An Empirical Evaluation of Early Time-Series Classification Algorithms

Evgenios Kladis  
NCSR "Demokritos"  
Athens, Greece  
eukladis@iit.demokritos.gr

Charilaos Akasiadis  
NCSR "Demokritos"  
Athens, Greece  
cakasiadis@iit.demokritos.gr

Evangelos Michelioudakis  
NCSR "Demokritos"  
Athens, Greece  
vagmcs@iit.demokritos.gr

Elias Alevizos  
NCSR "Demokritos"  
Athens, Greece  
alevizos.elias@iit.demokritos.gr

Alexandros Artikis  
NCSR "Demokritos"  
Athens, Greece  
a.artikis@iit.demokritos.gr

ABSTRACT

Early Time-Series Classification (ETSC) is the task of discerning the class of time-series observations, as accurately and fast as possible. Such approaches can be incorporated in forecasting, and this way assist on many research fields. However, available approaches are not suitable for all problems, since the shape and the nature of data can impact their performance. In the context of this work, we empirically evaluate five state-of-the-art ETSC algorithms on publicly available data, as well as on two newly introduced datasets, originating from the biological and maritime application areas. The first dataset refers to cancer simulation data, while the second consists of vessel geospatial information. The aim is to extensively evaluate ETSC algorithms, and provide intuition on how such approaches work, and what are the problem characteristics that may render each method successful. Also, the framework we used for the evaluation can serve as a benchmark for new related approaches.

1 INTRODUCTION

The evolution of computer systems and the proliferation of Internet of Things has significantly aided to the expansion of time-series data production, collection, and storage [35]. For instance, in the life sciences field, simulation frameworks aim to estimate how cellular structures respond to treatments, e.g. in the face of new experimental pharmaceuticals [9]. Such simulations require vast amounts of computational resources, as well as time, and produce gigabytes of data in each run. Thus, non-interesting cases should be detected at early stages and stopped as soon as possible, since they are not expected to provide any useful result.

However, many problems remain regarding the detection of situations and ailments ahead of time. Many effective methods for tackling such problems stem from the Early Time-Series Classification (ETSC) domain. Contrary to standard time-series classification, which requires all the time-points to make a decision, ETSC approaches aim to classify time-series as early in time as possible, always pursuing the smallest impact in the accuracy of the predictions. By providing results ahead of time, early classification can give e.g. critical data about a patient’s future health complications in the long-term within the predictive medicine field [20], or in the short-term, such as arrhythmia detection [16] or stroke prediction [5].

© 2021 Copyright for this paper by its author(s). Published in the Workshop Proceedings of the EDBT/ICDT 2021 Joint Conference (March 23-26, 2021, Nicosia, Cyprus) on CEUR-WS.org. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)
contemporary data analysis processes, focus has been put on multivariate time-series [8]. In a recent work presented in [7], authors examine the state-of-the-art methods for solely multivariate time-series classification and compare inter-dimensional dependencies among classifiers, providing this way further insight to this field of research.

A recent work was also conducted in ETSC, reviewing existing approaches related to the field [10]. The algorithms are categorized into four groups and the analysis is performed on a theoretical level, highlighting their strengths, limitations, and major concerns for each group. Our approach differs, since we follow an empirical approach. We categorize ETSC algorithms according to their internal structure and subcomponents and evaluate them on multiple datasets. Here, we distinguish three main types, i.e. those that rely on subsequence extraction and analysis, algorithms that are based on clustering of the time-series, and others that employ ANNs. Note that some algorithms could be considered to belong to more than one category. In this case, the type was chosen based on which component was the most significant for the classification task.

2.1 Subsequence-based Approaches

The algorithms that use subsequences, in principle, isolate smaller windows of the time-series—termed as subseries—and manipulate them accordingly, to maximize information gain and achieve the earliest and most accurate results. We distinguish five cases that belong to this category. An example of such an algorithm is Early Distinctive Shapelet Classification (EDSC) [34] where the classification task is based on subseries extraction. Initially, the user provides as input the minimum and maximum lengths of the windows that are to be extracted. Then, during the extraction step, all the corresponding sub-series are isolated from the full time-series and, by using Chebyshev Inequality on each extracted part, a threshold is calculated. This threshold controls the minimum similarity that a sub-series should have, so to be assigned in the same class. Next, each sub-series is transformed into a triplet containing the class of the time-series it was extracted from, the sub-series itself, and the threshold, forming this way a shapelet.

Having all the shapelets in hand, a utility function calculates a measure similar to the $F_1$-score for each one, which represents some sense the distinctive capability of the shapelet, i.e., how appropriate a shapelet is to be considered as a template for a particular class. Ranked by their utility values, the best shapelets are selected and stored into a “pool” that constitutes the classification basis. To ease the reader we incorporate a running example for the biological case. Suppose we have the time-series $t = [1137, 1227, 1205, 1082, 893, 736, 664, 639, 631]$ which is part of a time-series from our biological dataset. The user gives as input minimum length 2 and maximum, 4. Firstly, all the sub-series between sizes 2 and 4 are going to be extracted from the time-series. Let one sub-series be $t_{prime} = [1137, 1227, 1205]$. For $t_{prime}$, a threshold $\delta = 3.05$ is calculated based on Chebyshev’s inequality, which ensures the similarity of $t_{prime}$ only with time-series of distance less or equal than the value of $\delta$. After this step, $s_{prime} = \{class, t_{prime}, \delta\}$ is created, and then, the weighted $F_1$-Score of each shapelet is calculated and stored in a list. Assuming that the list is $[1.3, 3.67, 0.83]$ and 3.67 is the score of $t_{prime}$, the top K shapelets from the list are selected as the classifier shapelets. When the distance of the test time-series from a shapelet is less than $\delta$, then the shapelet’s class is assigned.

A similar approach to the EDSC is followed by the Mining Core Feature for Early Classification (MCFEC) [11] algorithm. The difference in this case lies in the criteria for shapelet selection, and also in that MCFEC has the capability to process multivariate time-series. Like EDSC, for each variable, all the sub-series of minimum and maximum lengths are extracted, but are additionally grouped using Silhouette Clustering [26]. Then, the best shapelets from each cluster are selected for classification based on an extension of the $F_1$-score metric.

Effective Confidence-based Early Classification (ECEC) [19] utilizes subsequence algorithm called WEASEL [29] that extracts windows (e.g. $(1137, 1227, 1205)$), transforms them to symbols that form words (e.g. $(abc, cba \ldots)$), and measures the frequency of their appearance in each time-series. In more detail, the user provides as input $(i)$ the desired window length and $(ii)$ the length of the words. Then, the frequency of each word (e.g. $(1, 2, 1, 0, 0)$ for a time-series and all window lengths), is passed on to a logistic regression classifier, which in turn produces probabilistic predictions. ECEC truncates the input into $t$ different prefix sizes, beginning from the $1/t$ of the time-series up to its full length. For each prefix, it trains a WEASEL classifier and performs a $5$-fold cross validation on the dataset as an evaluation. The predictions acquired are then passed into a cost function, which, based on the probability of the prediction being correct, calculates a threshold $\theta$. During the testing phase, using the classifier of the minimum prefix size, a prediction is made for each prefix, and the corresponding cost is calculated. If the cost is higher than the threshold $\theta$ then the prediction is produced, otherwise the algorithm waits until enough data have been accumulated to form the next prefix.

The WEASEL algorithm for transforming the input data combined with logistic regression is utilized by the Two-tier Early and Accurate Series Classifier (TEASER) method [30]. This algorithm truncates the time-series into $S$ prefixes, each containing length of time-series more time points than the previous prefix. $S$

For each prefix, TEASER applies the WEASEL - logistic regression pipeline to obtain respective probabilistic values. In order to ensure the reliability of the classification result, the outputs are passed into a One-Class SVM constructed for each prefix, that accepts or rejects a prediction. The final requirement for the approval of a classification is that a prediction should be consistently the same for $v$ out of $S$ prefixes, with $v$ being an algorithm’s hyper-parameter subject to optimization.

Early Classification framework for time series based on class Discriminativeness and Reliability (ECDIRE) [21] also uses part of the time-series to produce reliable and early predictions. The algorithm requires as input the set $E$ which contains time-series lengths at which there is high information value, in order to reduce granularity of data. Consequently, $10$ times $5$-fold cross-validation is done for each length using Gaussian Process Classifiers [24], storing the predictions and probabilities for each class. Based on the predictions, an earliness threshold is constructed, which indexes the time-point from which class instances differ from the other labeled instances. Using the probabilities from the cross-validation, a second probabilistic threshold is calculated, that ensures the reliability of the prediction. A classification result must satisfy both thresholds in order to be passed as a prediction.

A more recent approach for early time-series classification is the Distance Transformation based Early Classification (DTEC) algorithm [36]. DTEC relies on the transformation of data and the usage of probabilistic classifiers to achieve early and accurate
results. In particular, DTEC extracts subsequences of time-series and maps them to another space based on distances. The transformed data are then used to train a probabilistic classifier by forming the confidence area, which is drawn based on the probability strength of the class to be passed as a prediction, making this way the classification result more reliable.

We selected EDSC, TEASER, and ECEC for evaluation from this category. EDSC is considered a widely referenced method for ETSC, since most recently developed algorithms use it as a baseline. TEASER and ECEC are also interesting choices, since they follow different internal processes and have readily available implementations that provide good results.

2.2 Clustering-based Approaches

Clustering is the task of forming groups of data similar to each other based on some similarity measure. We refer to two distinct cases in this category. Early Classification on Time Series (ECTS) [33] is an algorithm that relies on clustering, as well as, the 1-Nearest Neighbors (1-NN) search, to perform accurate and early classification. In detail, first the 1-NN set of each time-series is calculated. Based on that set, the algorithm follows the Reverse Nearest Neighbors (R-NN) approach, which in a sense shows how many time-series consider the examined one as a nearest neighbor. Based on the consistency of the R-NN set for each different subsequence of the time-series, ECTS computes the minimum prediction length for each time-series that represents how many time-points are required for the time-series to act as a reliable classifier. First, the time-series are clustered using agglomerative hierarchical clustering [31] based on Euclidean distances. The minimum prediction length of each cluster is considered and appointed to each time-series in the cluster. During the test phase, time-series are paired with NN, using initially only the first time-point and increasing through each iteration, until the time-series they pair with has minimum prediction length less or equal to that time-point’s position.

TRIGGER [4] is another algorithm that uses clustering in combination with a cost function to estimate if a time-series can provide a trustworthy result. The time-series are grouped into clusters using \( k \)-means [31] based on Euclidean distances. For each cluster \( k \) and time-step \( t \in \{1, \ldots, T\} \), a Multilayer Perceptron [31] \( h^k_t \) is trained using time-series of length \( t \) from the cluster \( k \). When a new time-series arrives, a number of probability is calculated to appoint it to a cluster and, based on the cluster and the length of the time-series, a prediction is performed. The results of the classifier are handled with a cost function, which is calculated for every time-point between the current and the last of the time-series. If the cost function returns a value of zero the result of the classifier is deemed safe and is returned along with the size of the incomplete time-series up to the currently examined time-step. We choose to incorporate the ECTS algorithm in our evaluation since, similarly to EDSC, it is one of the oldest and most known approaches for early time-series classification.

The Multivariate LSTM-FCNs (MLSTM-FCN) [18] method utilizes neural networks and operates on multivariate time-series. This is actually an extension of the original algorithm LSTM-FCN [17], which supported only univariate datasets. Both algorithms duplicate the input and pass the data to two sub-models: The first sub-model consists of three convolutional neuron layers. The use of CNNs is widely adopted in regular time-series classification [37], since it extracts important features from sequences of data that often derive from imagery. In the described model, data that exit from each of the first two layers are first batch normalized [15] and then passed to an activation function, i.e. a Rectified Linear Unit (ReLU). In order to maximize the efficiency of the model on multivariate time-series, the activated output is also passed into a squeeze and excitation block [13]. A squeeze and excite network consists of a global pooling layer and two dense layers that give to each variable of the time-series a unique weight, so that to increase the sensitivity of predictions. The second sub-model, consists of a masking layer and the output is passed on an attention based LSTM. LSTM [12] is a Recurrent Neural Network model, popular in time-series classification, because of its ability to remember inter time-series dependencies with minimal computational cost and high accuracy for time-series of length less than a thousand time points [2]. Attention based LSTMs are variations of the normal LSTMs, with increased computational complexity, which nevertheless results to increased overall performance. The output of the two sub-models is concatenated and passed through a dense layer with as many neurons as the classes, and via a softmax activation function predictions as a probabilistic output are provided.

Another similar approach is the Multi-Domain Deep Neural Network (MDDNN) [14] algorithm, which utilizes simple neural networks based on the same principles as the previous one. However, this algorithm consists of two sub-models which are identical with respect to their structure, but differ on the input they receive. The first model takes as input the \( z \)-normalized raw time-series represented on the time domain, and the second takes as input the \( z \)-normalized fast-fourier transform of the time-series represented on the frequency domain. Both models consist of two CNN layers. After each layer the output is again batch normalized and passed onto a one-dimensional max pooling layer, activated with a ReLU layer. The output of each model is then concatenated, flattened, and passed through two dense layers. Finally a softmax layer is applied, just as in the MLSTM-FCN. We selected the MLSTM-FCN for our evaluation, since there are much more details available for this particular approach than others. Also, it is worth noting that the authors provide respective source-code and datasets to be used by the community.

3 DATASETS

In this section, we describe the datasets used in our evaluation, i.e., the publicly available UCR [5], as well as two new ones, from the domains of life-sciences and maritime. Note that, instances included from the publicly available collection are univariate, while the two new cases are both multivariate. Also, since not every selected algorithm is designed to work on multivariate time-series, we implemented a simple voting method for classification. This applies to the ECTS, EDSC, TEASER, and ECEC algorithms.
3.1 Dataset of cancer cell simulations

This dataset includes the count of tumor cells population during the administration of specific drug treatments as resulting from large-scale model exploration experiments. Each time-series represents one simulation experiment of a specific drug treatment configuration, which differs from the others based on a set of configurable parameters, such as the time of administration, the duration, and the drug concentration. A time-point of each instance corresponds to three different integer values, indicating the number of Alive, Necrotic and Apoptotic cells. The time-series are preclassified as interesting or non-interesting based on if the drug treatment was effective or not, according to a particular classification rule that was defined by domain experts. The dataset consists of 644 time-series, each having 48 time-points. The time-series instances included in the dataset were created by executing a parallel version of the PhysiBoSSv2 simulator.\(^1\)

Since the dataset originates from large-scale simulation experiments oriented to drug treatment discovery, the classes are rather imbalanced. Specifically, the configurations that produced interesting time-series constitute the 20% of the dataset, while the remaining 80% account for non-interesting cases. Moreover, many interesting and non-interesting examples tend to be very similar during the early stages of the simulation, until the drug treatment takes effect, which, as observed, is usually after the first 30% of the time-points of the time-series. This fact makes the dataset an interesting benchmark, since it is nearly impossible to obtain accurate predictions in less time than this.

3.2 Dataset of vessel position signals

The maritime dataset contains data of nine different sea vessels that cruise around the port of Brest, France. This dataset is derived from [23, 25]. Each measurement corresponds to a vector of values for longitude, latitude, speed, heading, course over ground of a vessel at a given time-point. The time-series are fragmented to a specific length, and divided into two classes, based on whether the ship entered or didn’t enter this particular port during the time course of the corresponding time-series fragment. In total, there are 5249 instances of 30 time-points each.

4 EXPERIMENTAL RESULTS

4.1 Evaluation Metrics

For our evaluation we use five well-known metrics.

- **Earliness.** Earliness counts how many time-points were required to make the final prediction for each time-series.

\[
\text{Earliness} = \frac{\text{consumed time series length}}{\text{total length of timeseries}}
\]

- **Accuracy.** Accuracy is translated to how many time-series were classified correctly during the testing process.

\[
\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{number of test instances}}
\]

- **F\(_1\)-score.** F\(_1\)-score is the harmonic mean of precision and recall:

\[
F_1\text{-score} = \frac{tp}{tp + \frac{1}{2}(fp + fn)}
\]

where \(tp, fp, \) and \(fn\) are the true positives, false positives, and false negatives respectively.

- **Harmonic Mean.** Harmonic mean is a metric proposed in [30] and it represents the Pythagorean harmonic mean between earliness and accuracy:

\[
\text{Harmonic Mean} = \frac{2 \cdot (1 - \text{earliness}) \cdot \text{accuracy}}{(1 - \text{earliness}) \cdot \text{accuracy}}
\]

Note that the algorithms were evaluated using 5-fold cross-validation on the two real-world datasets.

**Training and Testing times.** Measured in minutes and seconds respectively.

4.2 Comparison of ETSC Approaches

We present results from an evaluation of 5 state-of-the-art algorithms, i.e., EDS, ECTS, TEASER, ECEC, and MLSTM-FCN. The implementations were readily available, except for ECTS, which is a custom implementation. In particular, ECEC as well as TEASER are written in Java, while ECTS and MLSTM-FCN in Python, and, finally, EDS in C++. The algorithms were tested using 5-fold cross validation for the maritime and biological datasets. The results were run on a server with an Intel Xeon E5-2630 2.60GHz processor, with 25-cores and 252 GB RAM.

Since most algorithms don’t support multivariate time-series, a voting method was applied. According to the voting method, classifiers are trained, and tested on each of the folds, but separately for each dimension of the input vector shape. Upon collecting each of the outputs, the most frequent or voted prediction was chosen, however, assigned with the worst earliness among them. In the case of equal votes among classifiers, the earliest one made is selected. MLSTM-FCN is tested on the [40%, 50%, 60%] of the time-series length in each dataset, and the length with the best results based on harmonic mean is chosen, for each dataset individually. The number of LSTM cells is set to 8 for all experiments as default. For TEASER, S is set to 10 for the biological and maritime case whereas for the UCR dataset is fixed to 20. The code of our testing framework is available publicly, accompanied with the respective datasets and algorithm configurations.\(^2\)

It should be noted that fig. 1 presents for the biological and maritime cases using the 5-fold cross validation over the multivariate datasets. Also for the UCR datasets, each entry is a dataset and the F\(_1\)-Score bar represents the average F\(_1\)-score from all dataset’s classes. The training and testing time plots were made using log-scale on the y-axis.

EDS, as one of the older and simpler approaches in the ETSC algorithmic ‘arsenal’, has considerably high training times, an undesired characteristic that results to the algorithm being unable to provide results within several hours. In particular, for the maritime dataset, this algorithm did not progress after 48 hours of execution. This can be explained by the fact that EDS uses 1-NN as the main procedure behind the early classification. 1-NN requires to calculate Euclidean distances between all time-series, resulting to very high computation times for big datasets both in length and size. One notable example is the StarLightCurves dataset, which contained 1000 training time-series of length 1024, resulting to more than 65 hours of training and testing times. However, for the biological dataset (fig. 1a), the computation time was on average around 30 minutes per fold. It reaches at top accuracy and over 90% F\(_1\)-scores which shows that ECTS can successfully tackle the class imbalance.

EDSC, the first approach to propose the use of sub-sequences for early-time-series classification, is also struggling with time

\(^1\)https://github.com/xarakas/spheroid-tnf-v2-emews

\(^2\)https://github.com/Eukla/ETS
complexities. In fact, training and testing times are greatly influenced by the size of the dataset and, more importantly, the length of the time-series. Having lengths of more than a thousand time-points, and dataset sizes larger than a hundred time-series leads to increased computation times, e.g. the Haptics dataset, which has 155 time-series of size 1093 time-points and training times of more than 24 hours. For the available biological dataset the performance is high in terms of accuracy. However, the data usage is also very high. For the maritime (fig. 1b), the performance is suboptimal, since F1-score is relatively low and the earliness reaches to 100%. When EDSC is unable to find a prediction, returns the most frequent class label with the maximum data-usage. EDSC for both the biological and the maritime dataset was unable to make predictions and returned the default answer.

We can observe increased earliness values for ECTS and EDSC. Recall that the biological dataset consists of 3 variables, the Alive, Necrotic and Apoptotic cell counts at each time-step. While for the Alive and Necrotic variables numerical value differences exist between different class instances, for the Apoptotic, the number of cells is more consistent in both classes. This creates ambiguity for a simple classifier such as ECTS and EDSC, resulting in very delayed classification when using the Apoptotic variable as training. During a test instance, predictions with minimal data usage are produced by the classifiers trained with the Alive and Necrotic subseries but delayed classifications are made from the Apoptotic trained ones. The voting classification method chooses the most voted prediction, which is usually the one made from the Apoptotic classifier, and assigns as earliness the largest value among the three options, which in this dataset is the one from Apoptotic. For this reason earliness becomes worse.

Up next, we have the ECEC method. As already explained, the ECEC splits the dataset to prefixes and computes a confidence threshold based on the a priori probability of a prediction being correct. For the biological dataset, the ECEC has the highest accuracy with very low length usage, being the best one in harmonic mean among the evaluated algorithms. However, this could be considered as a misleading result due to the weakness of the confidence threshold as a measure of classifications reliability, and the strong predisposition of the algorithm to choose the majority class as a prediction over the minority. Because the 80% of the dataset is considered as non-interesting, even if all predictions are the same, high accuracy can be attained. Note though that the F1-Score does not suffer from this. For the Maritime dataset, where the classes are more balanced, the harmonic mean of ECEC is significantly low, with about 80% data usage. It should be noted that the F1-Score is on the higher end, at about 92% which is explained by increased data usage. Times increase significantly with the size of each dataset, since for datasets such StarLightCurves training needed more than 5 hours.

TEASER, uses WEASEL just like ECEC but is shown to perform differently. The computation times for the biological, maritime and UCR dataset are very low. Testing times are also the lowest. As one of the most promising presented algorithm in this paper, achieves steadily over 80% accuracy, with less than 50% earliness on all datasets. Also this was achieved by conducting multivariate time-series classification using the simple voting method. The 3-step prediction process containing, the transformation and logistic regression, the One-Class SVM, and the check of prediction consistency along with the hyperparameter searching step, give reliable results, very quickly, remaining unaffected from dataset sizes. A weakness of TEASER is the high data usage, in the Inilenskate dataset of UCR (fig. 1c) for example, which, nevertheless, can be tackled by increasing the available RAM. Also the F1-Score is high but still has room for improvement.

Lastly, MLSTM-FCN uses Deep Neural Networks to make predictions. Since we essentially run MLSTM-FCN 3 times for each
Weights are randomly initialized at the start of each run, and references leading to erratic results. Nevertheless, MLSTM-FCN shows potential in both unique datasets with very low computation times, good accuracy, and early predictions. The F1-Score is also high on average except for a few particular folds and datasets. Also, for the UCR datasets, MLSTM-FCN shows some low accuracies compared to other algorithms. Insufficient results derive from the lack of hyper parameter optimization that has not yet been integrated on the used MLSTM-FCN for univariate and multivariate time-series. For the LSTM layer, we used the default 8 cells. Using a grid search for cells might improve UCR results, but would drastically increase computation times.

5 SUMMARY AND FURTHER WORK

In this work, we evaluated state-of-the-art for ETSC on publicly available datasets and newly introduced ones originating from real-world applications. The results were very positive, since all the algorithms achieved over 80% accuracy, with the most recently introduced ones using less than half of the available time-points, i.e., producing accurate predictions quite early. Each algorithm performed slightly differently on each dataset, but TEASER and MLSTM-FCN provided a stable and good performance on both classification precision and earliness, in all datasets. We also provide a public repository with all of the algorithms and datasets used, therefore allowing to test and choose the most appropriate algorithm for new applications.

A possible extension of this suite is the integration of online data streams to preform time-series classification, allowing to test such algorithms in realistic settings.

ACKNOWLEDGEMENT

This work has received funding from the EU Horizon 2020 RIA program INFORE under grant agreement No 825070.

REFERENCES


