A Deep Dive into Alternatives to the Global Average Pooling for Time Series Classification

Cyril Meyer$^{1[0000-0001-7262-999X]}$ (✉), Ali Ismail-Fawaz$^{1[0000-0001-5385-3339]}$, Maxime Devanne$^{1[0000-0002-1458-3855]}$, Jonathan Weber$^{1[0000-0002-3694-4703]}$, and Germain Forestier$^{1,2[0000-0002-4960-7554]}$

[1] Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France
`{first-name.last-name}@uha.fr`
[2] DSAI, Monash University, Melbourne, Australia
`{first-name.last-name}@monash.edu`

Abstract. Global Average Pooling (GAP) has become a standard aggregation method in deep learning models for Time Series Classification (TSC), yet its effectiveness has recently been questioned by the research community. In this work, we conduct an extensive empirical investigation into the validity of GAP as an aggregation mechanism by comparing it to a diverse set of alternative methods. These include pooling-based, feature-based, and learnable aggregation techniques, evaluated across two well-established univariate (UCR) and multivariate (UEA) TSC benchmarks. Our results reveal that GAP remains highly competitive, consistently achieving strong classification performance with minimal computational overhead. Importantly, none of the alternative methods were able to statistically significantly outperform GAP, either in terms of accuracy or efficiency. Furthermore, we show that parametrized and complex aggregators, such as those based on Recurrent Neural Networks, often degrade performance, reinforcing the principle that simpler, non-parametric methods like GAP are not only sufficient but often preferable. This study reaffirms GAP as a robust and efficient choice for aggregation in deep neural networks for TSC tasks. All of our experimental results and source code are publicly available to ensure the reproducibility of our work and also to allow the community to use the raw results for further research.

Keywords: Time Series Classification · Deep Learning · Pooling Layer.

1 Introduction

Time series classification (TSC) [17, 8, 16] is an important task across domains such as finance, healthcare, and environmental monitoring. Deep learning models have emerged as effective solutions for TSC [12]. Although there are differences among current state-of-the-art approaches, they generally share a common architecture based on convolutional operations. In these models, reducing the dimensionality of intermediate representations is often necessary, typically achieved through aggregation layers. These layers convert variable-length sequences or high-dimensional feature maps into fixed-size representations suitable for downstream tasks.

The most common approach to this problem is Global Average Pooling (GAP), proposed in [15], which computes the mean value across a specific axis, often spatial or temporal. GAP is widely used because it enables explainability with methods such as Class Activation Maps [20]. Although GAP is simple and effective, researchers often question its utility and advantages for TSC tasks [14]. This is because GAP assumes that all temporal features contribute equally to the final decision.

However, a broader range of aggregation methods exists in the literature and can be applied to TSC tasks. We categorize them into three groups: (1) pooling-based, (2) feature-based, and (3) learning-based methods. Pooling-based methods include GAP as well as alternatives such as the Global Max Pooling (GMP). Additionally, static temporal versions of these methods apply pooling locally over temporal segments rather than globally. Feature-based methods have been primarily used in ROCKET [6] and its variants [18, 7]. These methods focus on extracting statistical features from the activations of convolution operations. Three types of features are considered: Proportion of Positive Values (PPV), Mean of Positive Values (MPV), and Mean Index of Positive Values (MIPV). We also consider three smoothed variants of these methods in our study. Finally, learning-based methods involve the use of Recurrent Neural Networks (RNNs) and their variants. RNNs remove the assumption of equal contribution across all time steps and instead learn aggregation representations through weighted accumulation over time. Some recent approaches, such as learnable dynamic temporal pooling [14], also fall into this category, enabling adaptive, data-driven pooling mechanisms.

While these alternatives have been used in various contexts, there is limited work that systematically compares their effectiveness within the same architectural backbone and experimental setup.

This paper presents an experimental study focused on the impact of alternatives to the GAP layer in TSC models. By evaluating a diverse set of aggregation techniques, we aim to shed light on how these layers influence classification accuracy and model robustness. To validate our findings, we use 128 univariate TSC datasets from the UCR archive [5] and 26 multivariate TSC datasets from the UEA archive [1].

2 Aggregation layers

In this section, we present the selected variety of aggregation strategies. We organize the aggregation layers into three categories, as mentioned in the previous section. Each category offers a different trade-off between simplicity, expressiveness, and computational cost. In the following subsections, we describe each category in detail and present the specific variants used in our experiments. To ensure consistency and clarity across all formulations, we use the following notations throughout this section:

- K: the number of feature channels (dimensions) in the input representation.
- $k \in \{1, \ldots, K\}$: index over the feature channels.
- T: the length of the time series.
- $t \in \{1, \ldots, T\}$: index over the time steps.
- $x_{k,t}$: the value at time step t in the k-th feature channel.
- T_k^+: the number of positive elements in the sequence $\{x_{k,1}, x_{k,2}, \ldots, x_{k,T}\}$.

2.1 Pooling-Based Methods

The **Global Average Pooling** (GAP) layer averages all values along the time dimension for each channel separately.

$$\text{GAP} = \left[\frac{1}{T} \sum_{t=1}^{T} x_{k,t} \right]_{k=1}^{K} \tag{1}$$

The **Global Max Pooling** (GMP) layer operates similarly to GAP, however instead of computing the average across the time dimension, it selects the maximum value.

$$\text{GMP} = \left[\max_{t=1}^{T} x_{k,t} \right]_{k=1}^{K} \tag{2}$$

Moreover, the **Static Temporal Average Pooling** (STAP) layer averages values over predefined segments of the time dimension, rather than across the entire temporal span such as GAP. This method divides the time series into equal-length segments and computes the average within each segment.

$$\text{STAP} = \left[\left[\frac{1}{S} \sum_{t=(i-1)\times S}^{i\times S} x_{k,t} \right]_{i=1}^{N} \right]_{k=1}^{K}, \tag{3}$$

where, N is the number of pooling operations and S is both the pool size and the stride of the pooling operation calculated as:

$$S = \left\lceil \frac{T}{N} \right\rceil$$

Whenever $\left\lceil \frac{T}{S} \right\rceil \leq N - 1$ (i.e., when T is too small to yield N equally-spaced pools), we reduce N to $N' = \left\lceil \frac{T}{S} \right\rceil$ and reshape the pooled output accordingly. This ensures that the layer remains valid even for small T relative to N.

The **Static Temporal Max Pooling** (STMP) layer, similarly to STAP, applies a maximum operation over the same predefined segments of the time dimension.

$$\text{STMP} = \left[\left[\max_{t=(i-1)\times S}^{i\times S} x_{k,t} \right]_{i=1}^{N} \right]_{k=1}^{K} \tag{4}$$

2.2 Feature-Based Methods

For feature-based methods, we examine techniques employed in the ROCKET models [6, 7, 18].
The **Proportion of Positive Values** (PPV) discards the actual values of the series and instead focuses on the number of positive values:

$$\text{PPV} = \left[\frac{1}{T} \sum_{\substack{t=1 \\ x_{k,t}>0}}^{T} 1 \right]_{k=1}^{K} = \left[\frac{T_k^+}{T} \right]_{k=1}^{K} \tag{5}$$

This simple yet effective measure provides insight into the overall positivity trend of the data. PPV was first introduced in Rocket [6] and was found to be an exceptional feature extractor for MiniRocket [7] and is still used in MultiRocket [18].
The **Mean of Positive Values** (MPV), unlike GAP, applies the average only over the set of positive values instead of all possible values:

$$\text{MPV} = \left[\frac{1}{T_k^+} \sum_{\substack{t=1 \\ x_{k,t}>0}}^{T} x_{k,t} \right]_{k=1}^{K} \tag{6}$$

By focusing only on positive values, this layer captures the magnitude of the positive trend, offering a different perspective compared to GAP.
The **Mean of Indices of Positive Values** (MIPV) captures information about the relative location of positive values. MIPV is computed by averaging the relative location (indices) of all positive values in the array:

$$\text{MIPV} = \left[\begin{cases} \dfrac{1}{T_k^+} \displaystyle\sum_{\substack{t=1 \\ x_{k,t}>0}}^{T} t-1 & \text{if } T_k^+ > 0 \\ 0 & \text{if } T_k^+ = 0 \end{cases} \right]_{k=1}^{K} \tag{7}$$

In the case where there are no positive values, ($T_k^+ = 0$), MIPV results in 0. This definition differs from the original, in which the result in this specific case will be -1[18]. The difference is motivated by the need to maintain consistency with the approach used in the smooth MIPV, which will be discussed subsequently.

Building upon the feature extraction techniques previously discussed, we introduce a **smooth threshold** alternative for each layer. This approach employs a parametrizable sigmoid function to replace the traditional threshold. Given that the preceding layer's output, which utilizes a ReLU activation, lies within the range $[0, +\infty[$, we incorporate a positive shifting constant to effectively manage this range. The parameters are fixed as described in Equation 8, with the behaviour illustrated in Figure 1.

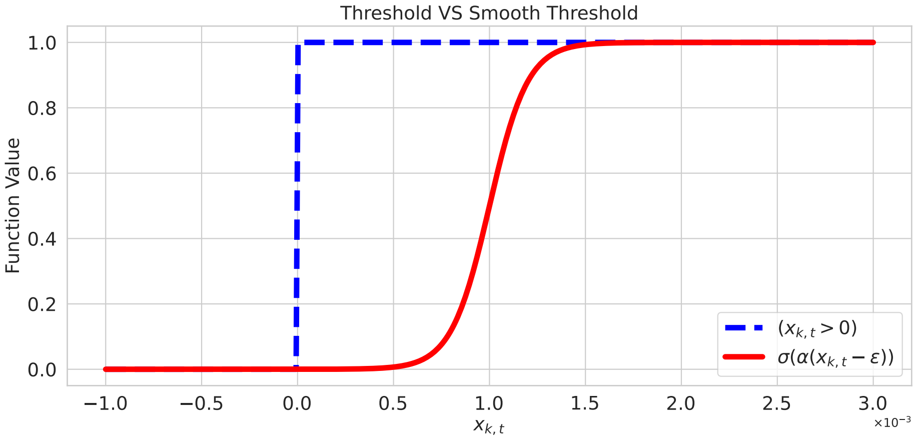


Fig. 1. Comparison of hard and soft thresholding using sigmoid approximation.

$$\sim (x_{k,t} > 0) = \sigma\left(\alpha\left(x_{k,t} - \epsilon\right)\right) \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function, defined as $\sigma(z) = \frac{1}{1+e^{-z}}$, α a hardness scaling factor (set as $\alpha = 10^4$) and ϵ a positive shifting constant (set as $\epsilon = 10^{-3}$)

We utilize the smooth threshold in feature-based aggregation methods. For instance, the smoothed version of PPV, denoted as $\sim$ PPV, is computed as:

$$\sim \mathrm{PPV} = \left[\frac{1}{T}\sum_{t=1}^{T}\sigma\left(\alpha\left(x_{k,t} - \epsilon\right)\right)\right]_{k=1}^{K} \tag{9}$$

Similarly, we utilize the smooth threshold in both MPV and MIPV and referr to them as $\sim$ MPV and $\sim$ MIPV, respectively.

Additionally, we denote the usage of the Straight-Through Estimator (STE) by the symbol "$\uparrow$", indicating a custom gradient that bypasses the operations in the backward pass while preserving forward behaviour. In this work, we apply STE only to the PPV variant, denoted as $\uparrow$ PPV. The approach follows the formulation introduced by Bengio et al. [2], in which the backward path avoids discontinuous functions as identity mappings to enable end-to-end training despite their non-differentiability.

2.3 Learning-Based Methods

Finally, to also explore aggregation methods capturing more complex temporal dependencies within the time series data, we consider a **Gated Recurrent Unit** (GRU) layer [3]. GRUs, introduced in 2014, are a type of Recurrent Neural Network (RNN) architecture. We refer the reader to the original work in [3] for further details about the functionality of GRUs.

3 Experimental Evaluation

In this section, we present the experimental setup and results obtained from evaluating the various aggregation layers discussed in the previous sections. The primary objective of our experiments is to assess the effectiveness of the aggregation layers in the context of TSC.

3.1 Experimental Setup

We conducted our experiments on two well-established benchmarks in the field of time series classification (TSC). For univariate TSC tasks, we used the 128 datasets from the UCR archive [4], and for multivariate TSC tasks, we used the 26 datasets from the UEA archive [1].

For each dataset, we implement a consistent experimental protocol to ensure fair and comparable results. We systematically evaluate each aggregation layer by integrating it into state-of-the-art deep learning models, trained with common hyperparameters. This approach allows us to isolate the impact of each aggregation layer on the classification performance. We compare the effect of the aggregation layers on 4 different convolution-based architectures, FCN [19], ResNet [19], Inception [13] and LITE [10] where for Inception and LITE we use a single model rather than an ensemble of models. For multivariate time series, we use LITEMV [11], the multivariate counterpart of LITE. The hyperparameter setup used for all our experiments is presented in Table 1.

Hyperparameter	**Value / Description**
Batch size	$\min\left(\frac{\lvert D\rvert}{10},\ 16\right)$ where $\lvert D\rvert$ is the number of training samples
Epochs	2 000
Model selection	Best model selected based on training set performance
Learning rate schedule	Reduce on plateau: factor $= \frac{1}{2}$ patience $= 50$ $\min_{LR} = 10^{-4}$

Table 1. Training hyperparameters and settings.

Additionally, we explore combinations of different aggregation layers to investigate potential complementary effects that could enhance models accuracy. With a comparison of 20 possible reducing methods or combinations, we end up with 80 comparable cases.

Each of these cases is trained and tested individually 11 times on the 128 UCR and 26 UEA datasets. We perform 11 trainings for more robust results. For the comparison, we keep the median accuracy, which, in opposition to an average, guarantees to correspond to a real performance. In total, 135 520 models ($11 \times 20 \times 4 \times (128+26)$) have been trained and evaluated to obtain the results presented in the following. In addition to the classification performance, we also compare the number of parameters and training time for each case. Our source code and raw results are available at https://github.com/MSD-IRIMAS/PoolParty-4-TSC.

3.2 Experimental Results

In this section, we present and analyse the experimental results obtained from evaluating the aggregation layers. In particular, we designed our experiments to compare the performance of GAP, the most used aggregation layer, against alternative methods.

GAP vs STAP First, we compare GAP and STAP to elucidate the differences in performance when using a global approach versus a segmented, static temporal approach for averaging time series features. In particular, we aim to understand how the temporal segmentation impacts the classification accuracy and the model's ability to capture local temporal features. We also assess the combination of both pooling methods.

The Multi Comparison Matrices (MCMs) [9] representing the comparative results are shown in Figure 2. We can notice that for Inception and ResNet models, the GAP layer gives the best performance on both UCR and UEA datasets. For smaller architectures like LITE and FCN, the STAP with 2 and 4 windows shows good results, but not significantly better than the GAP. Finally, the combination of both GAP and STAP often results in lower performance than a single pooling. However, focusing on LITE STAP cases, specifically STAP2 and STAP4, we notice that STAP2 wins on 47 datasets and STAP4 on 52, where 39 of them are shared across the two cases. This indicates a regularity in the performance improvement using those alternatives.

GAP vs Max Pooling Second, we compare the global and static temporal max pooling. We also include "GAP GMP" cases which combine both aggregation layers. The MCMs are in Figure 3. As before, GAP is still better on almost all datasets.

GAP vs Feature-Based Third, we focus on comparing GAP with feature-based methods. We individually compare each feature-based method, including its smoothed versions ($\sim$) and $\upharpoonright$ PPV. The results are shown in Figure 4. We can notice that none of the selected feature-based methods perform statistically significantly better than GAP. However, it is important to note that the smoothed versions improve the results for PPV and that $\upharpoonright$ PPV significantly outperforms both.

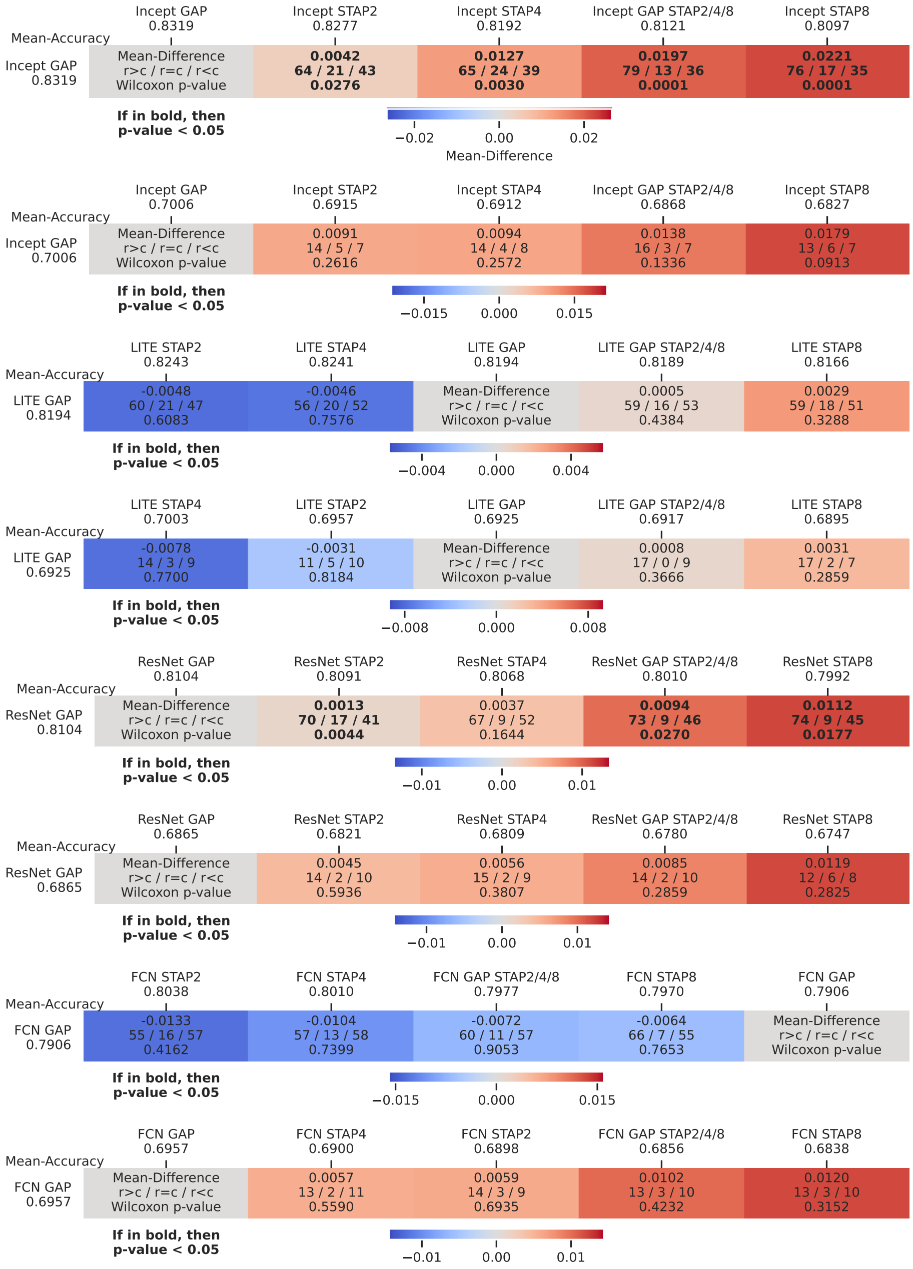
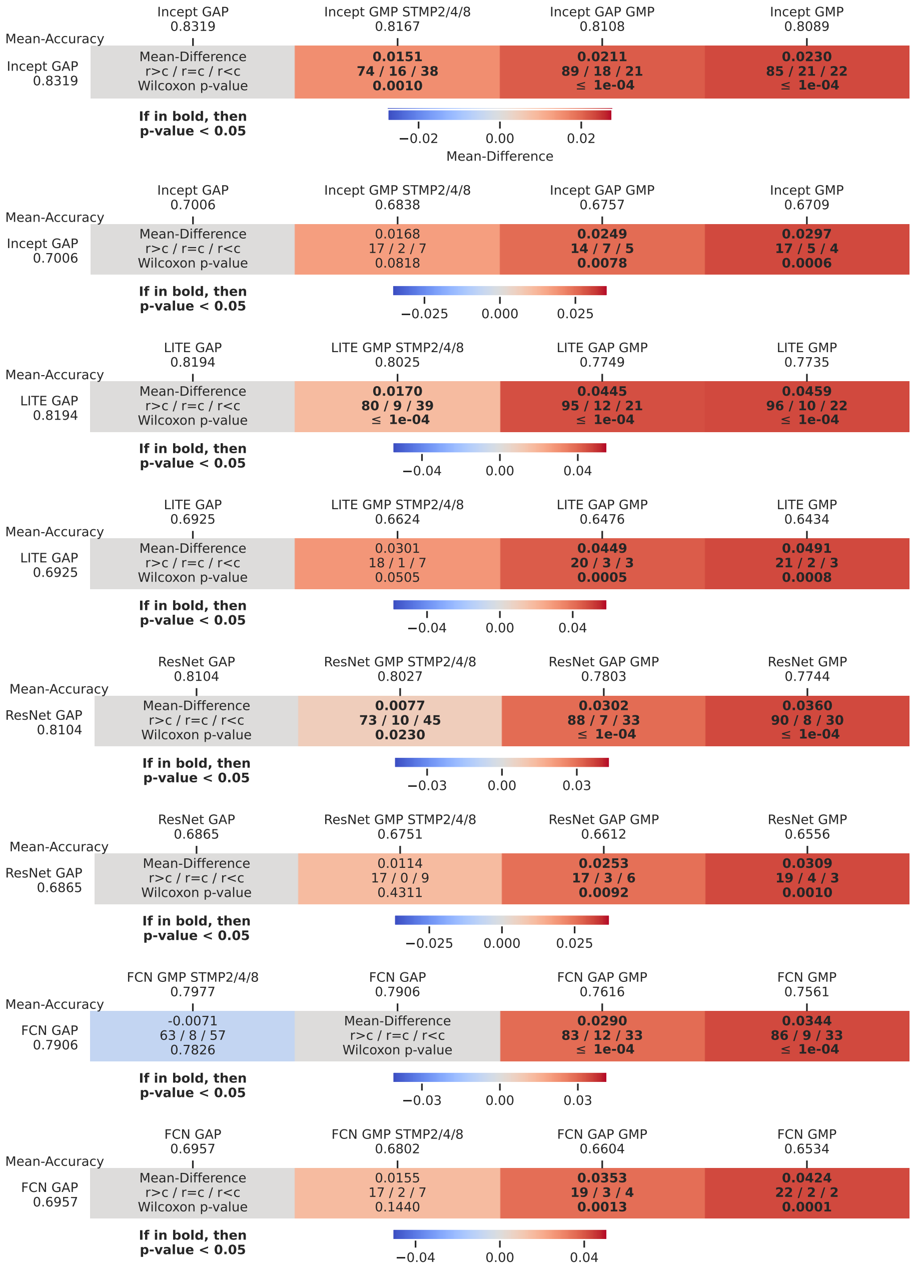


Fig. 2. Comparison of Static Temporal Average Pooling with Global Average Pooling on the UCR and UEA for the four selected models. From top to bottom : Inception UCR, Inception UEA, LITE UCR, LITE UEA, ResNet UCR, ResNet UEA, FCN UCR, FCN UEA.



Mean-Accuracy	Incept GAP 0.8319	Incept GMP STMP2/4/8 0.8167	Incept GAP GMP 0.8108	Incept GMP 0.8089
Incept GAP 0.8319	Mean-Difference r>c / r=c / r<c Wilcoxon p-value	**0.0151** **74 / 16 / 38** **0.0010**	**0.0211** **89 / 18 / 21** **≤ 1e-04**	**0.0230** **85 / 21 / 22** **≤ 1e-04**

If in bold, then p-value < 0.05

Mean-Accuracy	Incept GAP 0.7006	Incept GMP STMP2/4/8 0.6838	Incept GAP GMP 0.6757	Incept GMP 0.6709
Incept GAP 0.7006	Mean-Difference r>c / r=c / r<c Wilcoxon p-value	0.0168 17 / 2 / 7 0.0818	**0.0249** **14 / 7 / 5** **0.0078**	**0.0297** **17 / 5 / 4** **0.0006**

If in bold, then p-value < 0.05

Mean-Accuracy	LITE GAP 0.8194	LITE GMP STMP2/4/8 0.8025	LITE GAP GMP 0.7749	LITE GMP 0.7735
LITE GAP 0.8194	Mean-Difference r>c / r=c / r<c Wilcoxon p-value	**0.0170** **80 / 9 / 39** **≤ 1e-04**	**0.0445** **95 / 12 / 21** **≤ 1e-04**	**0.0459** **96 / 10 / 22** **≤ 1e-04**

If in bold, then p-value < 0.05

Mean-Accuracy	LITE GAP 0.6925	LITE GMP STMP2/4/8 0.6624	LITE GAP GMP 0.6476	LITE GMP 0.6434
LITE GAP 0.6925	Mean-Difference r>c / r=c / r<c Wilcoxon p-value	0.0301 18 / 1 / 7 0.0505	**0.0449** **20 / 3 / 3** **0.0005**	**0.0491** **21 / 2 / 3** **0.0008**

If in bold, then p-value < 0.05

Mean-Accuracy	ResNet GAP 0.8104	ResNet GMP STMP2/4/8 0.8027	ResNet GAP GMP 0.7803	ResNet GMP 0.7744
ResNet GAP 0.8104	Mean-Difference r>c / r=c / r<c Wilcoxon p-value	**0.0077** **73 / 10 / 45** **0.0230**	**0.0302** **88 / 7 / 33** **≤ 1e-04**	**0.0360** **90 / 8 / 30** **≤ 1e-04**

If in bold, then p-value < 0.05

Mean-Accuracy	ResNet GAP 0.6865	ResNet GMP STMP2/4/8 0.6751	ResNet GAP GMP 0.6612	ResNet GMP 0.6556
ResNet GAP 0.6865	Mean-Difference r>c / r=c / r<c Wilcoxon p-value	0.0114 17 / 0 / 9 0.4311	**0.0253** **17 / 3 / 6** **0.0092**	**0.0309** **19 / 4 / 3** **0.0010**

If in bold, then p-value < 0.05

Mean-Accuracy	FCN GMP STMP2/4/8 0.7977	FCN GAP 0.7906	FCN GAP GMP 0.7616	FCN GMP 0.7561
FCN GAP 0.7906	-0.0071 63 / 8 / 57 0.7826	Mean-Difference r>c / r=c / r<c Wilcoxon p-value	**0.0290** **83 / 12 / 33** **≤ 1e-04**	**0.0344** **86 / 9 / 33** **≤ 1e-04**

If in bold, then p-value < 0.05

Mean-Accuracy	FCN GAP 0.6957	FCN GMP STMP2/4/8 0.6802	FCN GAP GMP 0.6604	FCN GMP 0.6534
FCN GAP 0.6957	Mean-Difference r>c / r=c / r<c Wilcoxon p-value	0.0155 17 / 2 / 7 0.1440	**0.0353** **19 / 3 / 4** **0.0013**	**0.0424** **22 / 2 / 2** **0.0001**

If in bold, then p-value < 0.05

Fig. 3. Comparison of Global Max Pooling and Static Temporal Max Pooling with Global Average Pooling on the UCR and UEA for the four selected models. From top to bottom : Inception UCR, Inception UEA, LITE UCR, LITE UEA, ResNet UCR, ResNet UEA, FCN UCR, FCN UEA.

Given these results, we selected the best resulting layers, MPV and ↑ PPV and combined them with GAP. We also evaluated a ↑ PPV MPV MIPV combination, to be as close as possible to the MultiROCKET [18] framework. Those results are shown in Figure 5. Even though some examples outperform GAP, still no conclusion can be found on the statistically significant difference in performance with GAP.

GAP vs GRU Finally, we present the results of using GRU layers compared to GAP in the MCMs shown in Figure 6. These results reveal a significant degradation in performance when using GRU, compared to non-learnable methods such as GAP. This suggests that the use of recurrent architectures may not be well suited for this task, potentially due to their complexity, or that aggregation techniques should, in most cases, be non-learnable.

3.3 Efficiency Comparison

We present an efficiency comparison to explore the trade-off between performance and computational cost. For our experiments, we used a High Performance Computing (HPC) to handle the computational load. However, to maintain consistency and control over the variables affecting training time measurements, the timing experiments were conducted on a single computer setup. The computer setup consists of an RTX 4090 GPU and an i9-14900KF CPU, running Windows 11 with the Windows Subsystem for Linux 2 (WSL2). This configuration allowed us to accurately measure and compare the training times across different models and layer options. Moreover, in order to limit the number of trainings on a single computer, we chose to reduce the number of datasets used in order to obtain an average train time. In total, we use only 32 datasets for the efficiency comparison. We present this comparison for both number of parameters and training time in Figure 7. This comparison shows that there is almost no significant difference in model complexity when using non-learnable aggregation techniques. However, this is not the case with learnable layers such as GRU, where training time increases dramatically.

3.4 Discussion

Throughout our experiments, GAP consistently demonstrated superior or equal performance compared to other methods. These results underscore the effectiveness of GAP in capturing the essential features of the data while reducing dimensionality. The ability of GAP to summarize spatial information into a compact representation likely contributed to its robust performance across various tasks.

Interestingly, our results indicate that using GAP alone often yields better results than combining GAP with additional methods. These findings suggest that the simplicity and efficiency of GAP might be compromised when integrated

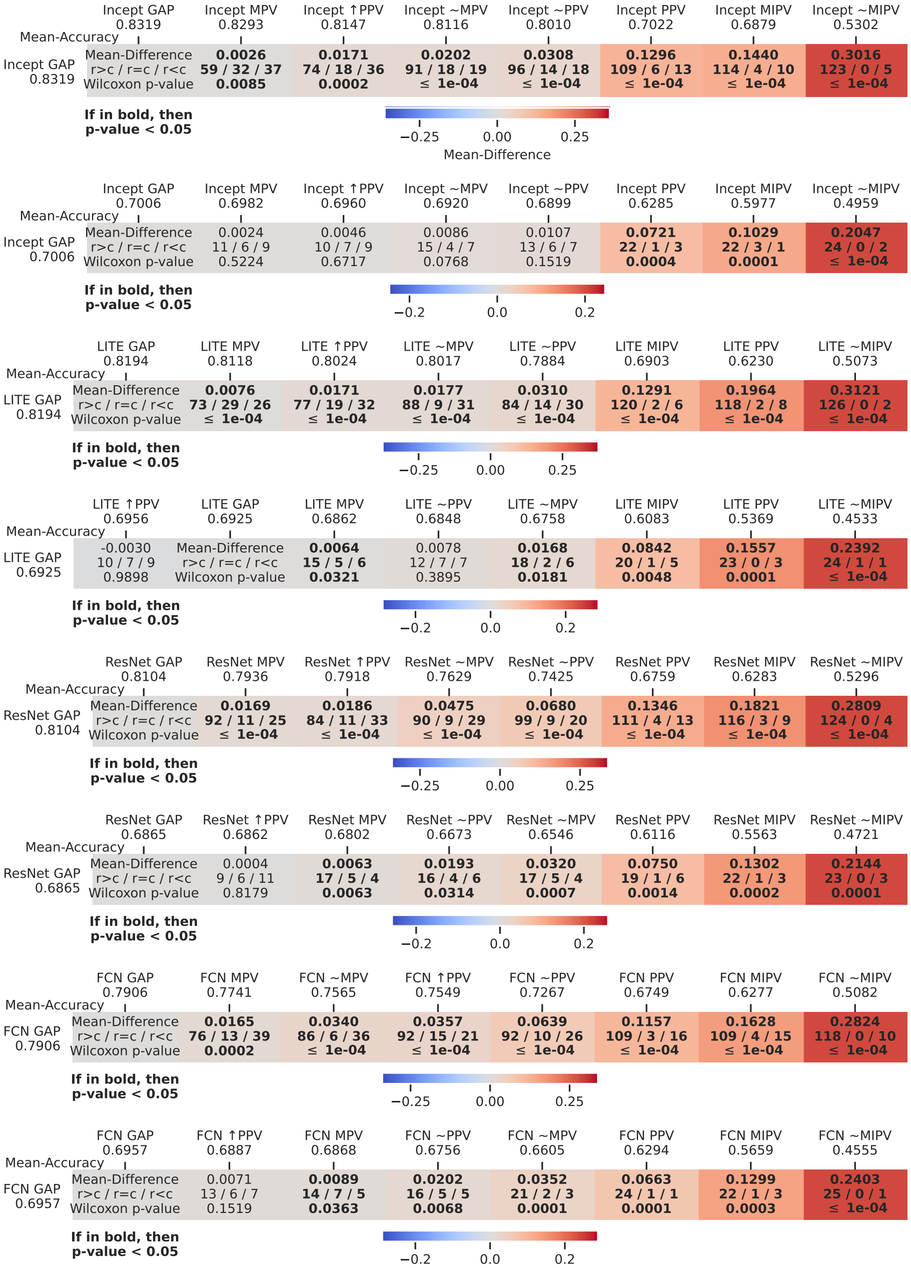


Fig. 4. Comparison of ROCKET inspired feature extraction layers with Global Average Pooling on the UCR and UEA for the four selected models. From top to bottom : Inception UCR, Inception UEA, LITE UCR, LITE UEA, ResNet UCR, ResNet UEA, FCN UCR, FCN UEA.

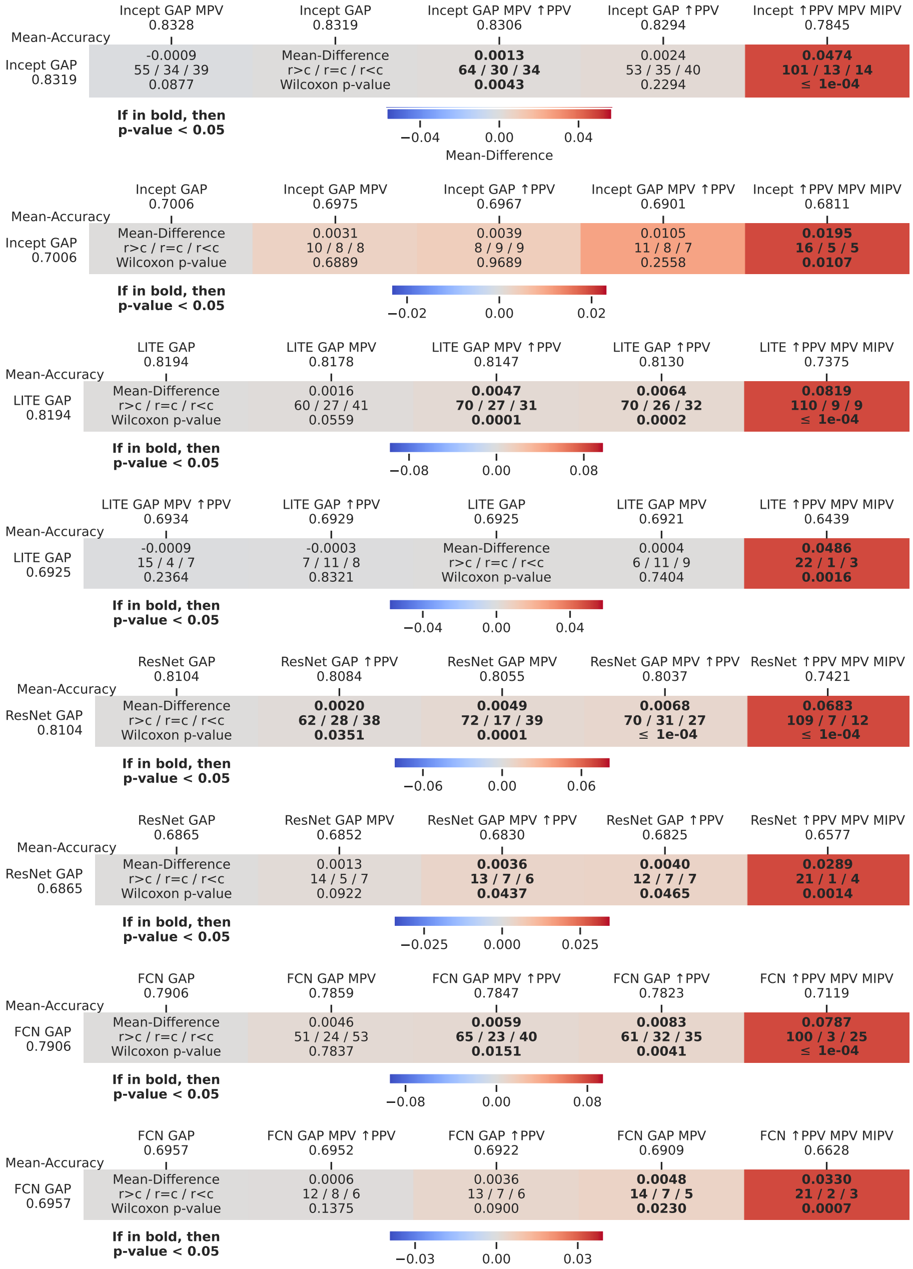
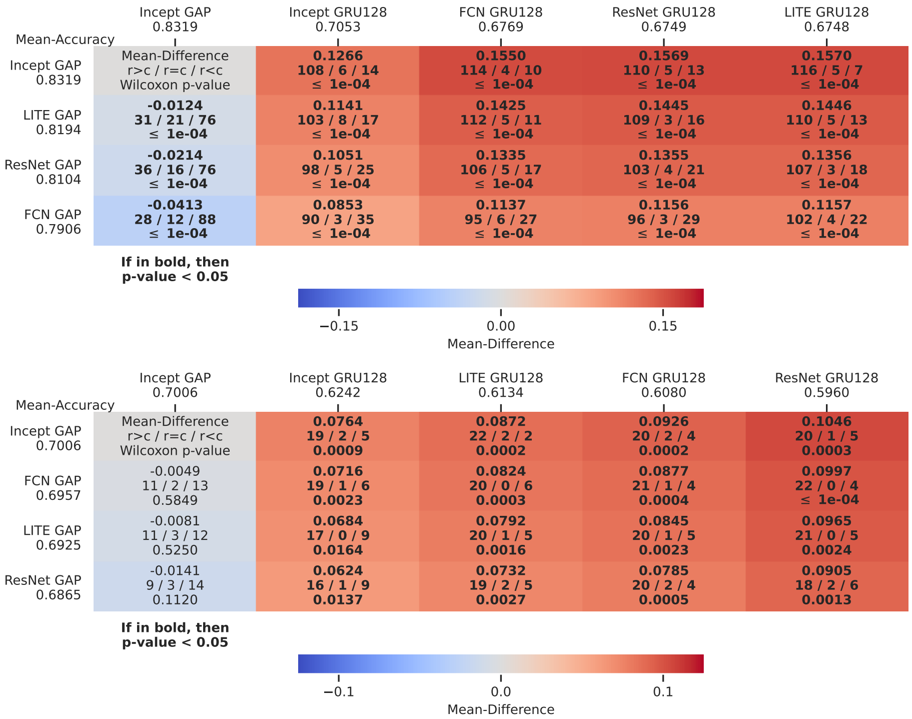


Fig. 5. Comparison of ROCKET inspired feature extraction layers in addition to GAP or mixed together with GAP on the UCR and UEA for the four selected models. From top to bottom : Inception UCR, Inception UEA, LITE UCR, LITE UEA, ResNet UCR, ResNet UEA, FCN UCR, FCN UEA.



Mean-Accuracy	Incept GAP 0.8319	Incept GRU128 0.7053	FCN GRU128 0.6769	ResNet GRU128 0.6749	LITE GRU128 0.6748
Incept GAP 0.8319	Mean-Difference r>c / r=c / r<c Wilcoxon p-value	**0.1266** **108 / 6 / 14** **≤ 1e-04**	**0.1550** **114 / 4 / 10** **≤ 1e-04**	**0.1569** **110 / 5 / 13** **≤ 1e-04**	**0.1570** **116 / 5 / 7** **≤ 1e-04**
LITE GAP 0.8194	**-0.0124** **31 / 21 / 76** **≤ 1e-04**	**0.1141** **103 / 8 / 17** **≤ 1e-04**	**0.1425** **112 / 5 / 11** **≤ 1e-04**	**0.1445** **109 / 3 / 16** **≤ 1e-04**	**0.1446** **110 / 5 / 13** **≤ 1e-04**
ResNet GAP 0.8104	**-0.0214** **36 / 16 / 76** **≤ 1e-04**	**0.1051** **98 / 5 / 25** **≤ 1e-04**	**0.1335** **106 / 5 / 17** **≤ 1e-04**	**0.1355** **103 / 4 / 21** **≤ 1e-04**	**0.1356** **107 / 3 / 18** **≤ 1e-04**
FCN GAP 0.7906	**-0.0413** **28 / 12 / 88** **≤ 1e-04**	**0.0853** **90 / 3 / 35** **≤ 1e-04**	**0.1137** **95 / 6 / 27** **≤ 1e-04**	**0.1156** **96 / 3 / 29** **≤ 1e-04**	**0.1157** **102 / 4 / 22** **≤ 1e-04**

If in bold, then p-value < 0.05

Mean-Accuracy	Incept GAP 0.7006	Incept GRU128 0.6242	LITE GRU128 0.6134	FCN GRU128 0.6080	ResNet GRU128 0.5960
Incept GAP 0.7006	Mean-Difference r>c / r=c / r<c Wilcoxon p-value	**0.0764** **19 / 2 / 5** **0.0009**	**0.0872** **22 / 2 / 2** **0.0002**	**0.0926** **20 / 2 / 4** **0.0002**	**0.1046** **20 / 1 / 5** **0.0003**
FCN GAP 0.6957	-0.0049 11 / 2 / 13 0.5849	**0.0716** **19 / 1 / 6** **0.0023**	**0.0824** **20 / 0 / 6** **0.0003**	**0.0877** **21 / 1 / 4** **0.0004**	**0.0997** **22 / 0 / 4** **≤ 1e-04**
LITE GAP 0.6925	-0.0081 11 / 3 / 12 0.5250	**0.0684** **17 / 0 / 9** **0.0164**	**0.0792** **20 / 1 / 5** **0.0016**	**0.0845** **20 / 1 / 5** **0.0023**	**0.0965** **21 / 0 / 5** **0.0024**
ResNet GAP 0.6865	-0.0141 9 / 3 / 14 0.1120	**0.0624** **16 / 1 / 9** **0.0137**	**0.0732** **19 / 2 / 5** **0.0027**	**0.0785** **20 / 2 / 4** **0.0005**	**0.0905** **18 / 2 / 6** **0.0013**

If in bold, then p-value < 0.05

Fig. 6. Comparison of Gated Recurrent Unit with 128 units with Global Average Pooling on the UCR and UEA for the four selected models.

with more complex aggregation methods. The additional methods could introduce noise or redundancy, thereby diminishing the overall performance. This insight is interesting for model design, as it highlights the importance of simplicity and the potential drawbacks of unnecessarily complex architectures.

Additionally, in contrast to the strong results of GAP, the performance of GRU is notably poor across all datasets. GRUs, which are typically valued for their ability to capture sequential dependencies, do not perform well in our specific context. This outcome might be attributed to the nature of our data or the tasks at hand, which potentially do not benefit from the sequential processing strengths of GRUs. This finding suggests that, while GRUs can be powerful in certain scenarios, they may not be universally applicable, especially on datasets like the UCR and UEA.

4 Conclusion

In this work, we address ongoing concerns within the research community regarding the use of Global Average Pooling (GAP) as an aggregation method in deep

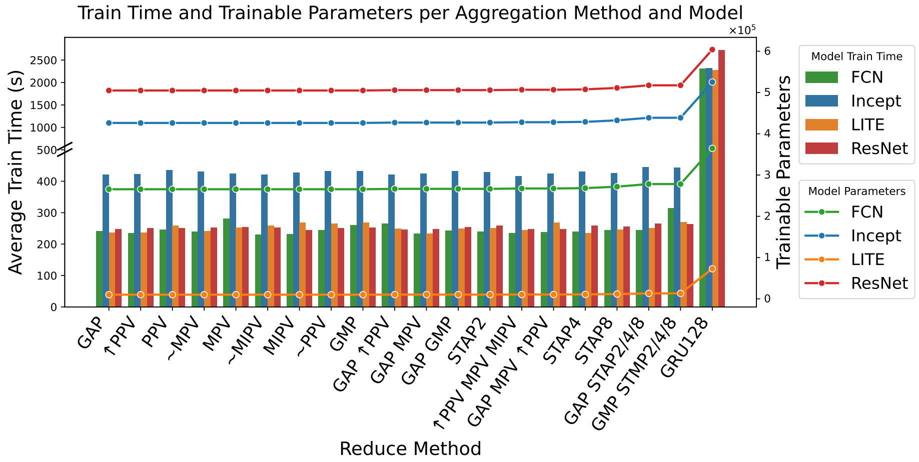


Fig. 7. Training time (bars) and trainable parameters (points) for each aggregation method and model.

neural networks for Time Series Classification (TSC) tasks. We present comprehensive experiments comparing GAP with both existing and newly adapted aggregation techniques, evaluated on two well-established univariate and multivariate TSC benchmarks. The alternative aggregation methods are categorized into three groups: pooling-based, feature-based, and learning-based approaches.

Our empirical results demonstrate that GAP remains the most accurate and efficient aggregation strategy, as no other method achieves statistically significant improvements over GAP in terms of both average classification performance and computational efficiency. In addition, we observe that STAP yields promising results specifically on smaller architectures such as LITE and FCN, highlighting its potential in resource constrained environments. This study underscores that GAP, a non-parametric, training-free, and computationally lightweight mechanism, is a fundamental component of effective deep learning models for TSC. Notably, complex learnable aggregators, such as Recurrent Neural Networks (RNNs), perform poorly, reinforcing the principle that in the context of TSC, simpler methods like GAP can indeed be superior.

As a final contribution, in an effort to promote transparency and support future work, all of our experimental results and source code are publicly available https://github.com/MSD-IRIMAS/PoolParty-4-TSC. This ensures the reproducibility of our study and allows the community to use the raw results for extended research.

Acknowledgments. We acknowledge the High Performance Computing Center of the University of Strasbourg for supporting this work by providing scientific support and access to computing resources. Part of the computing resources were funded by the Equipex Equip@Meso project (Programme Investissements d'Avenir) and the CPER

Alsacalcul/Big Data. This work was supported by the ANR DELEGATION project (grant ANR-21-CE23-0014) of the French Agence Nationale de la Recherche.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bagnall, A., Dau, H.A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., Keogh, E.: The UEA multivariate time series classification archive, 2018 (Oct 2018). https://doi.org/10.48550/arXiv.1811.00075
2. Bengio, Y., Léonard, N., Courville, A.: Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation (Aug 2013). https://doi.org/10.48550/arXiv.1308.3432
3. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: Moschitti, A., Pang, B., Daelemans, W. (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (Oct 2014). https://doi.org/10.3115/v1/D14-1179
4. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The UCR time series archive. IEEE/CAA Journal of Automatica Sinica **6**(6), 1293–1305 (Nov 2019). https://doi.org/10.1109/JAS.2019.1911747
5. Dau, H.A., Keogh, E., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G., Hexagon-ML: The UCR time series classification archive (Oct 2018)
6. Dempster, A., Petitjean, F., Webb, G.I.: ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery **34**(5), 1454–1495 (Sep 2020). https://doi.org/10.1007/s10618-020-00701-z
7. Dempster, A., Schmidt, D.F., Webb, G.I.: MiniRocket: A Very Fast (Almost) Deterministic Transform for Time Series Classification. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 248–257. KDD '21, Association for Computing Machinery, New York, NY, USA (Aug 2021). https://doi.org/10.1145/3447548.3467231
8. Dempster, A., Tan, C.W., Miller, L., Foumani, N.M., Schmidt, D.F., Webb, G.I.: Highly Scalable Time Series Classification for Very Large Datasets. In: Lemaire, V., Ifrim, G., Bagnall, A., Guyet, T., Malinowski, S., Schäfer, P., Tavenard, R. (eds.) Advanced Analytics and Learning on Temporal Data. pp. 80–95. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-77066-1_5
9. Ismail-Fawaz, A., Dempster, A., Tan, C.W., Herrmann, M., Miller, L., Schmidt, D.F., Berretti, S., Weber, J., Devanne, M., Forestier, G., et al.: An approach to multiple comparison benchmark evaluations that is stable under manipulation of the comparate set. arXiv preprint arXiv:2305.11921 (2023)
10. Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., Forestier, G.: LITE: Light Inception with boosTing tEchniques for Time Series Classification. In: 2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA). pp. 1–10 (Oct 2023). https://doi.org/10.1109/DSAA60987.2023.10302569

11. Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., Forestier, G.: Look into the LITE in deep learning for time series classification. International Journal of Data Science and Analytics (Jan 2025). https://doi.org/10.1007/s41060-024-00708-5
12. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: A review. Data Mining and Knowledge Discovery **33**(4), 917–963 (Jul 2019). https://doi.org/10.1007/s10618-019-00619-1
13. Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P.A., Petitjean, F.: InceptionTime: Finding AlexNet for time series classification. Data Mining and Knowledge Discovery **34**(6), 1936–1962 (Nov 2020). https://doi.org/10.1007/s10618-020-00710-y
14. Lee, D., Lee, S., Yu, H.: Learnable Dynamic Temporal Pooling for Time Series Classification. Proceedings of the AAAI Conference on Artificial Intelligence **35**(9), 8288–8296 (May 2021). https://doi.org/10.1609/aaai.v35i9.17008
15. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
16. Middlehurst, M., Bagnall, A.: Extracting Features from Random Subseries: A Hybrid Pipeline for Time Series Classification and Extrinsic Regression. In: Ifrim, G., Tavenard, R., Bagnall, A., Schaefer, P., Malinowski, S., Guyet, T., Lemaire, V. (eds.) Advanced Analytics and Learning on Temporal Data. pp. 113–126. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-49896-1_8
17. Middlehurst, M., Schäfer, P., Bagnall, A.: Bake off redux: A review and experimental evaluation of recent time series classification algorithms. Data Mining and Knowledge Discovery **38**(4), 1958–2031 (Jul 2024). https://doi.org/10.1007/s10618-024-01022-1
18. Tan, C.W., Dempster, A., Bergmeir, C., Webb, G.I.: MultiRocket: Multiple pooling operators and transformations for fast and effective time series classification. Data Mining and Knowledge Discovery **36**(5), 1623–1646 (Sep 2022). https://doi.org/10.1007/s10618-022-00844-1
19. Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International Joint Conference on Neural Networks (IJCNN). pp. 1578–1585 (May 2017). https://doi.org/10.1109/IJCNN.2017.7966039
20. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning Deep Features for Discriminative Localization. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2921–2929. IEEE Computer Society (Jun 2016). https://doi.org/10.1109/CVPR.2016.319