An Examination of the State-of-the-Art for Multivariate Time Series Classification

Bhaskar Dhariyal, Thach Le Nguyen, Severin Gsponer and Georgiana Ifrim

School of Computer Science, University College Dublin, Ireland
{bhaskar.dhariyal, thach.lenguyen, severin.gsponer, georgiana.ifrim}@insight-centre.org

2020 International Conference on Data Mining Workshops (ICDMW) | 978-1-7281-9012-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICDMW51313.2020.00042

Abstract—**The UEA Multivariate Time Series Classification (MTSC) archive released in 2018 provides an opportunity to evaluate many existing time series classifiers on the MTSC task. Nevertheless, although many new TSC approaches were proposed recently, a comprehensive overview and empirical evaluation of techniques for the MTSC task is currently missing from the time series literature. In this work, we investigate the state-of-the-art for multivariate time series classification using the UEA MTSC benchmark. We compare recent methods originally developed for univariate TSC, to bespoke methods developed for MTSC, ranging from the classic DTW baseline to very recent linear classifiers (e.g., MrSEQL, ROCKET) and deep learning methods (e.g., MLSTM-FCN, TapNet). We aim to understand whether there is any benefit in learning complex dependencies across different time series dimensions versus treating dimensions as independent time series, and we analyse the predictive accuracy, as well as the efficiency of these methods. In addition, we propose a simple statistics-based time series classifier as an alternative to the DTW baseline. We show that our simple classifier is as accurate as DTW, but is an order of magnitude faster. We also find that recent methods that achieve state-of-the-art accuracy for univariate TSC, such as ROCKET, also achieve high accuracy on the MTSC task, but recent deep learning MTSC methods do not perform as well as expected.**

1. Introduction

The proliferation of IoT and sensor technology has led to ease of collection of time series data, for example, a mobile phone sensor can record acceleration(A), orientation(O) and gyroscopic(G) data while a person is moving. This multi-channel data (e.g., [A, O, G] values) recorded over time is known as multivariate time series (MTS) data. Assigning a discrete label such as walking, running, jogging, to MTS data is called Multivariate Time Series Classification (MTSC). Much research in time series has focused on univariate (single channel) time series classification (UTSC). The MTSC task can be tackled by methods designed for univariate time series or by custom methods specifically designed for MTS data. The work of [1] has evaluated the adaptation of the classic 1NN-DTW UTSC method for the MTSC task.

While approaching the MTSC problems with UTSC techniques, the different dimensions of the multivariate time series (also named channels in related literature) are either converted into a single dimension by concatenation, or alternatively, each dimension is considered as a separate time series and the final prediction is computed based on the predictions for each dimension, for example by ensembling the predictions. These approaches could miss information which is only available while taking the interaction of multiple dimensions into account. Additionally, MTSC problems give rise to scalability challenges because with each additional dimension, the overall length of the time series is increased by $d * l$ where d and l stand for dimension and length respectively.

One of the major problems while comparing the various methods in the literature is that the published papers use different datasets to evaluate the proposed approaches, rendering a proper and fair comparison difficult. In this study, we attempt to fill this gap by evaluating and comparing existing methods on the UEA MTSC benchmark [1]. We first identify some of the most prominent TSC methods and evaluate their classification performance. In particular, we examine the state-of-the-art algorithms, DTW [1], ROCKET [2], MrSEQL [3] as well as two recent deep learning approaches, Tapnet [4] and MLSTM-FCN [5]. Furthermore, we introduce new simple baseline methods that rely on statistical features extracted over the time series. Second, we discuss advantages and drawbacks of the methods evaluated and give insights into which algorithm to use based on desired properties. For example, besides predictive accuracy, some of the compared methods only work on fixed-length time series, some have challenges with regard to memory consumption or runtime, and some of the methods are non-deterministic and are very complex, which makes it difficult to reproduce results or to provide a prediction explanation to an end user.

Our overarching goal is to first analyse the current state-of-the-art in MTSC and then investigate methods to improve their accuracy, with reduced computational complexity. Furthermore, we prefer methods that are conceptually simple to understand and to implement. The main contributions of this paper are as follows:

- We present an empirical study that compares current state-of-the-art MTSC methods on the UEA MTSC benchmark, in regards to both accuracy and computation time.

- We propose a simple baseline method for the MTSC task which is faster and more accurate than the classic DTW baselines.
- We show that simple and efficient methods that extract features independently from each time series dimension, concatenate them in a single feature space and train a linear classifier, are as accurate as recent, more complex deep learning methods. This raises interesting questions around whether there is any need to learn expensive and complex features or models that are able to capture complex dependencies across time series dimensions. From our experiments, we find that a linear combination of features extracted from each dimension independently, seems to work well in regard of accuracy and it is more efficient to train.

The rest of the paper is structured as follows. In Section 2 we introduce related work and describe the state-of-the-art methods compared in this paper. In Section 3 we introduce our new baseline methods and explain their different variants. In Section 4 we introduce the UEA MTSC benchmark used for the experiments and discuss the empirical results. We conclude in Section 5.

2. Related Work

From our analysis of the relevant literature, there are two main approaches for MTSC. The first approach turns the MTSC problem into an UTSC problem by either concatenating the dimensions or by ensembling UTSC models. As a result, any UTSC method can be used for MTSC. The second approach are methods that are specifically designed for MTSC (i.e., bespoke MTS classifiers). For this reason, in this section we separately discuss both UTSC and bespoke MTSC methods.

2.1. UTSC Methods

1NN-DTW [6], [7], a method that uses DTW as a distance measure and a one Nearest-Neighbor classifier, has been a reliable UTSC baseline and despite its simplicity is often hard to beat with regards to accuracy. However, recent work on UTSC has successfully outperformed 1NN-DTW both in terms of accuracy and efficiency. Notable methods include Shapelet Transform [8], Learning Shapelets [9] (shapelet-based classifiers), MrSEQL [3], SAX-VSM [10], BOSS [11], BOSS-VS [12], WEASEL [13] (symbol-based classifiers), EE [14], COTE [15], HIVE-COTE [16], TS-CHIEF [17] (ensemble classifiers), ResNet [18], FCN [19], (deep learning methods) and ROCKET [2] (linear classifier on top of random convolution kernels). These methods have explored a vast range of ideas to learn from time series data with great success [20]. The shapelet-based classifiers make use of shapelets, primitives of time series which are basically a set of subseries present in the data, that are used as features. The symbol-based classifiers first transform numeric time series into a symbolic representation (e.g., a numeric series to a string series), followed by applying techniques to extract discriminative patterns from the symbolic sequences. There are multiple different transformations that can be used, for example, the well known Symbolic Aggregate approXimation (SAX) [21] which relies directly on the values of the time series. Other transformations rely on signal processing techniques, e.g., the Discrete Fourier Transform in SFA (as used in BOSS, BOSS VS, WEASEL, and MrSEQL), autocorrelation, or power spectrum (used in TS-CHIEF and interval-based classifiers). Naturally, the ensemble classifiers and deep learning methods are known for their high accuracy, although their demands for computing resources are also unparalleled. Nevertheless, the introduction of WEASEL, MrSEQL, and recently, ROCKET, has demonstrated that linear classifiers can perform similarly in regard of classification accuracy, while requiring far less computational resources.

2.2. Bespoke MTSC Methods

We note that some of the MTSC methods presented here are actually UTS classifiers adapted or generalized for MTS data.

1NN-DTW In [22] the authors proposed two versions of DTW for MTS data, DTW_I and DTW_D, to study the impact of multi-dimensions in DTW. The main difference between the two versions is in the way they compute the distance between two multivariate time series. While DTW_I simply adds up the univariate DTW distances for each dimension, DTW_D computes the distance between two time steps by first summing up the distance across each dimensions. As also done in [1], we use both versions of DTW as baselines in this current study. The time and space complexity of DTW is $O(N^2)$. Given the quadratic time complexity, DTW is usually not considered a scalable technique.

MrSEQL [3] is a linear classifier that learns from symbolic features of time series. The method first transforms time series data to multiple symbolic representations of different domains (i.e., SAX [23] in the time domain and SFA [24] in the frequency domain) and different resolutions (i.e., different window sizes). The classifier then extracts discriminative subsequences from the symbolic data and uses them as features for training a classification model. As the authors opted to train with logistic regression, the output is a linear model which is basically a set of weighted symbolic subsequences. In [3], the authors showed that adding features from different representations types (e.g., SAX and SFA) successfully boosts the performance of the classifier to a level comparable to the state-of-the-art. The method was originally developed for univariate time series, but it can also work with multivariate time series by viewing each dimension as an independent representation of the time series data. Compared to UTSC ensemble classifiers, MrSEQL uses all the extracted features from the different representations to learn its final model, and hence is able to combine information from different dimensions. The original implementation for MrSEQL is done in C++, but for the experiments here we use the Python wrapper from sktime[1]. Note that MrSEQL's efficiency is bound by the time taken by the symbolic transformation. Because sktime only contains an efficient implementation for the SAX transform (which is implemented in C++ and wrapped in Python), while the SFA transform is very slow, we use MrSEQL only with the SAX transform for our experiments.

WEASEL-MUSE [25] works on the features extracted in the frequency domain using the SFA transformation to create sequences of SFA words. After this transformation is applied, the basic pipeline is to extract features from all dimensions by counting all the unigrams and bi-grams, select promising features via a feature selection method, and learn a linear classifier. Similar to Mr-SEQL, this approach takes into account the features extracted from different dimensions before learning the final model. Since there are potentially a lot of features, they perform feature selection using the Chi-squared method and then learn a Logistic Regression model.

ROCKET [2] was recently introduced as a novel approach for UTSC. Models produced by ROCKET were shown to be often highly accurate while keeping the computational burden low. The basic principle of ROCKET relies on convolutional kernels which are used in many recent neural network architectures to achieve state-of-the-art results, e.g., in image classification [26], instead of learning the weights of a few kernels through backpropagation, many (e.g., 10,000) convolutional kernels are generated at random. Thereby, not only the kernel weights, but also further kernel properties such as length, dilation, stride, bias and zero padding are generated at random. Such kernels are in principle just a simple linear transformation of the input time series producing a new time series. To extract the final features, global max pooling and the newly introduced proportion of positive values (ppv) is employed, resulting in two features for each random kernel. The resulting feature matrix can be used to train any classifier, i.e., a logistic regresison or a ridge regression classifier. The latest implementation of ROCKET, in the sktime package, can also be applied to MTSC problems. On top of the previously mentioned properties, each kernel gets assigned a random subsample of all dimensions. While applying the convolution, only these dimensions are taken into account.

Tapnet [4] is a very recent deep learning framework that deals with multivariate time series data. It proposes an attention prototype network that takes inspiration from traditional and deep learning frameworks. The authors propose Random Dimension Permutation (RDP) to reorganize the dimensions into groups. From these groups, a low dimension embedding for the sequences is learned using an LSTM and a one dimensional CNN. Using this embedding, a class prototype is learned, where a class prototype is a weighted combination of training samples belonging to the same class and the weights are learnt using an attention layer. We tried to reproduce the results of this approach and have contacted the authors with our queries, however, we were not able to obtain results for many datasets.

Multivariate LSTM-FCN [5] is an extension of LSTM-FCN and ALSTM-FCN. The proposed model consists of a convolution block and an LSTM block. In each convolution block, a convolution layer is followed by batch normalisation and the ReLU activation function. The differentiating part is a squeeze-and-excite block which adaptively calibrates the input feature maps. The squeeze operation exploits the contextual information, while the excite operation captures the inter-channel dependencies.

Besides these methods, one can approach the MTSC task by adapting UTSC ensemble methods for MTSC. The recent work in [27] presents some preliminary results showing that the UTSC method HIVE-COTE can successfully be adapted to the MTSC task. The accuracy results are similar to the methods discussed here, in particular the accuracy is similar to WEASEL-MUSE and 1NN-DTW, but the computational burden of ensemble methods is typically much higher. Hence, in this study we focus on empirically analysing the methods described above and omit ensemble methods for now.

3. Methods

The state-of-the-art MTS classifiers are usually complex and heavily engineered. Additionally, most of them suffer from scalability issues. In this part, we introduce some simpler techniques relying on basic statistics that can be directly extracted from the time series. We build our various methods with increasing complexity in order to understand the effect of data pre-processing steps on the accuracy and efficiency of these methods. Later we compare these approaches to the DTW baselines and to state-of-the-art MTSC methods.

3.1. Proposed Baselines

The fundamental principle of all the methods we propose here is to extract features from each dimension of the time series independently. The resulting features are then concatenated to generate a feature vector for the time series, which later can be used with any classification algorithm that accepts feature vectors as input. The reason for this approach is that time series dimensions can be processed in parallel for extracting basic statistics. Collecting the features extracted from each dimension into a single feature vector, allows us to learn dependencies between dimensions, without a heavy computational burden, as incurred by methods that try to

1. https://github.com/alan-turing-institute/sktime

directly model dimension dependencies.

Classifier For simplicity and scalability, when training TSC models we limit ourselves to linear models. These are known to be effective with large features spaces and can be trained efficiently. We employ the Ridge Regression Classification algorithm as implemented in the sklearn [28] library, because we found in our experiments that it is very efficient and no other classifier (e.g., Random Forest) outperforms its accuracy.

Our first approach computes four simple statistics over each dimension of the time series at hand. Namely, we compute the mean, minimum, maximum and standard deviation for each dimension independently. The resulting feature vector is $(4 * d)$-dimensional, where d is the number of dimensions in each time series. We henceforth call this approach 4_stat. To capture more patterns without greatly increasing the complexity of the method, we add five additional properties that can be computed efficiently from the time series. These are the median, kurtosis, number of peaks, 25 and 75 percentile resulting in a total of 9 features per dimension. Consequently, we term this method 9_stat. Finally, we use an even more extensive set of statistical features named Catch22; these features were found to be effective in the extensive empirical study [29]. We showcase the general pipeline of these simple approaches in Figure 1.

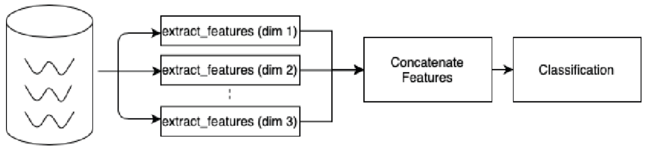


Figure 1: Feature extraction and classification in our proposed baselines.

Furthermore, we explore the effectiveness of the feature sets mentioned above while utilizing two techniques often employed in the time-series classification domain. First, to be able to capture details at a finer level, we use a rolling window approach.

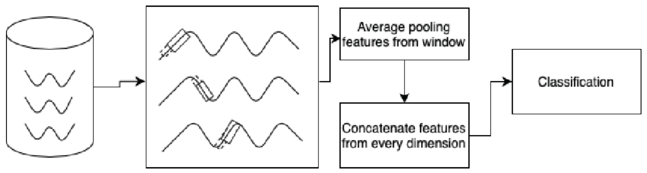

Average pooling features from window
Concatenate features from every dimension
Classification

Figure 2: Feature extraction and classification using a rolling window.

For each window, we compute the stats-features, followed by an average pooling step for each dimension and each feature. The average pooling step aims to get rid of noise and anomalies in the data. One thing to note is that the average (mean) feature will be exactly the same as if it is computed on the non-windowed time-series. We showcase this approach in Figure 2. Some of the Catch22 features do not make sense in such a setting, due to short windows

Method	4 stats	9 stats	Catch22	window	PAA
4_stat	x				
9_stat		x			
Catch22			x		
4_stat_window	x			x	
9_stat_window		x		x	
4_stat_PAA	x				x
9_stat_PAA		x			x
Catch22_PAA			x		x
4_stat_MulPAA	x				x
9_stat_MulPAA		x			x

TABLE 1: Summary of proposed baselines. All methods extract features from each time series dimension, concatenate all features in a single feature vector and train a linear classifier.

and dimension explosion, and hence we do not apply this technique while extracting Catch22 features. Secondly, we reduce the length of the time series using Piecewise Aggregate Approximation (PAA) [30]. PAA is a common dimensionality reduction technique used for time series and leads to shorter and smoother time series. In Figure 3 we show an example of extracting features from three different PAA transformed time series, where, for example, the number in PAA_30 means that we reduce the length of the time series to 30% of its initial length. This number is an input parameter to the PAA transform.

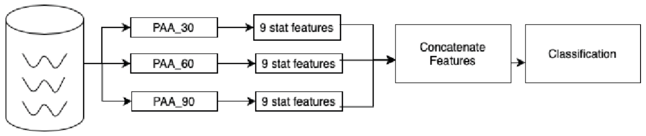


Figure 3: Feature extraction and classification using multiple PAA.

Besides the simplicity and ease of understanding of the features, a major advantage is that all of the features for our proposed baselines can be computed efficiently and in parallel, since the time series dimensions are considered independently during the feature extraction step. Table 1 provides a schematic overview of all the introduced methods and their corresponding names. The code for our methods is available on Github[2].

4. Experiments

In this section we describe the MTSC benchmark used for our evaluation and present the methods compared and their results.

4.1. MTSC Datasets

All approaches discussed in this paper were evaluated on the UEA MTSC archive [1]. There are a total of 30

2. https://github.com/mlgig/mtsc_benchmark

MTSC datasets which come from various domains, including human activity recognition, motion classification, ECG classification, EEG/MEG and audio spectra classification. The datasets have high variety in terms of the number of time series dimensions, sample size and length. The number of dimensions varies from 2 to 1,345, while the training sample sizes range from 12 to 30,000. The length of the longest time series is 17,894 while the smallest length is 8. Additionally, the archive also features some variable-length data. We use the data as is, although many of the compared methods have an intermediate step of normalising the data before training a model.

We use the critical difference (CD) diagram [31] to compare multiple MTSC methods. The CD diagram is a visual representation of the mean accuracy rank of multiple classifiers, computed over multiple datasets. It relies on pairwise statistical significance testing of the classification accuracy of compared methods, and connects with a thick horizontal line methods that are not found to have a statistically significant difference. As recommended in [31], [32], [33], we first perform a Friedmann test, followed by a Wilcoxon signed-rank test with Holm correction. We use the R library `scmamp` [34] to compute and plot all the CD diagrams.

4.2. Results and Discussion

In this study, we consider DTW_I and DTW_D as our baseline models. We were able to reproduce the results for these baselines on 26 datasets that have time series of equal-length. The DTW_I and DTW_D implementation[3] that we used in the experiments cannot handle variable-length time series. Some of the methods such as 9_stat_MulPAA and MrSEQL-SAX also work with variable-length time series. All comparative analysis in our experiments is done on these 26 datasets, unless mentioned otherwise (e.g., some methods ran into memory issues due to the scale of the data). All the compared methods' software implementations were run on a shared server. The server has 512GB memory with 72 cores and 32GB VRAM Nvidia GTX1080Ti. It operates on a Xeon E5-2695 processor with Ubuntu 18.04 LTS. We provide detailed accuracy and runtime results for all methods in the Github repository[4].

4.2.1. Comparing Variants of our Proposed Baselines. We first measure the quality of the models based on statistics-features, for classifying multivariate time series. As described in Section 3, we start by extracting 4 stats, 9 stats or Catch22 features from each dimension of the time series and compare the accuracy of the resulting models. From Figure 4, we see that the Catch22 feature set is the most effective in extracting the important information from the various dimensions. It is interesting to see that more stats features contribute to better accuracy.

3. https://github.com/uea-machine-learning/tsml
4. https://github.com/mlgig/mtsc_benchmark

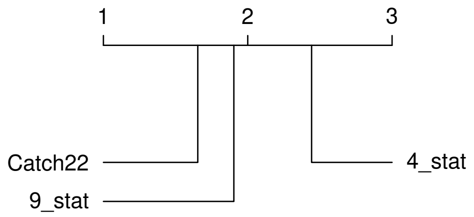


Figure 4: Critical difference diagram for methods based only on statistical features on 26 datasets.

In the next set of experiments, we investigate the effect of techniques used to smooth the input data (PAA), as well as to capture features at different resolutions (windowing). The results of all combinations can be found in Figure 5. We note that the best combination method is the one that uses 9 stats features with multiple PAA representations, although the difference to some of the other methods is not statistically significant.

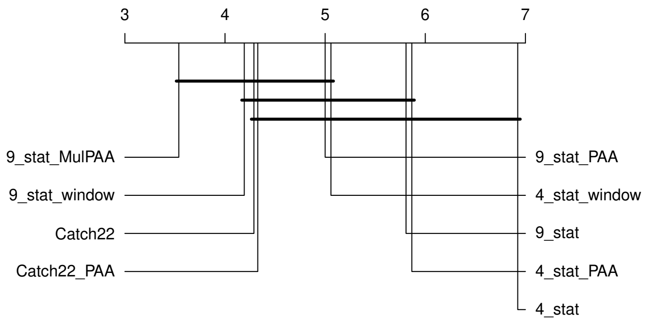
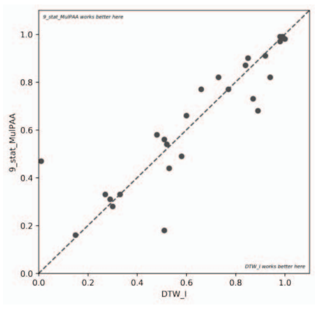


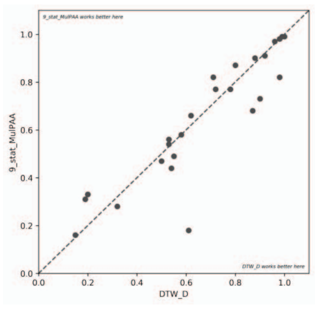
Figure 5: Comparison of all proposed stats-based baselines on 26 datasets.

Overall, the experiments show that both windowing and smoothing are effective as they both can improve the performance of their respective models. Furthermore, windowing appears to be more effective than PAA as it shows with the 4_stat and 9_stat models. Nonetheless, PAA is surprisingly ineffective when combining with Catch22 features. Finally, using multiple levels of smoothing ($\_MulPAA$ methods) had the strongest impact as it elevates the 9_stat model to the top of the average ranking. We do not combine Catch22 with multiple PAA due to efficiency issues, since computing the Catch22 features takes too long to run multiple times, and this runtime increases with the number of times PAA is being applied.

With regard to comparison to the state-of-the-art, we start by comparing our best method, 9_stat_MulPAA, to the DTW_D and DTW_I baseline methods introduced with the MTSC benchmark [1]. After performing a Friedmann test we found no statistically significant difference in accuracy between these three methods. Figure 6 gives a more detailed overview of the pairwise accuracy of the compared methods over the 26 datasets. The key difference though comes from the computational efficiency of the stats-based baseline 9_stat_MulPAA versus the DTW methods. The running time of these methods is reported and discussed in Section 4.2.3.



(a)



(b)

Figure 6: Accuracy comparison between (a) 9_stat_MulPAA vs DTW$_I$ and (b) 9_stats_MulPAA vs DTW$_D$

4.2.2. Comparison with State-of-the-art (SOTA). In this section, we compare our proposed method with the SOTA algorithms using a critical difference diagram between all the classifiers. All the SOTA methods (ROCKET-MTS[5], MLSTM-FCN[6], WEASEL-MUSE[7], MrSEQL-SAX[8](MrSEQL with SAX transformation only), TapNet[9]) results are obtained by running the original code provided by the authors. Unfortunately, although we followed the instructions in the original papers, we were unable to reproduce results for some of the methods compared. In the case of MLSTM-FCN, we opted to set the hyperparameter values recommended by [27] which resulted in better results. Some classifiers cannot handle variable-length time series (e.g., DTW, ROCKET, MLSTM-FCN), while others ran into memory issues (e.g., MUSE ran out of memory) or did not produce any results for some datasets (e.g., TapNet, MLSTM-FCN). The diagram in Figure 7 is based on 20 datasets for which we were able to obtain results with all methods. The summary of results for all methods and datasets can be found in our Github code repository.

ROCKET-MTS is the most accurate algorithm, followed by WEASEL-MUSE and MrSEQL-SAX. However, WEASEL-MUSE frequently runs out of memory. After a bit of a gap, Mr-SEQL comes third. Compared to ROCKET-MTS, the latter two methods have the advantage that they can also work with variable-length time series; nonetheless, they take more time for longer time series. The top three methods are followed by DTW_D, DTW_I and 9_stat_MulPAA, which are described in the earlier section. Surprisingly, the last ranked method is the deep learning approach MLSTM-FCN. The other deep learning method we analysed, TapNet, is not included in this comparison

	ROCKET-MTS	WEASEL-MUSE	MrSEQL-SAX	DTW$_D$	DTW$_I$	9_stat_MulPAA	MLSTM FCN
Average Rank	**2.57**	2.97	4.22	4.30	4.52	4.67	4.72
Wins/Ties	**9**	3	5	3	1	0	5

TABLE 2: Average rank and wins/ties for 7 compared methods across 20 datasets where all methods could run and deliver results.

since it only completed on 17 out of these 20 datasets. The results for Tapnet are given in our code repository.

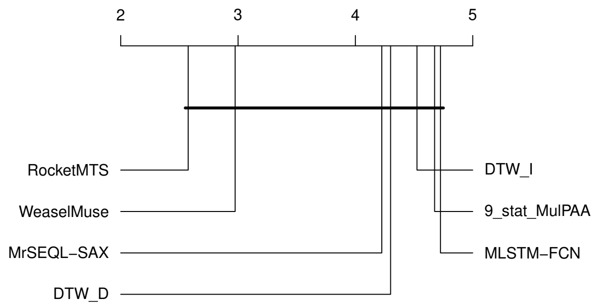


Figure 7: Comparison of mean rank classification accuracy for 7 methods across 20 datasets where all methods could deliver results.

4.2.3. Computation Time. Figure 8 presents the runtime comparison between the SOTA algorithms, baselines and our best approach, 9_stat_MulPAA, on 20 datasets of the UEA archive.

In our study, we found that ROCKET-MTS is one of the fastest (Figure 8) and most accurate MTSC algorithms, and strikes a good balance between runtime and accuracy. However, it currently does not work with variable-length time series (Table 3).

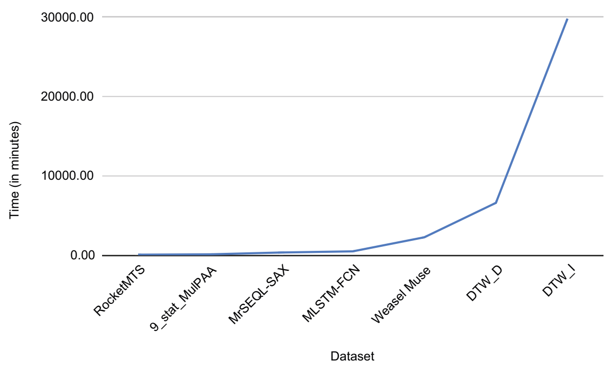


Figure 8: Runtime of 7 compared methods on 20 datasets.

WEASEL-MUSE, on the other hand, yields quite accurate results but it frequently runs out of memory and is the

5. https://github.com/alan-turing-institute/sktime/blob/master/examples/rocket.ipynb
6. https://github.com/titu1994/MLSTM-FCN
7. https://github.com/patrickzib/SFA
8. https://github.com/alan-turing-institute/sktime
9. https://github.com/xuczhang/tapnet

slowest algorithm in the top set. MrSEQL has the advantage that it runs well with variable-length time series, however, it has a high runtime especially for long time series. For TapNet we could not reproduce the results published. When we look at our proposed method it is clear that we can achieve similar accuracy to the DTW baselines, with a much better runtime. Along these lines, we see 9_stat_MulPAA as a faster alternative for the DTW approaches and encourage others to use it as an additional baseline.

4.2.4. Inter-dimension Dependency. The crucial aspect in the MTSC task is the data coming from different dimensions. In this section, we try to investigate if there is any useful interaction between the different dimensions of the time series. We chose DTW and ROCKET for this experiment since they support both univariate and multivariate time series. The DTW_I distance is simply the sum of each dimension's DTW distance while DTW_D computes the distance in multidimensional space. Intuitively, DTW_D is more capable to capture the inter-dimension dependency in the time series data. Similarly, ROCKET-MTS employs multiple kernels to capture the multidimensional features (one kernel per channel resulting in a random matrix where each row is a kernel corresponding to one channel) while ROCKET-UTS uses kernels that can only capture important information for each dimension independently. Surprisingly, we found no significant difference when comparing the two variants for each approach (Figure 9). This experiment suggests that dependencies across dimensions, usually complex and expensive to learn, are perhaps unnecessary. A simple linear combination of unidimensional features may already be sufficient for the MTSC task.

		ROCKET-MTS	MrSEQL-SAX	DTW_I	DTW_D	WEASEL-MUSE	MLSTM FCN
Challenges	Variable-length	C		C	C		C
	Scalability		C	C	C	C	
	Memory Consumption					C	
	Overfitting						C

TABLE 3: Challenges faced by evaluated state-of-the-art MTSC methods. 'C' marks the challenge this method does not yet address well.

5. Conclusion

In this paper we have evaluated recent state-of-the-art methods for multivariate time series classification on the UEA MTSC benchmark. We found that, while the methods compared do not achieve statistically significant difference in accuracy on a subset of 20 datasets where most methods can complete training and prediction, the recent method ROCKET appears to be the most accurate, with the linear

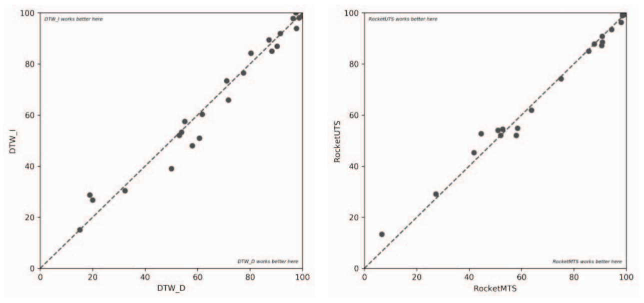


Figure 9: Accuracy comparison between (a) DTW_I vs DTW_D and (b) ROCKET-UTS vs ROCKET-MTS. There is no noticeable difference in accuracy between the uni-dimensional and the multi-dimensional variants of the same method.

classifiers Weasel-Muse and MrSEQL not far behind. Moreover, when considering running time, ROCKET appears to be the most appealing solution for MTSC. We also identified weaknesses in all of the SOTA methods which can hinder their applications to various problems. In particular, most methods do not work on variable-length time series (e.g., ROCKET) and run into memory issues or long runtime if the time series are very long (e.g., Weasel-Muse or MrSEQL). Recent deep learning methods proposed for MTSC, TapNet and MLSTM-FCN, do not achieve higher accuracy than the baselines. These challenges point out worthwhile directions for future work in this area.

In addition, we made a case for an alternate baseline method which is simple, fast to train and as accurate as DTW_D and DTW_I. Our approach mainly relies on some well-known statistical properties of time series, hence they are easy to implement and fast to compute. Furthermore, these statistical features can combine nicely with windowing or smoothing techniques (e.g., PAA) to enhance the performance of their respective models. Our best proposed method 9_stats_MulPAA relies only on nine statistical features and multiple levels of PAA smoothing and can perform comparably to DTW methods while running significantly faster.

From our study, we also want to question the necessity of capturing complex dimension dependencies in MTS data. Our experiments on two ROCKET and two DTW variants show that this extra knowledge has little impact on the overall accuracy of the models. While this is by no means conclusive evidence, we believe that this question should be considered when working on MTSC. A unified and ongoing comparison of MTSC algorithms is critical to maintain a clear overview of this research area, to identify strengths and weaknesses of existing approaches, and most importantly to facilitate and advance the research in this important topic which has a direct impact on a range of real-world applications.

Acknowledgments

This work was funded by Science Foundation Ireland through the VistaMilk SFI Research Centre

(SFI/16/RC/3835) and the Insight Centre for Data Analytics (12/RC/2289_P2). We would like to thank all the researchers that have contributed open source code and datasets to the UEA MTSC Archive and especially, we want to thank the groups at UEA and UCR that continue to maintain and expand the archive.

References

[1] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, "The uea multivariate time series classification archive, 2018," *arXiv preprint arXiv:1811.00075*, 2018.

[2] A. Dempster, F. Petitjean, and G. I. Webb, "Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, pp. 1–42, 2020.

[3] T. Le Nguyen, S. Gsponer, I. Ilie, M. O'Reilly, and G. Ifrim, "Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 1183–1222, 2019.

[4] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu, "Tapnet: Multivariate time series classification with attentional prototypical network," in *AAAI*, 2020.

[5] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.

[6] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, 2017.

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

[8] A. Bostrom and A. Bagnall, "A shapelet transform for multivariate time series classification," *arXiv preprint arXiv:1712.06428*, 2017.

[9] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *ACM SIGKDD*, 2014, pp. 392–401.

[10] P. Senin and S. Malinchik, "Sax-vsm: Interpretable time series classification using sax and vector space model," in *2013 IEEE 13th International Conference on Data Mining*, 2013, pp. 1175–1180.

[11] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Mining and Knowlegde Discovery*, vol. 29, no. 6, p. 1505–1530, Nov. 2015. [Online]. Available: https://doi.org/10.1007/s10618-014-0377-7

[12] P. Schäfer, "Scalable time series classification," *Data Mining and Knowledge Discovery*, vol. 30, pp. 1273–1298, 2015.

[13] P. Schäfer and U. Leser, "Fast and accurate time series classification with weasel," in *Proceedings of the 2017 ACM CIKM*, 2017, pp. 637–646.

[14] A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Mining and Knowledge Discovery*, vol. 29, 06 2014.

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

[16] J. Lines, S. Taylor, and A. Bagnall, "Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 1041–1046.

[17] A. Shifaz, C. Pelletier, F. Petitjean, and G. I. Webb, "Ts-chief: a scalable and accurate forest algorithm for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, pp. 742–775, 2020.

[18] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[19] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018.

[20] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Min. Knowl. Discov.*, vol. 31, no. 3, p. 606–660, May 2017. [Online]. Available: https://doi.org/10.1007/s10618-016-0483-9

[21] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2003, pp. 2–11.

[22] M. Shokoohi-Yekta, J. Wang, and E. J. Keogh, "On the non-trivial generalization of dynamic time warping to the multi-dimensional case," in *SDM*, 2015.

[23] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107–144, Oct 2007. [Online]. Available: https://doi.org/10.1007/s10618-007-0064-z

[24] P. Schäfer and M. Högqvist, "Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets," in *Proceedings of the 15th international conference on extending database technology*, 2012, pp. 516–527.

[25] P. Schäfer and U. Leser, "Multivariate time series classification with weasel+ muse," *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data (AALTD18), arXiv preprint arXiv:1711.11343*, 2017.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[27] A. P. Ruiz, M. Flynn, and A. Bagnall, "Benchmarking multivariate time series classification algorithms," *https://arxiv.org/abs/2007.13156*, 2020.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[29] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, "catch22: Canonical time-series characteristics selected through highly comparative time-series analysis," *Data Mining and Knowledge Discovery*, vol. 33, p. 1821–1852, 2019.

[30] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[31] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, Dec. 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1248547.1248548

[32] S. Garcia and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 12 2008.

[33] A. Benavoli, G. Corani, and F. Mangili, "Should we really use post-hoc tests based on mean-ranks?" *Journal of Machine Learning Research*, vol. 17, no. 5, pp. 1–10, 2016. [Online]. Available: http://jmlr.org/papers/v17/benavoli16a.html

[34] B. Calvo and G. Santafé, "scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems," *The R Journal*, vol. 8, no. 1, pp. 248–256, 2016. [Online]. Available: https://doi.org/10.32614/RJ-2016-017