advanced DNN architectures, such as InceptionTime [44] and OSCNN [42]. However, such methods are computationally intensive even with the acceleration of high-performance GPUs, since the DNNs incorporate large amounts of operations. DNNs also suffer from a deficiency of interpretability [45], which remains a challenge for existing research.

The *hybrid TSC approaches* combine different types of distance metrics [46], [47], features [48], or TSC methods [7], [26], [40], [41] to further improve the performance. The most representative work, namely the Hierarchical Vote Collective of Transformation-based Ensembles 2.0 (HIVE-COTE 2.0, HC2) [7], uses a generic meta-ensemble structure that can incorporate any TSC method discussed above as a component. Despite the SOTA accuracy the hybrid methods are achieving, they are remarkably time-consuming due to the computation overhead incurred by all components [10].

In this paper, we propose a **novel** TSC solution following the *kernel-based* methods, since they are *ultra-fast* compared to other competitive approaches that usually have high time complexity (e.g., it takes up to $\mathcal{O}(N^2T^4)$ time to build the shapelet-based features [30] where N and T represent the number and length of the time series, respectively). Meanwhile, they can also achieve accuracy comparable to SOTA methods [10]. However, *instead of designing novel transformations for feature extraction* (e.g., Rocket [8], MiniRocket [9] and MultiRocket [10]) *or deriving new feature types with kernels* (e.g., Hydra [39]) as in existing studies, we focus on an orthogonal direction, that is, *how to efficiently select high-quality transformations from the candidates generated by existing solutions, to achieve SOTA accuracy while guaranteeing the superior training speed.*

B. Kernel-based Transformation Selection.

Related work on kernel-based transformation selection can be divided into two categories, the *general variable or feature selection methods* and the *customized transformation selection approaches*.

In general, there are three types of variable selection solutions, including the filter, wrapper, and embedded methods [12]. The filter methods [13] incorporate a proxy metric (e.g., mutual information [13]) to evaluate the importance of each variable, which is a general solution independent of the models. Thus, the filter methods can be less accurate for specific applications. The wrapper methods [14] take the model as a black box and repeatedly evaluate the model performance using different subsets of candidates, which is computationally expensive. The embedded methods [12], [13] incorporate all possible candidates into a model training process to automatically select the subset of candidates during the training. However, the typical embedded methods, such as Least Absolute Shrinkage and Selection Operator (LASSO) [20], cannot scale well to a large number of candidates because the available numerical algorithm for solving it is computationally intensive.

To address the above limitations, previous work proposes improved methods tailored for transformation selection. POCKET [19] designs a variant of LASSO to improve the quality of the selected transformations. However, it leads to quadratic time and space complexities with respect to the total number of transformations to select, making it infeasible to

address hundreds of thousands of transformations as in our scenario. S-Rocket [17] and Detach-Rocket (D-Rocket) [21] are two wrapper methods designed based on evolution [18] and stepwise-backward selection [22] schemes, respectively. Although these approaches are faster than traditional wrapper methods, they are still *far from achieving fast TSC* comparable to the standard kernel-based approaches (e.g., MultiRocket [10]) and our SuperRocket (see § VI-B).

III. PRELIMINARIES

A. Kernel-based Time Series Classification

The goal of TSC is to create a function that maps from the space of input time series to the space of class labels [1]. A time series (sample) is defined as a sequence of data points $\boldsymbol{X} = (x_1, \ldots, x_p, \ldots, x_T)$ ordered by time, where x_p is the observation at timestamp p and T is the length. The class label y is a discrete variable with C possible values. i.e., $y \in \{c\}_{c=1}^{C}$ where $C \geq 2$. TSC is usually achieved by using a training data set $\mathcal{D}_{train} = \{(\boldsymbol{X}_j, y_j)\}_{j=1}^{N}$ to build a model that can predict the class values for previously unseen time series, where the instance $(\boldsymbol{X}_j, y_j)$ represents the pair of the j-th time series and the corresponding label.

This study focuses on the kernel-based TSC methods [8]–[10] due to their superior training efficiency and comparable accuracy, as discussed in § I. Generally, this kind of methods builds the TSC model in the following three steps, which *corresponds to the training phase*.

First, a large number K of transformations (e.g., 20K for Rocket [8] and 50K for MultiRocket [10]) are randomly generated to extract features from the time series, where each transformation is an operation sequence of {*preprocessing,convolution,pooling*} that produces one dimension of the feature. Formally, denote $f_k : \mathbb{R}^T \mapsto \mathbb{R}$ the k-th transformation ($k \in \{1, \ldots, K\}$). f_k is defined as follows.

$$f_k(\boldsymbol{X}) = Pool_k(Conv_k(Prep_k(\boldsymbol{X}))), \tag{1}$$

where $Prep_k$, $Conv_k$ and $Pool_k$ are the corresponding preprocessing, convolution and pooling operations, respectively. Different kernel-based methods contribute to different randomized strategies to generate these operations. We refer the interested readers to [8]–[10] for the details.

Next, all generated transformations are used to transform each time series $\boldsymbol{X}_j$ into a feature vector $\boldsymbol{Z}_j \in \mathbb{R}^K$. Denote $\mathcal{F}_K = (f_1, \ldots, f_K)$ the transformations generated given the number K and $z_{j,k} = f_k(\boldsymbol{X}_j)$ the k-th feature of $\boldsymbol{X}$ extracted using f_k. Then we have:

$$\boldsymbol{Z}_j = \mathcal{F}_K(\boldsymbol{X}_j) = (f_1(\boldsymbol{X}_j), \ldots, f_K(\boldsymbol{X}_j)) = (z_{j,1}, \ldots, z_{j,K}). \tag{2}$$

Finally, a lightweight linear classifier $\mathcal{G} : \boldsymbol{Z} \mapsto \boldsymbol{Y}$ (e.g., ridge regression [49]) is built using the transformed data $\{(\boldsymbol{Z}_j, y_j)\}_{j=1}^{N}$ to predict the distribution of classes for an input time series, i.e., $\hat{\boldsymbol{Y}}_j = \mathcal{G}(\boldsymbol{Z}_j)$. The unseen data are transformed in the same way using $\mathcal{F}_K$ for classification with the classifier $\mathcal{G}$ after training, where each time series is classified into the class that corresponds to the maximum prediction, i.e., $\hat{y}_j = \arg\max \hat{\boldsymbol{Y}}_{j,c}, \forall c = \{1, \ldots, C\}$.

B. Kernel-based Transformation Selection

Based on the observations in § I, the main focus of this work is to efficiently select a subset of high-quality kernel-based transformations from a large number of candidates to achieve fast and accurate TSC.

Given $\mathcal{F}_H$ the H transformation candidates generated by a kernel-based method (e.g., MultiRocket [10]), the goal of the transformation selection problem is to select the subset of K transformations $\mathcal{F}_K^*$ that optimizes a evaluation metric, e.g., minimizes a classification loss $\mathcal{L}_{cls}$ (e.g., cross-entropy [1]). The problem is formulated as follows.

$$\mathcal{F}_K^* = \underset{\mathcal{F}_K' \subset \mathcal{F}_H}{\arg\min} \sum_j \mathcal{L}_{cls}(\hat{\boldsymbol{Y}}_j, y_j), \tag{3}$$

The selected $\mathcal{F}_K^*$ is used to transform the raw time series into kernel-based features to build the classifier and make predictions, which is expected to achieve higher accuracy than using the randomly generated H or K transformations as shown in Fig. 1. Moreover, using only the selected K transformations will be faster than using the total H, provided that the time spent on selection is negligible.

However, as discussed in § I, the existing solutions are either inefficient or less accurate when dealing with the kernel-based transformation selection problem. To address this issue, our SuperRocket is a novel solution superior in both efficiency and accuracy. We elaborate on the methodology in § IV.

IV. METHODOLOGY

A. Workflow of SuperRocket

In this section, we comprehensively describe the workflow of our SuperRocket, which to the best of our knowledge is *the first end-to-end solution that addresses the challenge of efficiently selecting the high-quality kernel-based transformations for fast and accurate TSC*. The workflow is visually depicted in Fig. 2, which consists of four main steps. We briefly explain these steps in the following, while we discuss the key techniques in detail in § IV-B and IV-C, respectively.

Step ①: Time series to features. This step transforms the raw time series $\boldsymbol{X}_j \in \mathbb{R}^T$ into features $\boldsymbol{Z}_j \in \mathbb{R}^H$ using the transformations $\mathcal{F}_H$ randomly generated by a kernel-based TSC method (e.g., MultiRocket [10]), with the aim of evaluating the quality of the transformations according to how the transformed features perform in distinguishing class labels. The step is formulated as:

$$\boldsymbol{Z}_j = \mathcal{F}_H(\boldsymbol{X}_j), \forall j \in \{1, \ldots, N\}, \tag{4}$$

where $\mathcal{F}_H(\boldsymbol{X}_j)$ is computed following Eqs. (1)-(2).

Step ②: Transformation indication. This step introduces a learnable indicating vector $\boldsymbol{\alpha} \in (0, 1)^H$ with two constraints (detailed in § IV-B) to simultaneously determine whether each transformation in $\mathcal{F}_H$ should be selected or not, where each entry in $\boldsymbol{\alpha}$ represents the probability of selecting the corresponding transformation. To learn $\boldsymbol{\alpha}$ that represents high-quality transformations for TSC, the transformed features of the time series are multiplied by $\boldsymbol{\alpha}$ to achieve soft feature selection, i.e., $\boldsymbol{\alpha} \circ \boldsymbol{Z}_j, \forall j \in \{1, \ldots, N\}$, where $\circ$ represents

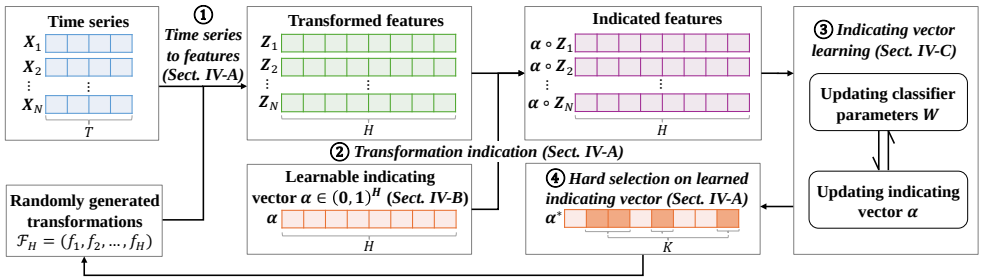


Fig. 2: Workflow of SuperRocket. The notation $\circ$ represents element-wise product.

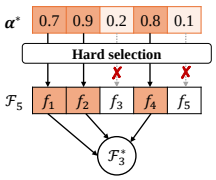


Fig. 3: A toy example of the hard selection using the learned indicating vector with $H = 5$ and $K = 3$.

element-wise product. The indicated features $\boldsymbol{\alpha} \circ \boldsymbol{Z}_j \in \mathbb{R}^H$ are then fed into the following indicating vector learning step to efficiently learn the optimal indicating vector for selecting the kernel-based transformations used for TSC.

Step ③: Indicating vector learning. In this step, we design an efficient learning algorithm to learn the optimal indicating vector $\boldsymbol{\alpha}^*$ that indicates the approximate K transformations resulting in the best performing classifier. It works by solving a derived unconstrained bi-level optimization problem in an end-to-end manner with (i) a learnable model combining the indicating vector $\boldsymbol{\alpha}$ that is regularized based on the constraints and the classifier parameters (denoted as $\boldsymbol{W}$) and (ii) a second-order approximation strategy that alternately updates $\boldsymbol{W}$ and $\boldsymbol{\alpha}$ (see details in § IV-C). The learned $\boldsymbol{\alpha}^*$ is then used to finally select the transformations in the next step.

Step ④: Hard selection on learned indicating vector. Given the learned $\boldsymbol{\alpha}^* \in (0,1)^H$ indicating the probability of selecting each transformation in $\mathcal{F}_H$, SuperRocket makes a hard selection to retain the K transformations corresponding to the highest probabilities, i.e.,

$$\mathcal{F}_K^* = \underset{\mathcal{F}_K' \subset \mathcal{F}_H}{\arg\max} \sum_{f_k' \in \mathcal{F}_K'} \boldsymbol{\alpha}^*_{I(f_{k'})}, \tag{5}$$

where the function $I(f_k) \in \{1, \ldots, H\}$ maps a transformation f_k to its original index in $\mathcal{F}_H$. Fig. 3 is a toy example of the hard selection with $H = 5$ and $K = 3$, where $\boldsymbol{\alpha}^*$, $\mathcal{F}_5$, and $\mathcal{F}_3^*$ represent the learned indicating vector, the whole transformations randomly generated by a kernel-based method, and the selected subset of transformations, respectively. The 1st, 2nd, and 4th transformations are selected because of the highest probabilities resulted in the corresponding entries of the learned indicating vector.

With the selected subset of transformations $\mathcal{F}_K^*$, SuperRocket transforms the training and test time series into K-dimensional feature vectors that are used in conjunction with the classifier $\mathcal{G}$ to achieve TSC. As can be seen, *SuperRocket is decoupled with the three steps of the standard kernel-based TSC methods discussed in § III-A*. Thus, it can be used as a plug-and-play module to seamlessly integrate into different kernel-based TSC approaches.

Next, we present the design details of the key techniques in SuperRocket, including the learnable indicating vector tailored for transformation selection and the efficient optimization algorithm for learning the indicating vector.

B. Learnable Indicating Vector for Transformation Selection

Considering the high-dimensional nature of the transformations before and after selection (e.g., H = 400K and K = 50K), to avoid the time-consuming repeated evaluation of different subsets of transformations as existing solutions, we propose an efficient method to select all K transformations simultaneously, by *directly learning an indicating vector of length H to indicate whether to select the corresponding transformations or not.* The design of our *customized* indicating vector is described as follows.

Continuous-valued vector with learnable parameters. Ideally, the indicting vector should be in $\{0,1\}^H$, where the entries corresponding to the selected K transformations have a value of 1, and 0 for the other $H - K$ entries. However, such a discrete form is hard to learn because the main optimization schemes in machine learning (e.g., stochastic gradient descent, SGD [50]) are gradient-based [50], which requires the learnable parameters to be continuous. To tackle this issue, we define each α_h ($h \in \{1, \ldots, H\}$) of the indicating vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_H)$ as an output of the Sigmoid function [51] w.r.t. to a learnable parameter $\beta_h \in \mathbb{R}$, i.e.,

$$\alpha_h = \frac{1}{1 + e^{-\beta_h}}. \tag{6}$$

By doing so, we can learn $\boldsymbol{\alpha}$ by learning β_h for $h \in \{1, \ldots, H\}$ using any gradient-based optimization algorithm. Based on the definition of the Sigmoid function, α_h can be interpreted as the probability of selecting the transformation f_h, which serves as the basis for the hard selection in Eq. (5). For ease of explanation, we denote $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_H)$ and $\boldsymbol{\alpha} = 1/(1+e^{-\boldsymbol{\beta}})$.

Constraints on $\boldsymbol{\alpha}$. However, Eq. (6) is still insufficient to represent the ideal indicating vector discussed above, because it cannot guarantee that K transformations are well indicated for selection among the H candidates. As an extreme case, the learned $\boldsymbol{\alpha}$ may have similar values in all entries, tending not to make a selection due to overfitting. In this case, the hard selection of Eq. (5) will degenerate into random selection. To solve this problem, we identify the following two constraints on $\boldsymbol{\alpha}$. The first one *forces the indicating vector to be selective*, i.e., with the values close to 0 or 1 as far as possible, which is equivalent to force the distances between the reciprocal of all values of $\boldsymbol{\beta}$ and 0 to be as small as possible, i.e.,

$$\sum_{h=1}^{H} \left(\frac{1}{\beta_h}\right)^2 < \epsilon, \tag{7}$$

where $\epsilon > 0$ is a positive value that should be as small as possible. On this basis, the second constraint *ensures that the number of selected transformations is* K, i.e.,

$$\sum_{h=1}^{H} \alpha_h = K. \tag{8}$$

We propose an end-to-end optimization algorithm to efficiently learn the indicating vector defined in Eq. (6) with the constraints of Eqs.(7)-(8), which contributes to the superior performance of SuperRocket in terms of both training efficiency and accuracy. We elaborate on the algorithm in § IV-C.

C. Efficient Algorithm for Indicating Vector Learning

Unconstrained bi-level learning objective. With the introduced indicating vector $\boldsymbol{\alpha}$, the transformation selection problem formulated in Eq. (3) is converted to the problem of searching for an $\boldsymbol{\alpha}$ defined in Eq. (6) and constrained by Eqs. (7)-(8) that can contribute to the optimal classifier, i.e.,

$$\begin{aligned} &\arg\min_{\boldsymbol{\alpha}} \sum_{j} \mathcal{L}_{cls}(\hat{\boldsymbol{Y}}_j, y_j), \\ &s.t. \quad \sum_{h=1}^{H} (\frac{1}{\beta_h})^2 < \epsilon, \ \sum_{h=1}^{H} \alpha_h = K, \end{aligned} \tag{9}$$

where y_j is the ground-truth label of $\boldsymbol{X}_j$ and $\hat{\boldsymbol{Y}}_j$ is the class distribution predicted by the classifier $\mathcal{G}$, i.e.,

$$\hat{\boldsymbol{Y}}_j = \mathcal{G}(\boldsymbol{\alpha} \circ \boldsymbol{Z}_j) = \mathcal{G}(\boldsymbol{\alpha} \circ \mathcal{F}_H(\boldsymbol{X}_j)). \tag{10}$$

Considering that Eq. (9) is a constrained optimization problem that is difficult to solve efficiently [52], we *transform it into an unconstrained form that can be optimized end-to-end*. This is achieved *by adding regularization terms to the objective function to serve as a penalty for violating the constraints*. The unconstrained problem is formulated as:

$$\arg\min_{\boldsymbol{\alpha}} \sum_{j} \mathcal{L}_{cls}(\hat{\boldsymbol{Y}}_j, y_j) + \lambda \mathcal{L}_{reg}, \tag{11}$$

where

$$\mathcal{L}_{reg} = \sum_{h=1}^{H} (\frac{1}{\beta_h})^2 + (\sum_{h=1}^{H} \alpha_h - K)^2 \tag{12}$$

is the regularization loss and λ controls its importance.

Note that to evaluate the performance of $\boldsymbol{\alpha}$, the classifier $\mathcal{G}$ in Eq. (10) should be a trained model that has the optimal parameters for the features indicated by $\boldsymbol{\alpha}$, which is an implicit constraint for Eq. (11) (and also Eq. (9)). Denote $\boldsymbol{W}$ the matrix of parameters of $\mathcal{G}$. Eq. (11) can be rewritten as

$$\begin{aligned} &\arg\min_{\boldsymbol{\alpha}} \sum_{\mathcal{D}_{val}} \mathcal{L}_{cls}\big(\mathcal{G}(\boldsymbol{\alpha} \circ \boldsymbol{Z}_j; \boldsymbol{W}^*), y_j\big) + \lambda \mathcal{L}_{reg}, \\ &s.t. \quad \boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} \sum_{\mathcal{D}_{train}} \mathcal{L}_{cls}\big(\mathcal{G}(\boldsymbol{\alpha} \circ \boldsymbol{Z}_j; \boldsymbol{W}), y_j\big), \end{aligned} \tag{13}$$

where $\boldsymbol{Z}_j = \mathcal{F}_H(\boldsymbol{X}_j)$. Eq. (13) is an unconstrained bi-level optimization problem [23], where the inner optimization is to learn the optimal classifier parameters $\boldsymbol{W}^*$ given an indicating vector $\boldsymbol{\alpha}$, and the outer optimization aims to learn the indicating vector $\boldsymbol{\alpha}^*$ by which the classifier built using the selected transformations achieves the best classification performance (measured as the minimum classification loss on validation set). Note that while the classifier parameters $\boldsymbol{W}$ are optimized using the training dataset $\mathcal{D}_{train}$, the indicating vector is learned using the validation set $\mathcal{D}_{val}$ to avoid overfitting of the classifier that can degrade the learning performance of $\boldsymbol{\alpha}$. The strategy is motivated by hyper-parameter optimization that is well studied in the literature [53].

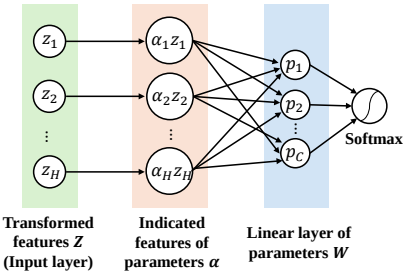


Fig. 4: Illustration of the end-to-end model combining the learnable indicating vector $\boldsymbol{\alpha}$ and classifier parameters $\boldsymbol{W}$.

Efficient end-to-end optimization. Although we have made the bi-level problem in Eq. (13) unconstrained, the nested form still makes it prohibitive to solve, because it needs to repeatedly perform the computationally expensive inner optimization to learn the classifier each step when updating $\boldsymbol{\alpha}$. For example, it takes $\mathcal{O}(HN \min\{H, N\})$ time to train the ridge regression classifier [49] commonly used by existing kernel-based methods for one time. Although there are heuristic approaches that can reduce the number of solving the inner optimization problem, such as random search [53], Bayesian optimization [54], and evolutionary algorithm [18], they still cannot avoid the time-consuming classifier training and may lead to sub-optimal solutions due to a lack of exploration of the necessary classifiers (see § VI-D).

To tackle the above issue, we propose *an efficient end-to-end optimization algorithm to avoid the time-consuming complete training of the classifier*. To this end, we combine the classifier parameters $\boldsymbol{W}$ and the indicating vector $\boldsymbol{\alpha}$ into *one learnable model* and use the second-order approximation strategy [55] to *approximately solve the inner problem with one training step*. In this way, the algorithm can optimize $\boldsymbol{W}$ and $\boldsymbol{\alpha}$ simultaneously. We detail the algorithm as follows.

We illustrate in Fig. 4 the model that combines the indicating vector $\boldsymbol{\alpha}$ and the classifier parameters $\boldsymbol{W}$. The model takes the transformed features $\boldsymbol{Z}$ as input and indicates the features using the indicating vector as $\boldsymbol{\alpha} \circ \boldsymbol{Z} = (\alpha_1 z_1, \ldots, \alpha_H z_H)$. Then, it feeds the transformed features into a linear classifier that consists of a linear (a.k.a. fully-connected) layer of the parameters $\boldsymbol{W}$ and a Softmax function [50] that maps the output of the linear layer to distributions of the C possible classes, which is consistent with the classifier used in the existing kernel-based TSC methods.

Given the combined model that contains the learnable parameters $\boldsymbol{\alpha}$ and $\boldsymbol{W}$, our optimization algorithm solves Eq. (13) by leveraging second-order approximation [55]. For ease of explanation, we simplify the notation of the loss functions in Eq. (13) to show only the variables that our algorithm will

learn, including $\boldsymbol{W}$ and $\boldsymbol{\alpha}$. Since the total loss in the outer optimization problem is a function with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{W}^*$, where $\boldsymbol{W}^*$ depends on $\boldsymbol{\alpha}$, we abbreviate it as

$$\sum_{\mathcal{D}_{val}} \mathcal{L}_{cls}\big(\mathcal{G}(\boldsymbol{\alpha} \circ \boldsymbol{Z}_j; \boldsymbol{W}^*), y_j\big) + \lambda \mathcal{L}_{reg} = \mathcal{L}_{val}(\boldsymbol{W}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}). \tag{14}$$

Similarly, the loss in the inner problem is abbreviated as

$$\sum_{\mathcal{D}_{train}} \mathcal{L}_{cls}\big(\mathcal{G}(\boldsymbol{\alpha} \circ \boldsymbol{Z}_j; \boldsymbol{W}), y_j\big) = \mathcal{L}_{train}(\boldsymbol{W}, \boldsymbol{\alpha}). \tag{15}$$

To avoid completely optimizing the inner problem, following [55], we adopt the commonly used SGD scheme [50] to update $\boldsymbol{W}$ for only one training step and approximate the solution $\boldsymbol{W}^*$ using the updated parameters, i.e.,

$$\boldsymbol{W}^*(\alpha) \approx \boldsymbol{W}^{OSU} = \boldsymbol{W} - \eta_{\boldsymbol{W}} \nabla_{\boldsymbol{W}} \mathcal{L}_{train}(\boldsymbol{W}, \boldsymbol{\alpha}), \tag{16}$$

where $\boldsymbol{W}^{OSU}$ is the classifier parameters after the one-step update (OSU) and $\eta_{\boldsymbol{W}}$ is the learning rate of the inner problem.

Given the approximate solution of the optimal classifier parameters in Eq. (16), the gradient of $\mathcal{L}_{val}(\boldsymbol{W}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$ that is required for updating $\boldsymbol{\alpha}$ becomes

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{W}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \approx \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{W}^{OSU}, \boldsymbol{\alpha}). \tag{17}$$

Applying the chain rule [56] to $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{W}^{OSU}, \boldsymbol{\alpha})$, we have

$$\begin{aligned} \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{W}^{OSU}, \boldsymbol{\alpha}) &= \frac{\partial \mathcal{L}_{val}(\boldsymbol{W}^{OSU}, \boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} \\ &- \eta_{\boldsymbol{W}} \nabla^2_{\boldsymbol{W},\boldsymbol{\alpha}} \mathcal{L}_{train}(\boldsymbol{W}, \boldsymbol{\alpha}) \frac{\partial \mathcal{L}_{val}(\boldsymbol{W}^{OSU}, \boldsymbol{\alpha})}{\partial \boldsymbol{W}^{OSU}}. \end{aligned} \tag{18}$$

Note that directly computing Eq. (18) takes $\mathcal{O}(||\boldsymbol{\alpha}|| \cdot ||\boldsymbol{W}||)$ time dominated by the term $\nabla^2_{\boldsymbol{W},\boldsymbol{\alpha}} \mathcal{L}_{train}(\boldsymbol{W}, \boldsymbol{\alpha})$. To improve efficiency, we adopt the finite difference approximation [55] to compute this term, which reduces the time to $\mathcal{O}(||\boldsymbol{W}|| + ||\boldsymbol{\alpha}||)$.

It is also noteworthy that $\boldsymbol{\alpha}$ is dependent on the learnable parameter $\boldsymbol{\beta}$ as defined in Eq. (6). Thus, updating $\boldsymbol{\alpha}$ requires updating $\boldsymbol{\beta}$. Based on the chain rule [56] and Eq. (17), we can derive the gradient of $\mathcal{L}_{val}(\boldsymbol{W}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha})$ with respect to $\boldsymbol{\beta}$ as:

$$\begin{aligned} \nabla_{\boldsymbol{\beta}} \mathcal{L}_{val}(\boldsymbol{W}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) &= \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{W}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \nabla_{\boldsymbol{\beta}} \boldsymbol{\alpha} \\ &\approx \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{W}^{OSU}, \boldsymbol{\alpha}) \nabla_{\boldsymbol{\beta}} \boldsymbol{\alpha}. \end{aligned} \tag{19}$$

Given the derived gradients, our proposed optimization algorithm alternately updates $\boldsymbol{\beta}$ (and also $\boldsymbol{\alpha}$) and $\boldsymbol{W}$ at each training step following the standard minibatch SGD [50] to learn the indicating vector, as outlined in lines 6-9 in Algo. 1.

D. Summary of SuperRocket

Based on the above discussion, we summarize the proposed SuperRocket method in Algo. 1. First, H transformations are generated by a kernel-based method (line 2) to transform the raw time series into features (lines 3-4). Next, the transformed data are randomly divided into a training set and a validation set (line 5), and the optimization algorithm works to learn the indicating vector that indicates the K transformations with the highest quality by iteratively updating $\boldsymbol{\beta}$ (and also $\boldsymbol{\alpha}$) and $\boldsymbol{W}$ to solve the bi-level problem in Eq. (13) (lines 6-9). Then a hard selection is made based on the learned indicating vector to retain the K best transformations $\mathcal{F}^*_K$ (line 10). Finally, the classifier $\mathcal{G}$ is built using the selected transformations $\mathcal{F}^*_K$ (lines 11-12). The returned $\mathcal{F}^*_K$ and $\mathcal{G}$ (line 13) are used in the same way as the standard kernel-based methods [8]–[10] to predict the class labels of the unseen time series.

Algorithm 1: SuperRocket

Input: Training set $\mathcal{D}_{train}$, number of transformations H, K, learning rate $\eta_{\boldsymbol{\beta}}$, $\eta_{\boldsymbol{W}}$, regularization coefficient λ, maximum training epoch E, batch size B
Output: Selected transformations $\mathcal{F}^*_K$, classifier $\mathcal{G}$

1 Initialize all parameters;
2 $\mathcal{F}_H = generate\_transformations(H)$;
3 $N = |\mathcal{D}_{train}|$;
4 Transform all $\boldsymbol{X}_j, j \in \{1, \ldots, N\}$ in $\mathcal{D}_{train}$ into $\boldsymbol{Z}_j$ using $\mathcal{F}_H$ as $\boldsymbol{Z}_j = \mathcal{F}_H(\boldsymbol{X}_j)$;
5 Randomly divide $\{(\boldsymbol{Z}_j, y_j)\}_{j=1}^N$ into training and validation sets;
6 **for** $epoch = 1$ *to* E **do**
  /* Iterate over minibatches of size B */
7 Update $\boldsymbol{\beta}$ as $\boldsymbol{\beta} = \boldsymbol{\beta} - \eta_{\boldsymbol{\beta}} \nabla_{\boldsymbol{\beta}} \mathcal{L}_{val}(\boldsymbol{W}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha})$ based on Eqs. (18)-(19);
8 Update $\boldsymbol{\alpha}$ based on Eq. (6);
9 Update $\boldsymbol{W}$ as $\boldsymbol{W} = \boldsymbol{W} - \eta_{\boldsymbol{W}} \nabla_{\boldsymbol{W}} \mathcal{L}_{train}(\boldsymbol{W}, \boldsymbol{\alpha})$ based on Eq. (16);
10 Select the transformations $\mathcal{F}^*_K$ based on Eq. (5);
11 Transform all $\boldsymbol{X}_j, j \in \{1, \ldots, N\}$ in $\mathcal{D}_{train}$ into $\boldsymbol{Z}_j$ using $\mathcal{F}^*_K$ as $\boldsymbol{Z}_j = \mathcal{F}^*_K(\boldsymbol{X}_j)$;
12 Build the classifier $\mathcal{G}$ using $\{(\boldsymbol{Z}_j, y_j)\}_{j=1}^N$;
13 **return** $\mathcal{F}^*_K$, $\mathcal{G}$

V. Theoretical analysis

In this section, we perform theoretical analysis on the efficiency and accuracy of the proposed SuperRocket.

A. Complexity analysis

Based on existing studies [8]–[10], each transformation in Eq. (1) takes $\mathcal{O}(T)$ time, where T is the length of the time series. Thus, the time complexity of transforming all time series using $\mathcal{F}_H$ is $\mathcal{O}(HNT)$. In the linear classifier, $\boldsymbol{W}$ is a matrix of size $(H+1) \times C$, where C is a negligible constant that represents the number of classes. Thus, the proposed indicating vector learning algorithm has a time complexity of $\mathcal{O}(EHN)$. $\mathcal{F}^*_K(\boldsymbol{X}_j)$ can be obtained in $\mathcal{O}(NK)$ time for all time series by selecting from $\mathcal{F}_H(\boldsymbol{X}_j)$, while training the classifier $\mathcal{G}$ using the resulting K-dimensional features takes $\mathcal{O}(KN \min\{K, N\})$ time, when $\mathcal{G}$ is a ridge regression classifier following the existing kernel-based methods [8]–[10]. Since $H >> E$, the time complexity of Algo. 1 can be simplified to $\mathcal{O}\big(N(HT + K \min\{K, N\})\big)$. Similarly, we can derive the space complexity of Algo. 1 as $\mathcal{O}(HN)$.

Moreover, we compare the time and space complexity of the transformation selection methods in our SuperRocket and the existing solutions (leaving outside the common process of transforming the time series and building the classifier after selection). As Table I shows, POCKET has poor scalability with respect to H for both time and space, which prevents it from addressing a large number of transformations. S-Rocket,

TABLE I: Time and space complexity of the transformation selection methods in the existing work and our SuperRocket.

Complexity	S-Rocket [17]	POCKET [19]	D-Rocket [21]	SuperRocket (Ours)
Time	$\mathcal{O}(SEHN)$	$\mathcal{O}(N^3+EH(H+N))$	$\mathcal{O}(EHN\min\{H,N\})$	$\mathcal{O}(EHN)$
Space	$\mathcal{O}(HN)$	$\mathcal{O}(H^2N)$	$\mathcal{O}(HN)$	$\mathcal{O}(HN)$

D-Rocket and our proposed SuperRocket have the same space complexity. However, D-Rocket has a higher time complexity with respect to N or H, while S-Rocket can also be less time-efficient than our proposal, especially when the additional variable S that represents the number of solutions for the selection problem [17] dominates the time consumption.

B. Theoretical guarantee of classification accuracy.

Since the ridge regression adopts $L2$ regularization to limit the complexity of the classifier, we can assume that the parameters are bounded, i.e., $||\boldsymbol{W}|| \leq B_W$ where $B_W > 0$ is the upper bound. Considering that the real-world time series, e.g., the sensor readings, usually have values in closed ranges and the features are usually preprocessed via standardization or normalization before input to the classifier, it is also a practical assumption that the features $\boldsymbol{Z} = \mathcal{F}_K(\boldsymbol{X})$ are bounded, i.e., $||\boldsymbol{Z}|| \leq B_Z$ for a constant $B_Z > 0$.

Meanwhile, the one-step approximation in Eq. (16) can empirically reach a fixed point with a suitable choice of $\eta_{\boldsymbol{W}}$ [55]. Therefore, we make the practical assumption that $\boldsymbol{W} \approx \boldsymbol{W}^*(\boldsymbol{\alpha})$ after the iteration of Algo. 1. With a suitable setting of λ, it is a fair assumption that the learned $\boldsymbol{\alpha}^*$ approaches $\boldsymbol{\alpha}_{\mathcal{F}_K^*} \in \{0,1\}^H$, where $\boldsymbol{\alpha}_{\mathcal{F}_K^*,h} = 1$ if $f_h \in \mathcal{F}_K^*$ and 0 otherwise, $\forall h \in \{1,\ldots,H\}$. Thus, the generalization error of SuperRocket, defined following Eq. (13) as

$$err(\boldsymbol{\alpha}, \boldsymbol{W}) \triangleq \mathbb{E}[\mathcal{L}_{cls}\Big(\mathcal{G}\big(\boldsymbol{\alpha} \circ \boldsymbol{Z}; \boldsymbol{W}(\boldsymbol{\alpha})\big), y\Big)], \tag{20}$$

is equivalent to the form as follows:

$$err(\mathcal{F}_K) \triangleq \mathbb{E}[\mathcal{L}_{cls}\Big(\mathcal{G}\big(\mathcal{F}_K(\boldsymbol{X})\big), y\Big)], \tag{21}$$

where $\boldsymbol{X}$ is the independent and identically distributed time series sampled from a distribution and y is its label.

Under the above assumptions, we propose *an upper bound on the difference in generalization error between using the transformations $\mathcal{F}_K^*$ selected by SuperRocket and using the oracle solution of transformation selection (denoted as $\mathcal{F}_K^o$) that contributes to the minimum classification error on the test data*, which is formulated as:

Proposition 1. For all $\delta > 0$, with probability at least $1-\delta$,

$$err(\mathcal{F}_K^*) - err(\mathcal{F}_K^o) \leq \mathcal{O}\big(\frac{\binom{H}{K}^{1/K}\sqrt{\pi K}}{2N} + \sqrt{\frac{\log\frac{1}{\delta}}{N}}\big). \tag{22}$$

Proof of Proposition 1. For the oracle solution $\mathcal{F}_K^o$, we have $0 \leq err(\mathcal{F}_K^o) \leq \mathcal{F}_K^*$. Then, based on Theorem 5 of [57], we obtain

$$err(\mathcal{F}_K^*) - err(\mathcal{F}_K^o) \leq err(\mathcal{F}_K^*) \leq \mathcal{O}\big(\mathcal{R}(\mathcal{H}_\mathcal{L}) + \sqrt{\frac{\log\frac{1}{\delta}}{N}}\big), \tag{23}$$

where $\mathcal{R}(\cdot)$ represents the Rademacher complexity and $\mathcal{H}_\mathcal{L} = \{\mathcal{L}_{cls}\Big(\mathcal{G}\big(\mathcal{F}_K(\cdot)\big), \cdot\Big) : \boldsymbol{X} \times y \mapsto \mathbb{R} \mid \forall \mathcal{F}_K, \mathcal{G}\}$ is the function class that corresponds to the classification losses regarding all possible transformations $\mathcal{F}_K$ and classifier $\mathcal{G}$. By applying the Dudley's entropy integral [57] to $\mathcal{R}(\mathcal{H}_\mathcal{L})$, we have

$$\mathcal{R}(\mathcal{H}_\mathcal{L}) \leq \mathcal{O}\big(\frac{1}{N}\int_0^\infty \sqrt{\log N_{cover}(\mathcal{H}_\mathcal{L}, \epsilon)}\ d\epsilon\big), \tag{24}$$

where $N_{cover}(\mathcal{H}_\mathcal{L}, \epsilon)$ represents the covering number of $\mathcal{H}_\mathcal{L}$ at radius ϵ. The loss function $\mathcal{L}_{cls}(\cdot,\cdot)$ of the ridge regression [49] corresponds to the square of difference between the prediction and the encoded label, that is, $\mathcal{L}_{cls}(\hat{\boldsymbol{Y}}, y) = ||\boldsymbol{Y} - \hat{\boldsymbol{Y}}||^2$, where $\hat{\boldsymbol{Y}} = \mathcal{G}\big(\mathcal{F}_K(\boldsymbol{X})\big) = \boldsymbol{Z}\boldsymbol{W}$ is the predicted class distribution and $\boldsymbol{Y}$ is the encoding of the label y as $\boldsymbol{Y}_c = 1$ if $y = c$ and -1 otherwise for $c = \{1,\ldots,C\}$.

Note that the loss function $\mathcal{L}_{cls}(\cdot,\cdot)$ is Lipschitz continuous with respect to the prediction $\hat{\boldsymbol{Y}}$ given the bounded inputs as $||\hat{\boldsymbol{Y}}|| = ||\boldsymbol{Z}\boldsymbol{W}|| \leq ||\boldsymbol{Z}|| \cdot ||\boldsymbol{W}| \leq B_Z B_M$ and $||\boldsymbol{Y}|| = \sqrt{C}$, i.e., the following inequality holds for any two predictions $\hat{\boldsymbol{Y}}_1$ and $\hat{\boldsymbol{Y}}_2$:

$$\begin{aligned}
&\big|\big|\mathcal{L}_{cls}(\hat{\boldsymbol{Y}}_1, y) - \mathcal{L}_{cls}(\hat{\boldsymbol{Y}}_2, y)\big|\big| \\
=&\big|\big|\ ||\boldsymbol{Y} - \hat{\boldsymbol{Y}}_1||^2 - ||\boldsymbol{Y} - \hat{\boldsymbol{Y}}_2||^2\ \big|\big| \\
=&\ \big|\big|(\hat{\boldsymbol{Y}}_1 - \hat{\boldsymbol{Y}}_2)(\hat{\boldsymbol{Y}}_1 + \hat{\boldsymbol{Y}}_2 - 2\boldsymbol{Y})\big|\big| \\
\leq&\ \big|\big|\hat{\boldsymbol{Y}}_1 + \hat{\boldsymbol{Y}}_2 - 2\boldsymbol{Y}\big|\big| \cdot \big|\big|\hat{\boldsymbol{Y}}_1 - \hat{\boldsymbol{Y}}_2\big|\big| \\
\leq&\ 2 \cdot \big|\big|\hat{\boldsymbol{Y}}_1\big|\big| \cdot \big|\big|\hat{\boldsymbol{Y}}_2\big|\big| \cdot \big|\big|\boldsymbol{Y}\big|\big| \cdot \big|\big|\hat{\boldsymbol{Y}}_1 - \hat{\boldsymbol{Y}}_2\big|\big| \\
\leq&\ L\big|\big|\hat{\boldsymbol{Y}}_1 - \hat{\boldsymbol{Y}}_2\big|\big|,
\end{aligned} \tag{25}$$

where $L = 2B_Z^2 B_M^2\sqrt{C}$ is the Lipschitz constant of the loss function. Moreover, the classifier $\mathcal{G}(\cdot) = \boldsymbol{Z}\boldsymbol{W}$ is Lipschitz continuous with respect to the parameters $\boldsymbol{W}$, i.e.,

$$||\boldsymbol{Z}\boldsymbol{W}_1 - \boldsymbol{Z}\boldsymbol{W}_2|| \leq ||\boldsymbol{Z}|| \cdot ||\boldsymbol{W}_1 - \boldsymbol{W}_2|| \leq B_Z||\boldsymbol{W}_1 - \boldsymbol{W}_2||, \tag{26}$$

for any $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. Recall that there are $\binom{H}{K}$ possible solutions of the transformations $\mathcal{F}_K$. Thus, under the conditions of Eqs. (25)-(26), the covering number $N_{cover}(\mathcal{H}_\mathcal{L}, \epsilon)$ is bounded as:

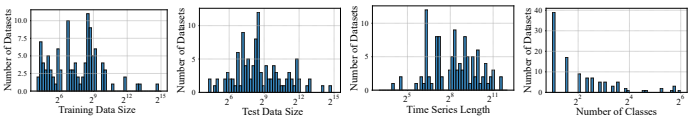
$$N_{cover}(\mathcal{H}_\mathcal{L}, \epsilon) \leq \mathcal{O}(\frac{\binom{H}{K}}{\epsilon^K}). \tag{27}$$

Without loss of generality, we set the upper bound of $N_{cover}(\mathcal{H}_\mathcal{L}, \epsilon)$ as A/ϵ^K, where $A = C_A\binom{H}{K}$ for a constant $C_A > 0$. Note that $N_{cover}(\mathcal{H}_\mathcal{L}, \epsilon) \geq 1$ by definition. Thus, we have $0 < \epsilon \leq A^{1/K}$. By substituting the two conditions to Eq. (24), we obtain

$$\mathcal{R}(\mathcal{H}_\mathcal{L}) \leq \frac{1}{N}\int_0^{A^{1/K}} \sqrt{\log A - K\log\epsilon}\ d\epsilon). \tag{28}$$

Let $t = \log A - K\log\epsilon$ and $u = t/K$, the integral of Eq. (28) becomes

$$\begin{aligned}
&\int_0^{A^{1/K}} \sqrt{\log A - K\log\epsilon}\, d\epsilon \\
=&\int_0^\infty \sqrt{t} \cdot \frac{A^{1/K}}{K} e^{-t/K}\ dt \\
=&\frac{A^{1/K}}{K}\int_0^\infty \sqrt{Ku} \cdot e^{-u} \cdot K\ du \\
=&A^{1/K}\sqrt{K}\int_0^\infty u^{1/2} e^{-u}\ du.
\end{aligned} \tag{29}$$



(a) Train size (b) Test size (c) Series length (d) # of classes

Fig. 5: Distribution of main statistics of the 109 UCR datasets.

By applying the gamma function [58], we have $\int_0^\infty u^{1/2}e^{-u}\,du = \Gamma(3/2) = \sqrt{\pi}/2$. Then, by substituting Eqs. (28)-(29) to Eq. (24) and ignoring the constant C_A, we obtain

$$err(\mathcal{F}_K^*) - err(\mathcal{F}_K^o) \leq \mathcal{O}\big(\frac{\binom{H}{K}^{1/K}\sqrt{\pi K}}{2N} + \sqrt{\frac{\log\frac{1}{\delta}}{N}}\big). \quad (30)$$

□

Proposition 1 indicates that the proposed SuperRocket is error-bounded in terms of selecting the best transformations. Note that the oracle solution $\mathcal{F}_K^o$ depends on the numbers of the total and selected transformations, i.e., H and K respectively, which are taken as hyper-parameters to control both accuracy and efficiency of our algorithm (see § VI-E).

VI. Evaluation

We conduct extensive experiments to evaluate the performance of the proposed SuperRocket, comparing it with a wide range of transformation selection and TSC approaches. The experiments aim to address four main research questions (RQs): ***RQ1:*** *How does SuperRocket perform compared to existing transformation selection solutions (§ VI-B)?* ***RQ2:*** *Is SuperRocket faster than the SOTA TSC methods while achieving competitive accuracy (§ VI-C)?* ***RQ3:*** *Are the key components of SuperRocket effective (§ VI-D)?* ***RQ4:*** *What is the impact of the key hyper-parameters (§ VI-E)?*

A. Experimental Setup

Datasets. To achieve consistency with baseline methods [7], [8], [10], [41], the experiments are mainly conducted on the popular 109 univariate datasets from the UCR archive [3] (without specification). The data are collected from various applications (e.g., ECG and motion recognition) and are highly heterogeneous in sample size, series length, number of classes, etc. The main statistics are summarized in Fig. 5. To show the generalization of our method, we also compare it with competitive baselines using 26 multivariate datasets from UEA [59] (see § VI-C).

Compared methods. We consider two types of baseline, including transformation selection methods and TSC approaches, to investigate RQ1 and RQ2 respectively.

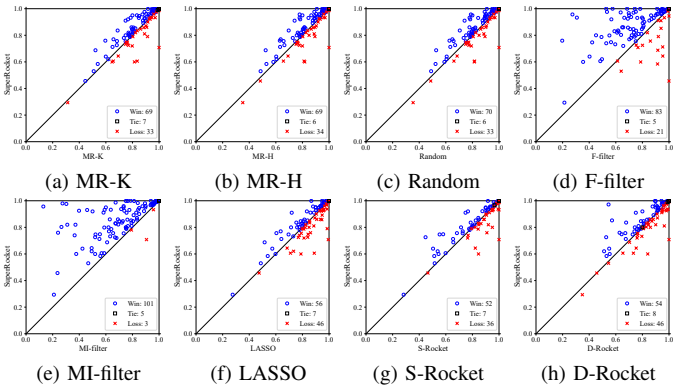
1) Transformation selection methods. To verify the superiority of SuperRocket, we compare it with a variety of methods that can be used for transformation selection, including three representative general variable selection approaches, two solutions tailored for kernel-based transformation, and an alternative random selection strategy. Note that we leave outside the tailored solution POCKET [19] because it runs out of memory on our 256GB memory server due to its quadratic space complexity with respect to the number of transformations H (e.g., it requires about 1.16TB of memory for $H = 400$K, which is too large to afford for a common server). We briefly describe the baseline approaches as follows.

- **F-filter** is a filter method that uses the ANOVA F-value [60] to score the importance between the features extracted with each transformation and the class labels, and select the transformations with the highest scores;
- **MI-filter** is similar to F-filter but uses mutual information (MI) [13] as the scoring metric;
- **LASSO** [20] is an embedded method that applies regression analysis and L_1-regularization to automatically learn the non-essential variables (i.e., transformations) for which the regression coefficients are close to 0;
- **S-Rocket (S-R)** [17] is a wrapper method that applies an evolutionary algorithm to iteratively search for the transformations that contribute to high accuracy;
- **D-Rocket (D-R)** [21] is an improved wrapper method designed based on stepwise-backward selection techniques. It iteratively estimates the importance of the remaining transformations using classifier parameters and greedily eliminates non-essential transformations;
- **Random** is a straightforward solution for transformation selection that directly selects K transformations from the H candidates at random, which serves as a lower bound for evaluating the selection methods.

2) TSC approaches. To investigate the performance of our SuperRocket in terms of efficiency and accuracy, we compare it with the SOTA TSC approaches of various types, including

- *Kernal-based* **MultiRocket (MR)** [10] that uses multiple random convolutional kernel-based transformations in conjunction with a ridge regression classifier for fast TSC;
- *Shapelet-based* **RDST** [34] that trains a ridge classifier using the features transformed from random dilated shapelets, and **ShapeFormer (SF)** [35] that capture shapelet-based features with class-specific and generic Transformer modules;
- *Interval-based* **DrCIF** [7] that builds a random forest classifier using multiple statistics computed over the time series segments in random intervals, and **QUANT** [28] that uses the distribution of values in each interval as features and extremely randomized trees [61] as the classifier;
- *Dictionary-based* **TDE** [37] that takes the frequency of the symbolic patterns appear in the time series as the features and uses an ensemble of nearest neighbor classifiers with a customized distance metric as the classifier, and **MR-Hydra** [39] that extracts features by counting the MultiRocket-style kernels representing the closest and farthest match with the time series to build a ridge classifier;
- *Deep-learning-based* **InceptionTime (IT)** [44] that designs a deep neural network for TSC based on the inception block, **OSCNN** [42] that proposes a lightweight multi-scale convolutional neural network for TSC, and **TS-GAC** [43] that designs a graph-aware contrasting method to capture the temporal and spatial features of the time series;
- *Hybrid* **HC2** [7] that is a meta-ensemble combining the class distributions predicted by different types of TSC models.

Implementations. We implement SuperRocket using Python



(a) MR-K (b) MR-H (c) Random (d) F-filter

(e) MI-filter (f) LASSO (g) S-Rocket (h) D-Rocket

Fig. 6: 1v1 accuracy comparison of SuperRocket against existing transformation selection solutions and two MR variants.

3.8 where the indicating vector learning algorithm is implemented using PyTorch 1.11.0. The experiments are run on a Ubuntu server with an Intel(R) Xeon(R) Platinum 8260 CPU and 256GB of memory. We adopt the SOTA kernel-based method MultiRocket [10] to generate the transformation candidates. We set $H = 400$K, a considerable value to generate enough high-quality transformations (see § VI-D), and $K =$ 50K that is consistent with MultiRocket. We set $\eta_{\boldsymbol{w}} = 100$, $\eta_{\boldsymbol{\beta}} = 0.01$, $\lambda = 0.01$, $E = 10$, $B = 32$. The above settings are used for all experiments without specification.

We implement F-filter, MI-filter and LASSO using the scikit-learn library [62] with the default configuration. The threshold of LASSO is selected from $\{10^2, 10^1, \ldots, 10^{-7}\}$ through 3-fold cross-validation. DrCIF, TDE, and HC2 are implemented using the sktime library [63], and the aeon toolkit [64] is used for reproducing InceptionTime. Other approaches are reproduced using their open source codes. For a fair comparison, we set H, K, and the transformation generation method the same as in SuperRocket, while other settings follow the authors' recommendation.

Evaluation metrics. Standard metrics are used for evaluation. We use the *accuracy rate (accuracy)* to evaluate the classification accuracy, which is measured as the proportion of samples that are correctly predicted (over the test dataset). For efficiency evaluation, we measure the *total running time* of the TSC algorithm for training the TSC model, including both feature extraction and classifier building processes. All results are averaged over 3 restarts. To complete the evaluation in a reasonable duration, we set a considerable time limit of 64 hours for each single run of a method on a dataset and omit the runs that are longer than the limit.

Since our experiments use abundant datasets, directly showing the accuracy scores is complicated to compare and is also a waste of space. Therefore, we adopt two commonly used tools to better analyze the results, including the *one-versus-one (1v1) comparison plot* that compares our proposal with the counterparts on each dataset [65], and the *critical difference (CD) diagram* based on the Wilcoxon-Holm test [5] that shows the average accuracy ranking of each algorithm on all tested datasets, where a higher ranking indicates better performance, and each thick horizontal line groups a set of methods that have **no significant difference** with a statistical level of 0.01.

B. Comparison against Transformation Selection Solutions

To investigate RQ1, we compare our SuperRocket with existing transformation selection solutions in terms of accuracy and efficiency. We also include two variants of MR, denoted MR-K and MR-H, that directly use the generated K and H transformations to train the classifier without selection, which serves as reference points for evaluating the effectiveness of transformation selection. Note that S-Rocket and D-Rocket run out of memory on 14 and 1 datasets, respectively, so we report the results of the remaining datasets for them.

Accuracy comparison. Fig. 6 shows the one-versus-one accuracy comparison between SuperRocket and the baseline transformation selection methods, from which we have three main findings. Firstly, SuperRocket performs better in most cases than the straightforward solutions that directly use the generated transformations without selection (Fig. 6a and 6b), indicating that it is necessary to make a selection on the transformations for accuracy improvement. Secondly, our SuperRocket outperforms Random (Fig. 6c), F-filter (Fig. 6d) and MI-Filter (Fig. 6e) for a large portion of the datasets. The results are as expected because the simple random and filter methods do not consider the correlation between different transformations. Thirdly, SuperRocket is slightly better than LASSO (Fig. 6f), S-Rocket (Fig. 6g), and D-Rocket (Fig. 6h), showing that it achieves higher accuracy on more than one-half of the datasets. The results show the superiority of our SuperRocket in selecting high-quality transformations.

To quantify the difference in accuracy, we plot the CD diagram for SuperRocket against the competitors in Fig. 7. As can be seen, SuperRocket achieves the highest averaged accuracy ranking among the transformation selection solutions. Although D-Rocket, LASSO and S-Rocket are not statistically significantly worse than SuperRocket, we show later that *our proposal is orders of magnitude faster than these three approaches*. It is also noteworthy that SuperRocket is *the only one that significantly outperforms* the baseline MR-H, MR-K and Random that make no or random selection.

Efficiency comparison. Table II shows the one-versus-one training time comparison between SuperRocket and the transformation selection competitors and the total time of each method on all datasets. As expected, MR-K is faster than any transformation selection method, with the cost of low accuracy compared to the well-performing selection methods discussed above. SuperRocket is more efficient than all transformation selection solutions except for Random. However, Random is only 1.09× faster than ours, which indicates that the proposed SGD-based optimization algorithm is highly efficient in selecting the transformations. Compared to the three approaches that are not significantly less accurate than our proposal, SuperRocket achieves 30.60× speedup over LASSO, and is more than 341.40× and 11.22× faster than S-Rocket and D-Rocket, respectively. Moreover, SuperRocket is only 4.08× slower than MR-K and is even 1.36× faster than MR-H, while it is statistically significant that SuperRocket is

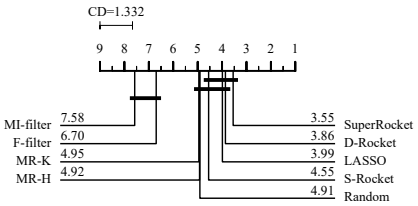


Fig. 7: CD diagram for SuperRocket against existing transformation selection solutions and two MR variants.

TABLE II: Training time comparison for SuperRocket against existing transformation selection solutions and the MR variants over 109 datasets. Bold and underlined values indicate the best and second best among the selection methods, respectively.

Metric	MR variants		Transformation selection method						
	MR-K	MR-H	Random	F-filter	MI-filter	LASSO	S-Rocket	D-Rocket	**SuperRocket**
Wins/Ties/Loses	0/0/109	33/0/76	20/0/89	92/0/17	109/0/0	109/0/0	95/0/0	108/0/0	-
Total time (min)	17.77	98.41	**66.53**	102.96	8111.18	2218.83	> 24757.65	> 813.65	<u>72.52</u>
Speedup	0.25×	1.36×	0.92×	1.42×	111.85×	30.60×	> 341.40×	> 11.22×	-

TABLE III: Training time comparison for SuperRocket against different types of SOTA TSC methods.

Metric	MR	RDST	SF	DrCIF	QUANT	TDE	MR-Hydra	IT	OSCNN	TS-GAC	HC2	**SuperRocket**
Wins/Ties/Loses	0/0/109	63/0/46	109/0/0	109/0/0	0/0/109	109/0/0	99/0/10	109/0/0	109/0/0	47/0/62	109/0/0	-
Total time (h)	<u>0.296</u>	1.83	>719.36	120.65	**0.115**	76.91	17.13	63.82	7.96	1.04	356.85	*1.21*
Speedup	0.25×	1.51×	594.51×	99.83×	0.10×	63.63×	14.16×	52.81×	6.59×	0.86×	295.25×	-

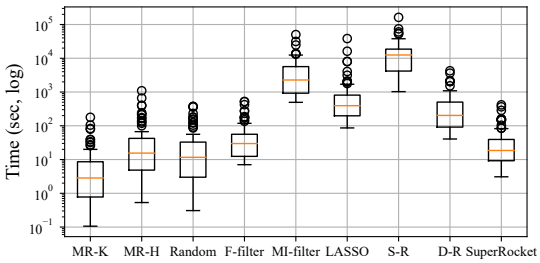


Fig. 8: Training time distributions of different transformation selection solutions and the MR variants without selection.

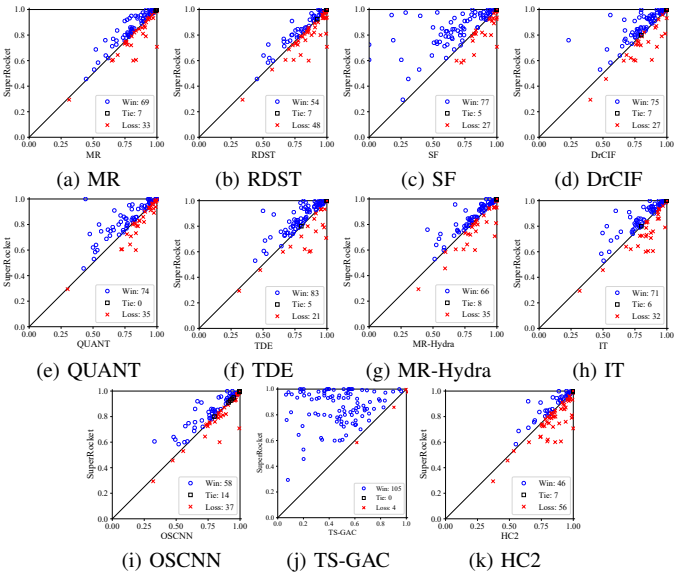


Fig. 9: 1v1 accuracy comparison of SuperRocket against different types of SOTA TSC methods.

more accurate. We also show the training time distributions of the transformation selection methods in Fig. 8. It is observed that our SuperRocket spends less than 100 seconds for most datasets and never runs more than 1000 seconds, which is comparable to MR-H, Random, and F-filter, and is clearly better than MI-filter, LASSO, S-Rocket, and D-Rocket.

In summary, the experimental results validate the superiority of our SuperRocket in *achieving higher accuracy with more efficient transformation selection.*

C. Comparison against TSC Methods

In this section, we further investigate how our SuperRocket performs compared to the SOTA TSC methods (RQ2).

Accuracy comparison. As Fig. 9 shows, SuperRocket consistently outperforms the existing TSC methods on most datasets, except for HC2 where the number of datasets that SuperRocket achieves higher accuracy is slightly less than half. However, we can observe from Fig. 10 that there is *no statistically significant difference* between SuperRocket and the SOTA approach HC2 in terms of mean accuracy ranking over the 109 datasets, which means that SuperRocket can also achieve SOTA accuracy. Note that HC2 is a general meta-ensemble framework that works by combining different types of TSC algorithm. Thus, it can also integrate SuperRocket to make use of our proposal for further improvement in future work. RDST, MR-Hydra and OSCNN are not significantly less accurate than ours (but are much less efficient as shown below), while the remaining competitors are all inferior to ours with statistical significance. Moreover, it is observed that the kernel-based MR that uses the generated transformations without selection cannot achieve the same statistical level as that of HC2 and SuperRocket in terms of mean accuracy ranking, which also indicates the effectiveness of the proposed transformation selection approach.

Efficiency comparison. We report the overall training time of different TSC methods in Table III. As can be seen, MR, QUANT and TS-GAC are faster in total than SuperRocket. However, they have been shown to be less accurate than SuperRocket on most datasets (see Fig. 9a, 9e, 9j) and achieve a mean ranking of accuracy statistically significantly worse than SuperRocket and also other competitive methods (see Fig. 10). Apart from these three methods, SuperRocket is consistently more efficient than the other competitors in total, especially OSCNN, RDST, MR-Hydra and HC2, the approaches that are as accurate as SuperRocket in statistics. Compared to HC2 which has the highest average accuracy ranking, SuperRocket *not only achieves the same level of accuracy (see Fig. 10), but also has 295.25× faster training speed*. Fig. 11 shows the training time distributions of the TSC methods. We can observe that SuperRocket rarely runs more

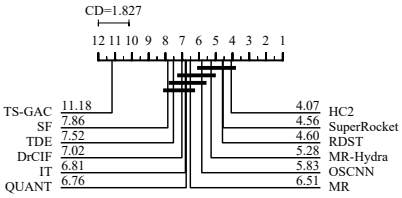


Fig. 10: CD diagram for SuperRocket against different types of SOTA TSC methods.

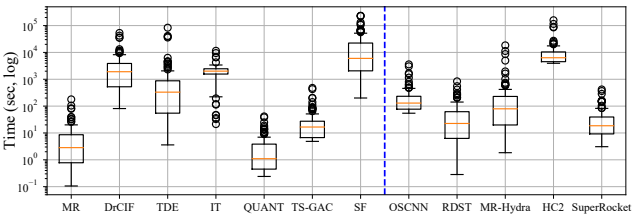


Fig. 11: Training time distributions of different TSC methods. Boxes in the right of the blue dashed line indicate the methods that have the highest level of mean accuracy in statistics as Fig. 10 shows.

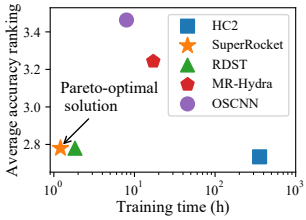


Fig. 12: Average accuracy ranking against training time of the TSC solutions that have the highest level of mean accuracy in statistics.

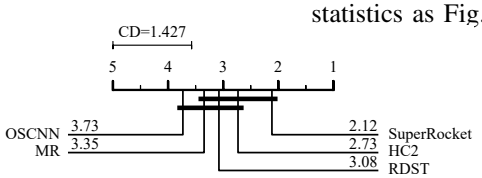


Fig. 13: CD diagram for SuperRocket against the baseline TSC methods on UEA multivariate datasets.

than 100 seconds on one dataset. In comparison, the baseline methods spend more than 100 seconds (e.g., OSCNN) and even 1000 seconds (e.g., HC2) on most cases.

To quantitatively evaluate the performance in terms of both efficiency and accuracy, we show the average accuracy ranking against the total training time of the significantly more accurate competitors in Fig. 12. We observe that no method can be faster than SuperRocket without degrading accuracy or more accurate than it without sacrificing efficiency, indicating that SuperRocket is a Pareto-optimal solution [66] for optimizing efficiency and accuracy.

The results verify that SuperRocket can *achieve competitive accuracy (without statistically significant difference) and superior training efficiency (295.25× faster) compared to the SOTA TSC method HC2*, providing a positive answer to RQ2.

To further assess the generalization of SuperRocket, we compare it with the baseline MR and the competitive methods HC2, RDST and OSCNN using the 26 UEA *multivariate* datasets [7]. We leave outside MR-Hydra because it is not directly applicable to multivariate scenarios. As Fig. 13 shows, SuperRocket consistently outperforms the competitors in terms of mean accuracy ranking, though the difference is not statistically significant compared to HC2, RDST and MR, which validating the effectiveness of our solution.

D. Ablation Study

To answer RQ3, we perform ablation studies on the key components of SuperRocket to thoroughly assess their effectiveness. The details are as follows.

Effectiveness of indicating vector learning algorithm. First, to study the effectiveness of the proposed indicating vector learning algorithm (see § IV-C), we design the following three variants that use alternative optimization techniques to solve the optimization problem in Eq. (13). For description, we denote the total loss of the outer optimization problem as $\mathcal{L}$.

- **Random Search (RS)** [53] repeatedly selects K transformations at random to train and validate the classifier, and retain the selection that achieves the minimum $\mathcal{L}$;
- **Bayesian Optimization (BO)** [54] solves Eq. (13) by taking the indicating vector $\boldsymbol{\alpha}$ as the input and the loss $\mathcal{L}$ as the output of a black-box function. Considering the high dimension of $\boldsymbol{\alpha}$ to which BO cannot scale, we use the random embedding strategy [67] to embed $\boldsymbol{\alpha}$ into a 5-dimensional vector to perform the optimization;
- **Evolutionary Algorithm (EA)** adapts the evolutionary strategy in S-Rocket [17] to take $\mathcal{L}$ as the objective for learning $\boldsymbol{\beta}$. The population size is set to a moderate value of 8 to balance the convergence speed and the solution quality.

To ensure optimization performance and to achieve fair comparison, we set the maximum number of iterations to 200 for the above methods. We report the results of accuracy and efficiency in Fig. 14 and Table IV, respectively. From Fig. 14, we observe that SuperRocket is statistically significantly more accurate than RS on average. EA and BO perform slightly worse than ours, but the difference is not statistically significant. However, Table IV shows that SuperRocket is consistently faster than the three variants on all datasets and achieves 13.65-84.53 times of speedup in total, which validates the effectiveness of the proposed optimization algorithm.

Effectiveness of indicating vector regularization. Next, we evaluate the effectiveness of the regularization loss $\mathcal{L}_{reg}$ derived from the constraints of the indicating vector in Eqs. (7)-(8). To achieve this goal, we compare SuperRocket with its variants that solve the alternative optimization problem similar to Eq. (13) but without using $\mathcal{L}_{reg}$ or its terms $\mathcal{L}_{reg,1} = \sum_{h=1}^{H} (\frac{1}{\beta_h})^2$ and $\mathcal{L}_{reg,2} = (\sum_{h=1}^{H} \alpha_h - K)^2$ ($\mathcal{L}_{reg} = \mathcal{L}_{reg,1} + \mathcal{L}_{reg,2}$). As Fig. 15 shows, the three variants are statistically significantly less accurate than SuperRocket, indicating that both terms in the derived regularized objective are important for achieving better learning performance in terms of selecting high-quality transformations.

E. Analysis of Key Hyper-parameters

To study the influence of the key hyper-parameters on our model (RQ4), including the number of transformations before and after selection, i.e., H and K, and the learning rate of the classifier $\eta_{\boldsymbol{w}}$ and the indicating vector $\eta_{\boldsymbol{\beta}}$, we vary them individually and record the accuracy and training time on the

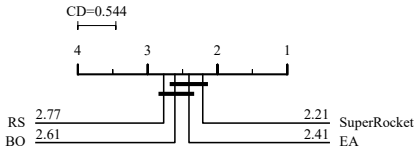


Fig. 14: CD diagram for SuperRocket against its variants using alternative optimization methods.

TABLE IV: Training time comparison for SuperRocket against its variants using alternative optimization methods.

Metric	RS	BO	EA	SuperRocket
Wins/Ties/Loses	109/0/0	109/0/0	109/0/0	-
Total time (h)	16.50	18.81	102.17	***1.21***
Speedup	13.65×	15.56×	84.53×	-

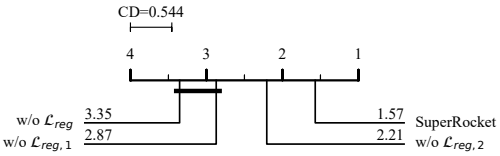
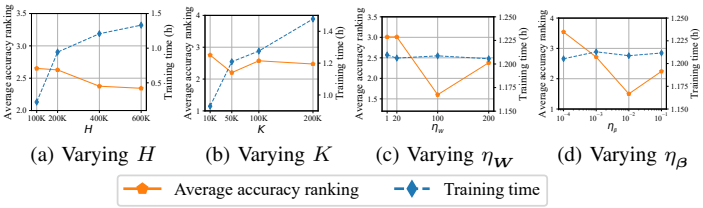


Fig. 15: CD diagram for SuperRocket against its variant without indicating vector regularization.



(a) Varying H (b) Varying K (c) Varying $\eta_{\boldsymbol{W}}$ (d) Varying $\eta_{\boldsymbol{\beta}}$

Fig. 16: Average accuracy ranking and total training time of SuperRocket with varying hyper-parameters.

109 datasets. For comparison, we visually report the average accuracy ranking and the total time in Fig. 16, where lower is better for both metrics. The results are discussed as follows.

Analysis of H and K. From Fig. 16a, we observe that increasing H can lead to higher accuracy on average. The reason is that the more transformations are generated at random, the higher the probability that high-quality candidates can be included. However, the model is more stable when $H > 400$K, which may indicate an upper bound of the number of high-quality transformations. Moreover, increasing H can significantly increase the training time, which is consistent with the theoretical complexity analysis in § V-A. Therefore, a moderate value of $H = 400$K can be a good choice to balance the accuracy and the training efficiency.

As Fig. 16b shows, increasing K will also increase the training time. However, increasing K cannot always lead to more accurate models, because more irrelevant transformations can be selected with a higher value of K, which will cause overfitting. In general, a value of around 50K can be suitable for achieving fast and accurate TSC.

Analysis of $\eta_{\boldsymbol{W}}$ and $\eta_{\boldsymbol{\beta}}$. As shown in Figs. 16c and 16d, varying the learning rate has almost no influence on the training time. This is because we use a fixed number of epochs during training, so the time complexity of the algorithm is independent of $\eta_{\boldsymbol{W}}$ and $\eta_{\boldsymbol{\beta}}$ (see Table I). Regarding their impact on accuracy, we observe in Fig. 16c that SuperRocket performs better when $\eta_{\boldsymbol{W}}$ is relatively large, and the best value may be in the range of 20 to 200. Similarly, Fig. 16d indicates that a value between 0.001 and 0.1 can be a good choice for $\eta_{\boldsymbol{\beta}}$ to achieve high accuracy.

VII. Conclusion

This paper presents SuperRocket, a novel solution for efficient selection of high-quality kernel-based transformations for fast and accurate TSC. To achieve efficient transformation selection, we specifically design a learnable indicating vector with two constraints to select all transformations simultaneously without iteratively building or evaluating the classifier. On this basis, we model the transformation selection problem as an unconstrained bi-level optimization problem and design an efficient SGD-based algorithm to learn the optimal indicating vector regularized based on the constraints. Theoretical analysis shows the effectiveness of the proposal. Extensive experiments on a total of 135 datasets demonstrate the superiority of our method over existing transformation selection solutions and SOTA TSC approaches.

AI-Generated Content Acknowledgement

This paper was entirely conducted by the authors without the use of any generative AI tools and technologies.

References

[1] Z. Liang, C. Liang, Z. Liang, H. Wang, and B. Zheng, "Units: A universal time series analysis framework powered by self-supervised representation learning," in *Companion of the 2024 International Conference on Management of Data*. ACM, 2024, pp. 480–483.

[2] E. Sylligardos, P. Boniol, J. Paparrizos, P. Trahanias, and T. Palpanas, "Choose wisely: An extensive evaluation of model selection for anomaly detection in time series," *Proc. VLDB Endow.*, vol. 16, no. 11, p. 3418–3432, Jul. 2023. [Online]. Available: https://doi.org/10.14778/3611479.3611536

[3] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," July 2015, www.cs.ucr.edu/~eamonn/time_series_data/.

[4] Z. Liang, D. Cai, C. Zhang, Z. Liang, C. Liang, B. Zheng, S. Qiu, J. Wang, and H. Wang, "Kdselector: A knowledge-enhanced and data-efficient model selector learning framework for time series anomaly detection," in *Companion of the 2025 International Conference on Management of Data*, 2025, pp. 175–178.

[5] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 31, pp. 606–660, 2017.

[6] T. Mu, H. Wang, S. Zheng, Z. Liang, C. Wang, X. Shao, and Z. Liang, "Tsc-automl: Meta-learning for automatic time series classification algorithm selection," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 1032–1044.

[7] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, "Hive-cote 2.0: a new meta ensemble for time series classification," *Machine Learning*, vol. 110, no. 11, pp. 3211–3243, 2021.

[8] A. Dempster, F. Petitjean, and G. I. Webb, "Rocket: exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1454–1495, 2020.

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

[10] C. W. Tan, A. Dempster, C. Bergmeir, and G. I. Webb, "Multirocket: Multiple pooling operators and transformations for fast and effective time series classification," *Data Mining and Knowledge Discovery*, pp. 1–24, 2022.

[11] C. Qian, Y. Yu, and Z.-H. Zhou, "Subset selection by pareto optimization," *Advances in neural information processing systems*, vol. 28, 2015.

[12] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM computing surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.

[13] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[14] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[15] R. Fu, Y. Wu, Q. Xu, and M. Zhang, "Feast: A communication-efficient federated feature selection framework for relational data," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–28, 2023.

[16] M. Yamada, W. Jitkrittum, L. Sigal, E. P. Xing, and M. Sugiyama, "High-dimensional feature selection by feature-wise kernelized lasso," *Neural computation*, vol. 26, no. 1, pp. 185–207, 2014.

[17] H. Salehinejad, Y. Wang, Y. Yu, T. Jin, and S. Valaee, "S-rocket: Selective random convolution kernels for time series classification," *arXiv preprint arXiv:2203.03445*, 2022.

[18] X. Yu and M. Gen, *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

[19] S. Chen, W. Sun, L. Huang, X. P. Li, Q. Wang, and D. John, "Pocket: Pruning random convolution kernels for time series classification from a feature selection perspective," *Know.-Based Syst.*, vol. 300, no. C, Nov. 2024.

[20] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.

[21] G. Uribarri, F. Barone, A. Ansuini, and E. Fransén, "Detach-rocket: sequential feature selection for time series classification with random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 38, no. 6, pp. 3922–3947, 2024.

[22] J. Friedman, "The elements of statistical learning: Data mining, inference, and prediction," *(No Title)*, 2009.

[23] L. N. Vicente and P. H. Calamai, "Bilevel and multilevel programming: A bibliography review," *Journal of Global optimization*, vol. 5, no. 3, pp. 291–306, 1994.

[24] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[25] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 275–309, 2013.

[26] J. Lines, S. Taylor, and A. Bagnall, "Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles," *ACM Transactions on Knowledge Discovery from Data*, vol. 12, no. 5, 2018.

[27] M. Middlehurst, J. Large, and A. Bagnall, "The canonical interval forest (cif) classifier for time series classification," in *2020 IEEE international conference on big data (big data)*. IEEE, 2020, pp. 188–195.

[28] A. Dempster, D. F. Schmidt, and G. I. Webb, "Quant: A minimalist interval method for time series classification," *Data Mining and Knowledge Discovery*, vol. 38, no. 4, pp. 2377–2402, 2024.

[29] Z. Liang, J. Zhang, C. Liang, H. Wang, Z. Liang, and L. Pan, "A shapelet-based framework for unsupervised multivariate time series representation learning," *Proceedings of the VLDB Endowment*, vol. 17, no. 3, p. 386–399, 2023.

[30] Z. Liang and H. Wang, "Fedst: secure federated shapelet transformation for time series classification," *The VLDB Journal*, vol. 33, no. 5, p. 1617–1641, Jul. 2024.

[31] Z. Liang, C. Liang, Z. Liang, H. Wang, and B. Zheng, "Timecsl: Unsupervised contrastive learning of general shapelets for explorable time series analysis," *Proceedings of the VLDB Endowment*, vol. 17, no. 12, pp. 4489–4492, 2024.

[32] G. Li, B. Choi, J. Xu, S. S. Bhowmick, D. N.-y. Mah, and G. L. Wong, "Ips: Instance profile for shapelet discovery for time series classification," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1781–1793.

[33] Z. Fang, P. Wang, and W. Wang, "Efficient learning interpretable shapelets for accurate time series classification," in *2018 IEEE 34th international conference on data engineering (ICDE)*. IEEE, 2018, pp. 497–508.

[34] A. Guillaume, C. Vrain, and W. Elloumi, "Random dilated shapelet transform: A new approach for time series shapelets," in *International Conference on Pattern Recognition and Artificial Intelligence*. Springer, 2022, pp. 653–664.

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

[36] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Mining and Knowledge Discovery*, vol. 29, no. 6, pp. 1505–1530, 2015.

[37] M. Middlehurst, J. Large, G. Cawley, and A. Bagnall, "The temporal dictionary ensemble (tde) classifier for time series classification," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 660–676.

[38] Z. Liang, Z. Liang, H. Wang, and B. Zheng, "Feddict: Towards practical federated dictionary-based time series classification," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[39] A. Dempster, D. F. Schmidt, and G. I. Webb, "Hydra: Competing convolutional kernels for fast and accurate time series classification," *Data Mining and Knowledge Discovery*, vol. 37, no. 5, pp. 1779–1805, 2023.

[40] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with cote: the collective of transformation-based ensembles," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2522–2535, 2015.

[41] A. Bagnall, M. Flynn, J. Large, J. Lines, and M. Middlehurst, "A tale of two toolkits, report the third: on the usage and performance of hive-cote v1. 0," *arXiv e-prints*, pp. arXiv–2004, 2020.

[42] W. Tang, G. Long, L. Liu, T. Zhou, M. Blumenstein, and J. Jiang, "Omni-scale CNNs: a simple and effective kernel size configuration for time series classification," in *International Conference on Learning Representations*, 2022.

[43] Y. Wang, Y. Xu, J. Yang, M. Wu, X. Li, L. Xie, and Z. Chen, "Graph-aware contrasting for multivariate time-series classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 14, 2024, pp. 15 725–15 734.

[44] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Min. Knowl. Discov.*, vol. 34, no. 6, pp. 1936–1962, 2020.

[45] A. B. Arrieta, N. D. Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Inf. Fusion*, vol. 58, pp. 82–115, 2019.

[46] J. Lines and A. J. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Min. Knowl. Discov.*, vol. 29, no. 3, pp. 565–592, 2015.

[47] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. A. Zaidi, B. Goethals, F. Petitjean, and G. I. Webb, "Proximity forest: an effective and scalable distance-based classifier for time series," *Data Min. Knowl. Discov.*, vol. 33, no. 3, pp. 607–635, 2019.

[48] A. Shifaz, C. Pelletier, F. Petitjean, and G. I. Webb, "TS-CHIEF: a scalable and accurate forest algorithm for time series classification," *Data Min. Knowl. Discov.*, vol. 34, no. 3, pp. 742–775, 2020.

[49] G. C. McDonald, "Ridge regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 93–100, 2009.

[50] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[51] T. M. Mitchell and T. M. Mitchell, *Machine learning*. McGraw-hill New York, 1997, vol. 1, no. 9.

[52] J. Kotary, F. Fioretto, P. Van Hentenryck, and B. Wilder, "End-to-end constrained optimization learning: A survey," *arXiv preprint arXiv:2103.16378*, 2021.

[53] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[54] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[55] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *International Conference on Learning Representations*, 2018.

[56] W. Cheney, "Analysis for applied mathematics," 2001.

[57] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory," in *Summer school on machine learning*. Springer, 2003, pp. 169–207.

[58] P. J. Davis, "Leonhard euler's integral: A historical profile of the gamma function: In memoriam: Milton abramowitz," *The American Mathematical Monthly*, vol. 66, no. 10, pp. 849–869, 1959.

[59] A. J. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. J. Keogh, "The UEA multivariate time series classification archive, 2018," *CoRR*, vol. abs/1811.00075, 2018.

[60] K. Fox, *Intermediate Economic Statistics: An integration of economic theory and statistical methods*, ser. Intermediate Economic Statistics. R. E. Krieger, 1980. [Online]. Available: https://books.google.co.jp/books?id=V6YrAAAAYAAJ

[61] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[62] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[63] M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király, "sktime: A unified interface for machine learning with time series," *arXiv preprint arXiv:1909.07872*, 2019.

[64] M. Middlehurst, A. Ismail-Fawaz, A. Guillaume, C. Holder, D. Guijo-Rubio, G. Bulatova, L. Tsaprounis, L. Mentel, M. Walter, P. Schäfer *et al.*, "aeon: a python toolkit for learning from time series," *Journal of Machine Learning Research*, vol. 25, no. 289, pp. 1–10, 2024.

[65] Z. Liang and H. Wang, "Efficient class-specific shapelets learning for interpretable time series classification," *Information Sciences*, vol. 570, pp. 428–450, 2021.

[66] P. Ngatchou, A. Zarei, and A. El-Sharkawi, "Pareto multi objective optimization," in *Proceedings of the 13th international conference on, intelligent systems application to power systems*. IEEE, 2005, pp. 84–91.

[67] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. De Freitas, "Bayesian optimization in high dimensions via random embeddings," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. IJCAI '13. AAAI Press, 2013, p. 1778–1784.