TAKING ROCKET ON AN EFFICIENCY MISSION: A DISTRIBUTED SOLUTION FOR FAST AND ACCURATE MULTIVARIATE TIME SERIES CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Nowadays, with the rising number of sensors in sectors such as healthcare and industry, the problem of multivariate time series classification (MTSC) is getting increasingly relevant and is a prime target for machine and deep learning solutions. Their expanding adoption in real-world environments is causing a shift in focus from the pursuit of ever higher prediction accuracy with complex models towards practical, deployable solutions that balance accuracy and parameters such as prediction speed. An MTSC solution that has attracted attention recently is ROCKET, based on random convolutional kernels, both because of its very fast training process and its state-of-the-art accuracy. However, the large number of features it utilizes may be detrimental to inference time. Examining its theoretical background and limitations enables us to address potential drawbacks and present LightWaveS: a distributed solution for accurate MTSC, which is fast both during training and inference. Specifically, utilizing a wavelet scattering transformation of the time series and distributed feature selection, we manage to create a solution which employs just 2,5% of the ROCKET features, while achieving accuracy comparable to recent deep learning solutions. LightWaveS also scales well with more nodes and large numbers of channels. In addition, it can give interpretability into the nature of an MTSC problem and allows for tuning based on expert opinion. We present three versions of our algorithm and their results on training time, accuracy, inference speedup and scalability. We show that we achieve speedup ranging from 8x to 30x compared to ROCKET during inference on an edge device, on datasets with comparable accuracy.

1 INTRODUCTION

Time series classification is the task of characterizing a series of values observed in sequential moments in time as belonging to one of two or more categories, or classes. There has been extensive work on univariate time series classification with machine and deep learning methods, as observed in surveys such as Bagnall et al. (2017). In practice, as noted in Ruiz et al. (2021), many problems are described by more than one channels of information, turning the task into multivariate time series classification (MTSC). Some recent contributing factors have been the development of smaller and cheaper sensors for the measurement of various quantities, the general advancement of Internet of Things and the engagement with inherently more complex problems in sectors such as healthcare and industry, which benefit from additional information channels. In addition, the rise of edge computing and deployment of models in such environments makes it necessary to also consider the conditions and resources of the computational nodes on which those models will execute. This change in perspective makes it important to include, apart from accuracy, variables such as prediction speed or throughput in the evaluation of a model's suitability for a given task.

In the recent evaluation of advances in MTSC (Ruiz et al., 2021), one solution that shows good performance is ROCKET (Dempster et al., 2019), both in terms of accuracy and training time. ROCKET utilizes random convolutions to transform the time series channels and then extracts two features per convolution that are used with a linear classifier. The evolution of ROCKET, MINIROCKET (Dempster et al., 2020), is proposed as the new default variant of ROCKET by its authors and utilizes only one feature per kernel (percentage of positive values), thereby halving

the features. It also utilizes other optimizations to speed up ROCKET in general, and uses a minimally random approach with a given set of kernels. Another variant, MultiRocket (Tan et al., 2021), uses more features per convolution to achieve better accuracy at the expense of transformation time. Both ROCKET and MINIROCKET have been developed for univariate time series. Although the authors of MINIROCKET characterize their provided repository code for multivariate time series as naive, it is the same algorithmic logic in the form of a ROCKET extension in the sktime (Löning et al., 2019) repository that achieves the impressive performance mentioned above.

In (MINI)ROCKET, the authors encourage research into more sophisticated approaches for multivariate time series. We can start by identifying potential points that can be improved. One point with (MINI)ROCKET that we try to address in our work is that the default number of features, although beneficial to accuracy, can be highly redundant, and extreme for some datasets, leading to unnecessarily high transformation and inference times during deployment. In addition, due to the fixed process and number of generated features, there is a large discrepancy in the representation (features per channel) across datasets with different dimensions. Moreover, the indiscriminate inclusion of all channels in the feature generation is susceptible to uninformative series. We present experiments to illustrate the above points in section 4.3. Finally, the stochastic element, especially the random channel combination, although beneficial to accuracy, does not offer significant interpretability.

Based on this analysis, we propose LightWaveS, a framework for fast, distributed transformation of multivariate time series based on convolutional kernels, wavelet scattering and feature selection, for lightweight and accurate classification with linear classifiers. Our solution aims to keep the successful aspects of the ROCKET model family, such as the short transformation time, the arbitrary convolutional kernel approach and the few descriptive features per kernel. On top of that, LightWaveS adds well-studied signal theory, multi-node[1] distribution and smart feature selection to address the drawbacks identified above. The trade-off that we propose compared to (MINI)ROCKET is clear: we take advantage of additional computational resources during training time to keep the short duration and also save time during inference. Our contributions with LightWaveS are:

- We introduce the concept of wavelet scattering based on arbitrary kernels
- We achieve accuracy comparable to state-of-the-art in the majority of the UEA datasets (Bagnall et al., 2018), using only a fraction of the number of features used by (MINI)ROCKET
- We distribute the framework to achieve training time comparable to (MINI)ROCKET on 1 node, and shorter on multiple nodes
- We achieve linear scaling of training time with the number of channels, tested in experiments with up to 900 channels
- Depending on the dataset size, we achieve good speedup with additional nodes, tested in experiments with up to 8 nodes
- We achieve inference speedups ranging from 8x to 30x compared to (MINI)ROCKET on an edge device in datasets with comparable accuracy

With this work, we take an efficiency-centric approach, focusing on the practicality and the inference speed of a model that may be deployed on an edge device, without necessarily trying to surpass the state-of-the-art in terms of accuracy.

2 Related Work

2.1 Multivariate time series classification

Due to the recent rise in popularity of the deep learning field, there is a multitude of models that can be easily adapted to incorporate the additional dimension of MTSC (Ismail Fawaz et al., 2019). A detailed evaluation of recent MTSC methods appears in Ruiz et al. (2021). Convolutions in particular, either 1-D or with more dimensions, are a very popular module when dealing with time series, as seen in models such as TapNet (Zhang et al., 2020), InceptionTime (Fawaz et al., 2020) and OS-CNN (Tang et al., 2021) among others. In these works, convolutions are combined in

[1] With the term *node* we refer to a computing node in a distributed (networked) system.

various architectures with other modules, such as fully connected networks, and achieve impressive classification accuracy. The majority of those deep learning models takes a significant amount of time and memory to train even on GPU nodes, ranging from hours to even days, depending on the dataset size (Ruiz et al., 2021). In contrast, our method takes less than 15 minutes to transform, train and test all 30 UEA datasets on a single CPU node with a linear classifier. If we distribute the solution on e.g. 8 CPU nodes, this time drops to under 4 minutes. An additional challenge is the interpretability of deep learning models. The usage of wavelets by LightWaveS and their method of application gives an advantage to interpreting the result based on signal theory, allows to incorporate domain expert knowledge and helps to extract useful conclusions about a given time series problem.

2.2 Feature extraction and selection

Deep learning models implicitly extract features from the input along their first layers. In contrast, explicitly extracting features to use with classifiers has also been visited in multiple ways. We have already described how the *ROCKET family extracts features from the convolutions of random kernels with the input, an idea that has also been explored before as noted in Dempster et al. (2019). Another example is tsfresh (Christ et al., 2018), which extracts a large number of predefined statistical features from the time series and then through feature selection reduces them to the most useful. Similarly, catch22 (Lubba et al., 2019) is a solution that uses only 22 predefined characteristics to transform time series, aiming for a very fast transformation. A different approach is presented in WEASEL+MUSE (Schäfer & Leser, 2018), which extracts features based on a bag-of-patterns approach and selects the most useful based on a χ^2 test. LightWaveS, similarly to MINIROCKET, depends on an arbitrary set of convolutions and only 4 statistical features, extracted however from the coefficients of wavelet scattering. Due to the speed of the convolution operation and the simplicity of the features, we can achieve extremely fast training and inference times, while giving the model enough complexity to accommodate difficult datasets, where predefined statistical features on the raw time series values may not be descriptive enough.

2.3 Distributed training

As the size of the deep learning models and the amount of input data increase, it is increasingly difficult to perform training on a single node. For that reason, distribution of the training process across multiple nodes is becoming increasingly necessary (Mayer & Jacobsen, 2020). This distribution can be implemented with methods such as data parallelism, where each node applies the same operations on different parts of the input data, with frequent communication among the nodes to update the model. Distribution can also be applied on solutions that do not depend on deep learning, such as tsfresh mentioned above, which supports operation on a cluster through Dask(Dask Development Team, 2016). LightWaveS is distributed using MPI in a data-parallel way, but in contrast to the communication heavy training of DL models, there is efficient and minimal communication between the worker nodes and the central coordinator.

2.4 Wavelets

Wavelets are localized waveforms (as seen in Figure 1 on the right) and are a well-studied method in signal processing that have been used extensively in the analysis of time series (Mallat, 1999). There is vast literature with methods and applications of wavelets on all types of problems, ranging from healthcare to audio analysis, and well-studied and developed families of wavelets suitable for specific applications (Addison, 2017; Merry & Steinbuch, 2005). There are also numerous approaches that combine wavelets with machine and deep learning methods, either as implicit or explicit feature extractors (Li et al., 2021; Wang et al., 2018). A seminal work is Bruna & Mallat (2013), where the concept of a wavelet scattering network using a Morlet wavelet is introduced, in combination with linear and support vector machine classifiers for hand-written digit classification and texture recognition. This method was constructed to be invariant to translations of the input and stable to small deformations. It has also been recently used in combination with deep learning networks for specific applications (Soro & Lee, 2019; Jin & Duan, 2020).

LightWaveS aims to combine the strong points of these works under a single generalized framework, with a focus on efficiency. We aim to bridge the gap between ROCKET and the wavelet theory, and we progress to the next logical step of wavelet scattering. We keep this approach lightweight, both

in depth and paths of the scattering, so we can apply it to time series channels on a massive scale in a very short time. The arbitrary base set of wavelets can be extended based on expert opinion, backed by the solid theory behind wavelets and their applications, making LightWaveS a suitable platform for experimentation on solutions for MTSC problems. Finally, the hierarchical feature filtering leads to the most relevant output features of the scattering coefficients being selected.

3 PROPOSED FRAMEWORK

3.1 PRELIMINARIES

3.1.1 MINIROCKET FUNDAMENTALS

(MINI)ROCKET is primarily based on the convolution operation, in which a kernel $\boldsymbol{k}$ with size l ($\boldsymbol{k} \in \mathbb{R}^l$), bias b and dilation factor d is used to calculate a sliding dot product with a 1-D input $\boldsymbol{x} \in \mathbb{R}^L$ of size L and produce an output $\boldsymbol{x}'$, where each element is calculated as (Dempster et al., 2019):

$$\boldsymbol{x}'_i = \sum_{j=0}^{l-1} \boldsymbol{k}_j \cdot \boldsymbol{x}_{i+d*j} + b \tag{1}$$

MINIROCKET generates a large number (10000 by default) of those kernels. The weights are selected from an empirically chosen subset of 84 kernels of length 9 with weights in $\{2, -1\}$. This has been chosen to limit the possible weight combinations but is not unique for the purpose, and other lengths, different values or weights drawn from $\sim \mathcal{N}(0, 1)$ are equally effective (Dempster et al., 2020). The only thing that is important is that the kernel weights have sum 0. The bias is drawn from the convolution output, and the dilation is drawn from the set $\mathbb{D} = \{\lfloor 2^e \rfloor\}$, where e$\sim \mathcal{U}(0, max)$, with max such that the length of the dilated kernel does not exceed that of the input. Finally, half of the kernels use padding, which appends zeros to either side of the input so that $\boldsymbol{x}'$ and $\boldsymbol{x}$ have the same dimension. The feature extracted from each of the 10000 outputs is the percentage of positive values, and is used for the final classification.

3.1.2 WAVELET SCATTERING

Wavelet scattering is the process of applying wavelet transforms in a cascading manner (Bruna & Mallat, 2013), combined with non-linearities and pooling. The wavelet transform is a method used to approximate a signal using a set of wavelets which originate from a "mother" wavelet $\Psi(t)$, scaled by s and shifted by b (Young, 2012). Each such wavelet can be described as:

$$\Psi_{s,b}(t) = \frac{1}{\sqrt{s}} \Psi(\frac{t-b}{s}) \tag{2}$$

Intuitively, we can relate these wavelets to the convolution filters that we discussed above, with the dilation being the scale of the wavelets (how "narrow" or "wide" they are), and the shift parameter b corresponding to the starting point of the convolution on the input ($i + d * j$). We can see an example of a wavelet convolution in Figure 1, where the response is strong at the points where the sliding wavelet "matches" the signal. In addition, the wavelet set from a mother wavelet is parallel to the way that MINIROCKET has a fixed set of kernel weights, for which different paddings and dilations are randomly selected, generating the "child" kernels.

In the wavelet scattering transform, on each level λ, the previous result is convoluted with each wavelet ψ_{λ_n} (kernel) and a complex modulus operator is applied before propagating the result to the next level, such that:

$$U[\lambda] = |U[\lambda - 1] * \psi_{\lambda_n}| \tag{3}$$

and the scattering coefficients that result from each level are

$$S[\lambda] = U[\lambda] * \phi \tag{4}$$

where ϕ is an averaging kernel. Since on every level of the scattering transform there can be multiple candidate wavelets (kernels), there is a geometric progression of potential paths, as can be seen in the graphic representation of the process in Figure 1.

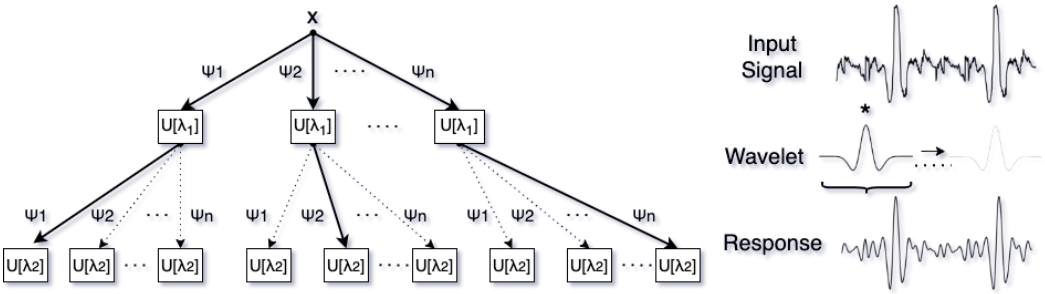


Figure 1: 2-level wavelet scattering (Bruna & Mallat, 2013) and wavelet convolution example

Throughout the above intuitive description, we connected the concepts of MINIROCKET with the wavelet theory. In MINIROCKET the prototype kernels have zero mean, which corresponds to a desirable constraint of mother wavelets (Young, 2012). Under this new context, we can reconsider MINIROCKET as a classifier based on convolutions with random child wavelets of an arbitrary set of mother wavelets. In addition, although not being wavelet scattering, MINIROCKET can be intuitively mapped on the first level, with its kernel set equivalent to the first level set, $\{\Psi_{0,1,s}, ..., \Psi_{0,n,s}\}$, with s drawn from $\mathbb{D}$ as we said above. Since each kernel is slid across the input, we can think of b as taking all discrete values from 0 to $l_{input} - l_{kernel} * s$ for each candidate wavelet, and the convolution output (response) as being the combined response of all shifted wavelets, as portrayed in Figure 1. However, MINIROCKET extracts the features before the application of any modulus operator, so it does not satisfy the rest of the wavelet scattering transform requirements.

3.2 Algorithm

Based on the above observations, we improve the approach and reach the crux of LightWaveS: Lightweight Wavelet Scattering based on random wavelets. The term lightweight refers both to the system optimizations of the framework that make it fast, such as the distribution, as well as the fact that not all paths of the wavelet scattering are considered, for computational efficiency. Specifically, we consider only the paths shown in bold in Figure 1. We generate and utilize the same set of kernels for both levels, so the bold paths mean that the same kernel is applied twice consecutively. In addition, we limit the depth to two layers, as a good balance point between computational speed and informative coefficients, as observed in Bruna & Mallat (2013).

Algorithm 1: LightWaveS distributed algorithm

Data: Input $\boldsymbol{X} \in \mathbb{R}^{NxCxL}$, Labels $\boldsymbol{y}$, Samples N, Channels C, Rank R in MPI WORLD, S nodes, Feature target size F, Feature pool size FP
Result: Transformed $\boldsymbol{X}' \in \mathbb{R}^{NxF}$

$\boldsymbol{X}_{node} \leftarrow GetDataSlice(S, R, C)$;
$\mathbb{K}_{node} \leftarrow GenerateKernels(S, R, C)$;
$\boldsymbol{X}'_{node} \leftarrow WaveletScatteringFeatures(\boldsymbol{X}_{node}, \mathbb{K}_{node})$;
$\boldsymbol{f}_{topnode} \leftarrow TopFeatureSelection(\boldsymbol{X}'_{node}, \boldsymbol{y}, FP)$;
if *R is 0* **then**
  $\boldsymbol{f}_{topglobal} \leftarrow GatherTopFeatures(FP)$;
  $\boldsymbol{X}_{top} \leftarrow FeatureTransform(\boldsymbol{X}, \boldsymbol{f}_{topglobal})$;
  $\boldsymbol{f}_{final} \leftarrow MRMRFeatureSelection(\boldsymbol{X}_{top}, \boldsymbol{y}, F)$;
  $\boldsymbol{X}' \leftarrow FeatureTransform(\boldsymbol{X}, \boldsymbol{f}_{final})$;
end

The main steps of LightWaveS appear in Algorithm 1. Initially, the number of channels is split among the nodes. Then the kernels are generated, which for our purpose are based on the same subset of 84 kernels that MINIROCKET uses. We limit randomness even more, so each kernel is

dilated with all dilations in the set $\{2^0, 2^1, ..., 2^{max}\}$. We also completely remove bias, and the padding is common to all kernels so that the output dimension is equal to the input.

All those kernels are applied to each of the input channels according to the wavelet scattering process described above. The generated kernels have no complex part, so the non-linearity modulus operator is equal to the absolute value of the convolution output. We also downsample the $U[\lambda 1]$ result by a factor of 2 before propagating it, while keeping the same scale for the kernel, so it operates in lower frequencies. This is according to the insight in Bruna & Mallat (2013), that not all paths are necessary and the most useful ones are frequency decreasing. As far as the pooling is concerned, since we want to limit the number of coefficients that will be used as features, we use max and min pooling, instead of the averaging kernel. In this way, although we accept the loss of more information, we have fewer features and we also save two additional convolutions per path. The features that are extracted from each scattering path are 8: the first four are the max and min value of $U[\lambda 1]$ and $U[\lambda 2]$. We selected those as the most straightforward non-linearities that can be quickly calculated during the convolution process. The other four are the percentage of positive values (ppv) and normalized longer sequence of positive values (ls) in $U[\lambda 1]$ and $U[\lambda 2]$ before the application of the modulus operator. We keep these features based on MINIROCKET and MultiROCKET respectively, since they have proved to be useful during classification.

After the feature extraction, the first selection phase is performed in a supervised way using ANOVA. The F-values are calculated on different subsets of the data, and the average score is taken, making the method more robust (Li et al., 2017). The main node gathers the top FP features and performs the final feature selection, using the minimum-redundancy, maximum-relevance algorithm (Peng et al., 2005), incrementally selecting the feature with the highest score. This score is determined by its F-statistic (relevance to the class), divided by its average correlation to the previously selected features (redundancy). In our case, we use the Spearman correlation coefficient as the correlation indicator. In both phases the feature selection methods are filter based, which are fast, classifier-independent and can be implemented to scale well with the number of features (Li et al., 2017). The design choices that are not guided or restricted by theory, such as the feature scoring function, are straightforward and focused on computational efficiency. We also performed empirical experimentation on a development subset, although without exhaustive parameter tuning of the method.

4 EXPERIMENTS

4.1 DATASETS

We select as benchmark the UEA collection of multivariate datasets (Bagnall et al., 2018), excluding *InsectWingbeat*, since due to its large size it presented issues when training ROCKET. The datasets are described in detail in Ruiz et al. (2021). Following the example of Dempster et al. (2019), we selected 15 of the 30 UEA datasets to work on when developing the method, in order to draw generic conclusions and avoid overfitting the whole collection. In addition, we prepare and use five machinery related datasets:

- **MAFAULDA**, from Machinery Fault Database (MAF) is a dataset with 8 sensor measurements on a machine fault simulator, taken under normal conditions and five different fault types. We downsample the measurements so that the input length is 1000 steps and we split the dataset in train and test with ratio 85-15 %.
- **TURBOFAN** (Saxena & Goebel, 2008), is an engine degradation simulation dataset collection, containing 4 datasets with operation simulations of engines that run until failure under different conditions, with measurements from 26 sensors. The goal is to predict the remaining useful life (RUL) of the engines. In order to turn the problem into binary classification, we prepare the dataset and the labels in a suitable way with the goal being classifying RUL as more or less than 20 operational cycles.

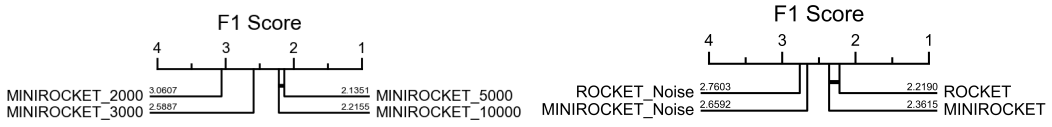
4.2 EXPERIMENTAL SETUP

All training experiments were run on nodes which have dual 8-core 2.4 GHz (Intel Haswell E5-2630-v3) CPUs and 64GB of RAM. The inference experiments are executed on a Jetson Xavier board which has an 8-core ARM CPU. We ran (MINI)ROCKET on the 29 UEA datasets and the 5

additional datasets using the default number of features (20 and 10 thousand respectively). As for LightWaveS, we present three variants of the model, termed L1,L2 and L1L2. These versions refer to keeping the features only from the scattering level 1, level 2 or both, although we consider the L1L2 version as the default. We select 500 features as the default variant, which has shown good balance between training time, inference speed and accuracy in the development set. We use a Ridge regression classifier from the Scikit-learn package (Pedregosa et al., 2011) for all methods. We use a critical difference diagram to present the results, a popular method of comparing classifier performance across multiple datasets (Ruiz et al., 2021; Ismail Fawaz et al., 2019). This shows the ranking of the classifiers and groups the ones that do not show statistical difference, based on pairwise comparisons using a Wilcoxon signed-rank test (Wilcoxon, 1992) with Holm's alpha correction (Holm, 1979; Garcia & Herrera, 2008). The grouped classifiers appear on the diagram connected by a thick horizontal line.

4.3 (MINI)ROCKET ANALYSIS

Using the *sktime* repository version of the code, we can see in Figure 2a that even keeping the top half features of MINIROCKET using an unsophisticated chi-squared test does not result in a statistically significant difference in the classifier performance (rather, it slightly increases the rank). As for the channel inclusion, as an extreme case, we can add noise channels to the UEA datasets (10% of the original channel number) and we see that both solutions are affected negatively when trained on them, belonging in a statistically different group.



(a) Selection of fractions of the default number of features

(b) Training on normal and noisy versions of the UEA dataset

Figure 2: Mean ranks of (MINI)ROCKET under different experiments, in terms of F1 score

4.4 ACCURACY RESULTS

We present the performance of LightWaveS in terms of accuracy in comparison with (MINI)ROCKET, as well as other recent solutions, namely (M)OSS-CNN (Tang et al., 2021), WEASEL+MUSE (Schäfer & Leser, 2018), TapNet (Zhang et al., 2020) and Catch22 (Lubba et al., 2019). Apart from LightWaveS and *ROCKET, the rest of the accuracy metrics are taken from the repositories of Tang et al. (2021)[2] and Dhariyal et al. (2020)[3]. Due to missing values in those metrics, we manage to compare the methods on 28 of the 30 UEA datasets.

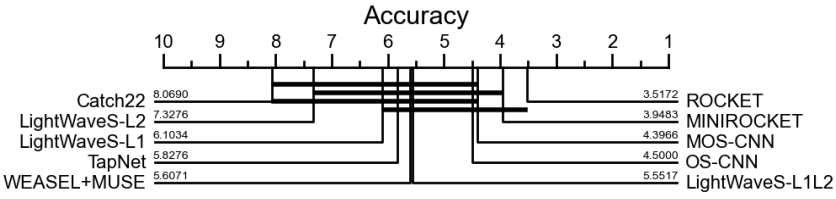


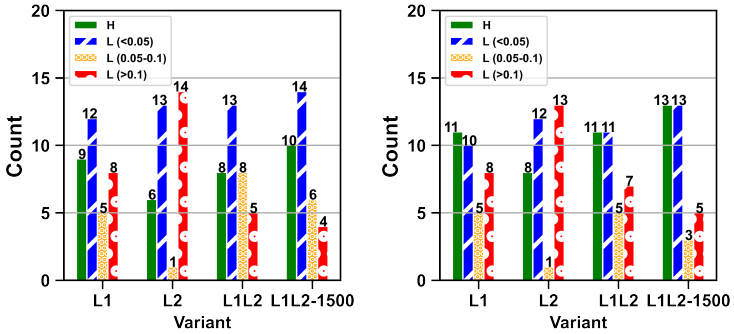
Figure 3: Mean rank of LightWaveS methods vs recent classifiers in terms of accuracy

In Figure 3 we can see the results. Although lower in rank, LightWaveS belongs in the same statistical group with (MINI)ROCKET, which have the best accuracy. In addition, it is among the ranks of more complex DL methods, such as TapNet and MOS-CNN. Out of the three variants, LightWaveS-L2 performs the worst, without the benefit of faster execution that L1 has. We

[2] https://github.com/Wensi-Tang/OS-CNN

[3] https://github.com/mlgig/mtsc_benchmark/

can focus on (MINI)ROCKET for a more detailed comparison, and also include the 5 additional datasets. Since the aim of LightWaveS is to approach their state-of-the-art accuracy with fewer features, not necessarily surpass it, we place the LightWaveS results in four accuracy bins compared to (MINI)ROCKET : higher or equal, lower within 0.05, lower within 0.05-0.10 and lower more than 0.1. We see in Figure 4 that in the majority of the datasets the accuracy stays in the first 3 categories for all variants apart from L2, and the datasets with large accuracy deviation are few.



(a) ROCKET (b) MINIROCKET

Figure 4: Distribution of average accuracy of LightWaveS variants compared with (MINI)ROCKET

In addition, we can see that just increasing the number of features to 1500 leads to significant improvement of the comparison results, leaving potential for successfully tackling the harder datasets.

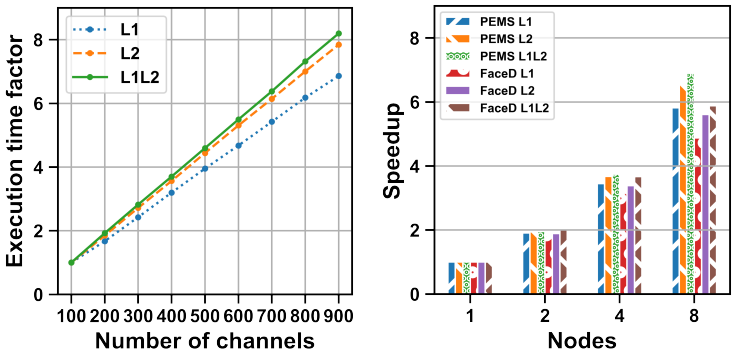
4.5 Channel and node scaling of training time

As a point of reference, ROCKET processes the whole UEA set (apart from *InsectWingbeat*) on a single node in approximately 9 minutes, MINIROCKET in 5 and LightWaveS (L1L2) in 14. (MINI)ROCKET has complexity $\mathcal{O}(kernels \cdot samples \cdot tslength)$. Since LightWaveS uses one more convolution per kernel it adds a fixed factor of two to the complexity, essentially keeping linear scalability with these variables. Thus, we can focus on the scalability of the LightWaveS training time with the additional channel dimension and with the number of distribution nodes.

We select *PEMS-SF* from UEA which has 963 channels, and we increment the channels by 100, starting from 100, training LightWaveS on 2 nodes. We see in Figure 5a that all variants show (sub)linear scaling with the number of channels, which is expected since each additional channel results in a fixed amount of additional convolutions, and there are no combinations or permutations of channels. For the node scaling, we also select *FaceDetection*, with 144 channels. As we see in Figure 5b, with PEMS-SF the gained training speedup is more close to ideal, since there are sufficient channels for each node to work on. In contrast to that, with FaceDetection that has fewer channels, the computational cost starts being dominated by the operations on the main node (application of kernels, feature selection, final transformation), so the gained speedup falls more quickly.

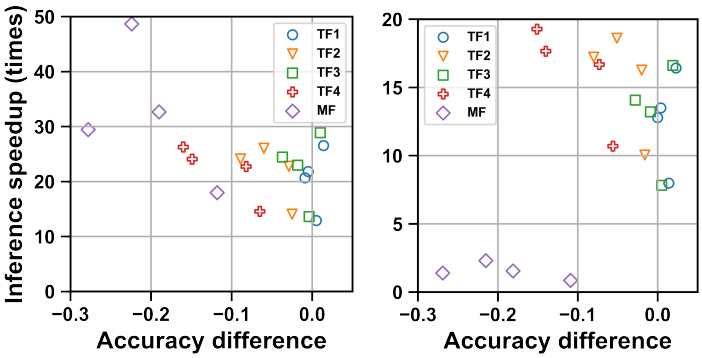
4.6 Inference results

We manage to achieve significant speedup during inference due to the small amount of features, which heavily reduces the amount of convolutions required to transform a test sample. We perform a qualitative analysis, focusing on measuring the inference on the machinery datasets, since they could correspond to a scenario of edge deployment of a model for e.g. fault detection of a machine. However, our conclusions hold for all datasets. We can see in Figure 6 the relation between the accuracy difference of the LightWaveS variants with (MINI)ROCKET and the speedup achieved. This figure frames clearly the trade-off that we propose: it is up to the end user to select their preferred balance of speedup and accuracy, in the cases where no ideal case is available. The MF dataset in the MINIROCKET comparison acts as an outlier in the graph due to its large length in combination with the optimizations of MINIROCKET. Our method of applying the kernels matches ROCKET's, but could also benefit from these optimizations, restoring the discrepancy.



(a) Channel scalability (b) Node scalability

Figure 5: Scalability of LightWaveS traing time with the number of (a) channels and (b) nodes



(a) ROCKET (b) MINIROCKET

Figure 6: Inference speedup-accuracy difference scatterplot for the 5 machinery datasets, for each of the LightWaveS variants (note the different y axis scale)

5 Discussions and Conclusion

Summarizing, LightWaveS offers comparable to state-of-the-art accuracy on the datasets tested, with training time similar to (MINI)ROCKET and inference time 8x-30x faster on an edge device. Apart from the advantages we have mentioned, a significant advantage of LightWaveS is that it is based on wavelet theory, which has a strong theoretical base. A promising future direction is to explore expert tuning of the framework, by preparing and including in the base set well tested wavelets, with different properties such as padding or dilations depending on the use case. Another secondary benefit is that due to the feature selection, LightWaveS can filter the channels to a subset of the originals. In the UEA, this reduction ranged from 0 to 94%, with an average of 20% across all datasets. This has multiple advantages: it can give insights into which channels contain useful information for the problem, leading to knowledge extraction. On edge devices, where resources are valuable, it can free up incoming signal channels. Finally, LightWaveS can act as an initial fast channel filtering method that precedes another deep learning solution, reducing the training data required. LightWaveS can also benefit from, and is indeed orthogonal to, recent works on feature selection, since additional descriptive features can be used on the scattering coefficients, improving accuracy. Future work can include an initial, more informed selection of the wavelets, so that the wavelet scattering network can extend to more paths with wavelet combinations, as well as explicit, informed combinations of different channels.

REPRODUCIBILITY STATEMENT

The code of LightWaveS, as well as the code used to train (MINI)ROCKET, run the inference experiments and preprocess the machinery datasets is provided as supplementary material. All accuracy and time experiments were repeated multiple times (30-100), with different seeds and the average of metrics was taken for a fairer representation of the results. All datasets used are publicly available. There has been an effort to explicitly set the random seed for all methods that accept such a parameter, in order to enable reproducibility of the results as closely as possible.

REFERENCES

Machinery Fault Database [Online]. http://www02.smt.ufrj.br/˜offshore/mfs/page_01.html.

Paul S Addison. *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance*. CRC press, 2017.

Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017. doi: 10.1007/s10618-016-0483-9.

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The UEA multivariate time series classification archive, 2018. *arXiv:1811.00075 [cs, stat]*, 2018.

Joan Bruna and S. Mallat. Invariant Scattering Convolution Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013. doi: 10.1109/TPAMI.2012.230.

Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing*, 2018. doi: 10.1016/j.neucom.2018.03.067.

Dask Development Team. *Dask: Library for dynamic task scheduling*, 2016. URL `https://dask.org`.

Angus Dempster, François Petitjean, and Geoffrey I. Webb. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *arXiv:1910.13051 [cs, stat]*, 2019.

Angus Dempster, Daniel F. Schmidt, and Geoffrey I. Webb. MINIROCKET: A Very Fast (Almost) Deterministic Transform for Time Series Classification. *arXiv:2012.08791 [cs, stat]*, 2020.

Bhaskar Dhariyal, Thach le Nguyen, Severin Gsponer, and Georgiana Ifrim. An Examination of the State-of-the-Art for Multivariate Time Series Classification. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pp. 243–250, Sorrento, Italy, 2020. IEEE. ISBN 978-1-72819-012-9. doi: 10.1109/ICDMW51313.2020.00042.

Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.

Salvador Garcia and Francisco Herrera. An extension on" statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of machine learning research*, 9(12), 2008.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pp. 65–70, 1979.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: A review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019. doi: 10.1007/s10618-019-00619-1.

Yang Jin and Yunling Duan. Wavelet scattering network-based machine learning for ground penetrating radar imaging: Application in pipeline identification. *Remote Sensing*, 12(21):3655, 2020.

Tianfu Li, Zhibin Zhao, Chuang Sun, Li Cheng, Xuefeng Chen, Ruqiang Yan, and Robert X. Gao. WaveletKernelNet: An Interpretable Deep Neural Network for Industrial Intelligent Diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–11, 2021. doi: 10.1109/TSMC.2020.3048950.

Yun Li, Tao Li, and Huan Liu. Recent advances in feature selection and its applications. *Knowledge and Information Systems*, 53(3):551–577, 2017. doi: 10.1007/s10115-017-1059-8.

Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J. Király. Sktime: A Unified Interface for Machine Learning with Time Series. *arXiv:1909.07872 [cs, stat]*, 2019.

Carl H Lubba, Sarab S Sethi, Philip Knaute, Simon R Schultz, Ben D Fulcher, and Nick S Jones. catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, 33(6): 1821–1852, 2019.

Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

Ruben Mayer and Hans-Arno Jacobsen. Scalable deep learning on distributed infrastructures: Challenges, techniques, and tools. *ACM Computing Surveys (CSUR)*, 53(1):1–37, 2020.

RJE Merry and M Steinbuch. Wavelet theory and applications. *literature study, Eindhoven university of technology, Department of mechanical engineering, Control systems technology group*, 2005.

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*, 12:2825–2830, 2011.

Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2):401–449, 2021. doi: 10.1007/s10618-020-00727-3.

A. Saxena and K. Goebel. Turbofan engine degradation simulation data set. http://ti.arc.nasa.gov/project/prognostic-data-repository, 2008.

Patrick Schäfer and Ulf Leser. Multivariate Time Series Classification with WEASEL+MUSE. *arXiv:1711.11343 [cs]*, 2018.

Bedionita Soro and Chaewoo Lee. A wavelet scattering feature extraction approach for deep neural network based indoor fingerprinting localization. *Sensors*, 19(8):1790, 2019.

Chang Wei Tan, Angus Dempster, Christoph Bergmeir, and Geoffrey I. Webb. MultiRocket: Effective summary statistics for convolutional outputs in time series classification. *arXiv:2102.00457 [cs, stat]*, 2021.

Wensi Tang, Guodong Long, Lu Liu, Tianyi Zhou, Jing Jiang, and Michael Blumenstein. Rethinking 1D-CNN for Time Series Classification: A Stronger Baseline. *arXiv:2002.10061 [cs, stat]*, 2021.

Jingyuan Wang, Ze Wang, Jianfeng Li, and Junjie Wu. Multilevel Wavelet Decomposition Network for Interpretable Time Series Analysis. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2437–2446, London United Kingdom, 2018. ACM. ISBN 978-1-4503-5552-0. doi: 10.1145/3219819.3220060.

Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pp. 196–202. Springer, 1992.

Randy K Young. *Wavelet theory and its applications*, volume 189. Springer Science & Business Media, 2012.

Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. Tapnet: Multivariate time series classi fication with attentional prototypical network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6845–6852, 2020.