Noname manuscript No.
(will be inserted by the editor)

MultiETSC: Automated Machine Learning for Early Time Series Classification

Gilles Ottervanger · Mitra Baratchi · Holger H. Hoos

Received: date / Accepted: date

Abstract Early time series classification (EarlyTSC) involves predicting a class label based on partial observation of a given time series. Most EarlyTSC algorithms consider the trade-off between accuracy and earliness as two competing objectives, using a single dedicated hyperparameter. To obtain insights into this trade-off requires finding a set of non-dominated (Pareto efficient) classifiers. So far, this has been approached through manual hyperparameter tuning. Since the trade-off hyperparameters only provide indirect control over the earliness-accuracy trade-off, manual tuning is tedious and tends to result in many sub-optimal hyperparameter settings. This complicates the search for optimal hyperparameter settings and forms a hurdle for the application of EarlyTSC to real-world problems. To address these issues, we propose an automated approach to hyperparameter tuning and algorithm selection for EarlyTSC, building on developments in the fast-moving research area known as automated machine learning (AutoML). To deal with the challenging task of optimising two conflicting objectives in early time series classification, we propose MultiETSC, a system for multi-objective algorithm selection and hyperparameter optimisation (MO-CASH) for EarlyTSC. MultiETSC can potentially leverage any existing or future EarlyTSC algorithm and produces a set of Pareto optimal algorithm configurations from which a user can choose *a posteriori*. As an additional benefit, our proposed framework can incorporate and leverage time-series classification algorithms not originally designed for EarlyTSC for improving performance on EarlyTSC; we demonstrate this property using a newly defined, "naïve" fixed-time algorithm. In an extensive empirical evaluation of our new approach on a benchmark of 115 data sets, we show that MultiETSC performs substantially better than baseline methods, ranking highest (avg. rank 1.98) compared to conceptually simpler single-algorithm (2.98) and single-objective alternatives (4.36).

Keywords Early Classification · Time Series Classification · Automated Machine Learning

Gilles Ottervanger
Leiden Institute of Advanced Computer Science, Leiden University
Leiden, The Netherlands
E-mail: gillesottervanger@hotmail.com

Acknowledgements We would like to extend our gratitude to Can Wang for her contributions to the conception and realisation of the ideas presented in this paper.

1 Introduction

The goal of time series classification (TSC) is to assign a class label to a given time series, i.e., to a sequence of observations that have been sampled over time. Practical applications of time series classification include the diagnosis of heart conditions from ECGs, identification of patterns in financial markets, and detection of anomalies in seismic activity. Many such time-critical applications can benefit from classification results being available as early as possible, preferably even before the full time series has been observed. As an example, cardiac surgical patients in postoperative care are monitored for postoperative complications during an extended period of time. For some of these complications, indications of increased risk can be made far in advance of the actual onset (Abdelghani et al., 2016). Being able to automatically detect these signals as soon as they occur, through a timely classification of the monitored time series, can mean the difference between life and death.

Early time series classification (EarlyTSC) addresses the problem of classifying time series based on partial observations while maintaining a reasonable level of accuracy. The problem has been first described by Rodríguez Diez and Alonso González (2002) and has received an increasing amount of attention since. EarlyTSC introduces a second criterion to the classification problem: classifications do not only need to be *accurate* but also *early*. This results in a natural trade-off between accuracy and earliness (Mori et al., 2019).

There are currently many algorithms available for EaryTSC with different performance on different datasets. To obtain the best performance on a given dataset, it is typically necessary to carry out algorithm selection as well as hyperparameter tuning. However, when solving an EarlyTSC problem, we want to optimise both earliness and accuracy simultaneously. If configuration A (i.e., algorithm choice and its hyperparameter settings) is better with respect to at least one of these objectives and just as good w.r.t. the other compared to configuration B, we say that A dominates B. However, if A is earlier than B but less accurate or vice versa, A and B do not dominate each other, but rather represent different points in the earliness-accuracy trade-off. In that case, we call them both non-dominated or Pareto-efficient with respect to each other. For any EarlyTSC problem, there are many (potentially infinite) possible non-dominated configurations that we want to identify to get an idea of the trade-off between earliness and accuracy. However, these configurations can be difficult to identify, since only a very small portion of all possible configurations is non-dominated, which makes automatic selection and configuration of EarlyTSC algorithms a challenging task.

We illustrate the status-quo in EarlyTSC and our proposed approach in Figure 1, which shows configurations obtained using different approaches and their performance on the GunPoint dataset from the UCR benchmark collection (see Section 6.2). Most EarlyTSC algorithms proposed so far provide control over the trade-off between earliness and accuracy by means of one dedicated hyperparameter. Each hyperparameter setting results in a classifier whose performance is represented by a single point in the earliness-accuracy space. Figure 1A shows the earliness and accuracy of a set of manually configured algorithms, for each of which five distinct values of the trade-off hyperparameter have been considered, evenly spread over the full range of the hyperparameter (we chose to consider five settings since that number corresponds closely to what is reported in literature (e.g., Mori et al., 2018; Schäfer and Leser, 2020; Mori et al., 2019) and it would be a realistic number of configurations to evaluate manually). Based on this plot we can make several observations.

First, we observe that the dedicated hyperparameters do not strictly trade off accuracy for earliness, meaning that in some cases, earliness and accuracy increase or decrease simultaneously. This can be observed for EDSC (in blue) and TEASER (in grey): some configurations are dominating other configurations of the same algorithm (i.e., are better w.r.t. both earliness and accuracy). The downside of not having a strict trade-off is that considering, for example, ten configurations might only lead to two or three non-dominated ones. This renders the process of finding desirable configurations more challenging, as there is no clear logic for predicting how tuning the hyperparameter can help find a desirable trade-off.

A second observation from Figure 1A is that, even when covering the full range of the trade-off hyperparameters, in many cases, a single algorithm can be limited to only a narrow part of the earliness-accuracy space or leave large gaps between reachable points. This stems

from the fact that these trade-off hyperparameters only provide indirect control over the earliness-accuracy trade-off. As a result, even after extensive hyperparameter tuning, the user might be left with a very limited set of trade-off points, or a choice between configurations that all perform very similarly. This renders finding a good solution to a specific EarlyTSC problem extremely difficult. Note that considering multiple algorithms simultaneously can mitigate some of the problems mentioned above. This is illustrated in Figure 1B. However, in this example, this involves evaluating 40 configurations, of which only six turn out to be non-dominated, and none of these has an error-rate between 0.33 and 0.02.

In summary, several challenges arise when manually optimising the performance of EarlyTSC algorithms:

- Exploring the earliness-accuracy trade-off requires manual tuning of at least one dedicated hyperparameter. This is tedious, time-consuming and error-prone.
- The earliness-accuracy hyperparameter does not always effectively trade off earliness against accuracy. As a result, considering many configurations might only result in a few non-dominated ones.
- The earliness-accuracy hyperparameter provides only indirect and limited control over the earliness and accuracy, possibly resulting in multiple configurations with very similar earliness and accuracy limiting the freedom of trade-off choice.

In this work, we address these challenges by automating the process of hyperparameter tuning, while considering multiple algorithms and multiple objectives simultaneously. This is done using an algorithm configurator, i.e., a fully automated procedure that repeatedly trains and evaluates algorithm configurations and optimises for earliness and accuracy. Our proposed automated approach falls into the area of automated machine learning (AutoML). Recent developments in this field have made collections of advanced machine learning tools easily accessible for non-experts (e.g., Thornton et al., 2013; Feurer et al., 2015; Koch et al., 2018), such that users do not have to deal with difficult design choices and performance optimisation tasks (e.g., the choice of algorithm, or hyperparameter settings). Particularly, when optimising a single objective function, the problem of combined algorithm selection and hyperparameter optimisation (CASH) has been addressed by Thornton et al. (2013) through the well-known SMAC (Sequential Model-based Algorithm Configuration) procedure (Hutter et al., 2011). However, so far, no such approach has been proposed for EarlyTSC. This is mainly due to the complex, multi-objective nature of this specific machine learning task. A way around this challenge would be to combine both objectives into a single objective. Within the context of EarlyTSC, using the harmonic mean of earliness and accuracy as an objective function has been proposed (Schäfer and Leser, 2020). However, this approach fails to capture the complex trade-off between earliness and accuracy.

In this paper, we address this issue by considering a multi-objective configurator, previously designed for hyperparameter optimisation, and expanding its use to the multi-objective CASH (MO-CASH). In particular, we introduce a novel EarlyTSC framework addressing the MO-CASH problem for EarlyTSC that achieves the following:

- Automating the process of acquiring a set of non-dominated solutions without the need for manual tuning.
- Enabling the user to make informed decisions through having fine-grained control over the earliness and accuracy trade-off, as provided by a set of non-dominated solutions.
- Substantially expanding the number of non-dominated solutions found for a given EarlyTSC problem by (i) optimising all hyperparameters and (ii) considering a wide range of EarlyTSC methods, simultaneously.

In Figure 1C, we illustrate the improvements that can be achieved by an automated search over an expanded space of multiple algorithms and all their hyperparameters. The result is a large set of non-dominated configurations from which a user can choose, with high resolution, the one that is best suited to the problem at hand. Using performance metrics developed for multi-objective optimisation, we can quantify the performance of sets of non-dominated configurations as a whole. Specifically, we can calculate the dominated hypervolume (HV, defined in Section 6.4), which takes values between 0 (worst) and 1 (best). Comparing Figure 1C against Figure 1B, it can be seen that automated configuration achieves higher HV and produces a larger, denser set of non-dominated configurations.

A strong advantage of our proposed approach is that it effectively leverages the complementary strengths of EarlyTSC algorithms. This means that any algorithm can potentially help in partially improving its overall performance. For example, an algorithm that achieves relatively

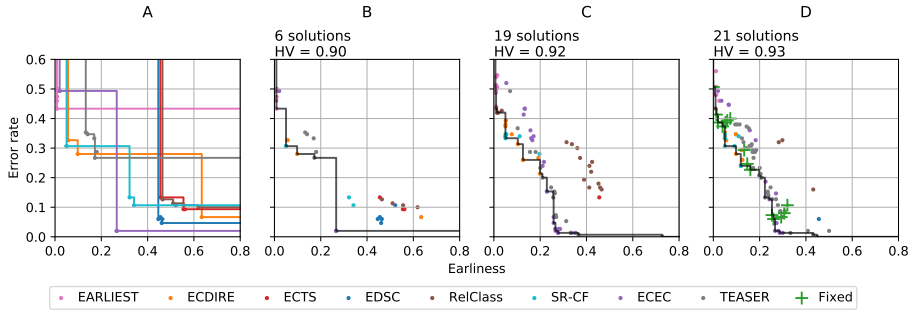


Fig. 1: Illustration of how an automated framework can improve the quality of EarlyTSC algorithms, in terms of the earliness-accuracy trade-off achieved on the GunPoint data set. Higher HV and a higher number of solutions show better performance. Each point represents an algorithm configuration in the earliness-accuracy space (coloured by the algorithm). **A**: Current approach: manual tuning of only the dedicated hyperparameter controlling the earliness-accuracy trade-off, separately for each algorithm. The Pareto set of configurations of a single algorithm provides a limited choice of solutions. The best Pareto front belongs to ECEC with $HV = 0.84$ and only 2 non-dominated solutions. **B**: By considering multiple manually configured algorithms, improved performance is achieved, but the choice of configurations remains limited, and manual configuration requires extensive effort. **C**: Combined automated algorithm selection and tuning of all hyper-parameters results in a more densely populated Pareto front of solutions. **D**: Adding the simple Fixed algorithm provides additional non-dominated configurations and improves the overall performance in terms of the dominated hypervolume.

good accuracy early on, but does not improve with more data, would not be considered a good EarlyTSC algorithm. Nevertheless, by considering multiple algorithms simultaneously within our multi-objective approach, the strength of this particular algorithm would be maximally exploited, while its weaknesses are compensated by other algorithms. This means that our approach can, additionally, open the door to exploring weaker EarlyTSC algorithms that would never be considered in isolation. These could be relatively simple algorithms, where the optimisation of earliness-accuracy trade-off is left to the algorithm configurator (as opposed to being done internally within the algorithm).

An example of such an algorithm is a naïve fixed-time early classifier (i.e., one that always classifies at a fixed timestep). The performance of such a fixed-time classifier heavily depends on the fixed point in time at which classification occurs (controlled by a hyperparameter). Ideally, one would want to consider a high number of possible classification times to obtain a detailed view of the earliness-accuracy trade-off. However, each setting requires retraining a model, which renders the process impractical, unless an automated configuration is used. By using automated configuration, however, favourable trade-offs between earliness and accuracy are easily identified – especially, because the effect of the hyperparameter on the performance of this simple algorithm is very direct. We will build upon this idea, by using the Fixed algorithm which is introduced in Section 5.1. The benefits of this step are illustrated in Figure 1D, where we added the Fixed algorithm to obtain better overall performance in terms of higher hypervolume and a larger number of non-dominated solutions.

Overall, our work presented here makes the following contributions to solving EarlyTSC:

– For the first time, we address the problem of EarlyTSC by systematically considering the trade-off between earliness and accuracy based on all relevant hyperparameters, using a multi-objective automated algorithm configuration approach.

- We propose MultiETSC, a framework for automatic algorithm selection and hyperparameter optimisation leveraging a wide range of existing EarlyTSC algorithms.
- Within our MultiETSC framework, we make use of an extension to an existing general-purpose algorithm configurator, exploiting prior knowledge of the structure of the MultiETSC search space.
- We demonstrate how MultiETSC framework can leverage regular time series classification algorithms to improve its performance on EarlyTSC.
- We perform an extensive empirical evaluation of MultiETSC on a benchmark of 115 data sets showing that MultiETSC can perform substantially better than baselines, ranking highest (avg. rank 1.98) compared to conceptually simpler single-algorithm (2.98) and single-objective alternatives (4.36).

The remainder of this article is structured as follows: after covering some preliminaries (Section 2), we formally describe the problem of MO-CASH for early time series classification in Section 3. Section 4 covers related work on both EarlyTSC and AutoML. In Section 5, we describe MultiETSC and cover its implementation details. Extensive experiments are described in Section 6, and their results discussed in Section 7. Finally, in Section 8, we draw some general conclusions and discuss directions for future work.

2 Preliminaries

A *time series* is a series of discrete observations over time. Time series can have varying sampling rates, but for the sake of simplicity, we consider only real-valued time series with a constant sampling rate. Note that for practical applications, the data could be transformed to fit this assumption. We denote a time series of l observations as $\mathbf{x} = [x_1, \ldots, x_l] \in \mathbb{R}^l$.

Time Series Classification (TSC) is the problem of determining a function $f : \mathbb{R}^l \to \mathcal{C}$ that maps a given time series $\mathbf{x} \in \mathbb{R}^l$ to a class label $f(\mathbf{x}) = y \in \mathcal{C}$, where $\mathcal{C}$ is a finite set of labels. Function f is obtained from a learning algorithm A based on a set of training examples $\{d_1, ..., d_n\}$, where each example is a pair of a time series and a class label $d_i = (\mathbf{x}_i, y_i) \in \mathbb{R}^l \times \mathcal{C}$. Note that we assume time series of uniform lengths. Additionally, most learning algorithms expose a set of hyperparameters λ that control some aspects of their inner workings; settings for these need to be chosen from a space $\mathbf{\Lambda}$ and often have a substantial impact on classification accuracy.

Early Time Series Classification (EarlyTSC) has been first mentioned in the literature by Rodríguez Diez and Alonso González (2002). The main difference to ordinary TSC occurs when performing the actual classification task on a given time series. While ordinary TSC assumes receiving the time series $\mathbf{x}$ as a single object, EarlyTSC considers multiple prefixes $\mathbf{x}_p = [x_1, \ldots, x_p] \in \mathbb{R}^p$ of $\mathbf{x}$ of increasing lengths ($p \leq l$). At each prefix length considered, the classifier either classifies or postpones classification to await more data. When the full time series has been observed, the classifier must produce a class output.

Hyperparameter optimisation (HPO) is the problem of finding the set of hyperparameter settings $\lambda^* \in \mathbf{\Lambda}$ with optimal generalisation performance for a given algorithm A_λ with hyperparameters λ and a set of training data $\mathcal{D}$. Generalisation performance can be estimated by repeatedly splitting $\mathcal{D}$ into non-overlapping training and validation subsets of $\mathcal{D}$, $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$ respectively, training on $\mathcal{D}_{train}$ and evaluating performance of the resulting classifier on $\mathcal{D}_{valid}$. Formally, hyperparameter optimisation involves determining

$$\lambda^* \in \operatorname*{argmin}_{\lambda \in \mathbf{\Lambda}} \mathcal{L}(A_\lambda, \mathcal{D}_{train}, \mathcal{D}_{valid}), \tag{1}$$

where $\mathcal{L}(A_\lambda, \mathcal{D}_{train}, \mathcal{D}_{valid})$ is the loss of algorithm A with hyperparameters λ when trained on $\mathcal{D}_{train}$ and evaluated on $\mathcal{D}_{valid}$. For a classification problem such as TSC, this loss usually is a measure of prediction accuracy, such as misclassification rate, but any performance metric could be chosen as an optimisation target. It is important to note that the combined hyperparameter space is a subset of the product of the permissible set of values for each individual hyperparameter $\mathbf{\Lambda} \subset \Lambda_1 \times \cdots \times \Lambda_m$. In most cases, this subset is strict, since some hyperparameters depend on the value of others (Hutter et al., 2014b). For example, for an algorithm with optional regularisation, there might be a hyperparameter λ_{reg} controlling whether regularisation is applied. The hyperparameter λ_{weight} controlling the regularisation weight is only *active* if the value of λ_{reg} is set to TRUE.

More formally, we say hyperparameter λ_i is *conditional* on hyperparameter λ_j if λ_i is only active when the value of λ_j is in a given set $V_i(j) \subsetneq \Lambda_j$. In this case we call λ_j a parent of λ_i. Conditional hyperparameters can themselves also be parents, resulting in a tree-structured search space (Bergstra et al., 2011).

When there are multiple learning algorithms to choose from, we might want to not only optimise hyperparameter settings but simultaneously select the best learning algorithm for a given data set. This problem is called *combined algorithm selection and hyperparameter optimisation or CASH* (Thornton et al., 2013). Given a set of algorithms $\mathcal{A}$, for each algorithm $A^{(j)} \in \mathcal{A}$, and a hyperparameter space $\mathbf{\Lambda}^{(j)}$, the goal is to optimise generalisation performance, i.e., to determine

$$A^*_{\lambda^*} \in \underset{A^{(j)} \in \mathcal{A}, \lambda \in \mathbf{\Lambda}^{(j)}}{\operatorname{argmin}} \mathcal{L}(A^{(j)}_\lambda, \mathcal{D}_{train}, \mathcal{D}_{valid}) \tag{2}$$

In their description of the CASH problem, Thornton et al. (2013) note that the choice of algorithm can be considered a top-level hyperparameter λ_r that selects an algorithm from $A^{(1)}, ..., A^{(k)}$. Thereby, the CASH problem can be reformulated as an HPO problem over the combined hyperparameter space $\mathbf{\Lambda} = \mathbf{\Lambda}^{(1)} \cup \cdots \cup \mathbf{\Lambda}^{(k)} \cup \{\lambda_r\}$, where each algorithm $A^{(i)}$ has its own subspace $\mathbf{\Lambda}^{(i)}$ that is conditional on λ_r being set to $A^{(i)}$. This places λ_r at the root of the tree-structured search space.

3 Problem Definition

So far, we have presented the single-objective CASH problem. To formulate the CASH problem for EarlyTSC, both accuracy and earliness objectives need to be considered. To accommodate that, we will introduce MO-CASH, the multi-objective extension to the CASH problem, and formulate this problem for the specific case of EarlyTSC. In Equations 1 and 2, we assumed a one-dimensional loss function defining a total order on the configuration space. When generalising to multi-objective optimisation, we can no longer speak of a total order. Let $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$ be vectors in objective space $\mathbb{R}^m$ with m objectives. We say $\mathbf{y}^{(1)}$ is dominated by $\mathbf{y}^{(2)}$ if the following two conditions are met: 1) $y^{(2)}_i \leq y^{(1)}_i$ for all $i \in 1, \ldots, m$ and 2) $y^{(2)}_i < y^{(1)}_i$ for at least one $i \in 1, \ldots, m$. We denote this relation with $\mathbf{y}^{(2)} \prec \mathbf{y}^{(1)}$. The domination relation is a partial order, meaning that some pairs of configurations are incomparable. In MO-CASH, we are interested in the *efficient set* of configurations, i.e., a set that consists solely of non-dominated, or Pareto-optimal, configurations. Since CASH can be formalised as a special case of HPO (as discussed in Section 2), we will be using the simpler notation of HPO Let $\mathcal{L}$ be a vector-valued loss function on $\mathbb{R}^m$; then the efficient set $\mathbf{\Lambda}^*$ and the Pareto-front $\mathcal{P}$ can be formalised as follows:

$$\mathbf{\Lambda}^* = \{\lambda^* \in \Lambda \mid \nexists \lambda \in \Lambda : \mathcal{L}(A_\lambda, \mathcal{D}_{train}, \mathcal{D}_{valid}) \prec \mathcal{L}(A_{\lambda^*}, \mathcal{D}_{train}, \mathcal{D}_{valid})\} \tag{3}$$

$$\mathcal{P} = \{\mathcal{L}(A_{\lambda^*}, \mathcal{D}_{train}, \mathcal{D}_{valid}) \in \mathbb{R}^m \mid \lambda^* \in \mathbf{\Lambda}^*\} \tag{4}$$

Note that the single-objective formulation from Equation 1 can be considered as a special case of a more general multi-objective problem. In the single-objective case, the efficient set is guaranteed to contain only a single configuration, since the ordering in a 1-dimensional solution space is guaranteed to be a total order. In the multi-objective case, on the other hand, there can be an arbitrarily large number of non-dominated configurations, and an optimiser needs to find improvements over a range of trade-off points in the objective space.

The problem we address in this work is the MO-CASH problem for the specific case of EarlyTSC, which can be defined as follows. Given the set $\mathcal{D} = \{d_1, ..., d_n\}$ of pairs of time series and class label $d_i = (\mathbf{x}_i, y_i) \in \mathbb{R}^l \times \mathcal{C}$, and given the combined space of EarlyTSC algorithms and their hyperparameter settings, in the form of a configuration space $\mathbf{\Lambda}$, find the best set of non-dominated configurations $\mathbf{\Lambda}^*$ in terms of earliness and accuracy.

Algorithm/Reference	Classification	Triggering	Pro/Con
ECTS (Xing et al., 2011b)	1NN ED	MPL	+ Good theoretical basis − Little control over earliness *vs* accuracy
EDSC (Xing et al., 2011a)	Shapelets	First match	+ Interpretable features − Computational complexity (w.r.t TS length)
RelClass (Parrish et al., 2013)	LDA/QDA	Reliability threshold	+ Good theoretical basis
ECDIRE (Mori et al., 2016)	GP	Probability threshold	+ Can theoretically use any probabilistic TS classifier − Computational complexity (w.r.t number of training samples)
SR-CF (Mori et al., 2018)	GP	Stopping rule	+ Optimised stopping rule − Computational complexity (w.r.t number of training samples)
TEASER (Schäfer and Leser, 2020)	WEASEL (Schäfer and Leser, 2017)	one-class SVM	+ Based on state-of-the-art TS classifier
ECEC (Lv et al., 2019)	WEASEL	Fused confidence	+ Based on state-of-the-art TS classifier − Not shown to outperform TEASER
EARLIEST (Hartvigsen et al., 2019)	RNN	RNN	+ Flexibility − Requires a large number of training examples and training time

Table 1: Overview of EarlyTSC algorithms

4 Related Work

In this section, we discuss work from the literature that has been fundamental to the algorithms we used as the basis of MultiETSC. Both the foundations of early time series classification as well as AutoML are covered.

4.1 EarlyTSC Methods

Many methods for solving the EarlyTSC problem have been proposed over the past two decades. Generally, these are adaptions of classification methods for full time series (for an overview of TSC methods we refer to Bagnall et al., 2016)). Most methods split the problem into two parts: one that addresses the classification of the partial data, aiming to maximise the classification accuracy; and a separate part that manages the trade-off between earliness and accuracy, by deciding whether enough data has been evaluated to base a reliable classification on. We will call this decision *triggering* and the function that controls it the *trigger function*. Table 1 shows a concise overview of the algorithms included in MultiETSC, their underlying classification approach and triggering mechanism as well as strengths and weaknesses. Next, we briefly discuss the existing literature on EarlyTSC algorithms.

Rodríguez Diez and Alonso González (2002) were the first to address the classification of time series based on partially observed data. They used a boosted set of simple interval-based

binary features using ADABoost (Freund and Schapire, 1999). To do early classification, the features that are not yet fully observed are simply ignored. Although this can be seen as the first attempt to solve EarlyTSC, this method does not make an explicit decision when to classify and consequently does not solve the EarlyTSC problem as we have defined it. Xing et al. (2011b) introduced **ECTS**, a method based on 1NN ED classification using the observed prefix of the time series. In the training phase, the *minimum prediction length* (MPL) of each time series is learned – the length at which the prediction based on the time series prefix is likely to be equal to the prediction on the full time series. The classification triggers as soon as the MPL of the closest match is equal to the observed prefix length.

EDSC, introduced by Xing et al. (2011a), is a shapelet-based EarlyTSC method. The shapelets are selected on a combination of their distinctiveness and the earliness of appearance in most time series. Classification is done as soon as a matching shapelet is found. **RelClass** (Parrish et al., 2013) explicitly estimates classification reliability, i.e., the probability of the early class prediction being equal to the classification of the complete time series. The trigger mechanism is simply a minimum reliability threshold that needs to be met.

Hatami and Chira (2013) proposed a method that based the triggering on the "agreement" among different classifiers in an ensemble. When the individual classifiers do not agree, the classification is rejected, and the method waits for more data. Antonucci et al. (2015) proposed a method based on "imprecise hidden Markov models", where a Markov model is fitted to the incoming data with some uncertainty. The classification is done if only a single model of a time series in the training set remains within the uncertainty bounds. Dachraoui et al. (2015) suggest a meta-algorithm that considers both the cost of classification quality and the cost of delaying the classification decision. Additionally, this method predicts in advance how much data will be needed to make a decision, and only triggers when the observed amount reaches or exceeds the required amount.

Mori et al. (2016) introduced the idea of prefix classifiers. Their algorithm, **ECDIRE**, trains a set of fully-fledged time series classifiers on increasing time series prefix lengths. The prefix classifiers are typically probabilistic classifiers to provide a confidence estimate. This confidence is then used in the triggering mechanism. **SR-CF** (Mori et al., 2018), **TEASER** (Schäfer and Leser, 2020) and **ECEC** (Lv et al., 2019) are all extensions of this idea using slightly different triggering mechanisms and prefix-classifiers.

EARLIEST (Hartvigsen et al., 2019) is based on a recurrent neural network (RNN) with LSTM cells. The base RNN produces a vector representation at each time step. This vector representation is used by two classifiers, one for classification and one binary classifier for triggering. The system is trained as a whole, minimising a loss function combining earliness and accuracy.

All methods discussed above employ sophisticated techniques to address the problem of EarlyTSC. The downside they all have in common is that they only produce a single classifier. Producing classifiers with different levels of accuracy and earliness requires changing one or more hyperparameters and retraining. These methods therefore do not fully address the multi-objective nature of the EarlyTSC problem.

Mori et al. (2019) proposed the first, and thus far the only, truly multi-objective approach to EarlyTSC. They proposed an adaptation of the SR-CF, to which we will refer to as MO-SR, where the internal parameters are optimised for both earliness and accuracy simultaneously. MO-SR produces a set of non-dominated classifiers in a single training phase. The approach we propose in this paper applies multi-objective optimisation at a higher level (hyperparameters instead of parameters) resulting in a more complex, tree-structured search space, and it uses a sophisticated optimiser that can search this space. Following this approach, we developed a more general method that can incorporate any existing or future EarlyTSC method. While it would be interesting to compare MO-SR with our proposed MultiETSC, no implementation has been made available by the original authors.

As is apparent from this overview, a diverse set of EarlyTSC algorithms can be found in the literature, each with its strengths and weaknesses. To the best of our knowledge, we are the first to attempt to combine the strengths of (a subset of) these algorithms into a single, integrated system for early time series classification. Additionally, while all EarlyTSC algorithms manage the trade-off between earliness and accuracy in some way, most do not provide insight into this trade-off. Our automated approach, described in the following, can produce this insight, without the need to understand the details of the underlying EarlyTSC algorithms, making EarlyTSC more accessible for non-experts.

4.2 Automated Machine Learning

The application of machine learning to a specific problem often encompasses decisions about data pre-processing, choice of algorithm and hyperparameter settings. Automated machine learning attempts to automate these decisions. Hutter et al. (2009) addressed the hyperparameter optimisation problem using sequential model-based optimisation (SMBO). Hutter et al. (2011) used SMBO as the basis for a general-purpose algorithm configuration procedure, SMAC (Hutter et al., 2011), which enables the efficient search of large and complex configuration spaces with categorical and numerical parameters.

Bergstra et al. (2011) applied two forms of SMBO, Gaussian process (GP) regression and the so-called Tree-structured Parzen Estimator (TPE), to HPO in deep belief networks. For this 32-dimensional configuration space, they achieved better results within 24 hours of computing time than had been achieved by manual configuration in earlier work. Snoek et al. (2012) built further on SMBO for AutoML by proposing Spearmint, an algorithm that takes the variable cost, in terms of training time, into account. Thornton et al. (2013) introduced Auto-WEKA, a software package based on SMAC that makes AutoML available for end-users familiar with the WEKA interface, being the first AutoML system addressing the full CASH problem. Feurer et al. (2015) introduced AUTO-SKLEARN, a SMAC-based AutoML system for Python. Additionally, AUTO-SKLEARN supports a meta-learning step before the SMBO phase and an ensemble building phase after optimisation, improving the efficiency of the configuration process and the quality of the results thus obtained.

Olson et al. (2016) introduced a Tree-based Pipeline Optimisation Tool, or TPOT, which optimises classification pipelines using decision trees and random forests. TPOT uses a tree representation for classification pipelines including feature selection, transformation and construction operators, as well as model selection and parameter optimisation elements. These pipelines are optimised using a Genetic Algorithm (GA). The authors introduce an extension of TPOT, called TPOT-Pareto, which not only considers classification accuracy but also pipeline complexity (i.e., number of pipeline operators). During optimisation, not just the best k performing pipelines are kept as the population for the GA, but a Pareto-front of non-dominated pipelines (in terms of accuracy and complexity) is used. However, the selection of the final pipeline is still based solely on accuracy and does therefore not address the more general MO-CASH problem.

Koch et al. (2018) developed the Autotune framework for the proprietary statistical software package SAS. Autotune uses a hybrid search strategy consisting of random search, Latin Hypercube Sampling (LHS), global and local search, GA and Bayesian optimisation using a GP surrogate. Autotune is implemented to maximally exploit parallel computation. The authors show competitive performance compared to only Bayesian optimisation and the Spearmint package. Gardner et al. (2019) extended the work on Autotune to address multi-objective optimisation. They reduced their hybrid search to only employ LHS, GA and Generating Set Search, a local search strategy. In their study, the authors address common conflicting objectives for binary classification, e.g., false-negative rate *vs* misclassification rate.

Jin et al. (2019) addressed the problem of Neural Architecture Search (NAS) by developing the open-source Keras-based system: Auto-Keras. NAS can be seen as a particularly interesting special case of AutoML since the search space of possible architectures is complex and highly hierarchical. Auto-Keras employs a custom GP kernel for SMBO, based on the edit-distance of the neural network architecture. The downside of Auto-Keras is that it only takes into account a single loss metric, without penalizing architecture complexity.

Some recent efforts have focused on the problem of AutoML for streaming applications, where the optimal algorithm and hyperparameters might change over time (e.g., Veloso et al., 2018; Carnein et al., 2020; Celik and Vanschoren, 2020). Although this might be interesting for EarlyTSC, in this paper we consider only static problems.

All the AutoML systems above are built upon existing machine learning packages and are aimed to provide end-users with easier access to advanced machine learning pipelines and algorithms and increased overall performance. For EarlyTSC, there does not yet exist a software package that allows for such a direct extension. This introduced the additional challenge of integrating all algorithms into a common framework with all required hyperparameters exposed. An important difference between these existing AutoML implementations and our AutoML system for EarlyTSC is that our system is not only meant to benefit possible end-users but should also help in the development of new EarlyTSC algorithms since it combines the strengths

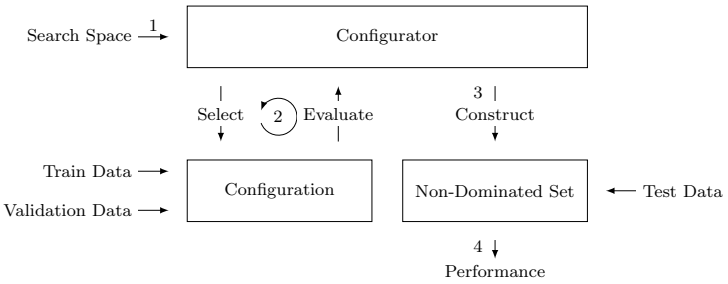


Fig. 2: Design of our Automated Machine Learning system. Numbers indicate the order of steps with step 2 being the inner loop of the system.

of multiple algorithms allowing the exploration of weaker EarlyTSC algorithms that would never be considered in isolation.

5 MultiETSC

In this section, we describe the approach developed for automatically configuring EarlyTSC algorithms and the system that implements our approach. At the core of our approach, we make use of a general-purpose automated algorithm configurator. It is the task of the configurator to efficiently search for the best performing configurations by searching a vast space of EarlyTSC algorithms and their hyperparameters. The inner loop of the search process consists of three steps: 1) selecting a candidate configuration (i.e., a combination of algorithm and its hyperparameters); 2) training the configuration on training data; 3) evaluating the configuration on validation data. The evaluation is fed back into the configurator enabling informed decisions for selecting new configurations. The final output thus obtained is a set of configurations that are mutually non-dominated, based on evaluation on the given validation data. The overall framework of our proposed approach is illustrated in Figure 2. In the remainder of this section, we will first describe in detail the space of EarlyTSC algorithms and hyperparameter settings that we search over. Second, we describe how configurations are evaluated. Third, a description of the algorithm configurator that we use to carry out the search.

5.1 Configuration Space

As discussed in Section 3, the configuration space is a tree-structured space defined by the choices of algorithm and hyperparameter settings. MultiETSC includes 9 EarlyTSC algorithms: ECTS, EDSC, RelClass, ECDIRE, SR-CF, TEASER, ECEC, EARLIEST (all described in Section 4.1) and a naïve fixed-time Euclidean 1NN algorithm, which we will refer to as 'Fixed' and is described in more detail below. A more detailed description of the search space, including hyperparameter descriptions and the size of the search space, is provided in Tables 5–7 in Appendix A.

These algorithms were chosen based on the fact that their implementations were made available by the original authors (except for Fixed, which we implemented), which helped us to ensure correctness and efficiency. There were several algorithms proposed in literature that we would have liked to include, but for which we were unable to acquire implementations (e.g., Hatami and Chira, 2013; Dachraoui et al., 2015; Wang et al., 2016; Martinez et al., 2018; He et al., 2019; Rußwurm et al., 2019). Note, however, that new algorithms can be added to MultiETSC with relative ease. No algorithms were excluded based on low expected performance. All EarlyTSC algorithms from the literature employ various techniques to find the best time to trigger classification. To put the merit of these techniques into perspective, we extended the algorithm portfolio with the naïve Fixed method that simply classifies at a fixed point in time that is determined by a hyperparameter. Without HPO, such a method would be too naïve to

consider. However, the responsibility of picking a favourable classification time is now shifted to the configurator, which is given more direct control. This will potentially lead to a better exploration of the full range of possible trade-off points.

All algorithms were modified and wrapped to provide a common command-line interface, where test data, training data, hyperparameters and, in the case of a stochastic algorithm, a random seed can be passed to the algorithm for reproducibility. To achieve this, the original implementations required a varying degree of modifications. In some cases, the needed modifications were quite considerable.

- **ECTS** (Xing et al., 2011b) (`C++`): is one of the oldest implementations of an EarlyTSC algorithm and still often used as a baseline in experimental evaluations of new algorithms.
- **EDSC** (Xing et al., 2011a) (`C++`): is the only explicitly shapelet-based EarlyTSC method.
- **RelClass** (Parrish et al., 2013) (`MATLAB` modified for `GNU Octave`): is the only EarlyTSC algorithm that is not based on any time series specific method and is still a very competitive EarlyTSC algorithm.
- **ECDIRE** (Mori et al., 2016) (`R`): uses a set of Gaussian process classifiers. This makes it theoretically better suited for situations where little training data is available.
- **SR-CF** (Mori et al., 2018) (`R`): is the first method introduced with an explicitly learned trigger function.
- **TEASER** (Schäfer and Leser, 2020) (`Java`): is designed based on the advanced WEASEL classifier (Schäfer and Leser, 2017) using a relatively simple triggering mechanism.
- **ECEC** (Lv et al., 2019) (`Java`): is designed based on the advanced WEASEL classifier (Schäfer and Leser, 2017) using a triggering mechanism based on reliability estimation.
- **EARLIEST** (Hartvigsen et al., 2019) (`Python`): is the only neural network based EarlyTSC method included. It is implemented in Python using the PyTorch module (Paszke et al., 2019).
- **Fixed** (authors)(`Python`): is a naïve Euclidean 1NN classifier on a fixed length prefix. A single hyperparameter controls the prefix length at which classification is done as a proportion of the full time series length. We theorise that any EarlyTSC algorithm with more control over "when to classify" should perform at least as well as this naïve algorithm, either by being more accurate with the same average earliness or by being earlier with the same level of accuracy or both. This algorithm is implemented by the authors for the purpose of this paper.

5.2 Algorithm Performance

As described earlier, the EarlyTSC algorithms are evaluated on both earliness and accuracy. For our setup, we need to define two metrics that represent the loss in both of these objectives. For the loss relating to accuracy, we used the *error rate* C_a defined as follows:

$$C_a = \frac{|\{\mathbf{x} \in \mathcal{D}_{test} | f(\mathbf{x}_{l^*_{\mathbf{x}}}) \neq Class(\mathbf{x})\}|}{|\mathcal{D}_{test}|} \tag{5}$$

Where $l^*_{\mathbf{x}}$ is the length at which the classification is triggered for time series $\mathbf{x}$, $f(\mathbf{x}_{l^*_{\mathbf{x}}})$ is the early class prediction and $Class(\mathbf{x})$ is the true class of $\mathbf{x}$.

The earliness C_e is quantified by the proportion of the time series needed to produce a classification averaged over the number of samples classified. It can be written as follows:

$$C_e = \frac{1}{|\mathcal{D}_{test}|} \sum_{\mathbf{x} \in \mathcal{D}_{test}} \frac{l^*_{\mathbf{x}}}{l_{\mathbf{x}}} \tag{6}$$

Where $l_{\mathbf{x}}$ is the length of time series $\mathbf{x}$.

5.3 Algorithm Configurator

Sequential Model-Based Optimisation (SMBO) has shown to be a promising approach to the single-objective CASH problem (Thornton et al., 2013). However, the problem of optimising for

multiple objectives is substantially more complex than the single-objective case. While there have been methods proposed for model-based optimisation for multi-objective problems (e.g., Emmerich et al., 2015), these methods are not able to handle the tree-structured search space that is typical for CASH problems.

We make use of the general purpose algorithm configurator MO-ParamILS (Blot et al., 2016) and customise it, in order to achieve increased efficiency in the context of the EarlyTSC problem. MO-ParamILS was originally designed for general-purpose multi-objective algorithm configuration. Here, we leverage its power to search tree-structured search spaces to extend this to MO-CASH (combined algorithm selection and hyper-parameter optimisation) by using the choice of algorithm as a top-level hyperparameter. MO-ParamILS uses iterated local search (ILS), a stochastic local search method, to find promising configurations. MO-ParamILS maintains a set of non-dominated configurations referred to as the *archive*. The *one-exchange neighbourhood* of a configuration λ is the set of configurations that is obtainable by changing a single parameter. This one-exchange neighbourhood of the configurations in the archive is used for local search steps. The local search strategy is complemented with random search steps to increase exploration of the search space, enabling to escape local optima.

MO-ParamILS handles the tree-structured search space by always keeping a value for each hyperparameter, whether it is active or not. Changing a higher-level hyperparameter (e.g., the algorithm choice) results in its dependent hyperparameters becoming active, retaining the values assigned to them earlier in the search. This means that the first time a specific algorithm is considered, its hyperparameters will be initialised at random.

It is known from earlier work on hyperparameter optimisation that, even in high-dimensional cases, most performance variation can be attributed to just a few hyperparameters (e.g., Hutter et al., 2014a; Bergstra and Bengio, 2012). However, the original version of MO-ParamILS treats each hyperparameter equally. Here, we implemented a variant of the MO-ParamILS method that is able to leverage prior knowledge about individual hyperparameters. As can be seen in Table 6 in Appendix A, all algorithms in our framework have a specific hyperparameter dedicated to controlling the earliness-accuracy trade-off. We will call these *trade-off hyperparameters*. In order to maximise coverage of the possible trade-offs (one of the objectives optimised by MO-ParamILS), these trade-off hyperparameters should be considered first when exploring the design space of EarlyTSC algorithms underlying our framework.

The original version of MO-ParamILS keeps track of a set of non-dominated candidates called the *archive*. In each local search step, the one-exchange neighbourhood of each configuration in the archive is searched uniformly at random, until either a neighbour is found that dominates at least one configuration in the archive, or all neighbours have been searched. In our customised version, we first consider neighbouring configurations obtained by changing a trade-off hyperparameter – a subset of the one-exchange neighbourhood – before considering the remaining neighbours. The neighbours resulting from changing a trade-off hyperparameter are expected to be less likely to dominate the current configuration but are at the same time more likely to be non-dominated. Thus, using this technique, we expect to obtain sets of configurations that are more diverse in terms of their earliness-accuracy trade-off. In Section 7.4, we show empirical support for the efficacy of this modification within the context of MultiETSC. We note that, in principle, this approach could be extended to other multi-objective problems, for which a limited number of (hyper)parameters controls the trade-off between different optimisation objectives.

One caveat of using MO-ParamILS is the requirement of a discrete search space requiring discretisation of continuous variables. This means that the true Pareto-set (the multi-objective optimum) might lie in between the chosen values in the discretisation. On the other hand, discretisation reduces the size of the search space facilitating optimisation.

In the context of reproducible research and to share our efforts with the community, we have made the source code for MultiETSC publicly available. [1]

6 Experimental Evaluation

We have designed our experiments to answer the following four questions:

[1] `https://github.com/Ottervanger/MultiETSC`

- What improvement can be achieved by solving the MO-CASH problem for EarlyTSC compared to multi-objective HPO of any single competitive EarlyTSC algorithm? (Subsection 7.1)
- What improvement can be achieved by solving the MO-CASH problem compared to solving the single-objective CASH problem optimising for the harmonic mean of accuracy and earliness? (Subsection 7.1)
- What would be the impact of adding other EarlyTSC algorithms to the search space (e.g., the naïve fixed method)? (Subsection 7.3)
- What improvement can be achieved by modifying the MO-ParamILS algorithm configurator to explore the search space of MultiETSC by prioritising the trade-off hyperparameters? (Subsection 7.4)

In the rest of this section, we will introduce our baselines, data sources, and evaluation protocol. Next, we will present the experimental results that will answer these questions.

6.1 Baselines

To answer the first question we compared algorithm selection, using MO-ParamILS on the previously defined search space, with hyperparameter optimisation of each individual algorithm in the search space by fixing the algorithm choice. This results in 9 baseline methods, one for each included algorithm: ECTS, EDSC, RelClass, ECDIRE, SR-CF, TEASER, ECEC, EARLIEST and Fixed.

To address the second question, we compared our multi-objective approach with a method that optimises a single objective (we refer to this baseline as SO-all). For this objective, we chose the harmonic mean of earliness and accuracy as suggested by Schäfer and Leser (2020):

$$HM = 1 - \frac{2 \cdot (1 - C_e) \cdot (1 - C_a)}{(1 - C_e) + (1 - C_a)} \tag{7}$$

Resulting in a value on the closed interval $[0, 1]$ which is to be minimised. This metric has the property that it will be low when both C_e and C_a are low and high when either is high. We will refer to this as the HM metric.

Although MO-ParamILS is capable of single-objective algorithm configuration, more advanced systems are available for this task. To make a fair comparison, we chose the state-of-the-art SMAC (Lindauer et al., 2017) algorithm configurator for its ability to efficiently search tree-structured search spaces. We will refer to the baseline method searching the full algorithm space using SMAC optimising the HM metric as SO-All. MO-ParamILS and SMAC differ in the optimisation method they are based on. In addition, for the single-objective case, the HM needs to be computed for each considered configuration. This means there are differences in the computational demands of these two methods. However, in this case, the computational costs of training and evaluating configurations heavily outweigh the costs of selecting the next configuration to evaluate. Furthermore, as we will discuss in Section 6.3, both configurators will be given the same overall time budget, to allow fair comparisons.

6.2 Data

As the main source of data, we will use the University of California, Riverside (UCR) Time Series Archive (Chen et al., 2015; Dau et al., 2018), last updated in 2018. As of 2018, the archive consists of 128 time series data sets for time series classification. Since its introduction in 2015, it has become the de facto standard for the evaluation of time series classification methods. The composers of the UCR archive recommend evaluating on all 128 data sets, and to clearly motivate excluding any. In our experiments we used 115 of these datasets in this paper due to the reason explained in Section 6.3. The data sets in the UCR archive contain real-world data and simulated data originating from various sources with varying degrees of complexity. Sources include ECG, EEG, spectrographs, image outlines, three-axis accelerometers and gyroscopes, and audio samples. In addition to real-world data, the archive contains 9 synthetic data sets that are specifically designed to evaluate time series classification methods. For data set details and descriptions, we refer to the Time Series Classification Website (Bagnall and Lines, 2020).

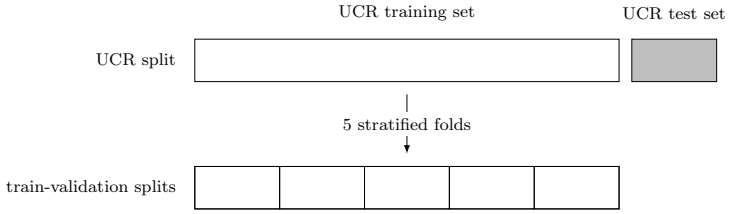


Fig. 3: Train, validate and test sets.

Configurator time budget	**Config. Validation time limit**	**Config. Test time limit**
120 min	3 min	15 min
Configurator runs	**Bootstrap sample size**	**Bootstrap samples**
25	10	1000

Table 2: Key numbers of the evaluation protocol.

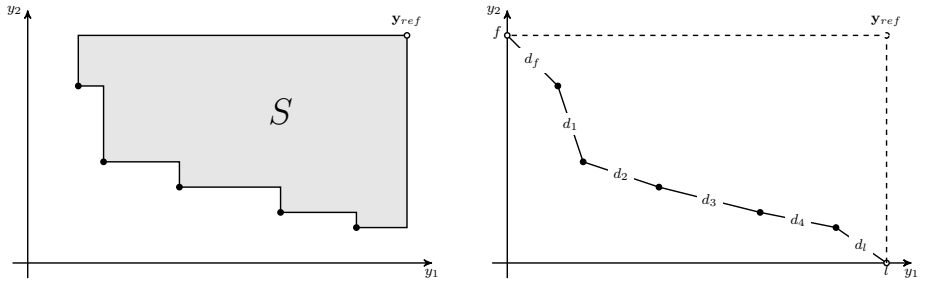
All datasets in the UCR archive consist only of univariate time series. Time series lengths vary between a few dozen samples to several thousand samples. For most data sets, all time series within one set are of equal length. 15 of the 128 data sets contain time series of varying lengths or with missing values. For these data sets the archive also includes same-length versions that are imputed using linear interpolation and padded up to the length of the longest time series with low amplitude noise. For this paper, we only used the same-length versions of these data sets.

6.3 Evaluation Protocol

Train and Validation splits: the UCR data sets have pre-defined train-test splits. The UCR defined train set is split again into five stratified train-validation splits. An algorithm is trained on 80% of the data and evaluated on 20% of the data. This is illustrated in Figure 3. For each repeated configurator run, a different set of cross-validation folds is generated. However, each experimental condition is run with the same set of cross-validation folds to keep the comparison fair. Due to the five-fold split requirement, we had to exclude 8 data sets that did not contain a sufficient amount of training examples per class for this split to be made. These were FiftyWords, Fungi, Phoneme, PigAirwayPressure, PigArtPressure, PigCVP, and Symbols. We evaluated the remaining 120 datasets.

Configurator runs: because a single configurator run is dependent on the random train/validation splits, and random initialisation, we performed multiple runs to get stable results. For each data set, 25 configurator runs were performed. For practical reasons, we set limits to run times of different stages of the experiment (presented in Table 2). The test evaluation is provided with a budget that is significantly bigger than that of the validation. The reason is that the test set can contain much more examples than the validation set. Additionally, we had to set a maximum amount of memory that can be used at any point in time by an algorithm implementation, which we set to 10GB. This made it impossible to process 5 more data sets: Crop, ElectricDevices, FordB, FordA, InsectWingbeatSound.

Bootstrapping: each configurator run will result in a Pareto-set of configurations based on the validation performance. Note that in the case of the SO configurator, this set will, by definition, contain only a single configuration. To create a distribution of the results, we took a bootstrapping approach by randomly sampling 10 runs from the 25 runs performed. For each subsample, the set of Pareto-sets (one for each run) is then combined into a single set of non-dominated configurations (based on validation performance). All configurations ending up in one or more subsample Pareto-sets are evaluated using test data resulting in a final Pareto-set



(a) A set of non-dominated solutions P (black points) in the objective space $\mathbb{R}^2$. S indicates the hypervolume dominated by P with respect to reference point $\mathbf{y}_{ref}$.

(b) The distances used for the computation of the Δ-spread of a set of non-dominated solutions P (black points). f and l are theoretical extreme solutions.

Fig. 4: Calculation of the Pareto-front metric.

for each subsample. Evaluation is based on these final Pareto-sets. By repeating this approach 1000 times we created a bootstrap distribution. This protocol was designed based on similar experiments presented in literature (e.g., Hutter et al., 2011; Thornton et al., 2013). Note the fact that the single-objective baseline is limited to producing only a single configuration per run. This means that a subsample of SO-configurator runs can, in the best case, result in a number of non-dominated configurations equal to the number of runs at most. To make the single-objective baseline slightly more competitive, we increased the number of runs from 4 in the original design to 10 runs per subsample for all methods. This way the SO baseline is less constrained by this upper limit. For our analysis, we will be using robust, non-parametric tests, which means we can keep the amount of bootstrap samples at a relatively low number of 1000.

6.4 Evaluation

We want to evaluate Pareto-sets of configurations in the earliness-accuracy space. Solutions that are both accurate and early are desirable, but we also prefer a "cheap" trade-off, getting much earlier predictions for only a small reduction in accuracy. To quantify the performance of a particular Pareto-set, we will turn to the theory of multi-objective optimisation. There is a multitude of Pareto-set performance metrics defined each capturing one or more desirable aspects (Audet, 2018). For our purpose, we will be using the dominated hypervolume, the Δ-spread and the HM-metric which we describe here in detail.

- **Hypervolume (HV)**: also called the S-metric, proposed by Zitzler, Deb and Thiele (Zitzler et al., 2000), HV is the hypervolume in the objective space that is dominated by the Pareto efficient solutions bounded by a reference point $\mathbf{y}_{ref} \in \mathbb{R}^m$ that is dominated by all feasible solutions. Figure 4a shows a Pareto-front and its dominated hypervolume S in $\mathbb{R}^2$. A commonly mentioned downside of the hypervolume is the need for a reference point that is larger than any point in the Pareto-set on all objectives but not too large since that would reduce the precision. In our case, however, it is clear that both error rate and earliness will never be larger than 100%, so we can safely use $(1, 1)$ as the reference point.
  The hypervolume metric has several properties that make it well suited for our purpose. First of all, it is intuitive. It is clear that dominating a large part of the objective space is desirable. As denoted in (Audet, 2018), the hypervolume metric is the only known unary Pareto-front performance indicator to be strictly monotonic. This means that if set A dominates set B (i.e., all points in B are dominated by at least one point in A) then the hypervolume metric of A is strictly larger than that of B. This is desirable since set domination is a strong indication that one set is preferable over the other and we want the metric to represent this property as well. Furthermore, the hypervolume metric is widely used in the performance evaluation of various multi-objective optimisation algorithms.

– **Δ-spread:** a desirable property that is not very well represented in the hypervolume is the distribution of configurations along the Pareto-front. To have control over the earliness-accuracy trade-off, these configurations should be evenly distributed along with a wide range of values. There can be two cases where the hypervolume metric is equal but in one case all hypervolume is dominated by a single configuration and in the other case there are 1000 unique configurations. In that case, it would be preferable to have a choice out of multiple options. We will be using the Δ-spread metric (Zitzler et al., 2000) to quantify the distribution of solutions in the objective space. This metric is based on the Euclidean distances between neighbouring solutions d_i and is formalised as follows:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1) \cdot \bar{d}} \tag{8}$$

  Where d_f and d_l are distances between the extreme solutions and the boundary solutions in the Pareto-front, and $\bar{d} = \frac{\sum_{i=1}^{N-1} d_i}{N-1}$. See Figure 4b for an illustration of the distances. An ideal distribution would make $d_f = d_l = 0$ and all distances d_i equal to $\bar{d}$, resulting in $\Delta = 0$. More clustering will result in higher values of Δ.
– ***HM* metric:** in addition to the hypervolume metric and Δ-spread, we will also look at the *HM* scalarisation as defined in Equation 7, to see how this metric compares to our proposed evaluation methods. Since each point in the Pareto-set will have its own *HM* value, we will look at the minimum value in each set.

7 Results

In this section, we cover the results of our extensive experiments. First, we focus on the relative performance of EarlyTSC and our baselines based on ranked performance. Next, we cover an illustrative example of one of the data sets in the UCR archive and show the performance of the produced classifiers. Finally, we analyse the contribution of the naïve Fixed algorithm to MutliETSC.

7.1 EarlyTSC vs Baselines

To compare relative performance between all methods, based on their performance across 115 data sets, we computed the average ranks based on the three selected metrics. Average ranks provide a robust method of comparison without making additional assumptions about normality and symmetry – assumptions we cannot safely make in general. This approach is similar to the empirical analysis done by Bagnall et al. (2016). The ranks are averaged over all subsamples and all datasets. With 1000 subsamples for each of the 115 data sets, these are the averages over 115 000 ranks. We first checked for significant differences between rank means using the Friedman test and subsequently applied the Nemenyi post-hoc test (Nemenyi, 1963) which checks for significant pairwise differences in average ranks. We visualised the outcome of this test using critical difference diagrams (Demšar, 2006), which are commonly used for the comparison of TSC methods to show the result of a statistical comparison of ranking results. These are shown in Figures 5, 6 and 7. Due to our subsampling method, we achieved high statistical power resulting in small CD intervals, which means that most observed differences are statistically significant.

According to the comparison of methods based on all metrics (shown in Figures 5,6,7), MultiETSC performs significantly better than any of the algorithms we compared against, finding configurations that together dominate a larger portion of the objective space and distributed more evenly across the trade-off according to the Δ-spread. This answers our first question mentioned earlier in Section 6. Similarly, *MultiETSC performs significantly better than the single-objective CASH method SO-All*, which answers our second question. We also observed that SO-All performs relatively well compared to our individual baseline algorithms. However, interestingly, SO-All performs worse than MultiETSC on the *HM* metric, even though this is the exact metric that the SO configurator optimises, while MultiETSC does not explicitly consider this metric during configuration. Overall, methods consistently rank according to different metrics.

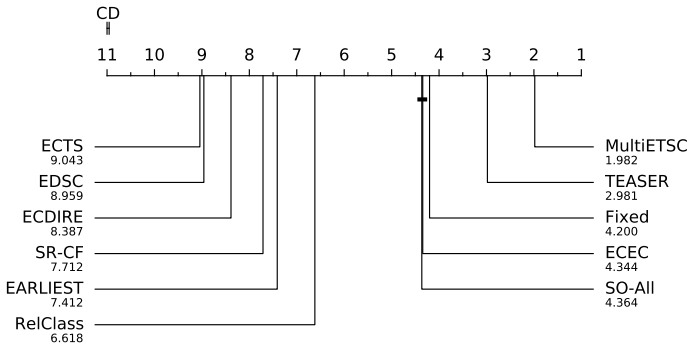


Fig. 5: CD diagram of the Nemenyi test on the Hypervolume. Numbers represent mean ranks (lower means better). Rank means with non-significant difference are connected with a horizontal line.

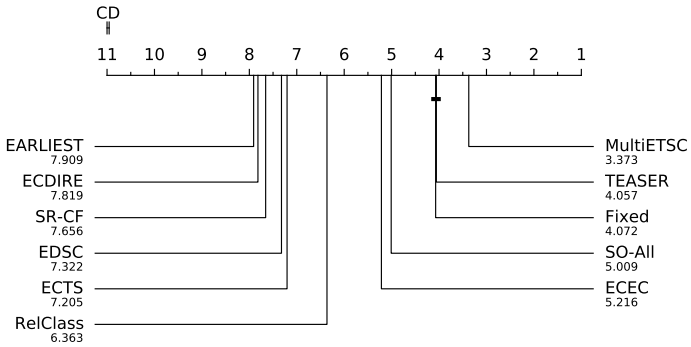


Fig. 6: CD diagram of the Nemenyi test on the Δ-spread. Numbers represent mean ranks (lower means better). Rank means with non-significant difference are connected with a horizontal line.

Table 3 shows the performance of the compared methods split out per problem (dataset) type. This table would show any method that is particularly suited or unsuited for a specific problem type. From these results, we observe that MultiETSC is consistently performing well across a broad range of problem types. The results for individual data sets are provided in Appendix B (Tables 5-7).

We found a significant number of cases with ties for the best method, most often between Fixed and MultiETSC. In these cases, the set of possible configurations using the Fixed algorithm dominates all other configurations. This set is found by both the Fixed method and MultiETSC, which results in equal hypervolume scores. This results in scenarios for which some rows in Table 3 add up to more than 100% (e.g., TRAFFIC).

As expected, the performance rank of each single algorithm method seems to correspond closely to the order in which these algorithms were originally introduced. However, there are some exceptions to this pattern. EARLIEST, the only neural-network-based algorithm, performs not as well as other methods proposed around the same time. In the original study, EARLIEST (Hartvigsen et al., 2019), is evaluated on data sets containing considerably more training examples than are available in typical UCR data sets and in many real-world applications. Additionally, the running time limits (3 minutes for configuration and 15 minutes for

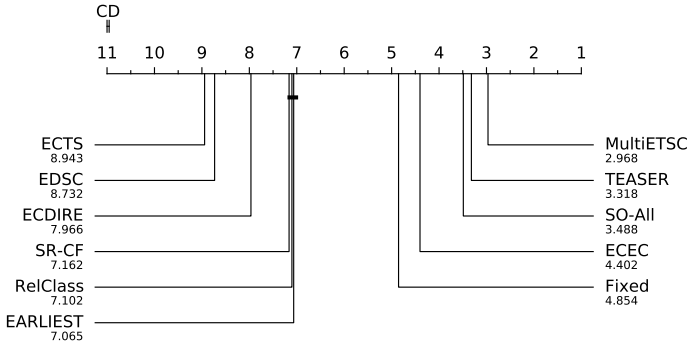


Fig. 7: CD diagram of the Nemenyi test on the *HM* metric. Numbers represent mean ranks (lower means better). Rank means with non-significant difference are connected with a horizontal line.

method	Fixed	ECTS	EDSC	ECDIRE	SR-CF	RelClass	TEASER	ECEC	EARLIEST	SO-All	MultiETSC	Counts
DEVICE	1.98	0.00	0.00	0.00	0.01	0.00	32.82	20.78	0.00	11.67	**32.77**	9000
ECG	28.31	0.00	0.00	0.00	0.00	0.00	12.93	3.23	0.00	0.61	**54.91**	7000
EOG	**69.70**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	34.85	2000
EPG	0.00	0.00	0.00	0.00	0.00	18.00	0.00	0.00	6.10	19.25	**76.95**	2000
IMAGE	4.39	0.00	0.64	0.00	0.12	14.60	28.81	2.93	0.26	1.96	**46.35**	28000
MOTION	4.74	0.00	0.00	0.00	0.00	0.08	34.37	5.58	0.00	5.38	**49.90**	25000
SENSOR	8.37	0.00	0.06	0.01	0.06	4.08	27.93	12.67	0.06	4.63	**46.84**	19000
SIMULATED	11.21	0.00	0.00	5.83	0.48	0.36	31.09	0.67	0.00	0.00	**50.37**	9000
SPECTRO	11.95	0.00	0.07	0.09	0.63	8.55	24.27	14.25	0.00	2.55	**37.81**	12000
TRAFFIC	50.00	0.00	0.00	0.00	0.00	0.00	29.70	0.00	0.00	0.00	**69.50**	2000
Overall	9.57	0.00	0.17	0.47	0.14	5.48	27.94	7.38	0.18	3.96	**46.82**	
Counts	11002	0	200	538	164	6301	32131	8488	207	4559	53842	115000

Table 3: Best performing algorithms by problem type. Entries represent percentages of subsamples that each method achieved the highest hypervolume on. Best performance is presented in bold face. Ties are counted as wins for both methods.

testing) might pose a challenge for the configurator to find viable parameter settings. This, however, is a challenge for all methods, not only EARLIEST.

7.2 Application of EarlyTSC

To provide some intuition on the produced Pareto-sets and their metrics we will discuss an illustrative example. A single subsample is constructed by combining ten different validation splits. Each method is run on each split and the resulting Pareto-sets are combined into one Pareto-set per method. Figure 8 shows these Pareto-sets for an arbitrary subsample of the BME data set. The BME data set is a synthetic data set created specifically for TSC research and is part of the UCR archive. It is a three-class problem with 10 training examples per class. The test set is balanced containing 150 items.

Looking at Figure 8, it is seen that generally, in Pareto-fronts, 66% seems to be the upper bound of the error rate at zero earliness (maximally early), which corresponds to the performance of a random or constant output classifier. With increasing earliness (later classification) most methods can produce classifiers that are increasingly accurate with clearly diminishing returns. Most methods level out at one-third of the time series observed, while the better-performing methods reduce the error rate up to 90% of the data observed achieving close to 5% error rate.

The ability of our approach to provide this visual representation of the trade-off between earliness and accuracy is a major advantage. Using any EarlyTSC algorithm requires the user to

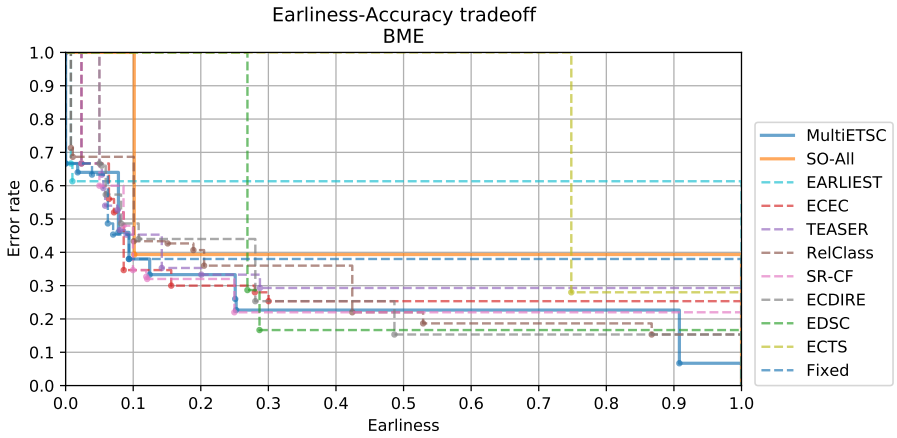


Fig. 8: Pareto-sets for one subsample on BME data set.

method	Fixed	ECTS	EDSC	ECDIRE	SR-CF	RelClass	TEASER	ECEC	EARLIEST	SO-All	MultiETSC
Hypervolume	0.600	0.181	0.607	0.711	0.708	0.698	0.655	0.693	0.384	0.545	**0.733**
Δ-spread	0.896	inf	0.925	0.853	0.902	0.834	0.837	**0.808**	0.966	inf	0.893
HM	0.264	0.627	**0.231**	0.267	0.233	0.291	0.262	0.235	0.444	0.276	0.240

Table 4: Pareto-set metrics for one subsample on the BME data set.

select a hyperparameter that only indirectly influences the trade-off. Using our approach, a user trying to solve a specific EarlyTSC problem can make an informed choice, making EarlyTSC more accessible for non-experts.

The metrics evaluating these Pareto-sets are shown in Table 4. In this particular example, the configurations found by MultiETSC dominate the largest hypervolume (0.733). Although the competing methods are close, it is clear that MultiETSC corresponds closely to the best of each individual baseline.

7.3 The Contribution of Fixed

In this section, we study the impact of adding other algorithms to the search space (e.g., the naïve fixed method). As seen in Figure 5, the naïve fixed-time method we introduced is highly competitive, consistently outperforming six out of the eight EarlyTSC algorithms proposed in the literature, according to all three metrics. More support for the merit of the Fixed method can be found by looking at how often each algorithm was selected by the configurator. Figure 9 shows the share of each algorithm in the final set of configurations. We observe that more than half of all the selected configurations are based on the Fixed algorithm. This supports our claim that giving more control to the configurator, in terms of optimising algorithmic choices, enables a better exploration of the earliness-accuracy trade-off. In turn, this exploration increases the resolution of choices over the trade-off, which gives the user more freedom and control in picking the best solution for the problem at hand.

So far we showed that Fixed contributes to the Pareto sets with a large number of configurations. However, this does not necessarily mean that Fixed provided benefits to the overall performance of MultiETSC. In other words, the inclusion of Fixed could even have a negative impact on performance by increasing the size of the search space. To show that adding Fixed provides tangible benefits to MultiETSC, we ran a series of experiments (using the same protocol as described earlier) comparing the performance of MultiETSC with and without Fixed on seven UCR data sets. Figure 10 shows the critical difference diagram comparing

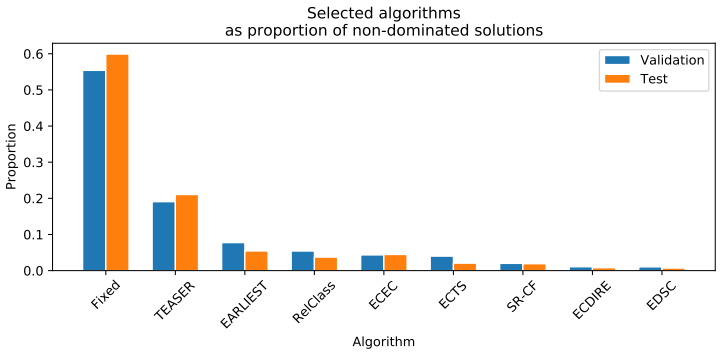


Fig. 9: Algorithms as selected by MultiETSC as proportion of non-dominated solutions as found based on validation and test evaluations.

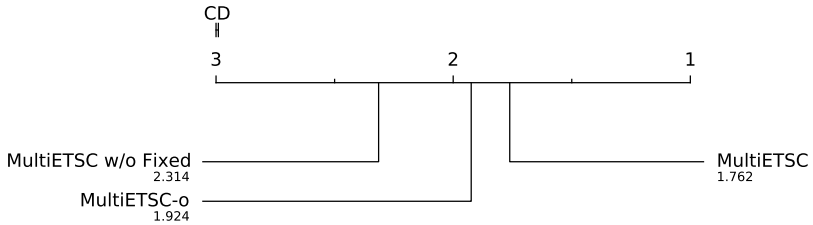


Fig. 10: CD diagram of the Nemenyi test on the Hypervolume comparing MultiETSC to MultiETSC without Fixed (MultiETSC w/o Fixed) and MultiETSC using the original version off MO-ParamILS (MultiETSC-o). Numbers represent mean ranks (lower means better). All rank differences are statistically significant.

MultiETSC to both MultiETSC without Fixed (MultiETSC w/o Fixed) and MultiETSC using the original version of MO-ParamILS (MultiETSC-o, which will be discussed in Subsection 7.4), showing their average ranks on all 115 data sets. Although MultiETSC with and without Fixed are both close to rank 2, the difference in rank is statistically significant.

These results provide evidence that MultiETSC is able to efficiently ignore ill-performing configurations and that performance tends to be improved by including Fixed.

7.4 The Contribution of our New Variant of MO-ParamILS

In Section 5.3, we theorised that modifying MO-ParamILS to be more explorative in the direction of the so-called trade-off hyperparameters could potentially result in improved overall performance. Figure 10 compares MultiETSC (using our new variant of MO-ParamILS) to MultiETSC-o, which uses the original version of MO-ParamILS. As seen in the figure, there is a significant difference in the average rank in favour of our new variant. The results shown in Table 7 in Appendix A further demonstrate that our modification also positively affects the average numbers of configurations searched and non-dominated configurations found compared to the original version.

8 Conclusions and Future Work

In this work, we have introduced MultiETSC, a systematic approach to automated early time series classification (EarlyTSC). MultiETSC performs automatic algorithm selection and hyperparameter optimisation to explore the full range of trade-offs between accuracy and earliness afforded by a broad set of EarlyTSC algorithms.

Our approach builds upon recently developed techniques in the area of automated algorithm configuration and uses them in combination with a broad range of well-known EarlyTSC algorithms. We have modified the general-purpose algorithm configurator MO-ParmaILS to exploit prior knowledge about the structure of the EarlyETSC problem, resulting in improved overall performance. MultiETSC enables users to explore and exploit trade-offs between earliness and accuracy, by producing a large and diverse set of non-dominated configurations from which a user can choose the one that is best suited to the problem at hand.

We performed an extensive empirical evaluation of our proposed method, using 115 data sets from the UCR Time Series Archive. We have shown that by leveraging the potential of many existing EarlyTSC algorithms, our approach can outperform any single algorithm, even when their hyperparameters are optimised. Our results also demonstrate that considering both earliness and accuracy separately and in a multi-objective fashion produces better results than combining the two into a single objective.

It should be mentioned that the benefits of automated algorithm configuration for EarlyTSC do come at a significant computational cost, which is primarily caused by the number of configurations that considered and evaluated. This can be illustrated by looking back at Figure 1 in the introduction. The evaluation of the solutions represented in Figure 1B (manual configuration) costs less than one hour of CPU time. In contrast, 50 hours of CPU time were required to produce Figure 1C and 1D (automated configuration). However, during this time, hundreds of other possible configurations are evaluated.

Therefore, MultiETSC essentially trades off the time spent by human experts on manual algorithm selection and configuration against computational effort, while also producing a more diverse set of qualitatively better results. Furthermore, in our experiments, we limited the running time of individual algorithms, in order to prevent spending too much time on a small set of configurations, and we limited the time budget of the automated configurator itself to two hours per run, in order to keep the computational cost of our experiments manageable. In practice, it would likely be best to adjust these limits to account for factors such as the size of the data set and the amount of CPU time the user can afford to spend on configuration.

We see numerous opportunities for future research. We have shown that our proposed approach can be extended and benefit from additional EarlyTSC algorithms (in particular, the naïve fixed-time classifier). Further research could focus on the development of additional EarlyTSC algorithms that improve the performance of MultiETSC. On the other hand, an algorithm might negatively impact the performance of MultiETSC by increasing the size of the search space without providing any beneficial configurations. Although MultiETSC can reasonably effectively ignore these ill-performing algorithms, the overall performance could possibly be further improved by excluding these algorithms altogether. This could be achieved by considering the marginal contribution or Shapley values of these algorithms in order to assess their contribution to overall performance (Fréchette et al., 2016).

This might make it possible to even more effectively construct a set of complementary algorithms that jointly represent the state-of-the-art in EarlyTSC.

For the area of automated machine learning, our work clearly demonstrates the potential of using multi-objective algorithm configurators within an integrated system leveraging many state-of-the-art techniques – an idea that can be extended to many other problems in machine learning in which multiple conflicting performance objectives arise. Our approach could also be extended to a larger number of objectives, which would enable the exploration of additional trade-offs, for instance, the resources required for processing given data. There are many situations where machine learning solutions need to be applied with limited computing resources. For such cases, it could be very valuable to know how much performance an ML solution is losing by constraining its access to resources.

Finally, we hope that this work will inspire further exploration in other directions, and ultimately lead to significant improvements in the state of the art in solving a broad set of machine learning problems that involve multiple competing performance objectives, as is the case in early time series classification.

Declarations

Funding

This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

Conflict of interest

The authors declare that they have no conflict of interest.

Availability of data and material

The data used in this work stems from the UCR archive. Instructions on acquiring these data can be found at `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

Code availability

The code used in our research is available at `https://github.com/Ottervanger/MultiETSC`.

References

Abdelghani SA, Rosenthal TM, Morin DP (2016) Surface electrocardiogram predictors of sudden cardiac arrest. The Ochsner Journal 16(3):280–289, URL https://pubmed.ncbi.nlm.nih.gov/27660578

Antonucci A, Scanagatta M, Mauá DD, de Campos CP (2015) Early classification of time series by hidden markov models with set-valued parameters. In: Proceedings of the NIPS Time Series Workshop, pp 1–5, URL https://sites.google.com/site/nipsts2015/home

Audet C (2018) Performance Indicators in Multiobjective Optimization. Les Cahiers du GERAD, GERAD HEC Montréal, URL https://books.google.nl/books?id=uKepzQEACAAJ

Bagnall A, Lines J (2020) The UEA TSC website. URL http://www.timeseriesclassification.com/

Bagnall A, Lines J, Bostrom A, Large J, Keogh E (2016) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery 31, DOI 10.1007/s10618-016-0483-9

Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13(1):281–305, URL http://dl.acm.org/citation.cfm?id=2503308.2188395

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. In: Proceedings of the 24th International Conference on Neural Information Processing Systems, Curran Associates Inc., USA, NIPS'11, pp 2546–2554, URL http://dl.acm.org/citation.cfm?id=2986459.2986743

Blot A, Hoos HH, Jourdan L, Kessaci-Marmion M, Trautmann H (2016) MO-ParamILS: A multi-objective automatic algorithm configuration framework. In: Proceedings of the 10th International Conference on Learning and Intelligent Optimization (LION 10), Springer, Lecture Notes in Computer Science, vol 10079, pp 32–47, DOI 10.1007/978-3-319-50349-3_3, URL https://doi.org/10.1007/978-3-319-50349-3_3

Carnein M, Trautmann H, Bifet A, Pfahringer B (2020) confstream: Automated algorithm selection and configuration of stream clustering algorithms. In: 14th Learning and Intelligent Optimization Conference (LION 2020), pp 80–95, DOI 10.1007/978-3-030-53552-0_10

Celik B, Vanschoren J (2020) Adaptation strategies for automated machine learning on evolving data. arXiv preprint arXiv:200606480 URL arxiv.org/abs/2006.06480

Chen Y, Keogh E, Hu B, Begum N, Bagnall A, Mueen A, Batista G (2015) The UCR time series classification archive. URL www.cs.ucr.edu/~eamonn/time_series_data/

Dachraoui A, Bondu A, Cornuéjols A (2015) Early classification of time series as a non myopic sequential decision making problem. In: Appice A, Rodrigues PP, Santos Costa V, Soares C, Gama J, Jorge A (eds) Machine Learning and Knowledge Discovery in Databases, Springer International Publishing, Cham, pp 433–447, DOI 10.1007/978-3-319-23528-8_27

Dau HA, Keogh E, Kamgar K, Yeh CCM, Zhu Y, Gharghabi S, Ratanamahatana CA, Yanping, Hu B, Begum N, Bagnall A, Mueen A, Batista G, Hexagon-ML (2018) The UCR time series classification archive. URL https://www.cs.ucr.edu/~eamonn/time_series_data_2018/

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7(1):1–30, URL http://jmlr.org/papers/v7/demsar06a.html

Emmerich M, Yang K, Deutz A, Wang H, Fonseca C (2015) A Multicriteria Generalization of Bayesian Global Optimization, vol 107, Springer International Publishing, pp 229–242. DOI 10.1007/978-3-319-29975-4_12

Feurer M, Klein A, Eggensperger K, Springenberg J, Blum M, Hutter F (2015) Efficient and robust automated machine learning. In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R (eds) Advances in Neural Information Processing Systems, Curran Associates, Inc., vol 28, pp 2962–2970, URL https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

Fréchette A, Kotthoff L, Michalak TP, Rahwan T, Hoos HH, Leyton-Brown K (2016) Using the shapley value to analyze algorithm portfolios. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16), AAAI Press, pp 3397–3403, URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12495

Freund Y, Schapire RE (1999) A short introduction to boosting. In: In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, pp 1401–1406, DOI 10.1051/matecconf/201713900222

Gardner S, Golovidov O, Griffin J, Koch P, Thompson W, Wujek B, Xu Y (2019) Constrained multi-objective optimization for automated machine learning. In: 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp 364–373, DOI 10.1109/

DSAA.2019.00051
Hartvigsen T, Sen C, Kong X, Rundensteiner E (2019) Adaptive-halting policy network for early classification. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 101–110, DOI 10.1145/3292500.3330974
Hatami N, Chira C (2013) Classifiers with a reject option for early time-series classification. 2013 IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL) pp 9–16, DOI 10.1109/CIEL.2013.6613134
He G, Zhao W, Xia X (2019) Confidence-based early classification of multivariate time series with multiple interpretable rules. Pattern Analysis and Applications 23:567–580, DOI 10.1007/s10044-019-00782-7
Hutter F, Hoos HH, Leyton-Brown K, Murphy K (2009) An experimental investigation of model-based parameter optimisation: SPO and beyond. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO 2009), pp 271–278, DOI 10.1145/1569901.1569940
Hutter F, Hoos HH, Leyton-Brown K (2011) Sequential model-based optimization for general algorithm configuration. In: Proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION 5), pp 507–523, DOI 10.1007/978-3-642-25566-3_40
Hutter F, Hoos H, Leyton-Brown K (2014a) An efficient approach for assessing hyperparameter importance. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014, JMLR.org, JMLR Workshop and Conference Proceedings, vol 32, pp 754–762, URL http://jmlr.org/proceedings/papers/v32/hutter14.html
Hutter F, Stützle T, Leyton-Brown K, Hoos HH (2014b) Paramils: An automatic algorithm configuration framework. CoRR abs/1401.3492, URL http://arxiv.org/abs/1401.3492, 1401.3492
Jin H, Song Q, Hu X (2019) Auto-keras: An efficient neural architecture search system. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, pp 1946–1956, DOI 10.1145/3292500.3330648
Koch P, Golovidov O, Gardner S, Wujek B, Griffin J, Xu Y (2018) Autotune: A derivative-free optimization framework for hyperparameter tuning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, New York, NY, USA, KDD '18, p 443–452, DOI 10.1145/3219819.3219837
Lindauer M, Eggensperger K, Feurer M, Falkner S, Biedenkapp A, Hutter F (2017) SMAC v3: Algorithm configuration in python. URL https://github.com/automl/SMAC3
Lv J, Hu X, Li L, Li P (2019) An effective confidence-based early classification of time series. IEEE Access 7:96113–96124, DOI 10.1109/ACCESS.2019.2929644
Martinez C, Perrin G, Ramasso E, Rombaut M (2018) A deep reinforcement learning approach for early classification of time series. In: European Signal Processing Conference, pp 2030–2034, DOI 10.23919/EUSIPCO.2018.8553544
Mori U, Mendiburu A, Keogh E, Lozano J (2016) Reliable early classification of time series based on discriminating the classes over time. Data Mining and Knowledge Discovery 31, DOI 10.1007/s10618-016-0462-1
Mori U, Mendiburu A, Dasgupta S, Lozano JA (2018) Early classification of time series by simultaneously optimizing the accuracy and earliness. IEEE Transactions on Neural Networks and Learning Systems 29(10):4569–4578, DOI 10.1109/TNNLS.2017.2764939
Mori U, Mendiburu A, Miranda I, Lozano J (2019) Early classification of time series using multi-objective optimization techniques. Information Sciences 492:204 – 218, DOI 10.1016/j.ins.2019.04.024, URL http://www.sciencedirect.com/science/article/pii/S0020025519303317
Nemenyi P (1963) Distribution-free Multiple Comparisons. Princeton University, URL https://books.google.nl/books?id=nhDMtgAACAAJ
Olson RS, Bartley N, Urbanowicz RJ, Moore JH (2016) Evaluation of a tree-based pipeline optimization tool for automating data science. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016, ACM, New York, NY, USA, GECCO '16, pp 485–492, DOI 10.1145/2908812.2908918
Parrish N, Anderson HS, Gupta MR, Hsiao DY (2013) Classifying with confidence from incomplete information. Journal of Machine Learning Research 14:3561–3589, URL http://jmlr.org/papers/v14/parrish13a.html
Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: An imperative style,

high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d' Alché-Buc F, Fox E, Garnett R (eds) Advances in Neural Information Processing Systems 32, Curran Associates, Inc., pp 8024–8035, URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Rodríguez Diez JJ, Alonso González CJ (2002) Boosting Interval-Based Literals: Variable Length and Early Classification, World Scientific, pp 149–171. DOI 10.1142/9789812565402_0007

Rußwurm M, Lefèvre S, Courty N, Emonet R, Körner M, Tavenard R (2019) End-to-end learning for early classification of time series. arXiv preprint URL https://arxiv.org/abs/1901.10681

Schäfer P, Leser U (2017) Fast and accurate time series classification with WEASEL. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Association for Computing Machinery, New York, NY, USA, CIKM '17, p 637–646, DOI 10.1145/3132847.3132980

Schäfer P, Leser U (2020) TEASER: Early and accurate time series classification. Data Mining and Knowledge Discovery 34:1336–1362, DOI 10.1007/s10618-020-00690-z

Snoek J, Larochelle H, Adams RP (2012) Practical bayesian optimization of machine learning algorithms. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ (eds) Advances in Neural Information Processing Systems 25, Curran Associates, Inc., pp 2951–2959, URL http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf

Thornton C, Hutter F, Hoos HH, Leyton-Brown K (2013) Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In: Dhillon IS, Koren Y, Ghani R, Senator TE, Bradley P, Parekh R, He J, Grossman RL, Uthurusamy R (eds) The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013, ACM, pp 847–855, DOI 10.1145/2487575.2487629, URL http://doi.acm.org/10.1145/2487575.2487629

Veloso B, Gama J, Malheiro B (2018) Self hyper-parameter tuning for data streams. In: Soldatova L, Vanschoren J, Papadopoulos G, Ceci M (eds) Discovery Science, Springer International Publishing, Cham, pp 241–255, DOI 10.1007/978-3-030-01771-2_16

Wang W, Chen C, Wang W, Rai P, Carin L (2016) Earliness-aware deep convolutional networks for early time series classification. arXiv preprint URL https://arxiv.org/abs/1611.04578

Xing Z, Pei J, Yu PS, Wang K (2011a) Extracting interpretable features for early classification on time series. In: Proceedings of the Eleventh SIAM International Conference on Data Mining, SIAM / Omnipress, pp 247–258, DOI 10.1137/1.9781611972818.22

Xing Zz, Pei J, Yu P (2011b) Early classification on time series. Knowledge and Information Systems 31, DOI 10.1007/s10115-011-0400-x

Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8(2):173–195, DOI 10.1162/106365600568202

A hyperparameter space of MultiETSC

This appendix provides a more detailed description of the search space that MultiETSC searches. Table 5 provides an overview of all hyperparameters and their descriptions. Note that some hyperparameters might be conditional on other, higher-level parameters. For example, `dSigma` is only set when `distance` is set to `EDR`. Note that to MultiETSC the algorithm choice is itself a high-level hyperparameter.

Algorithm	hyperparameter		Description
ECTS (Xing et al., 2011b)	`min_support`	int	Controls over-fitting
	`version`	cat	Strict or Loose version
EDSC (Xing et al., 2011a)	`minK`	int	Min. shapelet length
	`maxK`	int	Max. shapelet length
	`alph`	real	Accuracy-earliness trade off
	`method`	cat	Threshold learning method
	`boundThreshold`	real	Precision-recall trade-off of shapelets (method=ALL only)
	`probablityThreshold`	real	Precision-recall trade-off of shapelets (method=KDE only)
	`recallThreshold`	real	Min. threshold for shapelet distinctiveness
RelClass (Parrish et al., 2013)	`tau`	real	Min. reliability threshold
	`constr`	cat	Chebyshev, Quadratic or Box constraint
	`LDG`	bin	Use LDG dimensionality reduction or not
ECDIRE (Mori et al., 2016)	`acc_perc`	int	Desired level of accuracy as percentage of full data classification
	`doHPO`	bin	Controlling optimisation of kernel parameters of GPCs
Shared with SR-CF	`kernel`	cat	Gaussian process regression kernel
	`distance`	cat	Distance metric to use
	`dSigma`	real	Threshold value for the EDR metric
	`dN`	int	No. components for the Fourier distance metric
SR-CF (Mori et al., 2018)	`alpha`	real	Accuracy-earliness trade-off
	`optimiser`	cat	Method used for CF optimisation
	`sr`	cat	Stopping rule type
	`reg`	cat	Method used for CF regularisation
	`lambda`	real	Regularisation factor
TEASER (Schäfer and Leser, 2020)	`threshold`	int	No. required consistent classifications
	`svmKernelType`	cat	Kernel type to use for oc-SVM
	`nu`	real	Training parameter for oc-SVM
Shared with ECEC	`nClassifiers`	int	Number of prefix classifiers
	`minLen index`	int	Position of the first prefix classifier
	`maxLen index`	int	Position of the last prefix classifier
ECEC (Lv et al., 2019)	`ratio`	real	Accuracy-earliness trade off
	`nFolds`	int	Number of cross validation folds
EARLIEST (Hartvigsen et al., 2019)	`LAMBDA`	real	Accuracy-earliness trade off
	`lr`	real	Learning rate
	`lrf`	real	Learning rate decay factor
	`epochs`	int	
	`nLayers`	int	No. hidden layers
	`hiddenDim`	int	No. nodes per hidden layer
	`cellType`	cat	RNN cell type
1NN-Fixed	`percLen`	real	Prefix length as prop. of total TS length

Table 5: EarlyTSC algorithms and their hyperparameters. 'int', 'cat', 'real' and 'bin' stand for integer, categorical, real and binary valued parameters, respectively.

Table 6 shows all hyperparameter values that are used in MultiETSC. In order to be compatible with MO-ParamILS, real values were discretised. Where possible, the choice of these values was based on the values considered in the original papers.

hyperparameter	Values
algorithm	{ects, edsc, relclass, ecdire, srcf, teaser, ecec, earliest, fixed}
min_support	{.0, .01, ..., 1.0}
version	{strict,loose}
minK	{1, 2, 3, 4, 5, 10, 20, 50, 100, 200}
maxK	{3, 5, 10, 20, 50, 60, 70, 100}
alph	{20, 30, 50, 60, 70, 80, 100, 150}
method	{ALL, KDE}
boundThreshold	{1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0}
probablityThreshold	{.5, .7, .8, .9, .95, .975, .99}
recallThreshold	{.5, .2, .1, .05, .02, .01, .005, .002, .001, .0}
tau	{2^-1, 2^-2, ..., 2^-100}
constr	{boxco, Naive, Cheby}
LDG	{0, 1}
acc_perc	{0, 1, ..., 100}
doHPO	{TRUE, FALSE}
kernel	{gauss, iprod, cauchy, laplace}
distance	{euclidean, dtw, edr, fourier}
dSigma	{.01, .02, .05, .1, .2, .5, 1, 1.5, 2, 3}
dN	{1, 2, 5, 10, 20, 50}
alpha	{.0, .01, ..., 1.0}
optimiser	{ga, optim, sa, pso}
sr	{sr1, sr2}
reg	{none, L0, L1}
lambda	{.01, .02, .05, .1, .2, .5, 1, 2, 5, 10, 20, 50}
threshold	{1, 2, ..., 10}
svmKernelType	{LINEAR, POLY, RBF, SIGMOID}
nu	{.5, .2, .1, .05, .02, .01, .005, .002, .001}
nClassifiers	{10, 15, 20, 30, 50, 100}
minLen index	{3, 5, 10, 20, 50, 100}
maxLen index	{50, 75, 100, 150, 200, 300, 450}
ratio	{.01, .02, ..., 1.0}
nFolds	{1, 2, ..., 10}
LAMBDA	{.0, .0015, ..., .15}
lr	{1e-4, 1e-3, 1e-2, 2e-2, 3e-2}
lrf	{1., .999, .995, .99, .975, .95, .9}
epochs	{3, 10, 20, 40, 60, 80, 100, 150, 200, 400, 800, 1000}
nLayers	{1, 2, 3, 4}
hiddenDim	{1, 2, ..., 20}
cellType	{LSTM, GRU, RNN, RNN_TANH, RNN_RELU}
percLen	{.01, .02, ..., 1.0}

Table 6: All hyperparameter values considered in MultiETSC

Table 7 shows the number of possible configurations, the average number of configurations evaluated per run and the average number of non-dominated configurations per subsample of 10 runs. Note that the number of configurations for MultiETSC and SO-ALL is simply the sum of the number of configurations for each included algorithm. Also, note that for ECTS and Fixed all the configuration phase is consistently exhaustive, meaning all possible configurations are evaluated.

Algorithm	Configurations		
	Possible	avg. evaluated	avg. non-dominated
ECTS	200	200	2.18
EDSC	102 400	706.80	2.12
RelClass	606	79.69	4.44
ECDIRE	16 000	45.42	1.91
SR-CF	1 184 000	50.35	1.97
TEASER	90 720	61.27	6.98
ECEC	252 000	41.30	4.47
EARLIEST	16 800 000	55.78	1.65
Fixed	100	100	10.70
SO-All	18 446 026	63.83	3.25
MultiETSC w/o Fixed	18 445 926	134.74	6.18
MultiETSC-o	18 446 026	109.64	9.32
MultiETSC	18 446 026	159.40	9.75

Table 7: The number of possible configurations, the average number of configurations evaluated per run and the average number of non-dominated configurations per subsample of 10 runs.

B Additional, detailed results

This appendix provides tables of per-dataset median values of the three Pareto-set performance metrics that were computed. These metrics being hypervolume (see Section 6.4), Δ-spread (Eq. 8) and *HM* metric (Eq. 7). The reported values are the medians of the 1000 bootstrap subsamples of size 10, resampled from 25 runs (see Section 6.3). In some cases, not a single viable configuration was found in any of the subsamples. In these cases, the table entry is left blank.

method dataset	Fixed	ECTS	EDSC	ECDIRE	SR-CF	RelClass	TEASER	ECEC	EARLIEST	SO-All	MultiETSC
ACSF1	0.674	0.085			0.712		0.747	0.728	0.360	**0.748**	0.735
Adiac	0.562	0.162					**0.591**		0.076	0.526	0.563
AllGestureWiimoteX	0.100	0.000					**0.399**	0.332	0.160	0.306	0.365
AllGestureWiimoteY	0.100	0.011					**0.472**	0.409	0.185	0.350	0.421
AllGestureWiimoteZ	0.100	0.007					**0.379**	0.306	0.150	0.317	0.353
ArrowHead	0.661	0.129	0.523	0.639	0.551	0.655	0.731	**0.733**	0.381	0.601	0.688
BME	0.616	0.185	0.604	**0.732**	0.706	0.683	0.724	0.701	0.351	0.642	0.722
Beef	0.503	0.125	0.430	0.487	0.603	0.614	**0.744**	0.684	0.399	0.680	0.743
BeetleFly	0.747	0.203	0.463	0.617	0.617	0.549	**0.827**	0.645	0.748	0.633	0.697
BirdChicken	0.710	0.223	0.530	0.630	0.586	0.731	**0.792**	0.692	0.549	0.665	0.778
CBF	0.748	0.192	0.683	0.763	0.808	0.763	0.820	0.810	0.442	0.780	**0.835**
Car	0.678	0.179	0.557	0.658	0.602	0.507	**0.863**	0.819	0.283	0.763	0.839
Chinatown	**0.945**	0.169	0.763	0.927	0.925	0.854	0.827	0.821	0.931	0.933	**0.945**
ChlorineConcentration	0.593	0.191				0.627	**0.667**	0.654	0.533	0.648	0.666
CinCECGTorso	0.701	0.394		0.556	0.596	0.177	0.835	0.779	0.327	0.788	**0.843**
Coffee	0.893	0.280	0.826	0.818	0.844	**0.898**	0.862	0.874	0.534	0.816	0.896
Computers	0.575	0.049		0.593	0.593	0.511	0.744	0.738	0.531	0.741	**0.754**
CricketX	0.484	0.143				0.102	0.503	0.373	0.153	0.443	**0.520**
CricketY	0.490	0.124				0.315	**0.515**	0.419	0.166	0.446	0.506
CricketZ	0.493	0.134				0.102	**0.557**	0.343	0.148	0.464	0.501
DistalPhalanxOutlineAgeGroup	0.632	0.105	0.616			0.671	0.685	0.667	0.667	0.652	**0.711**
DistalPhalanxOutlineCorrect	0.693	0.107	0.650	0.661	0.661	0.665	0.744	0.713	0.648	0.675	**0.758**
DistalPhalanxTW	0.564	0.129	0.549			**0.631**	0.596	0.596	0.298	0.601	0.625
DodgerLoopDay	0.150						0.420	0.406		0.399	**0.438**
DodgerLoopGame	0.522						**0.527**	0.516			0.522
DodgerLoopWeekend	0.739						0.901	**0.927**		0.875	0.739
ECG200	0.854	0.362	0.737	0.812	0.817	0.795	0.849	0.832	0.633	0.846	**0.862**
ECG5000	0.904	0.070	0.718			0.851	0.900	0.893	0.610	0.902	**0.914**
ECGFiveDays	0.612	0.215	0.529	0.583	0.604	0.613	0.759	0.668	0.570	0.600	**0.762**
EOGHorizontalSignal	**0.306**	0.102					0.128	0.101	0.102	0.160	0.302
EOGVerticalSignal	0.295	0.068					0.086	0.082	0.086	0.213	**0.297**
Earthquakes	**0.748**	0.008		0.711	0.711	0.747	0.736	0.744	0.747	0.747	0.747
EthanolLevel	0.272	0.028				0.264	**0.382**	0.380	0.252	0.376	0.358
FaceAll	0.698	0.229				0.093	**0.785**	0.723	0.258	0.738	0.745
FaceFour	0.763	0.189	0.618	0.539	0.759	0.808	0.765	0.758	0.306	0.749	**0.810**
FacesUCR	0.647	0.077	0.521			0.351	**0.781**	0.723	0.192	0.705	0.760
Fish	0.696	0.218			0.353	0.581	**0.844**	0.777	0.308	0.798	0.768
FreezerRegularTrain	0.937	0.146	0.822	0.911	0.912	0.762	0.940	0.933	0.521	0.942	**0.951**
FreezerSmallTrain	0.724	0.209	0.828	0.642	0.636	0.763	0.900	**0.926**	0.750	0.750	0.920
GestureMidAirD1	0.226	0.018					**0.281**	0.177	0.153	0.262	0.234
GestureMidAirD2	0.161	0.001					**0.233**	0.138	0.069	0.229	0.215
GestureMidAirD3	0.139	0.003					0.138	0.085	0.084	0.134	**0.155**
GesturePebbleZ1	0.243	0.046					**0.446**	0.413	0.180	0.369	0.433
GesturePebbleZ2	0.249	0.040					0.375	0.292	0.164	0.308	**0.386**
GunPoint	0.881	0.491	0.742	0.825	0.816	0.776	0.856	0.873	0.662	0.803	**0.898**
GunPointAgeSpan	0.936	0.461	0.819	0.894	0.886	0.808	0.937	0.927	0.503	0.905	**0.963**
GunPointMaleVersusFemale	0.934	0.615	0.898	0.895	0.867	0.906	0.944	0.939	0.522	0.908	**0.963**
GunPointOldVersusYoung	0.976	0.675	0.956	0.917	0.936	0.974	0.965	0.967	0.520	0.952	**0.986**
Ham	0.635	0.071	0.424	0.664	0.656	0.713	**0.719**	0.715	0.513	0.653	0.719
HandOutlines	0.843	0.109		0.664	0.811		0.657	0.659	0.640	0.826	**0.845**
Haptics	0.378	0.016			0.355	0.211	0.381	**0.385**	0.260	0.358	0.382
Herring	0.594	0.058	0.521	0.564	0.623	**0.643**	0.611	0.621	0.593	0.593	0.624
HouseTwenty	0.703	0.055		0.735	0.716	0.487	0.914	**0.917**	0.580	0.880	0.907
InlineSkate	0.265	0.037		0.083	0.259	0.120	0.357	**0.364**	0.218	0.319	0.315
InsectEPGRegularTrain	0.995	0.555	0.991	0.926	0.950	0.998	0.946	0.931	0.998	0.998	**0.999**
InsectEPGSmallTrain	0.995	0.280	0.995	0.933	0.950	0.998	0.837	0.839	0.998	0.998	**0.999**
ItalyPowerDemand	0.791	0.200	0.666	0.787	0.793	0.794	0.712	0.709	0.658	0.682	**0.821**
LargeKitchenAppliances	0.464	0.070				0.341	**0.537**	0.528	0.333	0.439	0.529
Lightning2	0.685	0.079	0.663	0.573	0.639	0.591	**0.694**	0.665	0.671	0.671	0.672
Lightning7	0.452	0.060	0.443	0.194	0.449	0.490	**0.577**	0.509	0.355	0.495	0.549
Mallat	0.684	0.230		0.380	0.554	0.230	0.585	0.550	0.125	0.587	**0.693**
Meat	0.861	0.482	0.708	0.870	0.836	0.897	0.914	0.926	0.333	0.851	**0.928**
MedicalImages	0.693	0.291	0.546			0.587	0.691	0.664	0.509	0.678	**0.710**
MelbournePedestrian	0.100						**0.719**	0.684		0.654	0.715
MiddlePhalanxOutlineAgeGroup	0.514	0.107	0.561			0.553	0.574	0.526	0.564	0.557	**0.583**
MiddlePhalanxOutlineCorrect	0.689	0.130	0.613	0.642	0.711	**0.768**	0.729	0.674	0.563	0.647	0.748
MiddlePhalanxTW	0.467	0.081	0.453			**0.553**	0.508	0.497	0.526	0.494	0.527
MixedShapesRegularTrain	0.801	0.221				0.136	**0.850**	0.820	0.276	0.833	0.841
MixedShapesSmallTrain	0.715	0.240		0.567	0.708	0.182	**0.833**	0.731	0.262	0.725	0.778
MoteStrain	0.810	0.109	0.544	0.728	0.754	0.801	0.842	0.794	0.674	0.670	**0.868**
NonInvasiveFetalECGThorax1	**0.787**	0.142							0.063	0.741	0.764
NonInvasiveFetalECGThorax2	**0.849**	0.172							0.104	0.821	0.795
OSULeaf	0.498	0.077			0.324	0.373	**0.684**	0.617	0.313	0.596	0.654
OliveOil	0.800	0.384	0.673	0.672	0.789	0.874	**0.923**	0.866	0.399	0.852	0.916
PLAID	0.544	0.002					**0.715**	0.647	0.272	0.676	0.674
PhalangesOutlinesCorrect	0.706	0.120			0.617	**0.745**	0.718	0.703	0.633	0.707	0.745
PickupGestureWiimoteZ	0.505	0.001					**0.670**	0.632	0.180	0.637	0.636
Plane	0.933	0.560	0.808		0.923	0.867	0.930	0.942	0.586	0.899	**0.948**
PowerCons	0.807	0.199	0.606	0.801	0.805	0.745	0.830	0.764	0.563	0.713	**0.853**
ProximalPhalanxOutlineAgeGroup	0.766	0.140	0.790			0.778	0.818	0.800	0.780	0.820	**0.841**
ProximalPhalanxOutlineCorrect	0.768	0.131	0.678	0.714	0.771	0.746	0.817	0.793	0.702	0.725	**0.835**
ProximalPhalanxTW	0.693	0.115	0.726			0.654	0.770	0.756	0.347	0.761	**0.780**
RefrigerationDevices	0.434	0.032			0.466	0.272	**0.552**	0.550	0.333	0.549	0.547
Rock	0.456	0.145		0.487	0.380		**0.629**	0.459	0.420	0.555	0.519
ScreenType	0.409	0.022				0.309	0.402	**0.412**	0.338	0.400	0.402
SemgHandGenderCh2	0.793	0.129		0.810	0.820	0.673	0.694	0.695	0.551	0.800	**0.856**
SemgHandMovementCh2	0.501	0.059					0.321	0.327	0.233	0.452	**0.502**
SemgHandSubjectCh2	**0.617**	0.082				0.222	0.350	0.335	0.266	0.536	0.617
ShakeGestureWiimoteZ	0.595	0.000					0.671	0.629	0.239	0.623	**0.687**
ShapeletSim	0.505	0.018	0.446	0.475	0.491	0.504	**0.705**	0.651	0.499	0.503	0.620
ShapesAll	**0.647**	0.230				0.037			0.070	0.541	0.647
SmallKitchenAppliances	0.560	0.026				0.450	0.719	0.718	0.333	0.721	**0.722**
SmoothSubspace	0.702	0.069		0.518	0.730	0.758	0.671	0.659	0.311	0.614	**0.794**
SonyAIBORobotSurface1	0.761	0.152	0.586	0.750	0.676	**0.833**	0.800	0.810	0.618	0.745	0.800
SonyAIBORobotSurface2	0.754	0.366	0.502	0.700	0.657	0.730	0.748	0.717	0.660	0.657	**0.772**
StarLightCurves	0.887	0.104					0.883	0.884	0.826	0.882	**0.892**
Strawberry	0.928	0.449		0.824	0.881	0.859	0.931	0.922	0.641	0.908	**0.942**
SwedishLeaf	0.714	0.139				0.290	**0.785**		0.260	0.719	0.776
SyntheticControl	0.788	0.068	0.593			0.819	0.889	0.860	0.643	0.858	**0.900**
ToeSegmentation1	0.624	0.057	0.702	0.521	0.525	0.567	**0.827**	0.803	0.507	0.513	0.765
ToeSegmentation2	0.829	0.205	0.559	0.774	0.623	0.664	0.789	0.834	0.483	0.658	**0.850**
Trace	0.745	0.334	0.668	0.622	0.712	0.622	**0.865**	0.729	0.667	0.736	0.862
TwoLeadECG	0.776	0.214	0.727	0.802	0.827	0.720	0.819	0.843	0.611	0.829	**0.879**
TwoPatterns	**0.516**	0.121				0.456	0.456	0.362	0.259	0.425	0.483
UMD	0.576	0.158	0.532	0.589	0.682	0.594	**0.742**	0.637	0.338	0.593	0.714
UWaveGestureLibraryAll	0.784	0.156					0.386		0.285	0.674	**0.785**
UWaveGestureLibraryX	0.574	0.069				0.111	0.478	0.262	0.291	0.523	**0.584**
UWaveGestureLibraryY	0.519	0.056				0.147	0.443		0.279	0.462	**0.523**
UWaveGestureLibraryZ	0.545	0.064				0.152	0.436		0.284	0.489	**0.548**
Wafer	0.989	0.558		0.920	0.920	0.975	0.979	0.975	0.886	0.986	**0.997**
Wine	0.640	0.471	0.670	0.623	0.581	**0.717**	0.694	0.694	0.498	0.554	0.688
WordSynonyms	0.468	0.068				0.048	0.460	0.374	0.219	0.425	**0.475**
Worms	0.499	0.070			0.494	0.311	0.637	0.593	0.428	0.644	**0.648**
WormsTwoClass	0.711	0.078		0.538	0.565	0.571	0.689	0.568	0.571	0.596	**0.724**
Yoga	0.769	0.201		0.748	0.745	0.575	0.815	0.802	0.545	0.775	**0.820**

Table 8: Median hypervolume metric for compared methods on different dataset in the UCR Archive. Best values are marked. Cases where no configurations are found are left blank.

method dataset	Fixed	ECTS	EDSC	ECDIRE	SR-CF	RelClass	TEASER	ECEC	EARLIEST	SO-All	MultiETSC
ACSF1	0.663				**0.530**	0.965	0.942	0.901	0.832	0.924	0.989
Adiac	1.019	0.938					**0.656**		0.975	0.894	0.976
AllGestureWiimoteX							0.762	0.838	0.949	0.735	0.794
AllGestureWiimoteY		0.983					0.750	0.848	0.921	0.974	0.727
AllGestureWiimoteZ							0.805	0.863	0.964	0.771	0.872
ArrowHead	0.918	0.996	0.822	0.872	**0.813**	0.887	0.838	0.825	0.982	0.892	0.828
BME	0.896	0.919	0.925	0.799	0.863	**0.737**	0.764	0.786	0.976	0.898	0.863
Beef	0.784		0.915	0.764	**0.758**	0.792	0.818	0.922	0.956	0.769	0.770
BeetleFly	0.942	0.892	0.936	0.941		**0.440**	0.718	0.939	0.859	0.941	0.859
BirdChicken	0.913	0.908	0.859	0.877	**0.704**	0.723	0.711	0.921	0.970	0.882	0.869
CBF	0.895	0.875	0.917	0.975	0.918	0.905	**0.823**	0.909	0.995	0.876	0.877
Car	**0.750**	0.981	0.933	0.895	0.936	0.924	0.797	0.851	0.946	0.898	0.753
Chinatown	**0.638**	0.956	0.958			0.989	0.978	0.977	0.870	0.951	**0.638**
ChlorineConcentration	0.975	0.998				**0.683**	0.956	0.906	0.998	0.944	0.909
CinCECGTorso	0.869	0.971		0.874	0.957	1.000	**0.867**	0.934	0.996	0.950	0.870
Coffee	0.853		0.845	0.852	**0.827**	0.838	0.853	0.858	0.957	0.959	0.906
Computers	**0.960**	0.974		0.988	0.975	0.998	1.125	0.970	0.971	0.965	0.962
CricketX	0.728	0.953				**0.419**	0.867	0.888	0.996	0.937	0.833
CricketY	0.754					**0.752**	0.876	0.858	0.983	0.921	0.839
CricketZ	0.793	0.977				**0.363**	0.788	0.897	0.993	0.900	0.832
DistalPhalanxOutlineAgeGroup	0.907	0.943	0.913			0.947	0.758	**0.682**	0.878	0.884	0.774
DistalPhalanxOutlineCorrect	0.941	0.958	0.844	0.876	**0.629**	1.018	0.841	0.796	0.981	0.954	0.809
DistalPhalanxTW	0.872	0.989	0.875			0.930	0.851	**0.655**		0.938	0.754
DodgerLoopDay							0.748	0.893		0.853	0.779
DodgerLoopGame							0.865	0.791		0.861	0.589
DodgerLoopWeekend							0.852	0.953		0.994	0.854
ECG200	0.933	0.962	0.934	0.948	0.917	**0.847**	0.930	1.024	0.994	0.938	0.920
ECG5000	0.900	0.988	**0.856**			0.868	1.098	0.968	0.993	0.934	1.000
ECGFiveDays	0.977	0.856	0.918	0.940		**0.785**	0.890	1.037	0.991	0.943	0.951
EOGHorizontalSignal	0.828	0.966				0.998	0.940	0.982	0.987	0.960	**0.818**
EOGVerticalSignal	0.912	0.990					0.953	0.983	0.996	0.964	**0.899**
Earthquakes		0.995					0.772				0.579
EthanolLevel	0.978					0.995	1.021	**0.958**	0.996	0.979	0.964
FaceAll	0.766	0.955				**0.364**	0.774	0.991	0.966	0.987	0.777
FaceFour	0.879		0.864	0.817	**0.661**	0.887	0.826	0.919	0.978	0.905	0.846
FacesUCR	**0.729**	0.968	0.773			0.906	0.775	0.933	0.997	0.778	0.753
Fish	0.773	0.949				0.835	**0.771**	0.939	0.891	0.863	0.788
FreezerRegularTrain	0.878		0.943	0.889	1.035	**0.800**	0.962	1.037	0.986	0.922	0.923
FreezerSmallTrain	0.802		0.898	0.975	0.950	0.998	1.027	0.963	0.996	0.815	**0.726**
GestureMidAirD1	0.896	0.968					**0.817**	0.873	0.965	0.835	0.918
GestureMidAirD2	0.896						**0.790**	0.901	0.979	0.834	0.905
GestureMidAirD3	0.956	0.996					**0.807**	0.875	0.984	0.871	0.981
GesturePebbleZ1	0.958	0.917					**0.673**	0.808	0.980	0.718	0.804
GesturePebbleZ2	0.945	0.929					**0.680**	0.877	0.991	0.737	0.882
GunPoint	**0.841**	0.944	0.873	0.913	0.926	0.872	0.889	0.872	0.971	0.907	0.842
GunPointAgeSpan	1.040	**0.700**	0.897	0.977	0.956	0.912	1.070	0.910	0.992	0.961	0.994
GunPointMaleVersusFemale	0.930	0.987	0.881	**0.845**	0.952	1.001	1.031	0.983	0.969	0.934	0.907
GunPointOldVersusYoung	1.051		**0.950**	0.988	0.969	0.975	1.079	0.985	0.971	0.990	1.019
Ham	0.936		**0.834**	0.910	0.954	0.902	0.957	0.876	0.982	0.946	0.877
HandOutlines	0.928	0.948		0.975	0.836	0.997	0.990	0.987	**0.835**	0.934	0.914
Haptics	0.896				0.958	**0.880**	0.965	0.954	0.972	0.931	0.923
Herring			0.727		0.991	0.890	0.757	0.959			0.952
HouseTwenty	0.946	0.947		0.953	0.878	0.972	**0.847**	0.917	0.904	0.924	0.921
InlineSkate	0.910	0.978			**0.828**	0.999	0.945	0.988	0.963	0.904	0.906
InsectEPGRegularTrain	**0.736**		0.915				1.026	0.979			0.737
InsectEPGSmallTrain	**0.736**						0.868	0.987	0.797		0.737
ItalyPowerDemand	**0.742**	0.966	0.782	0.899	0.953	0.897	0.809	0.834	0.995	0.979	0.819
LargeKitchenAppliances	**0.860**	0.953					0.979	0.994	0.991	0.941	0.914
Lightning2	0.805	0.943	0.834	0.958	0.982	**0.673**	0.899	0.963	0.981	0.952	0.960
Lightning7	0.823	0.991	0.756	0.905	0.962	0.787	**0.739**	0.965	0.991	0.805	0.773
Mallat	0.889	0.961		0.942	0.912	0.929	0.904	0.930	0.999	0.926	**0.779**
Meat	**0.706**	0.892	0.883	0.942	0.928	0.931	1.061	0.895	0.806	0.983	0.903
MedicalImages	0.911	0.889	**0.835**			0.868	0.850	0.938		0.911	0.959
MelbournePedestrian							0.686	0.938		0.913	0.851
MiddlePhalanxOutlineAgeGroup	0.897	0.962	0.978			0.886	0.959	**0.761**	0.767	0.793	0.805
MiddlePhalanxOutlineCorrect	0.828	0.968	0.828	0.961	0.885	0.945	0.772	0.867		0.941	**0.737**
MiddlePhalanxTW	0.985		0.961			1.028	0.879	0.953	0.920	0.852	**0.749**
MixedShapesRegularTrain	**0.940**	0.950				0.997	1.034	0.963	0.967	0.965	0.942
MixedShapesSmallTrain	**0.831**	0.918		0.920	0.988	0.994	0.937	0.962	0.991	0.977	0.864
MoteStrain	0.951		0.937	0.953	0.871	0.997	0.943	0.944	0.995	**0.834**	0.872
NonInvasiveFetalECGThorax1	1.099	**0.971**							0.994	1.163	1.067
NonInvasiveFetalECGThorax2	1.221	0.925							0.965	0.981	**0.744**
OSULeaf	0.884	0.926			0.953	0.881	0.906	0.890	0.929	0.874	**0.816**
OliveOil	0.830	0.949	0.811	0.828	0.856	0.889	**0.740**	0.859		0.920	0.806
PLAID	0.988	0.999					**0.795**	0.949	0.926	0.824	0.934
PhalangesOutlinesCorrect	0.912	0.972			0.960	0.874	**0.808**	0.842	0.999	0.944	0.813
PickupGestureWiimoteZ	0.739						0.699	0.832	0.974	0.667	**0.645**
Plane	0.944	0.981	0.911		0.995	0.965	1.008	**0.831**	0.966	0.903	0.943
PowerCons	1.006	0.882	0.811	0.862	0.867	0.965	0.767	0.892	0.960	0.872	**0.750**
ProximalPhalanxOutlineAgeGroup	0.928		0.982			0.833	0.958	0.914	0.832	0.942	**0.823**
ProximalPhalanxOutlineCorrect	0.922	0.959	0.845	0.987	0.905	0.878	0.843	0.854	0.992	0.971	**0.814**
ProximalPhalanxTW	0.878	0.982	0.938			0.950	0.943	0.918	**0.778**	0.895	0.819
RefrigerationDevices	0.935	0.995			0.983	0.911	1.025	0.966	0.957	**0.875**	0.970
Rock	0.915	**0.843**		0.923	0.939		0.936	0.980	0.975	0.863	0.938
ScreenType	0.984					**0.683**	0.938	0.972	0.996	0.953	0.979
SemgHandGenderCh2	0.909	0.956		0.871	0.935	0.999	0.993	0.977	0.887	**0.825**	0.840
SemgHandMovementCh2	**0.802**	0.956					0.976	0.980	0.988	0.842	0.856
SemgHandSubjectCh2	0.793	0.960					1.010	0.978	0.962	**0.739**	0.827
ShakeGestureWiimoteZ	0.780						0.677	0.831	0.932	0.769	**0.573**
ShapeletSim	0.969	0.979	0.946		0.969	0.871	**0.642**	0.738	0.997	0.706	0.883
ShapesAll	0.782	0.895							0.967	0.945	**0.773**
SmallKitchenAppliances	0.968						1.082	0.989	0.998	**0.835**	0.938
SmoothSubspace	**0.506**			0.888	0.926	0.725	0.794	0.871	0.871	0.898	0.808
SonyAIBORobotSurface1	0.830	0.955	0.957	0.940	**0.454**	0.972	0.874	0.793	0.986	0.922	0.785
SonyAIBORobotSurface2	0.950	0.977	0.973	0.987	**0.912**	0.987	0.993	0.974	0.993	0.993	0.923
StarLightCurves	**0.950**	0.997					1.168	0.983	0.999	0.975	0.959
Strawberry	**0.868**	0.870		0.944	0.959	0.911	1.124	0.911		0.905	0.899
SwedishLeaf	0.872	0.994				1.129	0.899		0.908	0.970	**0.848**
SyntheticControl	0.716	0.980	0.905			0.838	0.792	0.897	0.955	0.855	**0.666**
ToeSegmentation1	0.912		0.854	0.803	0.957	**0.764**	0.797	0.935	0.997	0.937	0.907
ToeSegmentation2	0.920	0.954	0.861	0.887	**0.625**	0.849	0.694	0.848	0.844	0.931	0.808
Trace	0.646	0.891	0.845	0.748	0.942	0.753	**0.619**	0.966	0.942	0.948	0.712
TwoLeadECG	0.846	0.835	0.850	**0.724**	0.766	0.940	0.920	0.774	0.993	0.833	0.855
TwoPatterns	**0.588**	0.998				0.850	0.833	0.917	0.997	0.840	0.994
UMD	0.937	0.941	0.901	0.854	0.858	0.699	0.703	0.808	0.977	0.837	**0.699**
UWaveGestureLibraryAll	0.616	0.970					0.867	0.993	0.997	0.997	**0.613**
UWaveGestureLibraryX	**0.682**	0.984				0.996	0.913	0.951	0.995	0.901	0.797
UWaveGestureLibraryY	**0.739**	0.996				0.998	0.855	0.879	0.999	0.919	0.850
UWaveGestureLibraryZ	**0.766**	0.984				1.000	0.915		0.992	0.948	0.841
Wafer	0.951	0.989		0.998	0.991	**0.939**	1.198	0.992		0.954	0.951
Wine	0.936	**0.577**	0.887	0.882	0.876	0.950	0.856	0.883	0.966	0.950	0.887
WordSynonyms	0.770	0.965				**0.304**	0.854	0.964		0.817	0.865
Worms	0.951				0.951	0.935	0.960	0.933		**0.897**	0.924
WormsTwoClass	0.908			0.907	**0.754**		0.978	0.936	0.914	0.914	0.889
Yoga	1.040	0.970		0.937	0.981	0.915	0.955	0.919	0.994	0.926	**0.905**

Table 9: Median Δ metric for compared methods on different dataset in the UCR Archive. Best values are marked. Cases where no configurations are found are left blank.

method dataset	Fixed	ECTS	EDSC	ECDIRE	SR-CF	RelClass	TEASER	ECEC	EARLIEST	SO-All	MultiETSC
ACSF1	0.194	0.759			0.162		0.153	0.158	0.471	**0.144**	0.152
Adiac	0.330	0.630					**0.262**		0.858	0.325	0.330
AllGestureWiimoteX	0.818	0.992					**0.501**	0.533	0.724	0.539	0.534
AllGestureWiimoteY	0.818	0.937					**0.454**	0.471	0.687	0.471	0.469
AllGestureWiimoteZ	0.818	0.949					**0.526**	0.556	0.739	0.535	0.545
ArrowHead	0.300	0.733	0.354	0.303	0.314	0.280	**0.243**	0.250	0.447	0.292	0.287
BME	0.264	0.627	0.231	0.267	**0.231**	0.291	0.253	0.238	0.479	0.265	0.240
Beef	0.465	0.707	0.379	0.349	0.317	0.355	0.192	0.198	0.429	0.196	**0.170**
BeetleFly	0.150	0.603	0.332	0.228	0.228	0.291	0.148	0.219	**0.144**	0.221	0.188
BirdChicken	0.303	0.551	0.312	0.277	0.324	0.213	**0.177**	0.188	0.291	0.205	0.189
CBF	0.210	0.643	0.216	0.193	**0.179**	0.243	0.218	0.189	0.385	0.185	0.186
Car	0.303	0.642	0.287	0.330	0.269	0.369	0.172	0.200	0.559	**0.168**	0.181
Chinatown	**0.034**	0.701	0.144	0.037	0.038	0.076	0.091	0.094	0.035	**0.034**	**0.034**
ChlorineConcentration	0.305	0.590				0.306	0.277	0.286	0.303	0.300	**0.276**
CinCECGTorso	0.337	0.411		0.382	0.328	0.700	0.126	0.136	0.507	0.124	**0.123**
Coffee	0.104	0.562	0.157	0.175	0.149	0.112	0.119	0.118	0.303	0.109	**0.100**
Computers	0.280	0.852		0.247	0.247	0.323	0.154	0.155	0.306	0.153	**0.146**
CricketX	0.415	0.653				0.814	0.343	0.439	0.733	0.408	**0.333**
CricketY	0.418	0.687				0.541	**0.341**	0.385	0.714	0.409	0.356
CricketZ	0.385	0.672				0.814	**0.309**	0.470	0.741	0.361	0.354
DistalPhalanxOutlineAgeGroup	0.245	0.731	0.238			0.279	0.217	0.216	**0.198**	0.216	0.216
DistalPhalanxOutlineCorrect	0.217	0.752	0.229	0.197	0.197	0.281	0.205	0.205	0.212	0.197	**0.187**
DistalPhalanxTW	0.300	0.684	0.335			**0.251**	0.267	0.258	0.537	0.265	0.262
DodgerLoopDay	0.739						0.443	0.451		0.459	**0.416**
DodgerLoopGame	**0.314**						0.330	0.317			**0.314**
DodgerLoopWeekend	0.150						0.056	**0.055**		0.065	0.150
ECG200	0.138	0.446	0.165	0.134	0.125	0.197	0.129	0.131	0.223	**0.124**	0.128
ECG5000	0.092	0.862	0.156			0.102	**0.069**	0.070	0.243	0.092	0.092
ECGFiveDays	0.246	0.599	0.344	0.255	0.239	0.335	**0.186**	0.311	0.272	0.248	0.246
EOGHorizontalSignal	**0.571**	0.700					0.771	0.815	0.815	0.653	**0.571**
EOGVerticalSignal	**0.552**	0.756					0.842	0.847	0.842	0.568	0.558
Earthquakes	**0.144**	0.978		0.163	0.163	0.145	0.153	0.146	0.145	0.145	0.145
EthanolLevel	0.581	0.855				0.582	**0.449**	0.450	0.597	0.456	0.474
FaceAll	0.236	0.573				0.774	0.174	**0.157**	0.588	0.236	0.236
FaceFour	0.198	0.640	0.268	0.314	**0.155**	0.165	0.178	0.170	0.531	0.173	0.182
FacesUCR	0.319	0.819	0.409			0.542	**0.217**	0.229	0.676	0.232	0.236
Fish	0.266	0.591			0.466	0.301	0.190	0.196	0.529	**0.185**	0.189
FreezerRegularTrain	0.114	0.717	**0.112**	0.116	0.115	0.284	0.129	0.127	0.315	0.115	0.115
FreezerSmallTrain	0.201	0.587	0.149	0.211	0.215	0.158	0.145	**0.137**	0.143	0.142	0.142
GestureMidAirD1	0.622	0.894					**0.573**	0.680	0.733	0.599	0.615
GestureMidAirD2	0.710	0.980					**0.624**	0.742	0.871	0.630	0.638
GestureMidAirD3	0.741	0.959					**0.724**	0.832	0.844	0.752	0.731
GesturePebbleZ1	0.615	0.806					0.502	0.503	0.695	0.503	**0.495**
GesturePebbleZ2	0.611	0.827					**0.553**	0.559	0.717	0.593	0.563
GunPoint	0.171	0.331	0.184	**0.168**	0.169	0.254	0.177	0.180	0.202	0.187	0.172
GunPointAgeSpan	**0.063**	0.360	0.231	0.073	0.085	0.180	0.096	0.099	0.329	0.065	**0.063**
GunPointMaleVersusFemale	0.132	0.235	0.164	0.122	0.149	0.145	**0.094**	0.119	0.313	0.105	0.101
GunPointOldVersusYoung	0.037	0.193	0.046	0.052	0.052	0.034	**0.025**	0.026	0.314	0.027	0.027
Ham	0.257	0.806	0.365	0.256	0.256	0.254	0.221	**0.216**	0.321	0.218	0.226
HandOutlines	**0.172**	0.780		0.201	0.176		0.207	0.206	0.219	0.191	**0.172**
Haptics	0.477	0.924			0.464	0.652	0.447	**0.443**	0.588	0.480	0.462
Herring	0.255	0.825	0.292	0.269	**0.224**	0.271	0.251	0.237	0.255	0.255	0.244
HouseTwenty	0.207	0.854		0.213	0.173	0.334	**0.084**	0.084	0.266	0.092	0.099
InlineSkate	0.626	0.832		0.825	0.593	0.786	0.476	**0.466**	0.642	0.516	0.520
InsectEPGRegularTrain	0.005	0.286	0.004	0.038	0.026	**0.001**	0.039	0.037	0.001	**0.001**	**0.001**
InsectEPGSmallTrain	0.005	0.563	0.003	0.034	0.026	**0.001**	0.090	0.087	0.001	**0.001**	0.001
ItalyPowerDemand	0.226	0.653	0.272	**0.186**	0.192	0.213	0.251	0.251	0.201	**0.186**	0.192
LargeKitchenAppliances	0.371	0.778				0.491	**0.303**	0.309	0.500	0.389	0.310
Lightning2	0.298	0.806	0.226	0.261	0.213	0.271	0.215	0.211	**0.197**	**0.197**	0.210
Lightning7	0.411	0.828	0.408	0.619	**0.378**	0.389	0.389	0.391	0.475	0.401	0.389
Mallat	**0.284**	0.600		0.432	0.294	0.624	0.312	0.328	0.778	0.288	**0.284**
Meat	0.119	0.354	0.218	0.122	0.112	0.108	0.084	0.077	0.500	0.090	**0.075**
MedicalImages	**0.216**	0.478	0.317			0.335	0.229	0.237	0.323	0.225	**0.216**
MelbournePedestrian	0.817						0.264	0.267		**0.262**	0.270
MiddlePhalanxOutlineAgeGroup	0.351	0.710	**0.265**			0.318	0.289	0.309	0.277	0.309	0.296
MiddlePhalanxOutlineCorrect	0.259	0.722	0.251	0.231	**0.224**	0.241	0.238	0.270	0.277	0.238	0.237
MiddlePhalanxTW	0.382	0.762	0.401			**0.287**	0.351	0.336	0.308	0.336	0.336
MixedShapesRegularTrain	0.230	0.611				0.760	**0.094**	0.105	0.568	0.100	0.101
MixedShapesSmallTrain	0.279	0.566		0.319	0.241	0.691	**0.156**	0.160	0.585	0.163	0.165
MoteStrain	0.147	0.782	0.297	0.170	0.162	0.163	0.133	**0.130**	0.193	0.196	0.140
NonInvasiveFetalECGThorax1	**0.152**	0.715							0.881	0.153	**0.152**
NonInvasiveFetalECGThorax2	0.126	0.680							0.811	**0.125**	0.130
OSULeaf	0.388	0.762			0.501	0.454	0.273	0.269	0.522	0.277	**0.268**
OliveOil	0.205	0.426	0.258	0.240	0.164	**0.094**	0.104	0.101	0.429	0.097	0.098
PLAID	0.308	0.975					**0.219**	0.224	0.573	0.224	0.240
PhalangesOutlinesCorrect	0.240	0.738			0.229	0.227	0.230	0.232	**0.223**	0.230	0.240
PickupGestureWiimoteZ	0.331	0.970					**0.300**	0.314	0.695	0.310	0.308
Plane	0.087	0.279	0.126		**0.076**	0.112	0.086	0.101	0.259	0.098	0.088
PowerCons	0.171	0.657	0.281	0.172	0.176	0.311	0.186	0.192	0.279	0.176	**0.165**
ProximalPhalanxOutlineAgeGroup	0.144	0.712	0.116			0.208	0.116	0.120	0.123	0.119	**0.113**
ProximalPhalanxOutlineCorrect	0.188	0.729	0.229	**0.167**	0.171	0.243	0.175	0.177	0.173	0.175	0.175
ProximalPhalanxTW	0.217	0.733	**0.166**			0.284	0.167	0.169	0.482	0.169	0.169
RefrigerationDevices	0.396	0.868			0.353	0.542	0.294	0.304	0.500	**0.293**	0.294
Rock	0.391	0.636		0.345	0.437		**0.254**	0.370	0.409	0.292	0.320
ScreenType	0.437	0.898				0.520	0.432	**0.427**	0.494	0.436	0.437
SemgHandGenderCh2	0.212	0.751		**0.182**	0.184	0.195	0.201	0.195	0.289	0.194	**0.182**
SemgHandMovementCh2	**0.384**	0.830					0.515	0.504	0.622	0.390	**0.384**
SemgHandSubjectCh2	**0.311**	0.814				0.636	0.486	0.499	0.579	0.318	**0.311**
ShakeGestureWiimoteZ	0.265	0.995					**0.251**	0.275	0.613	0.253	0.269
ShapeletSim	0.333	0.934	0.349	0.345	0.331	0.329	0.330	**0.316**	0.334	0.334	0.333
ShapesAll	**0.310**	0.566				0.929			0.869	**0.310**	**0.310**
SmallKitchenAppliances	0.285	0.876				0.379	0.164	0.167	0.500	**0.163**	0.164
SmoothSubspace	**0.277**	0.865		0.306	0.282	0.286	0.291	0.301	0.509	**0.277**	**0.277**
SonyAIBORobotSurface1	0.142	0.672	0.235	0.141	0.183	**0.108**	0.124	0.124	0.234	0.155	0.139
SonyAIBORobotSurface2	0.201	0.427	0.301	**0.176**	0.200	0.178	0.181	0.181	0.204	0.205	0.195
StarLightCurves	0.109	0.787					**0.075**	0.076	0.095	0.078	0.095
Strawberry	0.111	0.368		0.131	0.116	0.142	0.097	0.109	0.218	0.106	**0.095**
SwedishLeaf	0.259	0.710				0.665	**0.173**		0.585	0.223	0.259
SyntheticControl	0.201	0.859	0.308			0.192	**0.146**	0.164	0.235	0.169	0.150
ToeSegmentation1	0.299	0.852	0.257	0.323	0.301	0.317	0.243	**0.242**	0.326	0.319	0.260
ToeSegmentation2	**0.102**	0.629	0.316	0.187	0.300	0.316	0.208	0.114	0.348	0.218	**0.102**
Trace	0.154	0.442	0.274	0.225	0.208	0.384	**0.144**	0.156	0.199	0.156	0.199
TwoLeadECG	0.207	0.584	0.232	0.198	0.206	0.229	**0.186**	0.209	0.239	0.198	0.187
TwoPatterns	0.508	0.764				0.455	**0.446**	0.511	0.586	0.463	0.461
UMD	0.302	0.672	0.286	0.275	**0.253**	0.372	0.257	0.300	0.493	0.290	0.310
UWaveGestureLibraryAll	**0.249**	0.718					0.435		0.556	0.250	**0.249**
UWaveGestureLibraryX	0.384	0.833				0.799	**0.367**	0.568	0.549	0.384	0.384
UWaveGestureLibraryY	0.422	0.848				0.743	**0.391**		0.563	0.423	0.422
UWaveGestureLibraryZ	**0.393**	0.829				0.735	0.401		0.556	0.393	**0.393**
Wafer	**0.013**	0.282		0.041	0.041	0.060	0.021	0.022	0.060	**0.013**	**0.013**
Wine	0.263	0.365	0.209	0.234	0.272	**0.207**	0.208	0.257	0.334	0.269	0.233
WordSynonyms	0.471	0.811				0.867	**0.402**	0.451	0.640	0.433	0.426
Worms	0.365	0.774			0.328	0.526	0.243	0.259	0.400	**0.241**	**0.241**
WormsTwoClass	**0.216**	0.797		0.297	0.318	0.273	0.231	0.277	0.273	0.261	**0.216**
Yoga	0.184	0.621		0.182	0.179	0.293	0.164	0.174	0.294	0.169	**0.163**

Table 10: Median *HM* metric for compared methods on different dataset in the UCR Archive. Best values are marked. Cases where no configurations are found are left blank.