Utilizing Data Fingerprints for Privacy-Preserving Algorithm Selection in Time Series Classification: Performance and Uncertainty Estimation on Unseen Datasets

Lars Böcking*
University of Bayreuth
& Fraunhofer FIT
lars.boecking@uni-bayreuth.de

Leopold Müller*
University of Bayreuth
& Fraunhofer FIT
leopold.mueller@uni-bayreuth.de

Niklas Kühl
University of Bayreuth
& Fraunhofer FIT
kuehl@uni-bayreuth.de

Abstract

The selection of algorithms is a crucial step in designing AI services for real-world time series classification use cases. Traditional methods such as neural architecture search, automated machine learning, combined algorithm selection, and hyperparameter optimizations are effective but require considerable computational resources and necessitate access to all data points to run their optimizations. In this work, we introduce a novel data fingerprint that describes any time series classification dataset in a privacy-preserving manner and provides insight into the algorithm selection problem without requiring training on the (unseen) dataset. By decomposing the multi-target regression problem, only our data fingerprints are used to estimate algorithm performance and uncertainty in a scalable and adaptable manner. Our approach is evaluated on the 112 University of California riverside benchmark datasets, demonstrating its effectiveness in predicting the performance of 35 state-of-the-art algorithms and providing valuable insights for effective algorithm selection in time series classification service systems, improving a naive baseline by 7.32% on average in estimating the mean performance and 15.81% in estimating the uncertainty.

Keywords: time series classification, performance estimation, quantification of model risk

1. Introduction

Time series classification involves analyzing sequences of data points, indexed in time order, to categorize them into predefined classes. It is crucial in various services such as health record analysis (W. K. Wang et al., 2022), predictive maintenance (Rudolph et al., 2020), cyber-security (MontazeriShatoori et al., 2020), and earthquake prediction (Arul & Kareem, 2021), reflecting its wide-ranging impact in both scientific and practical applications.

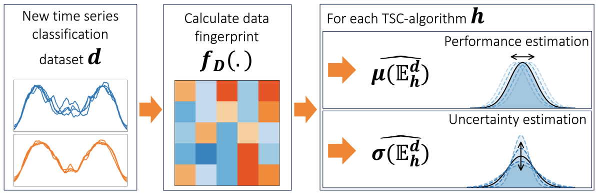
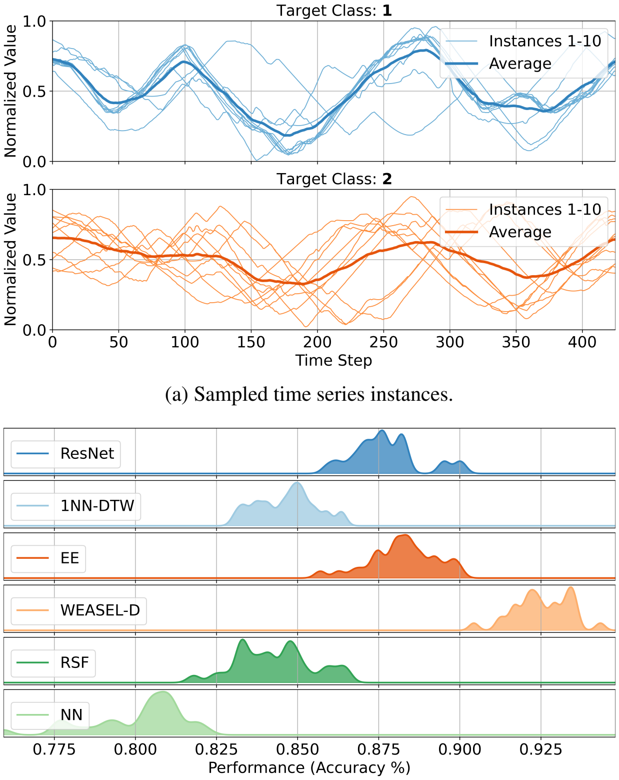

New time series classification dataset $\boldsymbol{d}$
Calculate data fingerprint
$\boldsymbol{f_D}(.)$
For each TSC-algorithm $\boldsymbol{h}$
Performance estimation
$\widehat{\boldsymbol{\mu}(\mathbb{E}_h^d)}$
Uncertainty estimation
$\widehat{\boldsymbol{\sigma}(\mathbb{E}_h^d)}$

Figure 1: Approach for performance estimation in time series classification. Inspired by Amini et al., 2020.

The plethora of algorithms developed for time series classification presents a significant challenge: the selection of the most appropriate algorithm in a service to achieve the best performance for a specific dataset—and, therefore, the related domain problem—is a complex task (Oreski & Redep, 2018). This task is formalized by Rice and is known as the algorithm selection (AS) problem (Rice, 1976). According to the "no free lunch" theorem, under certain assumptions, no algorithm uniformly dominates all others (Wolpert & Macready, 1997). It is not known which algorithm will perform best on the new dataset. Therefore, we present an approach to estimate these performances based on a data fingerprint, describing the dataset in an aggregated manner Figure 1. As a result, our approach allows us to differentiate variance in classification accuracies among different algorithms on real-world datasets, such as the UCR *Yoga*-dataset (Bagnall et al., 2017), an observation highlighted in Figure 2b.

Organizations, researchers and service provider alike are constrained by limited resources—time,

[*] These authors contributed equally to this work



(a) Sampled time series instances.

(b) Histogram on the classification performance.

Figure 2: Problem statement: Mapping the time series instances to the algorithm performance. Figure 2a: Ten sampled time series instances of each target class and their averages in the *Yoga*-dataset. Figure 2b: Histogram of the classification performance of different algorithms on the *Yoga*-dataset across 30 cross validation folds. Achieved accuracy on the x-axis and density on the y-axis.

computation, data, and expertise. Other methods, such as neural architecture search (NAS) (Elsken et al., 2019), automated machine learning (AutoML) (Feurer et al., 2022; Hutter et al., 2019), transfer learning (TL) (Lu et al., 2015), hyperparameter optimization (HPO) (Bischl et al., 2023) and algorithm configuration (Lindauer et al., 2015) engage in broad algorithmic experimentation or adaptation of pre-trained models for similar tasks. These methods, despite their effectiveness, face substantial resource demands and complexity in selecting the most appropriate algorithm for possible deployment within an AI service (Elsken et al., 2018). In addition, they necessitate access to data points to run their optimizations, compromising data privacy. In scenarios where data privacy is a concern, the AI service provider needs a solution that allows for informed decision-making without requiring full access to the dataset. For a new, unseen time series classification dataset, the service provider wants to assess which state-of-the-art algorithm is most promising without training the algorithms themselves, running HPO or NAS. Instead, this can be achieved by analyzing the dataset's characteristics, enabling the service to act as an assistant in the AI development process while maintaining data privacy.

To address these shortcomings, we want to answer the following research questions (RQs): **(RQ1)** How can diverse time series datasets be translated into a standardized input format to facilitate comparison and analysis? **(RQ2)** To what extent can this standardized input format be used to estimate the expected performance of various state-of-the-art classification algorithms? **(RQ3)** How can the uncertainty associated with the predicted algorithm performance on these standardized inputs be estimated?

In our work, we propose a data fingerprint to characterize datasets. We translate the algorithm selection (AS) problem into a multi-target regression problem that estimates algorithm performance and uncertainty, as illustrated in Figure 1. We train various regressors on data fingerprints from benchmark datasets to predict the performance of time series classification algorithms on these benchmarks. Once trained, these regressors can be applied to new, unseen data fingerprints to predict how each algorithm within a service system will perform on the new datasets. As a result, our approach effectively suggests the most suitable algorithm for any new dataset in a privacy-preserving manner, as only the data fingerprint is shared, not the actual data points. Therefore, a service provider does not need to access the dataset but can assist in the AI development process by suggesting the most promising classification algorithm—only by processing the fingerprint, which does not expose any data points. Our approach is highly customizable and can provide tailored suggestions by predicting other target variables in addition to accuracy and uncertainty to provide a basis for informed decision-making for AS in AI services.

Our contributions are threefold:

1. We introduce novel data fingerprints to form feature maps for AS representing whole time series datasets. They capture the essential attributes of any time series dataset, making it easier to compare datasets and providing a standardized input for regression models.

2. We present a customizable approach that utilizes the fingerprints and benchmark results to decompose any multi-target regression problem related to AS. This includes estimating algorithm performance and its uncertainty in the work at hand—but can easily be adapted to other target

objectives,to select the algorithm that will deliver the highest performance on a given dataset. Our approach enables efficient AS without the need for extensive training and testing of multiple algorithms, thus streamlining the process of achieving optimal model performance.

3. We extensively experiment on 112 benchmark datasets and the expected performance of 35 state-of-the-art classification algorithms on unseen datasets. The results of our analysis provide insights for researchers and practitioners navigating AS in time series classification. Surpassing a naive baseline by an average of 7.32% in estimating the mean performance and 15.81% in estimating the uncertainty, it shows the approach's potential for future AS problems. We offer an out-of-the-box framework that predicts algorithm performance on any (unseen) dataset using the fingerprints.

2. Related Work

In this section, we review various methods for algorithm selection, highlighting the computational challenges and privacy concerns associated with these methods to give an overview of the state-of-the-art in the field.

Algorithm selection: AS generally describes the selection of the most suitable algorithms for novel tasks (Rice, 1976). Unlike the broader approach that does not distinguish between general and machine learning-specific algorithms, our focus is squarely on the latter. We adopt a meta-learning perspective, utilizing machine learning algorithms not for direct selection but for estimating performance and uncertainty of potential algorithm choices. Based on these results, we use the predictions of the best algorithm selectors to make a final prediction about the expected performance and uncertainty. However, the predictions could also be used for the selection itself. For example, the algorithm with the highest performance could be selected. Other target values, such as the expected running time, could also be estimated and used as a basis for decision-making. A distinction can be made between online and offline AS (Degroote, 2017). Online AS describes the case where no training data is available in advance, and the selection is made iteratively. Our approach is to be considered offline AS.

Distinction from other methods: The selection of hyperparameters for an algorithm can also be optimized, which is referred to as HPO (Bischl et al., 2023); if they are generalized for a set of tasks, this is called algorithm configuration (Schede et al., 2022). As we use benchmarks as a data basis, we assume that the benchmarks utilized already incorporate potential performance improvements achievable through HPO or algorithm configuration. This is also an advantage over other methods such as NAS (Elsken et al., 2019) or AutoML (Hutter et al., 2019), which have also been applied to the algorithm selection problem in time-series classification (Mu et al., 2023; Parmentier et al., 2021). These methods are not only computationally intensive, as they attempt to find an optimal architecture or algorithm through targeted experimentation, but they also require full access to data points, compromising data privacy. There is ongoing research addressing the limitation of data privacy in NAS and AutoML (F. Wang et al., 2022; Yan et al., 2022; Zhang et al., 2021). These works, although primarily focused on domains other than time series classification, highlight the importance of privacy-preserving techniques in algorithm selection and model training. If both AS and HPO are carried out, it is named the combined algorithm selection and hyperparameter optimization (CASH) problem (Thornton et al., 2013). A large number of algorithms already exist for time series classification. Our work is intended to help estimate the performance and uncertainty for new datasets and thus support the selection process instead of searching for new architectures in a computationally intensive manner. TL (Lu et al., 2015) is also based on existing algorithms, which are adapted to the new task. However, here too, the expected performance is unknown in advance.

3. Approach

Consider an AI service that supports the development of time series classification solutions by recommending the most suitable algorithms based on data characteristics, enabling more informed AS and solutions tailored to the specific data of new clients. While the current state-of-the-art time series classification algorithm can be employed, this approach often overlooks the complexities and nuances inherent in real-world data. These algorithms typically rely on benchmark datasets that may not fully capture the intricacies of diverse datasets. Instead, our approach identifies the most promising algorithm based on the unique characteristics of the new client's dataset, thereby delivering a truly smart service as defined by Jussen et al., 2020.

To formalize our approach, we start by defining the time series classification problem, its performance assessment, and key concepts of our fingerprint aggregation. For a given time series x, a time series classification algorithm predicts its target class y. Each task is represented by an individual dataset d with an arbitrary number of instances x^i to be classified. Let $\mathbf{x}^{i,d}$ denote a $(1 \times T)$ vector of an univariate time series. The classification task is to learn $h(.)$: $x^{i,d} = \left(x_1^{i,d}, \dots x_T^{i,d}\right) \xrightarrow{h} y^{i,d}$ with: d : datasets, $1 \dots D, i$: instance, $1 \dots I, t$: time, $1 \dots T$.

For an algorithm $h(.)$, its classification performance on a dataset d is defined as $\mathbb{E}_h^d\left[\mathbf{1}\{y^{i,d} = h(x^{i,d})\} \mid \tau\right]$, where τ relates to all training instances and their respective labels (Hastie et al., 2009). Predicting how the algorithms will perform on this task can provide insight into which algorithm should be chosen for the task. To do this, we first translate the dataset into a standardized data fingerprint $f_D^d(.)$ of fixed size, so that it can be used as input to a regressor. Then we learn a mapping $f_D^d(.) \rightarrow \mathbb{E}_h^d$ between the data fingerprint and the expected performance $\mathbb{E}_h^d$.

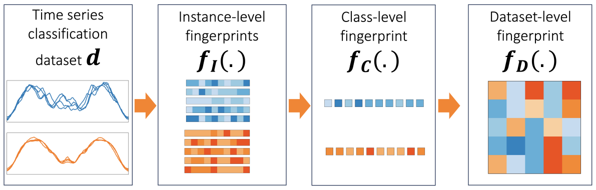

Time series classification dataset d
Instance-level fingerprints $f_I(.)$
Class-level fingerprint $f_C(.)$
Dataset-level fingerprint $f_D(.)$

Figure 3: Approach step-wise aggregation function.

More precisely, we map the data fingerprint to the mean $\mu(\mathbb{E}_h^d)$ and the standard deviation $\sigma(\mathbb{E}_h^d)$ of the performance, which occurs over multiple training runs, using k-fold cross-validation. The standardized data fingerprint $f_D^d(.)$ is derived by aggregating information about the time series dataset on three levels, motivated by balancing the discriminating power of characteristics while consolidating information about individual time series into a vector of fixed size. This step-wise aggregation is shown in Figure 3 and determines the structure of the remaining section: Instance-level fingerprint (see Section 3.1), class-level fingerprint (see Section 3.2), and the dataset-level fingerprint (see Section 3.3).

3.1. Instance-level fingerprint

On the first level, we want to describe each instance by representations of fixed size instead of the actual values themselves. Thus, we introduce an instance-level fingerprint. This instance-level fingerprint is described by $f_I(.)$ and is calculated for each instance $x^{i,d}$ of a dataset d separately: $\forall i \in I : f_I(x^{i,d})$. One exemplary representation could be to identify the deviation of change in one time step to the average deviation given by $\overline{\Delta x^{i,d}}$, such as $f_I(x^{i,d}) = \sqrt{\frac{1}{T-2}\sum_{t=1}^{T-1}\left(x_{t+1}^{i,d} - x_t^{i,d} - \overline{\Delta x^{i,d}}\right)^2}$.

Other statistical measures can also be used to capture different aspects of the time series data. The descriptive statistics are combined in a $(1 \times L_I)$ vector, where L_I is the number of statistical measures used, to form the instance-level fingerprint $f_I(x^{i,d})$. An overview of the proposed statistical measures like Skewness γ_1 and Kurtosis $Kurt[X]$ can be found in the code of this work.

3.2. Class-level fingerprint

On the next level, we want to aggregate the instance-level fingerprints of each class into class-level fingerprints of fixed size. So each class is assigned a class-level fingerprint, which results from transforming all instance-level fingerprints belonging to that class.

To aggregate the instance fingerprint for all instances of a given class c in a dataset d, we first define the set of indexes related to a specific class as $I^c = \{i | y^{i,d} = c\}$. We can then derive $f_C(.)$ as: $\forall c \in \mathcal{C} : f_C^c\left(\{f_I(x^{i,d})\}_{i\in I^c}\right)$. One option is to calculate the average of a given fingerprint across all instances of a class, so $f_C^c(.) = \frac{1}{|I^c|}\sum_{i\in I^c} f_I\left(x^{i,d}\right)$. Another option is to take the median value as a representation for the class instances. Again, $f_C(.)$ can be independently selected from $f_I(.)$ or the later defined data fingerprint $f_D(.)$. Our approach provides a generalizing concept that can be easily extended and adapted by choosing different aggregation functions.

3.3. Dataset-level fingerprint

On the last level, we aggregate the previously calculated class-level fingerprint $f_C(.)$ and extend them by meta characteristics to form a standardized dataset-level fingerprint that describes any time series classification dataset as a function of $f_D(.)$, as: $\forall d \in D : f_D^d\left(f_C^1(.), \dots, f_C^{|C|}(.)\right)$. One example aggregation function is the standard deviation of each class-level fingerprint across the available classes: $\sqrt{\frac{\sum_{c\in C}\left(f_C^c(.)) - \overline{f_C^c}\right)^2}{|C|}}$.

Besides the aggregated $f_C(.)$, we also add meta characteristics of our dataset, such as the total number of training and test instances, the length of each instance expressed as $||x^{i,d}||$, number of target classes. Moreover, the distribution of instances across classes is characterized by the minimum and maximum number of instances in any class, represented by $\min(||I_c||)$ and $\max(||I_c||)$, respectively as well as the average number of instances per class and the standard deviation of the number of instances per class. This fixed size $(1 \times L_D)$ vector, where L_D is determined by the dataset-level aggregation type and the number of meta characteristics, provides a comprehensive dataset description and serves as the input for our multi-target regression problem. Details on the decomposition of this multi-target regression are discussed in the following subsection.

3.4. Performance estimation

There is no single algorithm that performs best on all available tasks—the "no free lunch" theorem (Wolpert & Macready, 1997). Our approach addresses this by mapping our proposed fingerprint $f_D^d(.)$ to any multi-objective performance measures defined by the multi-target regression problem. It does so by decomposing the performance and its uncertainty, as well as the algorithm $h(.)$, learning a regressor $r(.)$ separately as shown in Figure 1 on page 1.

Motivated by the central limit theorem (Rosenblatt, 1956) and the asymptotic characteristics of k-folds (Li, 2023), we derive the estimation of the performance and its uncertainty for our approach. We learn a regressor $r(.)$ such that $f_D^d(.) \xrightarrow{r(.)} \widehat{\mu(\mathbb{E}_h^d)}$, estimating the mean classification performance of algorithm $h(.)$ on a dataset d as well as the observed standard deviation in performance $f_D^d(.) \xrightarrow{r(.)} \widehat{\sigma(\mathbb{E}_h^d)}$. Note that our approach and code allow us to estimate various characteristics of an algorithm's performance, e.g., lower percentiles of the k-folds for estimating lower bounds in a risk-averse setting like earthquake prediction (Arul & Kareem, 2021). Details on the regressors $r(.)$ applied to the multi-target regression AS problem can be found in Section 4.4.

4. Experiment & results

Our approach estimates the performance of an algorithm $h(.)$ on a dataset d, described by $\mathbb{E}_h^d$, solely through the computation of select characteristics that describe the dataset. To train and test this mapping, we need various datasets and the related performance of multiple classification algorithms. We evaluate our approach on the 112 univariate time series datasets established in the UCR classification benchmark (Bagnall et al., 2017; Dau et al., 2019). The performances are established by Middlehurst et al., 2023 in their back-off paper and available as part of the time series machine learning package (Middlehurst et al., 2023). We run our evaluation on all 35 algorithms $h(.)$ referenced in this most recent benchmark, such as BOSS (Schäfer, 2015), HC2 (Middlehurst et al., 2021), InceptionT (Ismail Fawaz et al., 2020), ROCKET (Dempster et al., 2020), among others [1]. For each dataset $d \in D$ we calculate the instance fingerprint $f_I(.)$, the class fingerprint $f_C(.)$, and finally accumulate the data fingerprint $f_D^d(.)$. Our approach estimates the classification performance $\widehat{\mu(\mathbb{E}_h^d)}$ and uncertainty $\widehat{\sigma(\mathbb{E}_h^d)}$ of algorithms $h(.)$ based on this final fingerprint.

We split the 112 datasets of the UCR benchmark $d \in [1, ..., D]$ by a $.2/.2/.6$ train-validation-test split. For each of the individual datasets in D_{train}, $D_{validation}$ and D_{test}, we calculate their fingerprint $f_D^d(.)$ and pair them with the achieved performance of each classification algorithm h.

The performance regressors $r(.)$ are trained on the fingerprint and classification performances of $h(.)$ on all datasets in D_{train}. The regressors $r(.)$ are selected based on their accuracy in performance estimation of classification algorithms $h(.)$ on $D_{validation}$. We evaluate our approach by running the regressor $r(.)$ on the fingerprints of D_{test} and compare the estimated performances and uncertainty to the benchmark results. The code of this work is publicly available[2].

4.1. Naive baseline

We derive a naive baseline $\ddot{\mu}_h^d$ for the mean performance $\mathbb{E}_h^d$ of an algorithm h on a dataset d building upon the common concept of a single best solver (Bischl et al., 2016). We define $\ddot{\mu}_h^d = \frac{1}{|D_{train}|} \sum_{d \in D_{train}} \mu(\mathbb{E}_h^d)$. It reflects the average performance of algorithm h on D_{train}, the datasets used for training our performance estimator. Correspondingly a naive baseline $\ddot{\sigma}_h^d$ for the expected uncertainty in performance can be calculated by $\ddot{\sigma}_h^d = \frac{1}{|D_{train}|} \sum_{d \in D_{train}} \sigma(\mathbb{E}_h^d)$. Our approach estimates

[1] 1NN-DTW, BOSS (Schäfer, 2015), Catch22 (Lubba et al., 2019), cBOSS (Middlehurst et al., 2019), CIF (Middlehurst, Large, & Bagnall, 2020), CNN (Ismail Fawaz et al., 2019), EE (Lines & Bagnall, 2015), FreshPRINCE (Middlehurst & Bagnall, 2022), HC1 (Bagnall et al., 2020), Arsenal, DrCIF and HC2 (Middlehurst et al., 2021), Hydra and Hydra-MR (Dempster et al., 2023), InceptionT (Ismail Fawaz et al., 2020), Mini-R (Dempster et al., 2021), MrSQM (Nguyen & Ifrim, 2022), Multi-R (Tan et al., 2022), PF (Lucas et al., 2019), RDST (Guillaume et al., 2022), ResNet (Z. Wang et al., 2017), RISE (Flynn et al., 2019), RIST, ROCKET (Dempster et al., 2020), RSF (Karlsson et al., 2016), RSTSF (Cabello et al., 2021), ShapeDTW (Zhao & Itti, 2018), Signatures, STC (Hills et al., 2014), STSF (Cabello et al., 2020), The Temporal Dictionary Ensemble (TDE) (Middlehurst, Large, Cawley, & Bagnall, 2020), TS-CHIEF (Shifaz et al., 2020), TSF (Deng et al., 2013), TSFresh (Christ et al., 2018), WEASEL (Schäfer & Leser, 2017), WEASEL-D.

[2] https://github.com/LarsBoecking/time_series_fingerprint

$\widehat{\mu(\mathbb{E}_h^d)}$ and $\widehat{\sigma(\mathbb{E}_h^d)}$ for any $d \in D_{test}$ is benchmarked against this baseline.

4.2. Fingerprints

Instance-level fingerprints: The instance-level fingerprint $f_I(.)$ describes instances of any length by a fixed-size vector. Accurately differentiating each individual instance just by its fingerprint seems challenging, as shown in Figure 2a. Still underlying patterns can be identified, e.g. instances of target class 1 in the *Yoga*-dataset have higher Skewness γ_1 while instances of target class 2 have higher mean change $\overline{\Delta x^{i,d}}$, as shown in Figure 4.

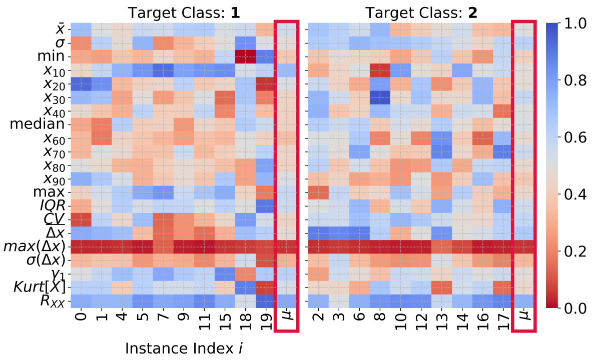


Figure 4: Instance fingerprint $f_I(x^{i,d})$ for ten sampled instances of each class in the *Yoga*-dataset and the class aggregation $f_C(.)$ via μ aggregation. As indicated by their different values, Skewness γ_1 and Kurtosis $Kurt[X]$ are promising characteristics to differentiate individual instances as well as aggregated class fingerprints.

Class-level fingerprint: Our approach aggregates instance-level fingerprint $f_I(.)$ across all instances I^c for each target class c. Figure 4 provides a fingerprint for the first ten individual instances of each target class in the *Yoga*-dataset, as well as their class-level aggregation $f_C(.)$. Note that this visualization highlights the concept for a reduced number of instances (ten in this case). The actual class-level fingerprint $f_C(.)$ is aggregated on all instances I^c in the training subset of the given dataset d. Still, only these ten instances result in a class fingerprint with certain distinguishing characteristics, e.g., Kurtosis and Skewness.

Dataset-level fingerprint: Finally, to build a fixed-size fingerprint that can be utilized to describe any time series classification task, the class-level fingerprints $f_C(.)$ are aggregated on dataset granularity. For the dataset-level aggregation, standard deviation, interquartile range, and the range between the minimum and maximum value are calculated. Each of those fixed-sized fingerprints representing an individual dataset is then mapped to an algorithm performance $f_D^d(.) \to \mathbb{E}_h^d$.

4.3. Performance estimation for a given algorithm

We evaluate various regression models on the decomposed multi-target of estimating an algorithm's (h) mean performance on a dataset d, described by $\mu(\mathbb{E}_h^d)$ and the uncertainty across the k-folds, described by the standard deviation $\sigma(\mathbb{E}_h^d)$. The mean performance $\widehat{\mu(\mathbb{E}_h^d)}$ is shown in Figure 5 and the estimated uncertainty $\widehat{\sigma(\mathbb{E}_h^d)}$ is shown in Figure 6 for h 1NN-DTW (exemplarily). Algorithms ridge and random forest are selected based on their performance on D_{val} shown in the upper half and evaluated on D_{test} shown in the lower half. Reporting average relative improvement to account for the different baseline levels.

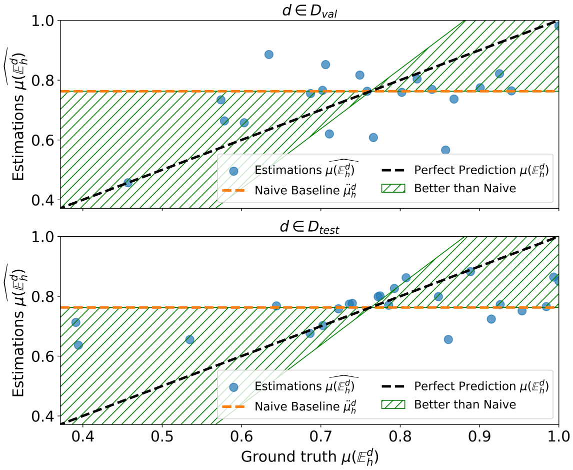


Figure 5: Ridge regression estimations $\widehat{\mu(\mathbb{E}_h^d)}$ for h 1NN-DTW, achieving an average improvement in MAE of 18.13% on D_{val} and 18.61% on D_{test} compared to $\ddot{\mu}_h^d$.

The hatched area indicates predictions with a relative improvement, while un-hatched areas cover points where the performance estimation is further off than the naive baseline. Our performance estimation achieves a relative improvement, if $|\widehat{\mu(\mathbb{E}_h^d)} - \mu(\mathbb{E}_h^d)| < |\ddot{\mu}_h^d - \mu(\mathbb{E}_h^d)|$. This dynamic applies both to comparing the ground truth mean performance to the estimated mean performance $\widehat{\mu(\mathbb{E}_h^d)}$ as well as the ground truth standard deviation across the k-folds compared to the estimated uncertainty $\widehat{\sigma(\mathbb{E}_h^d)}$. Algorithms ridge and random forest are selected based on their performance on D_{val} shown in the upper half and evaluated on D_{test} shown in the lower half.

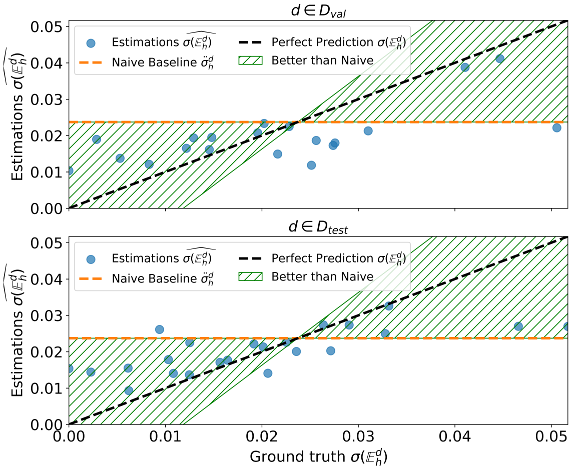


Figure 6: Random forest regression estimations $\widehat{\sigma(\mathbb{E}_h^d)}$ for h 1NN-DTW, achieving an average improvement in MAE of 37.44% on D_{val} and 37.31% on D_{test} compared to $\ddot{\sigma}_h^d$.

4.4. Estimation improvements benchmark

In Table 1, we report the absolute error level on the test set as the mean absolute error (MAE) and the relative improvement compared to the naive baseline. For each algorithm (row) we report: **(1)** Which model is selected based on its MAE performance on the validation set. **(2)** The performance of naive baselines $\ddot{\mu}_h^d$ and $\ddot{\sigma}_h^d$. **(3)** MAE in predicting the mean performance $\widehat{\mu(\mathbb{E}_h^d)}$ and the std. across the k-folds $\widehat{\sigma(\mathbb{E}_h^d)}$. **(4)** The relative change of MAE in %.

For example, when a ridge regressor estimates the performance of the *1NN-DTW* algorithm, as shown in Table 1, we observe a significant improvement: Just by analyzing the fingerprint $f_D^d(.)$ of the unseen test datasets, our approach is 18.61% more accurate in estimating the mean ground truth performance on these datasets $\mathbb{E}_h^d$, compared to the naive baseline measured by the MAE. Estimating the uncertainty of the *1NN-DTW*-algorithm $\widehat{\sigma(\mathbb{E}_h^d)}$ our approach improves the naive baseline by 37.31% in MAE. A visual interpretation of these results as well as a comparison of the performance on the validation datasets D_{val} and the test datasets D_{test} is given in fig. 7. In summary, based on the validation set performance for each algorithm $h(.)$, our approach outperforms the naive baseline by an average of -7.32% for the MAE when predicting the mean performance and by -15.81% in the MAE of the std. deviation. Our approach is capable of differentiating performances on individual datasets. Instead of selecting an algorithm based on its average performance on some publicly available benchmark, our approach allows for precise estimation of the exact performance each algorithm will achieve on a specific dataset. This enables a more informed and tailored algorithm selection process, ensuring that the chosen algorithm is the most suitable for the unique characteristics of the new dataset.

	Mean μ				**Std.** σ			
		MAE				**MAE**		
$h(.)$	$r(.)$	$\ddot{\mu}_h^d$	$\widehat{\mu(\mathbb{E}_h^d)}$	$\Delta\%$	$r(.)$	$\ddot{\sigma}_h^d$	$\widehat{\sigma(\mathbb{E}_h^d)}$	$\Delta\%$
1NN-DTW	Ri	0.1277	0.1039	**-18.61**	RF	0.011	0.0069	**-37.31**
Arsenal	RF	0.1046	0.1039	**-0.72**	GB	0.0094	0.0105	12.23
BOSS	Ri	0.1242	0.1099	**-11.55**	GB	0.0134	0.0104	**-22.68**
CIF	RF	0.1072	0.1152	7.44	GB	0.0125	0.0085	**-32.07**
CNN	Ri	0.1786	0.144	**-19.37**	AB	0.0184	0.0149	**-19.02**
Catch22	Ri	0.1205	0.098	**-18.67**	GB	0.0122	0.0083	**-31.82**
DrCIF	GB	0.1054	0.1	**-5.09**	RF	0.011	0.0086	**-21.69**
EE	Ri	0.1159	0.1013	**-12.64**	RF	0.0127	0.0103	**-18.57**
FreshPRINCE	Ri	0.1157	0.095	**-17.92**	GB	0.0122	0.0089	**-27.19**
HC1	AB	0.1055	0.1017	**-3.60**	RF	0.0109	0.0103	**-5.58**
HC2	RF	0.0957	0.0905	**-5.45**	RF	0.0104	0.0102	**-1.82**
Hydra-MR	RF	0.0954	0.0925	**-2.95**	RF	0.0101	0.0106	4.70
Hydra	KN	0.1058	0.0966	**-8.70**	RF	0.0104	0.01	**-4.08**
InceptionT	RF	0.0943	0.0901	**-4.52**	RF	0.0108	0.0084	**-22.77**
Mini-R	RF	0.1021	0.1029	0.77	RF	0.0098	0.0091	**-7.51**
MrSQM	RF	0.1116	0.1112	**-0.37**	RF	0.0139	0.0132	**-4.80**
Multi-R	Ri	0.0973	0.0878	**-9.76**	GB	0.0104	0.0104	0.12
PF	Ri	0.1134	0.0911	**-19.64**	RF	0.0123	0.0104	**-16.00**
RDST	RF	0.1001	0.1037	3.63	AB	0.0099	0.0105	6.65
RISE	Ri	0.1429	0.1241	**-13.14**	GB	0.011	0.0078	**-28.85**
ROCKET	KN	0.1019	0.094	**-7.80**	RF	0.0093	0.0093	**-0.38**
RSF	KN	0.1327	0.1227	**-7.56**	RF	0.0127	0.0067	**-46.95**
RSTSF	AB	0.1008	0.0943	**-6.47**	GB	0.0113	0.011	**-2.71**
ResNet	AB	0.1214	0.1096	**-9.71**	KN	0.0167	0.0111	**-33.25**
STC	Ri	0.1136	0.1051	**-7.42**	AB	0.0132	0.0143	8.47
STSF	RF	0.1143	0.1132	**-0.89**	GB	0.0118	0.0088	**-25.07**
ShapeDTW	Ri	0.1562	0.1252	**-19.86**	GB	0.0106	0.0066	**-37.79**
Signatures	GB	0.1342	0.107	**-20.28**	RF	0.0121	0.0084	**-30.12**
TDE	KN	0.1086	0.1095	0.88	Ri	0.0139	0.0111	**-19.73**
TS-CHIEF	AB	0.1009	0.1113	10.20	GB	0.0111	0.0125	12.84
TSF	Ri	0.1324	0.113	**-14.59**	RF	0.0137	0.0078	**-43.37**
TSFresh	KN	0.1501	0.1349	**-10.12**	GB	0.0405	0.0338	**-16.49**
WEASEL-D	RF	0.109	0.1052	**-3.44**	RF	0.0091	0.009	**-1.73**
WEASEL	RF	0.1199	0.1141	**-4.86**	RF	0.0126	0.0101	**-19.91**
cBOSS	AB	0.1248	0.1328	6.42	Ri	0.0134	0.0108	**-18.99**
Mean	-	0.1167	0.1073	**-7.32**	-	0.0127	0.0106	**-15.81**

Table 1: Applying AdaBoost (AB), GradientBoosting (GB), KNeighbors (KN), RandomForest (RF), Ridge (RID) as regressors $r(.)$ in our multi-target regression AS problem for all 35 algorithms $h(.)$ referenced in the benchmark (Middlehurst et al., 2023). Error measured by MAE and relative performance improvements (lower better $\Downarrow$) for both mean performance and uncertainty estimation. Improvements highlighted **bold**.

5. Limitations & Future Work

While our current approach demonstrates significant advancements, it also has certain limitations that opens various directions for future work. The decomposition of the multi-target regression overlooks the intricate dependencies between algorithms and the collective objectives (Lorena et al., 2008). In future work, this can be investigated by a regressor that predicts the performance of multiple algorithms at once. The selection requires domain experts to balance different objectives, such as mean performance and uncertainty, which can complicate the AS process. Further development into an AI service could incorporate

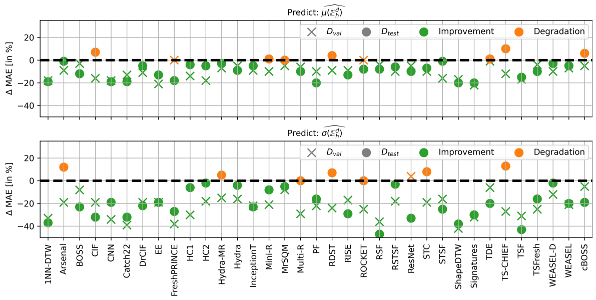


Figure 7: Performance improvement for all 35 algorithm $h(.)$ in predicting the mean performance $\widehat{\mu(\mathbb{E}_h^d)}$ and uncertainty $\widehat{\sigma(\mathbb{E}_h^d)}$ on validation and test set. Reporting the relative change compared to the naive baseline $\ddot{\mu}_h^d$ and $\ddot{\sigma}_h^d$ (lower better). Algorithms on the x-axis are sorted by name.

relaxations of multiple objectives and include domain experts in a human-in-the-loop manner. Our approach tests a range of regressors, including those with transparent internal mechanics, but does not inherently prioritize regressors based on their interpretability. This may restrict its usefulness when understanding a model's internal decision-making process, which is often crucial in real-world applications. The predictions and properties, such as the interpretability, could be combined in a decision rule for the final AS that meets the user's preferences. The effectiveness of our estimation strategy depends on the chosen metric, with MAE showing different levels of robustness and volatility in performance improvements across validation and test sets (as documented in Table 1). To improve the robustness of our approach, we encourage researchers to share their data fingerprint and the corresponding performances (Koester et al., 2020). For the next steps, our performance estimations can guide service providers in the domain, so instead of auto-correcting human decisions, we can provide feedback on which algorithm would be more suited (Balla et al., 2023). Such an extension follows the trajectory of leveraging technology to advance service, as suggested by Ostrom et al., 2010. Further developments of our approach can follow up on the ongoing discussion about which additional objectives to assess (e.g., expected running time (Bossek & Trautmann, 2019)). Our adaptable and extensible approach allows us to estimate such objectives instead or aside from the performance and uncertainty.

6. Conclusion

This paper introduces a novel data fingerprint for time-series classification, offering an approach to support more effective and privacy-preserving AI development without having access to all data points. We predict algorithmic performance and associated uncertainties by strategically decomposing the multi-target regression problem.

Our assessment across 112 datasets of the University of California riverside benchmark showcases its capability in accurately forecasting the outcomes of 35 state-of-the-art algorithms, surpassing a naive baseline by an average of 7.32% in estimating mean performance and 15.81% in quantifying uncertainty. Our approach will assist researchers and professionals in the field of algorithm selection in time series classification to set up successful AI services. We encourage other researchers and practitioners to use and extend the approach with the proposed fingerprints for further objectives. A promising field of research lies ahead.

References

Amini, A., Schwarting, W., Soleimany, A., & Rus, D. (2020). Deep evidential regression. *Advances in Neural Information Processing Systems*.

Arul, M., & Kareem, A. (2021). Applications of shapelet transform to time series classification of earthquake, wind and wave data. *Engineering Structures*.

Bagnall, A., Flynn, M., Large, J., Lines, J., & Middlehurst, M. (2020). On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (hive-cote v1. 0). *ECML PKDD*.

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *DMKD*.

Balla, N., Setzer, T., & Schulz, F. (2023). Feeding-back error patterns to stimulate self-reflection versus automated debiasing of judgments. *HICSS*.

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., Deng, D., & Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *WIREs Data Mining and Knowledge Discovery*.

Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, M., Malitsky, Y., Fréchette, A., Hoos, H., Hutter, F., Leyton-Brown, K., Tierney, K., et al. (2016). Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*.

Bossek, J., & Trautmann, H. (2019). Multi-objective performance measurement: Alternatives to par10 and expected running time. *Learning and Intelligent Optimization*.

Cabello, N., Naghizade, E., Qi, J., & Kulik, L. (2020). Fast and accurate time series classification through supervised interval search. *ICDM*.

Cabello, N., Naghizade, E., Qi, J., & Kulik, L. (2021). Fast, accurate and interpretable time series classification through randomization. *arXiv:2105.14876*.

Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*.

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., & Keogh, E. (2019). The ucr time series archive. *Journal of Automatica Sinica*.

Degroote, H. (2017). Online algorithm selection. *IJCAI*.

Dempster, A., Petitjean, F., & Webb, G. I. (2020). Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels. *DMKD*.

Dempster, A., Schmidt, D. F., & Webb, G. I. (2021). Minirocket: A very fast (almost) deterministic transform for time series classification. *ACM SIGKDD*.

Dempster, A., Schmidt, D. F., & Webb, G. I. (2023). Hydra: Competing convolutional kernels for fast and accurate time series classification. *DMKD*.

Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*.

Elsken, T., Metzen, J. H., & Hutter, F. (2018). Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint 1804.09081*.

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*.

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., & Hutter, F. (2022). Auto-sklearn 2.0: Hands-free automl via meta-learning. *The Journal of Machine Learning Research*.

Flynn, M., Large, J., & Bagnall, T. (2019). The contract random interval spectral ensemble (c-rise): The effect of contracting a classifier on accuracy. *HAIS*.

Guillaume, A., Vrain, C., & Elloumi, W. (2022). Random dilated shapelet transform: A new approach for time series shapelets. *ICPRAI*.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction*.

Hills, J., Lines, J., Baranauskas, E., Mapp, J., & Bagnall, A. (2014). Classification of time series by shapelet transformation. *DMKD*.

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated machine learning: Methods, systems, challenges*. Springer Nature.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: A review. *DMKD*.

Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P.-A., & Petitjean, F. (2020). Inceptiontime: Finding alexnet for time series classification. *DMKD*.

Jussen, P., KuÌĶhl, N., & Maleshkova, M. (2020). *Smart service management: Design guidelines and best practices*.

Karlsson, I., Papapetrou, P., & Boström, H. (2016). Generalized random shapelet forests. *DMKD*.

Koester, A., Baumann, A., Krasnova, H., Avital, M., Lyytinen, K., & Rossi, M. (2020). Panel 1: To share or not to share: Should is researchers share or hoard their precious data?

Li, J. (2023). Asymptotics of k-fold cross validation. *Journal of Artificial Intelligence Research*.

Lindauer, M., Hoos, H. H., Hutter, F., & Schaub, T. (2015). Autofolio: Algorithm configuration for algorithm selection. *AAAI*.

Lines, J., & Bagnall, A. (2015). Time series classification with ensembles of elastic distance measures. *DMKD*.

Lorena, A. C., De Carvalho, A. C., & Gama, J. M. (2008). A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review*.

Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S., & Zhang, G. (2015). Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems*.

Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., & Jones, N. S. (2019). Catch22: Canonical time-series characteristics: Selected through highly comparative time-series analysis. *DMKD*.

Lucas, B., Shifaz, A., Pelletier, C., O'Neill, L., Zaidi, N., Goethals, B., Petitjean, F., & Webb, G. I. (2019). Proximity forest: An effective and scalable distance-based classifier for time series. *DMKD*.

Middlehurst, M., Large, J., Cawley, G., & Bagnall, A. (2020). The temporal dictionary ensemble

(TDE) classifier for time series classification. *ECML PKDD*.

Middlehurst, M., & Bagnall, A. (2022). The freshprince: A simple transformation based pipeline time series classifier. *ICPRAI*.

Middlehurst, M., Large, J., & Bagnall, A. (2020). The canonical interval forest (cif) classifier for time series classification. *international conference on big data*.

Middlehurst, M., Large, J., Flynn, M., Lines, J., Bostrom, A., & Bagnall, A. (2021). Hive-cote 2.0: A new meta ensemble for time series classification. *Machine Learning*.

Middlehurst, M., Schäfer, P., & Bagnall, A. (2023). Bake off redux: A review and experimental evaluation of recent time series classification algorithms. *preprint arXiv:2304.13029*.

Middlehurst, M., Vickers, W., & Bagnall, A. (2019). Scalable dictionary classifiers for time series classification. *IDEAL*.

MontazeriShatoori, M., Davidson, L., Kaur, G., & Lashkari, A. H. (2020). Detection of doh tunnels using time-series classification of encrypted traffic. *Conf. on Dependable, Autonomic and Secure Computing*.

Mu, T., Wang, H., Zheng, S., Liang, Z., Wang, C., Shao, X., & Liang, Z. (2023). Tsc-automl: Meta-learning for automatic time series classification algorithm selection. *ICDE*.

Nguyen, T. L., & Ifrim, G. (2022). Fast time series classification with random symbolic subsequences. *International Workshop on Advanced Analytics and Learning on Temporal Data*.

Oreski, D., & Redep, N. B. (2018). Data-driven decision-making in classification algorithm selection. *Journal of Decision systems*, *27*.

Ostrom, A. L., Bitner, M. J., Brown, S. W., Burkhard, K. A., Goul, M., Smith-Daniels, V., Demirkan, H., & Rabinovich, E. (2010). Moving forward and making a difference: Research priorities for the science of service. *Journal of service research*.

Parmentier, L., Nicol, O., Jourdan, L., & Kessaci, M.-E. (2021). Autotsc: Optimization algorithm to automatically solve the time series classification problem. *ICTAI*.

Rice, J. R. (1976). The algorithm selection problem. In M. Rubinoff & M. C. Yovits (Eds.).

Rosenblatt, M. (1956). A central limit theorem and a strong mixing condition. *Proceedings of the national Academy of Sciences*.

Rudolph, L., Schoch, J., & Fromm, H. (2020). Towards predictive part quality and predictive maintenance in industrial machining - a data-driven approach. *HICSS*.

Schäfer, P., & Leser, U. (2017). Fast and accurate time series classification with WEASEL. *Information and Knowledge Management*.

Schäfer, P. (2015). The boss is concerned with time series classification in the presence of noise. *DMKD*.

Schede, E., Brandt, J., Tornede, A., Wever, M., Bengs, V., Hüllermeier, E., & Tierney, K. (2022). A survey of methods for automated algorithm configuration. *Journal of Artificial Intelligence Research*.

Shifaz, A., Pelletier, C., Petitjean, F., & Webb, G. (2020). Ts-chief: A scalable and accurate forest algorithm for time series classification. *DMKD*.

Tan, C. W., Dempster, A., Bergmeir, C., & Webb, G. I. (2022). Multirocket: Multiple pooling operators and transformations for fast and effective time series classification. *DMKD*.

Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. *ACM SIGKDD*.

Wang, F., Zhang, L. Y., Pan, L., Hu, S., & Doss, R. (2022). Towards privacy-preserving neural architecture search. *ISCC*.

Wang, W. K., Chen, I., Hershkovich, L., Yang, J., Shetty, A., Singh, G., Jiang, Y., Kotla, A., Shang, J. Z., Yerrabelli, R., et al. (2022). A systematic review of time series classification techniques used in biomedical applications. *Sensors*.

Wang, Z., Yan, W., & Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. *IJCNN*.

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. *Transactions on Evolutionary Computation*.

Yan, C., Zhang, Y., Zhang, Q., Yang, Y., Jiang, X., Yang, Y., & Wang, B. (2022). Privacy-preserving online automl for domain-specific face detection. *CVPR*.

Zhang, C., Yuan, X., Zhang, Q., Zhu, G., Cheng, L., & Zhang, N. (2021). Privacy-preserving neural architecture search across federated iot devices. *TrustCom*.

Zhao, J., & Itti, L. (2018). Shapedtw: Shape dynamic time warping. *Pattern Recognition*.