An Empirical Evaluation of Multivariate Time Series Classification with Input Transformation across Different Dimensions

Pantiskas, Leonardos; Verstoep, Kees; Hoogendoorn, Mark; Bal, Henri

published in
2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)
2023

DOI (link to publisher)
10.1109/ICMLA55696.2022.00012

document version
Publisher's PDF, also known as Version of record

An Empirical Evaluation of Multivariate Time Series Classification with Input Transformation across Different Dimensions

Leonardos Pantiskas
Vrije Universiteit Amsterdam
0000-0002-4898-5334

Kees Verstoep
Vrije Universiteit Amsterdam
0000-0001-6402-2928

Mark Hoogendoorn
Vrije Universiteit Amsterdam
0000-0003-3356-3574

Henri Bal
Vrije Universiteit Amsterdam
0000-0001-9827-4461

***Abstract*—In current research, machine and deep learning solutions for the classification of temporal data are shifting from single-channel datasets (univariate) to problems with multiple channels of information (multivariate). The majority of these works are focused on the method novelty and architecture, and the format of the input data is often treated implicitly. Particularly, multivariate datasets are often treated as a stack of univariate time series in terms of input preprocessing, with scaling methods applied across each channel separately. In this evaluation, we aim to demonstrate that the additional channel dimension is far from trivial and different approaches to scaling can lead to significantly different results in the accuracy of a solution. To that end, we test seven different data transformation methods on four different temporal dimensions and study their effect on the classification accuracy of five recent methods. We show that, for the large majority of tested datasets, the best transformation-dimension configuration leads to an increase in the accuracy compared to the result of each model with the same hyperparameters and no scaling, ranging from 0.16 to 76.79 percentage points. We also show that if we keep the transformation method constant, there is a statistically significant difference in accuracy results when applying it across different dimensions, with accuracy differences ranging from 0.23 to 47.79 percentage points. Finally, we explore the relation of the transformation methods and dimensions to the classifiers, and we conclude that there is no prominent general trend, and the optimal configuration is dataset- and classifier-specific.**

***Index Terms*—time series, classification, multivariate, input preprocessing, scaling**

I. Introduction

Due to the rising availability of data sources in sectors such as industry, healthcare, and finance, time series classification datasets and problems are increasingly consisting of multiple channels of information [1], representing for instance readings from multiple sensor types. Following this trend, machine and deep learning solutions have shifted to try to more effectively address these multivariate problems [1].

An aspect that does not usually get much focus when describing a novel machine or deep learning solution is the preprocessing of the input data. With the terms preprocessing, scaling, and transformation we refer to the methods that modify a set of values to deal with issues such as outliers or to shift them to a predefined range, e.g. standardization or min-max scaling. Especially in the deep learning landscape, it is now a standard approach at the implementation level to implicitly transform all values with some method such as normalization, in order to bring them to the same numerical scale prior to propagating them through the network. In the time series classification field, there has already been extended research for the univariate datasets [2], so it is a natural direction to try and transfer the concepts to the multivariate cases. This can regard the models themselves, e.g. by applying a model to each univariate channel of a multivariate problem and then aggregating the classification decisions using some ensembling or voting method [1]. Similarly to this approach, the preprocessing of the input also follows the same pattern. For example, if in the case of univariate classification, where the dataset dimensions are (*samples*,*timesteps*), all observations are standardized, it seems natural to apply the same transformation to each channel of the multivariate dataset which has dimensions (*samples*,*channels*,*timesteps*), treating it as a separate univariate entity. However, as has been noted by Ruiz et al. in [1], the transformation of the input data in the case of multivariate datasets is not a trivial problem. The additional data dimension presents several options even in the fundamental handling of this prior input scaling.

In this work, we want to empirically explore the opportunities that can arise from scaling the temporal data across different dimensions and the effect this can have on classification accuracy. In order to achieve this, we experiment with five recent multivariate time series classification models, which are a mix of deep learning and other methods and are representative solutions with results equal to or sufficiently close to the state of the art. We choose seven different transformation methods and apply them to four distinct slices of the temporal data. Our contributions are:

- We show that in the large majority of datasets tested, the best combination of transformation method and dimension it is applied to leads to better accuracy than that of the models with the same functional hyperparameters and no input scaling, ranging from 0.16 to 76.79 percentage points.
- We show that in the majority of the best configurations, there is a statistically significant difference among the

This work has been conducted as part of the Just in Time Maintenance project funded by the European Fund for Regional Development.

accuracy results of the models when the same transformation method is applied to different temporal dimensions, ranging from 0.23 to 47.79 percentage points.
- We explore the relation of the best transformation methods and dimensions to the models and we find that there are no distinct general trends and that the best configuration depends on the dataset and classifier used.

II. Related Work

In fields such as computer vision, data-centric approaches that aim to increase model accuracy have been widely utilized, with data augmentation methods such as cropping and rotation applied to images to increase the amount of the training data, tackle class imbalances, and make the model more robust to perturbations of the input [3]. In contrast to that, data augmentation techniques in the time series domain have been less extensively employed. Although the three-dimensional nature of the multivariate time series problem may initially resemble image data, the temporal dependencies and dynamics of channels, as noted in [4], lead to a qualitative difference between the challenges.

In recent surveys of time series data augmentation [4], [5] the methods presented range from basic ones inspired by the computer vision field, such as flipping and slicing samples, to more advanced ones utilizing deep generative models. In a recent work specifically focused on multivariate time series classification [6], the authors showed that basic time series augmentation methods can be beneficial to the task, counteracting the overfitting of models, especially on smaller datasets.

Our exploration of the transformation of data across different dimensions, although not strictly a data augmentation method, can be considered a data-centric approach, in the sense that we are trying to achieve better classification accuracy only by modifying the input data in a specific manner. It is orthogonal to the above data augmentation methods and is conceptually placed at an earlier stage. It stems from a consideration of the inherent nature of the time series datasets and what the intrinsic relationship of each dimension slice is to the real-world problem.

III. Methodology

A. Models

The landscape of multivariate time series classification models is continuously evolving, with ever-more complex and accurate models and methods [1]. We can, however, distinguish two broad categories which encompass multiple recent methods: machine learning models based on extracted features and deep learning models. In the first case, several features are extracted from either the input values or a transformation of them, and those features are then used with a linear classifier for the final classification. In the second category, the input values are propagated to a deep learning architecture, usually after normalization to bring them to a similar scale. The architecture then internally performs the end-to-end transformation and classification of the input. We select three recent methods belonging to the first category and two belonging to the second. A short description of the methods follows, starting with the feature extraction ones:

ROCKET [7] is a method based on random convolutional kernels which not only achieves the best results in terms of accuracy according to a recent evaluation [1] but is also the fastest approach in terms of training time. Two features are extracted from the output of the convolution of the input with each random kernel.

WEASEL+MUSE [8] extracts features from windows of the input, utilizing a truncated Fourier transform and bag-of-patterns approach. It also applies statistical filtering of these features using a χ^2 test.

LightWaveS [9] utilizes lightweight wavelet scattering with arbitrary wavelets. Four statistical features are extracted from each of the scattering coefficients and are filtered with a hierarchical feature selection approach.

Our selected models in the deep learning category are:

ResNet [10], which has been proposed as a strong deep learning baseline for the time series classification task, with its architecture consisting of convolutional layers, residual connections, and global average pooling layers.

InceptionTime [11], which utilizes a more complex architecture of convolutional blocks and bottleneck layers, in the form of Inception modules [12], with residual connections.

The models were selected on the basis of them being recent time series classification methods, which were designed to handle multivariate data and are not just ensembles of univariate methods. Moreover, the selected models include the best classifiers for 20 out of the 26 equal-length UEA problems according to the reported accuracy metrics in [1], so they are a very representative sample of the current state of the art. Another factor that was taken into account was the computational cost of each method. Since we have to perform multiple resamplings of multiple transformation methods across four data slices, we have to limit the model selection in order for the experiments to finish within a reasonable amount of time. Thus, although solutions such as HIVE-COTE [13] and CIF [14] may rank higher in accuracy for some problems, their very long training time makes it impractical to fairly include them in our evaluation.

B. Transformation methods

Data scaling is of course a standard method during the exploratory data analysis phase of a problem. However, as we mentioned above, it is often overlooked when presenting novel time series classification approaches, especially in the recent deep learning environment, where the model architecture is expected to reach the correct weight values regardless of the input format. In our experiments, we test seven different well-known transformation methods [15], ranging from simple linear to more complex non-linear functions. We present those below, with a short description for the sake of completeness:

Normalization The values are transformed so that their L2 norm is 1.

Standardization The values are transformed so that they have zero mean and unit variance.

MinMax The values are scaled to the [0,1] range.

MaxAbs The values are scaled based on their maximum absolute value, but their sign is retained. On positive data, this method is equivalent to MinMax.

Robust The values are scaled based on their median and interquartile range, which are robust against outliers.

Power Transformation The values are non-linearly transformed with a power transformation (Yeo-Johnson method in our experiments) in order to approach a Gaussian distribution, minimizing skewness and stabilizing variance.

Quantile Transformation The values are non-linearly transformed so that their probability density function is mapped to a uniform distribution with a [0,1] range.

C. Dimensions

The 3-dimensional nature of multivariate time series problems, namely samples, channels, and timesteps, presents a multitude of options for selecting data slices across which the appropriate transformation method can be applied. As we said, as a result of the mapping of concepts from the univariate time series research, it may seem natural to transform all values of each channel as a separate set. In this work, we claim that this choice is not standard or trivial and that different configurations may lead to considerably different results. For instance, it is generally accepted that a large part of the added value in multivariate datasets comes from the interplay and associations among different channels. Thus, by selecting data slices that include values across different channels, we introduce such associations even before the processing of the input by the models, which may be able to help them perform better in some datasets. We denote the original dataset as D with N samples, C channels, and T timesteps. Below we present the four distinct data slices that we selected for experimentation, along with an intuitive explanation:

Channels This is the configuration more closely related to the univariate paradigm, as all values of each channel across all samples are considered a separate set $S_i = \{D_{*,i,*}\}, 1 \leq i \leq C$.

Timesteps In this configuration, the values of each timestep across all samples and all channels are considered a separate set $S_i = \{D_{*,*,i}\}, 1 \leq i \leq T$.

Both This configuration is a combination of the above, where for each channel, the values of each timestep across all samples are considered a separate set $S_{ij} = \{D_{*,i,j}\}, 1 \leq i \leq C, 1 \leq j \leq T$.

All In this configuration, all values of the dataset are taken as a single set $S = \{D_{*,*,*}\}$.

This non-exhaustive selection of dataset slices is based on the rationale of capturing the intuitive, real-world meaning of each dimension. For example, if the different channels represent sensors with significantly different value ranges, it would make sense to transform them separately. On the other hand, if all sensors are of the same type, but for instance, their readings come from fixed points of a process, then each timestep is potentially a more important dimension to consider for classification. By combining these concepts, we end up with the four slices mentioned above. We see that two of our selected slices do not include information sharing across time series (**channels** and **both**) while the other two do.

IV. Experiments

A. Datasets

We experiment on the 26 of the 30 datasets of the UEA collection [16] that have equal-length samples and can thus be handled easily by all models. Moreover, this is the same subset for which there are detailed metrics in [1], so we have a robust point of reference. For the WEASEL+MUSE method, we also exclude the datasets *DuckDuckGeese*, *EigenWorms*, *FaceDetection*, *MotorImagery*, *PEMS-SF*, and *PhonemeSpectra*, due to its inability to successfully complete the training on these, as also noted in [1].

B. Experimental setup

All experiments were run on the DAS-6 infrastructure [17], on nodes with 24-core AMD EPYC-2 (Rome) 7402P CPUs, NVIDIA A6000 GPUs, and 128 GB of RAM. We implement ROCKET and MUSE using *sktime* [18] and InceptionTime and ResNet using its deep learning extension, *sktime-dl*. For LightWaveS, we use its provided code.

Regarding the method parameters, we tried to follow as closely as possible the ones reported in [1] and used the default settings for ROCKET and LightWaveS. We present those parameters in detail in Table I. As baseline, we use the models on the unmodified UEA datasets and in addition, we disable all data preprocessing in the methods that allow this.

We used scikit-learn [19] to implement all scaling methods. We repeat each experiment 20 times with different starting seeds and get the mean accuracy. For each model and dataset, we sort the different configuration results by descending mean accuracy and then ascending standard deviation among resamples. In this way, we find the scaling method - dimension combination which yields the highest mean accuracy and most stable results. We present the mean-accuracy difference from the baselines only when it is statistically significant.

In order to distinguish the value of dimension selection from that of the transformation method, we present another set of results: For each of the datasets and models, we keep the transformation method of the best-performing configuration fixed and we apply it to all four data slices. We then do pairwise testing to determine the statistical difference between all possible pairs of the four accuracy results.

We do not make any assumptions about the result distributions, so in both cases, we use the Wilcoxon signed-rank test [20] with p-value of 0.05 and Holm's alpha correction when needed [21]. The code for the experiments as well as the detailed metrics are made available at https://github.com/lpphd/mtsscaling to facilitate reproducibility of the results.

V. Results

We present the best-achieved accuracy, as well as the difference from the baseline result for each model in Table II.

TABLE I
METHOD PARAMETERS

Method	Parameters
ROCKET	Ridge regression classifier, 10000 kernels
WEASEL+ MUSE	Default sktime parameters (anova=True, bigrams=True, window_inc=2, p_threshold=0.05, use_first_order_differences=True)
LightWaveS	Ridge regression classifier, 500 features
ResNet	Epochs: 1500, Batch size: 16, Learning rate: 1e-3 and halved after no improvement for 50 epochs Three residual blocks each with three conv layers with kernel sizes [8, 5, 3] Filters per conv layer for each block [64, 128, 128] Training ends after no improvement for 150 epochs Weights with lowest training loss are used for testing
InceptionTime	Epochs: 1500, Batch size: 16, Learning rate: 1e-3 and halved after no improvement for 50 epochs Two residual blocks each with three Inception modules with kernel sizes per module [10, 20, 40] Plus bottleneck filters for all conv layers 32 Training ends after no improvement for 150 epochs Weights with lowest training loss are used for testing

TABLE II
ACCURACY UNDER BEST TRANSFORMATION-DIMENSION CONFIGURATION AND SIGNIFICANT DIFFERENCES FROM BASELINE ACCURACY

	ROCKET	MUSE	LightWaveS	ResNet	IT
AWR	99.8 (+0.48)	99.3 (-)	99.7 (-)	98.2 (+0.35)	98.7 (+0.3)
AF	46.7 (+26.67)	40.7 (+15.0)	46.7 (+13.33)	38 (+8.0)	36 (+15.67)
BM	100 (-)	100 (-)	100 (-)	100 (-)	100 (-)
CR	100 (-)	100 (+0.62)	97.2 (+4.17)	99.9 (+1.18)	99.4 (+0.83)
DDG	69.1 (+7.4)	N/A	52 (+8.0)	69 (+7.7)	66.8 (+7.7)
ER	98.6 (-)	97 (+1.52)	96.7 (-0.37)	93.1 (+6.48)	91.2 (+2.74)
EW	96.5 (+6.07)	N/A	96.2 (-)	93.9 (+76.79)	94.9 (+10.53)
EP	100 (-)	100 (-)	97.8 (+1.45)	99.2 (+0.43)	97.9 (+1.01)
EC	44.7 (-)	37.8 (-)	63.5 (-0.38)	30.4 (+5.91)	29.6 (+2.45)
FD	66.1 (+2.3)	N/A	64.5 (+3.2)	65.7 (+7.49)	66.7 (+1.83)
FM	59.8 (+5.7)	57 (+3.8)	59 (+4.0)	56 (+2.2)	59.2 (-)
HMD	50.1 (-)	34.6 (+3.78)	43.2 (+9.46)	36.2 (-)	44.7 (+7.57)
HW	59 (+0.46)	38.2 (+9.76)	38.2 (+1.18)	61 (+1.85)	58.2 (-)
HB	78.9 (+1.9)	79.5 (+6.2)	80.5 (+7.32)	77.5 (+20.78)	76.7 (+7.68)
LSST	67.8 (+3.49)	64.7 (+4.04)	47.8 (+13.76)	61.6 (+10.0)	64.5 (+13.53)
LIB	96.2 (+2.33)	91.9 (-)	90 (+5.0)	96.1 (+0.5)	90.1 (+1.06)
MI	59.6 (+5.45)	N/A	64 (+13.0)	54.6 (+2.1)	52.4 (+2.05)
NATO	95.1 (+1.14)	93.7 (+1.53)	76.7 (+13.33)	97.4 (+1.06)	95.7 (-)
PEMS	88.3 (+15.32)	N/A	92.5 (+11.56)	91.4 (+13.5)	87.7 (+12.31)
PD	98.3 (+0.16)	94.8 (+0.22)	95.4 (-0.05)	98.6 (-)	98.9 (-)
PS	31.6 (+0.65)	N/A	22.7 (+7.58)	31.6 (-)	30.7 (+0.63)
RS	93 (+3.16)	89.9 (+2.53)	88.8 (+5.26)	93 (+1.97)	92.9 (+3.88)
SRS1	93 (+2.2)	77.5 (-)	88.4 (+9.22)	78.6 (+1.57)	86.3 (+1.43)
SRS2	57.5 (+6.64)	55.4 (+2.97)	53.9 (+7.22)	52.3 (+5.19)	54 (+4.25)
SWJ	51.3 (+3.33)	53.3 (+11.0)	60 (+6.67)	41 (+12.33)	70 (+34.0)
UW	94.3 (+0.61)	93 (+1.62)	94.1 (+3.12)	86.6 (+0.98)	90 (+0.5)

We can see that there is an increase in accuracy in the large majority of the datasets for all models, ranging from 0.16 to 76.79 percentage points, with the median increase being 3.88 percentage points. Broken down by model, the median increase in accuracy is 2.75 for ROCKET, 3.38 for MUSE, 7.27 for LightWaveS, 3.7 for ResNet, and 2.74 for InceptionTime. It is remarkable that not only is the accuracy increased compared to the baseline experiments, but the best accuracy across all models is higher than the best accuracy presented in [1] for 13 out of the 26 datasets, showing that this input preprocessing exploration can result in new state-of-the-art results without modifying the base model at all.

A point that merits explanation is that this approach of no input scaling as baseline differs from the default behavior of models such as ROCKET and LightWaveS, which employ scaling as part of their pipeline, or the usual normalization for deep learning models. The reason we follow it is to get as fair results as possible and create a reference point based only on the mechanics of the models rather than any scaling effect. However, we can confirm that the same trends and conclusions hold true for the default behavior of our selected classifiers by getting their reported accuracy metrics on the same datasets from [1], [9] and performing mean-accuracy comparison. Again, for the majority of datasets and classifiers, there is an increase in accuracy, ranging from 0.1 to 40.0 with a median of 3.25.

There are also a few negative results, which indicate that the application of no transformation gives better accuracy for specific models and datasets. These results do not affect our conclusions, since we are considering the transformation method and dimension as hyperparameters, and we can include additional configurations in this hyperparameter search to increase the chances of achieving the optimal result.

We also aim to distinguish the value of dimension selection from that of the transformation method. Although in practice the scaling method and dimension would be co-selected based on their interplay and the dataset characteristics, we want to demonstrate that even for more complex transformation methods, the dimension selection can significantly affect the outcome. In Table III we see whether or not there is a statistically significant difference in the accuracy results between any two dimensions under the optimal scaling method, and if so, what the difference is in the mean accuracy between the

optimal and worst dimension.

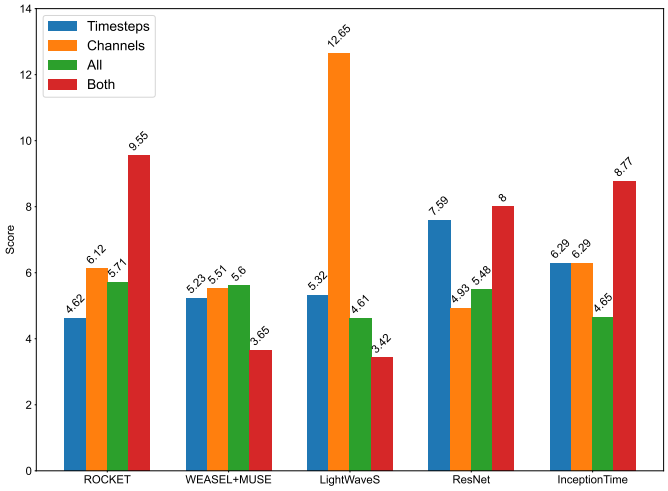
TABLE III
DIFFERENCE (OF SIGNIFICANTLY DIFFERENT RESULTS) IN MEAN ACCURACY BETWEEN BEST AND WORST DIMENSION FOR FIXED TRANSFORMATION METHOD

	ROCKET	MUSE	LightWaveS	ResNet	IT
AWR	0.42	1	1.33	0.63	-
AF	33.33	15	26.67	7.33	12.33
BM	-	-	-	-	-
CR	1.39	2.78	6.94	3.82	1.53
DDG	8	N/A	32	10.6	3.1
ER	1.76	2.46	1.48	6.85	3.41
EW	47.79	N/A	14.5	3.05	5.27
EP	-	0.72	1.45	0.65	4.13
EC	10.32	5.21	20.91	2.49	-
FD	1.02	N/A	-	1.77	-
FM	6.05	4.45	12	5.35	-
HMD	6.35	-	24.32	10	10.95
HW	11.74	3.2	13.29	16.99	19.25
HB	3.73	-	5.85	-	2.56
LSST	0.41	13.42	0.66	3.73	-
LIB	1.72	1.36	2.22	-	-
MI	7.95	N/A	15	2.45	2.25
NATO	2.39	8.14	19.44	3.78	1.11
PEMS	5.84	N/A	8.67	6.94	9.48
PD	0.31	0.69	-	0.23	-
PS	0.56	N/A	7.15	1.94	1.76
RS	0.63	3.06	10.53	1.51	2.47
SRS1	0.84	-	11.95	-	2.83
SRS2	3.31	3.31	5.56	4.67	3.14
SWJ	10.33	11.33	13.33	7	33
UW	0.42	5.5	1.88	-	-

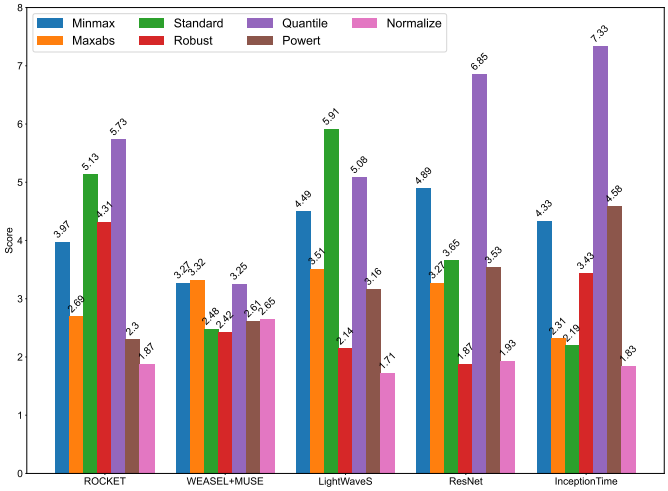
These results reinforce our conclusions, as we can see that for the majority of datasets and models there is a statistically significant difference between the results of at least two out of the four dimensions and the mean-accuracy differences range from 0.23 to 47.79 percentage points, with the median being 3.82 points. This shows that a significant part of the accuracy increase compared to the baselines stems from the selection of the most suitable dimension for a given dataset and classifier.

We can also study the configurations that achieve the best performances to discover potential trends in the dimension or transformation method selection. To achieve this, for each classifier and dataset we consider the group of configurations that help achieve either the top accuracy or within 1 percentage point of it. We then calculate a score for each dimension and transformation method that appears in these configurations, which is defined as the number of times it appears divided by the total number of the group members. For example, if the top configurations are [minmax_both, standard_both, quantile_all], the dimension 'Both' would get a score of (1+1)/3 = 2/3, while each of 'MinMax', 'Standard', 'Quantile' methods would get a score of 1/3. By summing this normalized score across all datasets, we get a "usefulness" profile of the dimensions and transformation methods for each classifier. We can see these scores in Fig. 1.

Regarding the dimension scores in Fig. 1a, we can see that there is no universal trend and the results are classifier-specific. One result that stands out is the high utility of "Channel" dimension for the LightWaveS method. This seems to be related to the method's mode of operation, which is extracting features from individual input channels, without combining them in any way. ROCKET does combine channels when generating features, and its top dimension is "Both". This dimension is also the top one for the two deep learning models, although for ResNet the "TimeSteps" dimension is also valuable. On the contrary, in WEASEL+MUSE we observe that "Both" has the lowest value, with a balance across the other dimensions.



(a) Dimension scores



(b) Transformation method scores

Fig. 1. Utility scores of dimensions and transformation methods for each classifier

Considering the transformation method scores in Fig. 1b, we see again that the results vary across classifiers. The two deep learning models show a common behavior in that "Quantile" and "MinMax" are valuable methods in both, which seems to validate the common approach of scaling the input to the [0,1] range. However, they also show deviation in the scores of methods such as "Robust" and "Standard". "Quantile" is also the top method for ROCKET, followed by "Standard", while for LightWaveS this order is reversed. MUSE seems to have a relative balance across transformation

methods. A general trend that we can observe is that the "Quantile" method seems to be useful for all classifiers, while the simple normalization method has a low score in all cases.

These figures show that there is no clear winner either in dimension or transformation methods, and the best configuration depends on the classifier and the dataset under consideration. The conclusion we can draw from this is that the inclusion of data dimension and transformation method in the hyperparameter search of a model is the most certain method of discovering the configuration that gives the optimal result in terms of accuracy.

VI. Discussions and Conclusion

In summary, in this paper, we empirically explore the input preprocessing possibilities presented by the format of multivariate time series datasets, namely *(samples, channels, timesteps)* and their effect on classification accuracy. We test seven data transformation methods across four distinct data slices and their effect on the accuracy of five recent machine and deep learning methods. We show that the optimal configuration of data slice and transformation method leads to an increase in the classification accuracy in almost all cases, in comparison with the baselines without any preprocessing. We also show that the correct dimension selection can lead to a large accuracy increase compared to a sub-optimal selection.

The above empirical results affect topics on a broad spectrum of time series analysis and classification, from the evaluation of novel methods to computational cost savings. In research, these results indicate that data-centric approaches are a fruitful research direction and can have significant benefits in terms of classification accuracy, on par or even better than new methods, without incurring the additional model complexity, especially in the landscape of deep learning. In this case, the novel approaches should be evaluated based on additional aspects, such as interpretability or deployment suitability. Similarly, in the more applied industry sector, it points practitioners to the possibility of increasing accuracy for a specific use case through data-centric means, obviating the need to switch to more computationally expensive or communication-intensive models, especially in the edge intelligence applications. A natural research direction stemming from our work is to formalize the discovery of the most suitable transformation method and dimension. Although for the faster methods such as ROCKET and LightWaveS it is easy to quickly search for the best configuration, it is quite impractical for the slower methods such as MUSE. Thus, a desirable approach would indicate the optimal dimension for each dataset, possibly based on the statistical properties of each data slice, and also explain this choice. In terms of more applied directions, it would be interesting to experiment with additional models which may have more markedly different approaches than the ones presented, such as shapelet-based ones [1]. Finally, additional data slices could be explored, such as grouping channels depending on the underlying problem and data source type, e.g., sensors of similar type in an IoT problem.

References

[1] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, "The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 35, no. 2, pp. 401–449, 2021.

[2] 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.

[3] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[4] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 8 2021, pp. 4653–4660.

[5] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *Plos one*, vol. 16, no. 7, p. e0254841, 2021.

[6] H. Yang and T. Desell, "Robust augmentation for multivariate time series classification," *arXiv preprint arXiv:2201.11739*, 2022.

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

[8] P. Schäfer and U. Leser, "Multivariate Time Series Classification with WEASEL+MUSE," *arXiv:1711.11343 [cs]*, 2018.

[9] L. Pantiskas, K. Verstoep, M. Hoogendoorn, and H. Bal, "Taking ROCKET on an Efficiency Mission: Multivariate Time Series Classification with LightWaveS," *arXiv:2204.01379 [cs]*, 2022.

[10] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 1578–1585.

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

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

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

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

[15] scikit-learn developers, "Compare the effect of different scalers on data with outliers." [Online]. Available: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

[16] 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:1811.00075 [cs, stat]*, 2018.

[17] H. Bal, D. Epema, C. de Laat, R. van Nieuwpoort, J. Romein, F. Seinstra, C. Snoek, and H. Wijshoff, "A medium-scale distributed system for computer science research: Infrastructure for the long term," *Computer*, vol. 49, no. 5, pp. 54–63, 2016.

[18] M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király, "sktime: A Unified Interface for Machine Learning with Time Series," in *Workshop on Systems for ML at NeurIPS 2019*, 2019.

[19] 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.

[20] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in statistics*. Springer, 1992, pp. 196–202.

[21] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.