LITE: Light Inception with boosTing tEchniques for Time Series Classification

Ali Ismail-Fawaz
IRIMAS, Université de Haute-Alsace
Mulhouse, France
ali-el-hadi.ismail-fawaz@uha.fr

Maxime Devanne
IRIMAS, Université de Haute-Alsace
Mulhouse, France
maxime.devanne@uha.fr

Stefano Berretti
MICC, University of Florence
Florence, Italy
stefano.berretti@unifi.it

Jonathan Weber
IRIMAS, Université de Haute-Alsace
Mulhouse, France
jonathan.weber@uha.fr

Germain Forestier
IRIMAS, Université de Haute-Alsace
Mulhouse, France
DSAI, Monash University
Melbourne, Australia
germain.forestier@uha.fr

***Abstract*—Deep learning models have been shown to be a powerful solution for Time Series Classification (TSC). State-of-the-art architectures, while conducting promising results on the UCR archive, present a high number of trainable parameters. This can lead to long training with a high CO2, Power consumption and possible increase in the number of FLoat-point Operation Per Second (FLOPS). In this paper, we present a new architecture for TSC, the Light Inception with boosTing tEchnique (LITE) with only 2.34% of the state-of-the-art model InceptionTime's number of parameters, while preserving performance. This architecture, with only $9,814$ trainable parameters due to the usage of DepthWise Separable Convolutions (DWSC), is boosted by three techniques: multiplexing, custom filters, and dilated convolution. The LITE architecture, trained on the UCR, is 2.78 times faster than InceptionTime and consumes 2.79 times less CO2 and Power.**

***Index Terms*—Time Series Classification, Neural Networks, Convolutional Neural Networks, DepthWise Separable Convolutions**

I. Introduction

Time Series Classification (TSC) has been widely investigated by researchers in recent years. Some TSC tasks include the classification of surgical evaluation [1]–[3], action recognition of human motion [4], [5], cheat detection in video games [6], interpretability [7], Entomology [8], *etc.*. Thanks to the availability of the UCR archive [9], the largest archive for TSC datasets, a significant amount of work has been done in this domain. Deep learning models have been proposed in the time series context for classification [10]–[14], clustering [15], averaging [16], representation learning [17]–[19], adversarial attacks [20], [21], *etc.*. Even though deep learning approaches proven to be very powerful for TSC, they present a large amount of trainable parameters, which often leads to a long training time, inference time and storage usage.

For this reason, some work started to question the need of such a large complexity in deep learning models for TSC such as ROCKET and its variants [22]–[24]. Like for images, deep learning also presents a large complexity, which limits the usage of the models on small devices such as mobile phones and robots [25], [26]. Furthermore, Large Language Models (LLM) also shown to be very effective [27], and that their complexity can be decreased, while preserving performance [28].

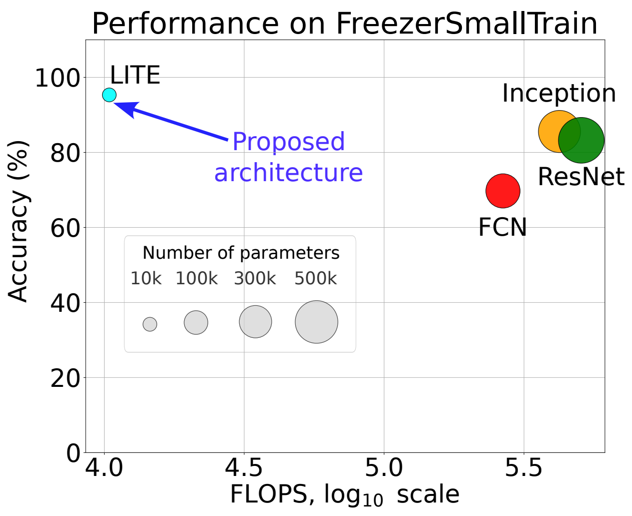

Fig. 1. For each model, the accuracy on the FreezerSmallTrain dataset is presented on the y-axis and the number of FLoat-point Operations Per Second (FLOPS) is presented on the x-axis in $\log_{10}$ scale. The diameter of the circles represents the number of trainable parameters of the model. The smallest model is LITE (**ours**) with only $10k$ trainable parameters and the lowest number of FLOPS (4 in $\log_{10}$ scale); it also presents the highest accuracy score on the test set in this comparison.

In this paper, we address the methodology of reducing the complexity of deep learning models, while preserving the performance of the TSC task. We argue that a large complex model may not be necessary in order to perform well on the UCR archive. However, simply removing layers and or parameters to reduce complexity may not guarantee to preserve the performance. For this reason, the neural network architecture often requires additional techniques in order to balance between complexity and performance. In this work, we borrow existing techniques that have been efficiently used in state-of-the-art architectures on time series data. These techniques are multiplexing convolutions [12], [29], dilated convolutions [19], and custom filters [13]. By combining these three techniques with a modified version of a small non complex model, the Fully Convolution Network (FCN) [11], we propose a new architecture named Light Inception with boosTing tEchniques (**LITE**). The proposed model uses only 2.34% of Inception's number of parameters, while being competitive with state-of-the-art architectures. For instance, Figure 1 shows that on the FreezerSmallTrain dataset, the classification accuracy of **LITE** is much higher than other approaches with way less trainable parameters. The reduction in number of parameters is made possible thanks to the usage of DepthWise Separable Convolutions (DWSC) [25]. The additional techniques used in this proposed architecture, multiplexing, dilated, and custom convolutions, have the advantage of only slightly increasing the number of parameters by about $1,000$.

To position the proposed architecture among the state-of-the-art, we compare not only the accuracy but also the training time and number of parameters. A comparison of CO2 and Power consumption[1] is also presented. The main contributions of this work are:

- A new architecture for TSC, the LITE model with only 2.34% of Inception's number of parameters.
- Extensive experiments showing that LITE achieves state-of-the-art results.
- A comparison based on the number of trainable parameters, number of FLOPS, training time, CO2 and Power consumption.
- A deeper analysis presented as an ablation study to show the impact of each technique added to boost the proposed model.

In what follows, we present some related work in Section II, discuss the details of our proposed architecture in Section III, present some results compared to other approaches in Section IV, and conclude by drawing future work in Section V.

II. Background and Related Work

Time Series Classification (TSC) was widely investigated in the literature. Some work addressed this problem using machine learning algorithms by comparing similarity metrics between the time series [30], decisions based on random forest algorithm [31], *etc.*. The problem of most of those algorithms is that they require huge amount of CPU time to perform their calculations, and can not be parallelized on a cluster of GPUs. For these reasons, deep learning for TSC is being used in the recent years. Even though the performance of deep learning is better than machine learning algorithms, the number of parameters to be optimized may be very high. In what follows, we first define the problem at hand, then we present some work that tackled the TSC problem using machine and deep learning techniques. Finally, we present some work that addressed the training time problem of deep learning models and the large number of parameters.

[1]https://codecarbon.io/

A. Definitions

Let $\mathbf{x}$ be a univariate *time series* of length L, and let $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=0}^{N-1}$ be a *dataset* of N univariate time series $\mathbf{x}_i$ with their corresponding labels y_i. The goal of this work is to construct an algorithm to learn how to correctly classify each input time series to its corresponding label.

B. Machine Learning for TSC

The basic approach to solve TSC tasks is by using the Nearest Neighbour (NN) algorithm. In order to use this algorithm, a specific similarity metric for time series data should be defined. In [30], the authors used the Dynamic Time Warping algorithm (DTW) in order to define a similarity metric for the NN algorithm. This algorithm has some limitations given that DTW does not have the ability to extract features from the input samples. The work in [32] also used the same algorithm but with an upgraded version of DTW called shapeDTW that aligns a local neighborhood around each point. The main limitation of the DTW algorithm and its variants is the time complexity of the algorithm, which is a function of the time series length, *i.e.*, $\mathcal{O}(L^2)$.

C. Deep Learning for TSC

In this section, we present the work done on TSC using deep learning approaches. The simplest architecture is the Multi Layer Perceptron (MLP) proposed in [11] that uses fully connected layers and dropout operations. This architecture is limited given the fact that it ignores the temporal dependency in a time series. The Fully Convolution Network (FCN) was also proposed in [11] that uses 1D convolution operations. In this model, the backpropagation algorithm finds the best filters to extract features from the time series, and correctly classifies the samples. In this model, convolutions account for temporal dependencies in time series data, and they are also independent of the input time series *length*. The authors in [11] also proposed the ResNet model, which uses the residual connections [33] to solve the vanishing gradient problem. A comparative study in [10] shows that using convolutions, especially ResNet, outperforms other models that use multi-scale transformation or pooling layers with convolutions [29], [34]. ResNet and FCN use Batch Normalization and ReLU activation instead of pooling operations after each convolution layer to avoid overfitting. The state-of-the-art model in deep learning for TSC on the UCR archive [9] is InceptionTime [12], where the authors adapted the original Inception model on images for time series data. InceptionTime has the ability to detect multiple patterns of different length given to the multiplexing technique. This technique comes down to learning multiple convolution layers on the same input but with different characteristics. It is important to note that InceptionTime is an ensemble of five Inception models each trained separately.

D. Reducing Model Complexity

Even though deep learning for TSC shown to be very effective, it still has some issues. One of these issues we address in this work is the large number of parameters to be optimized, which increases the training time as well. Recently, in [22] a new approach was proposed, called ROCKET, that also includes convolution operations, but is way faster than InceptionTime. They proposed not to learn few filters with a backpropagation algorithm, but instead randomly generate a large number of filters with different characteristics. These characteristics include the length of the filter, the bias value, the dilation rate, *etc.*. It has been shown that on the UCR archive, no statistical significant difference can be found between InceptionTime and ROCKET. The main advantage of ROCKET compared to InceptionTime is the training and inference time. Some adaptations of ROCKET were proposed in order to boost its performance even more such as MiniROCKET [23] and MultiROCKET [24]. More recently, knowledge distillation [35] was also approached for the TSC model called FCN [36]. In this study, the authors proposed a smaller variant of FCN with a lower number of convolution layers and filters to learn.

Furthermore, the work in [13] proposed to hand-craft some custom convolution filters instead of randomly generate them. Those hand-crafted filters are constructed in a way to get activated on increasing and decreasing intervals as well as peaks in the time series. By using these filters, the authors were able to construct a Hybrid FCN (H-FCN). Results on the UCR archive shown that H-FCN is statistically significantly better than FCN and is competitive with InceptionTime. The H-FCN model uses the custom filters in parallel to the learned filters in the first layer.

Some work to optimize the complexity of large models were proposed in Computer Vision as well. For instance, the authors in [25] proposed the usage of DWSC instead of standard ones. The MobileNet architecture as proposed in [25] proven to be very competitive with state-of-the-art models on ImageNet [37] with way less complexity.

In Natural Language Processing, some work proposed the usage of Small Language Models (SLM) as a one-shot learning approach [28]. The authors showed that with way less parameters than GPT-3 [27], their model can have no significant difference in performance.

III. Proposed approach

A. Convolutions for TSC

Multiple Convolution Neural Networks (CNNs) were proposed for the task of TSC and they all proved how they outperform other methods. The Fully Convolution Network (FCN) is a simple three layered network, where each layer is composed of 1D convolutions followed by a batch normalization and a ReLU activation function. As FCN is only composed of simple 1D convolution layers, its performance generally lags behind more advanced architectures using residual connections (ResNet) or Inception modules (InceptionTime). In this paper,

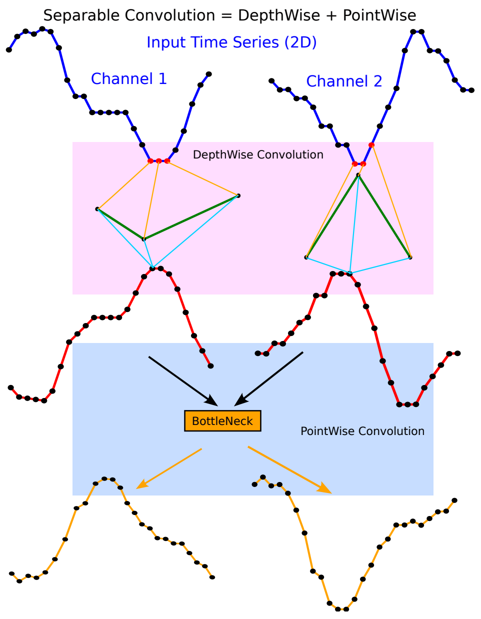

Fig. 2. DWSC for time series represented in its two phases: (1) DepthWise convolution (purple block), and (2) PointWise convolution (blue block).

we present an adaptation of the FCN architecture that only has 2.34% of Inception's number of parameters. Given this significant drop in the number of parameters, boosting techniques are used in order to preserve the performance of Inception.

First, we discuss about two ways of applying convolutions with less number of parameters, while preserving the performance. The first approach uses standard convolutions with BottleNecks (BN), and the second uses DWSC.

1) Standard Convolutions with BottleNecks: Many approaches that use CNN based architectures suffer from the problem of high number of parameters such as ResNet [11]. For this reason, the authors of InceptionTime [12] proposed to use a BottleNeck operation in order to reduce the number of parameters. This BottleNeck operation is made of 1D convolutions with a unit kernel size.

The following example shows the impact of this operation on reducing the number of parameters. Suppose at a depth d in the network, the input number of channels is C_{in}. The following convolution layer of depth $d+1$ projects the input into a new space with a number of channels C_{out} using a kernel of size k. On the one hand, without a BottleNeck operation, the number of learned parameters is $C_{in} * C_{out} * k$. On the other hand, with a BottleNeck operation that uses C_{bn} filters of size 1, the number of learned parameters is $C_{in} * C_{bn} * 1 + C_{bn} * C_{out} * k$. This operation reduces the number of parameters if and only if the following inequality is true:

$$C_{in} * C_{bn} + C_{bn} * C_{out} * k < C_{in} * C_{out} * k, \qquad (1)$$

which indicates that the condition on the BottleNeck operation is:

$$C_{bn} < \frac{C_{in} * C_{out} * k}{C_{in} + C_{out} * k}. \qquad (2)$$

The goal of the BottleNeck operation is to learn the same number of filters in the output channels (C_{out}), while reducing, at the same time, the intermediate learned filters between the input and output channels ($C_{in} * C_{out}$).

2) DepthWise Separable Convolution DWSC: DWSC can be divided into two phases: DepthWise convolution (*Phase 1*), and PointWise convolution (*Phase 2*). A visual representation of the DWSC operation is presented in Figure 2.

In standard convolutions, if the input sample of length L has C_{in} channels and the desired output is a space with C_{out} channels using a kernel of size k, then the number of learned parameters is $C_{in} * C_{out} * k$. The number of multiplications is $L * Cout * C_{in} * k$.

a) DepthWise convolution: In this phase, if the convolution is done using a kernel of size k, then the number of learned filters is C_{in}, and the output number of channels will be C_{in} (*such as the input*). In other words, for each dimension of the input time series, one filter is learned.

b) PointWise convolution: This phase projects the output of the DepthWise convolution into a space with a desired number of channels C_{out}. This is done by applying a standard convolution with C_{out} filters of kernel size 1 (*a BottleNeck*).

Hence, by combining these two phases, the number of learned parameters in a DWSC becomes $C_{in} * k + C_{in} * C_{out}$.

The following calculation finds the condition that the DWSC have less parameters to learn compared to the standard one:

$$\begin{aligned}
\underbrace{C_{in} * C_{out} * k}_{\text{standard Conv}} &> \underbrace{C_{in} * k + C_{in} * C_{out}}_{\text{separable Conv}} \\
C_{in} * C_{out} * k &> C_{in} * (k + C_{out}) \\
C_{out} * k &> k + C_{out} \\
k * (C_{out} - 1) &> C_{out} \\
k &> \frac{C_{out}}{C_{out} - 1} \xrightarrow{C_{out} \to \infty} \boxed{k > 1}.
\end{aligned} \tag{3}$$

This means that if the number of desired output channels is high enough (if $C_{out} >= 3$ the previous equation holds), then DWSC have less parameters to learn compared to the standard convolutions.

The number of multiplications performed in the DWSC is $C_{in} * L * k + Cin * C_{out} * L * 1$. The following calculation finds the second condition for when DWSC have less multiplications to perform compared to standard convolutions:

$$\begin{aligned}
\underbrace{C_{in} * C_{out} * L * k}_{\text{standard convs}} &> \underbrace{C_{in} * L * k + C_{in} * C_{out} * L * 1}_{\text{separable convs}} \\
C_{in} * C_{out} * L * k &> C_{in} * L * (k + C_{out}) \\
C_{out} * k &> k + C_{out} \\
k(C_{out} - 1) &> C_{out} \\
k &> \frac{C_{out}}{C_{out} - 1} \xrightarrow{C_{out} \to \infty} \boxed{k > 1}.
\end{aligned} \tag{4}$$

This concludes that DWSC have less parameters to learn with less number of multiplications to perform compared to standard convolutions. In this work, we present a comparison between the usage of DWSC or standard ones + BottleNecks. Our results demonstrate that with the techniques added to boost DWSC, we can maintain performance, while significantly reducing the number of parameters to optimize.

After defining two techniques to use convolutions in a more optimized way concerning number of parameters and multiplications, some other techniques should be defined as well. These techniques aim to minimize the impact of parameters reduction in convolutions operations explained above.

B. Boosting Techniques

The following techniques are borrowed from previous work from the literature.

1) Multiplexing: Multiplex convolution was proposed in the architecture of Inception [12]. Its main idea is to learn in parallel different convolution layers of different kernel size. A multiplexing example is shown in Figure 3.

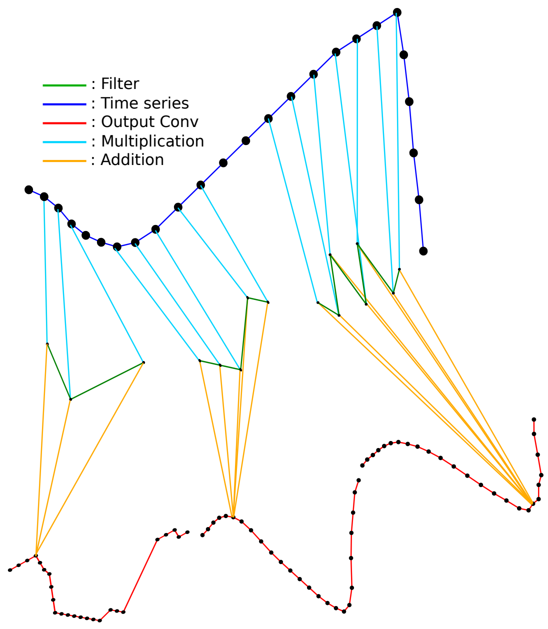

Fig. 3. Multiplexing one dimensional convolution on the input time series (in blue) using filters (in green) with three different kernel sizes, respectively, 3, 5, and 7. The output of the convolutions (in red) is different for each filter.

2) Dilation: Dilated convolutions were not very explored for deep supervised learning on TSC but they were used in self-supervised models showing to be very effective [19]. Dilation in convolutions filters defines the number of empty cells in the kernel. Suppose a kernel of length 3 has the following parameters $k = [k_0, k_1, k_2]$, a dilation of rate 2 will produce the following kernel $k = [k_0, skip, k_1, skip, k_2]$. The $skip$ parameter indicates that the convolution layer will not use the values of the input aligned with this index of the kernel. A visualization of the dilation effect on convolution can be seen in Figure 4. Dilation will help increasing the receptive field of a model without having to add deeper layers because the dilated kernel will find the deeper combinations in the same layer.

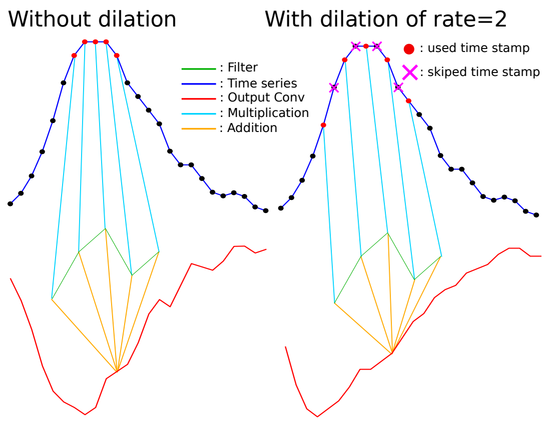

Fig. 4. One dimensional convolution performed w/o dilation on the left (*rate*=1), and with dilation on the right (*rate*=2).

3) Custom Filters: Custom filters were proposed in [13]. The authors hand-crafted some kernels in order to detect specific patterns in the input time series. These filters were then added to Inception and results on the UCR archive have shown that such filters can help with the generalization and boost the performance. This is due to the fact that these filters are generic and fixed (not learned). This allows the model to focus on learning new patterns harder to detect.

C. Proposed architecture

1) Light Inception with boosTing tEchniques (LITE): In our proposed architecture, we reduce the number of parameters, while preserving performance. This is obtained by using DWSC and the previously explained boosting techniques. First, custom filters are used in the first layer parallel to the first layer. Second, in this first layer, multiplexing convolution is used in order to detect different patterns of different characteristics (three parallel convolution layers). Third, the second and third layer present the usage of dilation in their kernels. It is important to notice that for the first layer, standard convolutions are used instead of DWSC. This is due to the fact that the input time series is univariate and DWSC will learn only one filter. A summary of the architecture is given in Figure 5.

2) Ensemble: Ensemble learning is a technique of combining the prediction of multiple models in order to reduce the variance, and it has been shown to be very effective in the literature [12], [38], [39]. Applying an ensemble of multiple classification models is equivalent to find the average predicted distribution of all the models. This average distribution is finally used for choosing the predicted class. This motivated us to build an ensemble of multiple LITE models to form LITE-Time (by adding the suffix *Time* following [12]). Moreover, the usage of an ensemble in the case of LITE is also motivated by its small architecture and the fact that it can boost less complex architectures even more.

IV. Experimental Evaluation

A. Datasets

The UCR archive [9] is the largest directory for the TSC problem. It is publicly available [2]. The archive contains 128 datasets of univariate TSC tasks. Some tasks involves Electrocardiography (ECG) time series data and some are observations of Sensors, *etc.*. Each dataset is split into a training and a testing set. The labels are available for all the samples. In order to train the model on a normalized dataset, we apply Z-normalization over all the samples independently. This normalization technique reduces the time series samples to a zero mean and unit variance sequence.

B. Implementation Details

Our results were obtained on the UCR archive using a GTX 1080 GPU with 8GB of VRAM. In the experiments, we accounted for the training time, inference time (testing time), CO2 and Power consumption. The model used for testing is the best model during training, chosen by monitoring the training loss. The Adam optimizer was used with a Reduce on Plateau learning rate decay method by monitoring the training loss. The Adam optimizer's parameters are the default set up of the Tensorflow Python model [3]. The model is trained with a batch size of 64 for 1500 epochs.

For the parameters of the LITE architecture presented in Figure 5, the following setup is used: $N_i = 6$, variations in kernel sizes $= [2, 4, 8, 16, 32, 64]$; $N_d = 6$, variations in kernel sizes $= [2, 4, 8, 16, 32, 64]$; $N_p = 5$, variations in kernel sizes $= [6, 12, 24, 48, 96]$; $N = 32$; $K = 40$; $D0 = 1$ in order to start with no dilation and increase with depth ($D1 = 2$ and $D2 = 4$). The source code is publicly available [4].

C. Results and Discussion

1) Number of Parameters, FLOPS Training Time, Testing time, CO2 and Power Consumption: Table I summarizes the number of parameters, the number of FLoating-point Operations Per Second (FLOPS), training time, inference time, CO2 and Power consumption over the 128 datasets of the UCR archive. The number of parameters shown is the number of trainable parameters of the architecture without the last classification Fully Connected layer because it depends on each dataset (number of classes). The rest of the information is summed over the 128 datasets of the UCR archive and average over five different runs.

First, the table shows that the smallest model in terms of number of parameters is the LITE with $9,814$ parameters. This is mainly due to the usage of DWSC instead of standard ones. Compared to FCN ResNet and Inception, LITE has only 3.7% 1.95% and 2.34% of their number of parameters respectively. Second, the fastest model in the training phase is LITE, with a training time of 0.62 days. LITE is 2.79, 3.08 and 2.71 time faster than FCN, ResNet and Inception respectively.

[2]https://www.cs.ucr.edu/~eamonn/time_series_data_2018/
[3]https://keras.io/api/optimizers/adam/
[4]https://github.com/MSD-IRIMAS/LITE

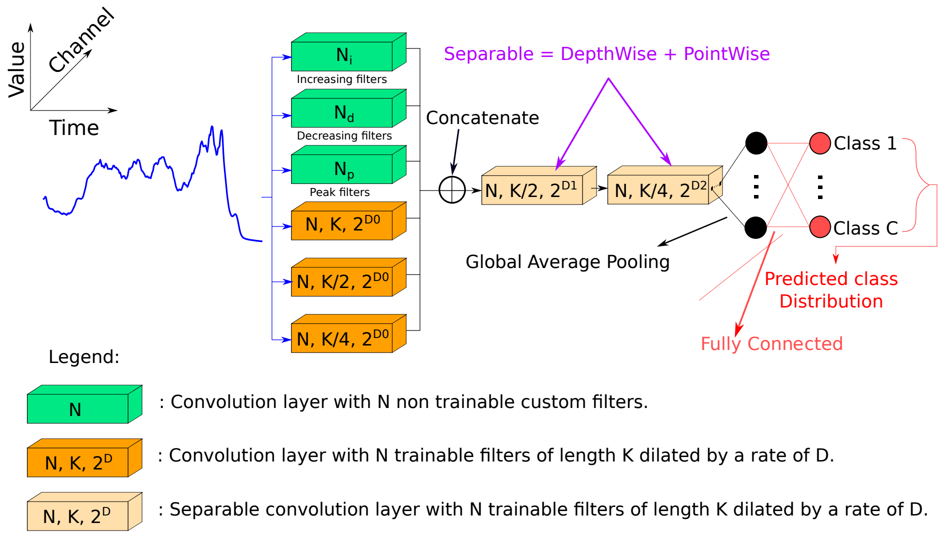
Value
Channel
Time
N_i
Increasing filters
N_d
Decreasing filters
N_p
Peak filters
N, K, 2^{D0}
N, K/2, 2^{D0}
N, K/4, 2^{D0}
Concatenate
Separable = DepthWise + PointWise
N, K/2, 2^{D1}
N, K/4, 2^{D2}
Global Average Pooling
Class 1
Class C
Predicted class Distribution
Fully Connected
Legend:
N
: Convolution layer with N non trainable custom filters.
N, K, 2^D
: Convolution layer with N trainable filters of length K dilated by a rate of D.
N, K, 2^D
: Separable convolution layer with N trainable filters of length K dilated by a rate of D.

Fig. 5. The proposed LITE architecture that uses multiplexing convolutions in the first layer (three convolution blocks in orange) with custom filters (in green). The second and third layer are composed of DWSC (in beige). The last layer is followed by a Global Average Pooling (GAP) on the time axis and finished by a classification Fully Connected layer to approximate the class distribution.

Third, LITE is the model that consumes the smallest amount of CO2 and Power, 0.1048 g and 0.2468 W respectively. Compared to the other approaches, LITE presents the fastest and most ecologic model for TSC compared to FCN, ResNet and Inception. We believe, given the factors explained above, that LITE can be used for the deployment of deep learning for TSC in small machine such as mobile phones. In what follows we present the performance of the proposed LITE model compared to the state-of-the-art in terms of accuracy metric on test evaluation.

2) Accuracy Performance: In what follows, a one-vs-one comparison is presented between the models in order to show that LITE can preserve the performance of the more complex architectures. This one-vs-one comparison comes down to a Win/Tie/Loss count on the 128 datasets of the UCR archive between two classifiers. This comparison is visualized in Figures 6 and 7. Each point in these plots represents one dataset of the UCR archive. The x-axis contains the accuracy value on the test set using classifier-x and the y-axis the ones using classifier-y.

In order to evaluate the significance of the Win/Tie/Loss comparison, a statistical Wilcoxon Signed Rank Test [40] is used. This test will return a statistical value, the P-Value, representing how significant the difference is between the two classifiers. If the P-Value is low, this would mean that the difference in performance between the two classifiers is statistically significant. If the last condition is not true, it means that there are not enough examples (datasets) to find a statistical significant difference between the classifiers. This Wilcoxon test needs a P-Value threshold, usually in the literature a 0.05 (or 5%) threshold is used.

On the one hand, the results presented in Figure 6 show that LITE beats FCN significantly (low P-Value), and is statistically not significant than ResNet (high P-Value). The results compared to ResNet are impressive given the small complexity of LITE (almost 1.95% of ResNet's number of parameters). On the other hand, the comparison shows that LITE still is not significantly close to Inception. To study more the reason why LITE performs not as good as Inception with a large margin (more than 10%), we presented some characteristics of those datasets in Table II. This table shows that some of the datasets have either long time series or small training set. Firstly, this indicates that Inception is better than LITE on long time series given its large receptive field (deeper architecture). Secondly, this points out that Inception generalizes better better in the case where the dataset presents a small training set so it can generalize better.

Given that Inception still beats LITE, an ensemble comparison shows the real performance of the proposed architecture. This is due to the fact that LITE has way less parameters (2.34% of Inception's number of parameters) which can make it sensitive to a higher variance when training with different initialization. Applying an ensemble removes this variance as explained before.

The comparison between LITETime with InceptionTime and ROCKET is presented in Figure 7. This comparison shows that, given the 128 datasets of the UCR archive, there are not enough datasets to find a statistical significance in the difference of performance with InceptionTime and ROCKET. We included ROCKET in the ensemble comparison with LITETime because the motivation in ROCKET is to replace the ensemble technique by using random filters instead of learning them starting with different initialization.

Those last results suggest that in order to get a good performance on the UCR archive, a large complex architecture with a high number of parameters is not always needed.

For a multi-classifier comparison, the average rank of each model is shown in a Critical Difference Diagram [41] (CD

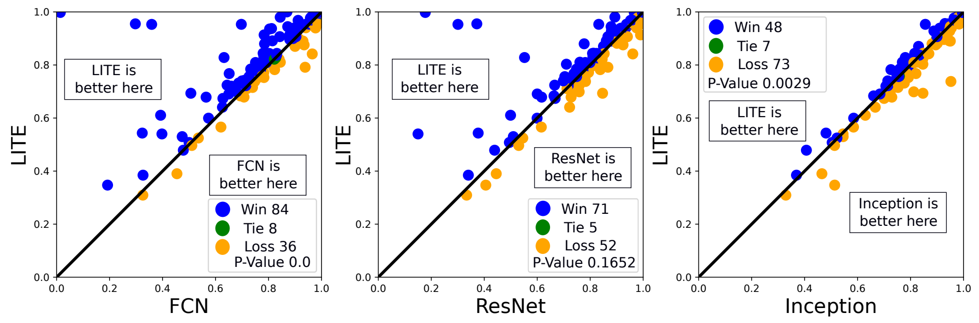

Fig. 6. One-vs-One comparison between LITE with three different models: FCN, ResNet and Inception over the 128 datasets of the UCR archive.

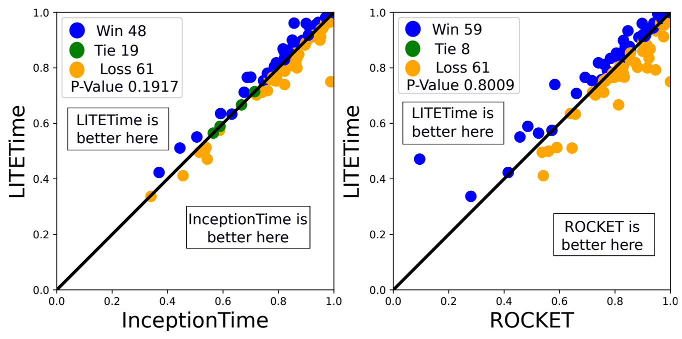

Fig. 7. One-vs-One comparison between LITETime with two different models: InceptionTime and ROCKET over the 128 datasets of the UCR archive.

Diagram) based on the ranking classifiers given the average rank over multiple datasets, the two tailed Wilcoxon Signed-Rank Test with the Holm multiple test correction [42]. To generate the CD Diagram, we used the publicly available code [5]. This diagram also presents connections between classifiers, when the difference in performance is not statistically different following the Wilcoxon Signed Rank Test. A CD Diagram is presented in Figure 8 and shows that LITETime comes $3rd$ on the average rank between ROCKET and InceptionTime. The diagram also shows that FCN performs statistically significantly worse than LITE on the average rank. Furthermore, on the UCR archive, no statistical significance can be observed between ResNet and LITE. This last comparison shows the real impact of this work where LITE has almost 1.95% (Table I) of ResNet's number of parameters. These conducted results show that on a large amount of cases, there is no need for a complex model with high number of parameters to achieve good performance.

Furthermore, a new Multi-Comparison Matrix (MCM) evaluation tool was proposed that is stable to the variation of the addition and removal of classifiers [43]. The MCM is presented in Figures 9 and 10 to compare LITE and LITETime, respectively, to other approaches in the literature. The MCM uses the average accuracy on the UCR as an ordering metric instead of the average rank. As presented in the MCMs, LITE and LITETime perform better than FCN and ResNet on the average accuracy and is closer to the performance of InceptionTime, which is not significantly different than LITETime (high p-value). MCM has an advantage over the usage of the CD Diagram of being stable with the addition and removal of classifiers, given that the average performance would not change in this scenario unlike the average rank. Another advantage is not using a multiple test correction for the p-value significance test.

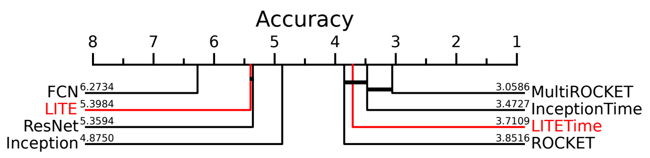

Fig. 8. A Critical Difference Diagram (CD Diagram) showing the average rank of each classifier over the 128 datasets of the UCR archive with the significance in difference of performance.

D. Ablation Study - LITE

1) Impact of Additional Techniques: The proposed LITE architecture, uses multiples techniques in order to improve the performance. In order to show the impact of each technique on the proposed architecture, an ablation study is presented in this section. First, the LITE is stripped down from the three used techniques: dilation, multiplexing and custom filters. Second, given that for the multiplexing convolutions performed in the first layer there are a total of three layers with n filters, the Striped-LITE learns a total of $3n$ filters for the first layer. The rest of the architecture is the same using DWSC without dilation. We then add each boosting technique separately on the stripped LITE model and evaluate its performance. The results of this ablation study are visualized in Figure 11 in the form of a Heat Map. Each cell of the Heat Map contains the Win Tie Loss count when evaluating the test accuracy on the UCR archive. The addition of the P-Value statistics is presented in each cell in order to assess the significance in difference of performance. The P-Value is emphasized in **bold** when it is lower than the specified threshold (0.05). The colors of the Heat Map represent the difference in the average accuracy.

Results show that adding custom filters in the first layer as well as using multiplexing convolution in the first layer

[5]https://github.com/hfawaz/cd-diagram

TABLE I

COMPARISON BETWEEN THE PROPOSED METHODS WITH FCN, RESNET AND INCEPTION WITHOUT ENSEMBLE.

Models	Number of parameters	FLOPS	Training Time	Testing Time	CO2 Consumption	Power Consumption
Inception	420,192	424,414	145,267 seconds 1.68 days	81 seconds 0.0009 days	0.2928 g	0.6886 W
ResNet	504,000	507,818	165,089 seconds 1.91 days	62 seconds 0.0007 days	0.3101 g	0.7303 W
FCN	264,704	266,850	149,821 seconds 1.73 days	27 seconds 0.00031 days	0.2623 g	0.6176 W
LITE	**9,814**	**10,632**	**53,567 seconds 0.62 days**	**44 seconds 0.0005 days**	**0.1048 g**	**0.2468 W**

TABLE II

DATASETS WHERE INCEPTION BEATS LITE BY MORE THAN 10% OF ACCURACY.

Dataset Name where Inception is better than LITE	Difference in Accuracy (%)	Series Length	Number of Samples
EthanolLevel	11.32	1751	504
OliveOil	15.34	570	30
PigAirwayPressure	16.64	2000	104
ShapeletSim	21.44	500	20

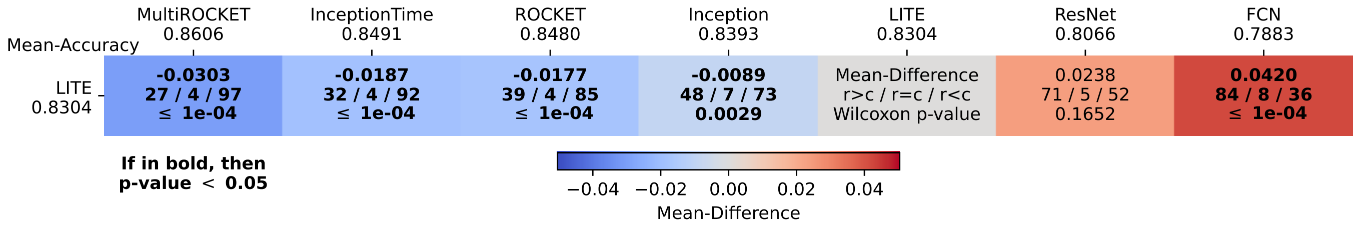

Fig. 9. The Multi-Comparison Matrix applied to show the performance of LITE compared to other approaches.

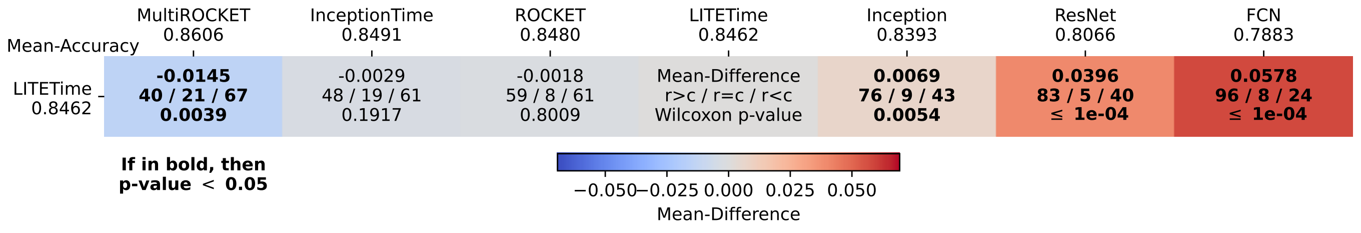

Fig. 10. The Multi-Comparison Matrix applied to show the performance of LITETime compared to other approaches.

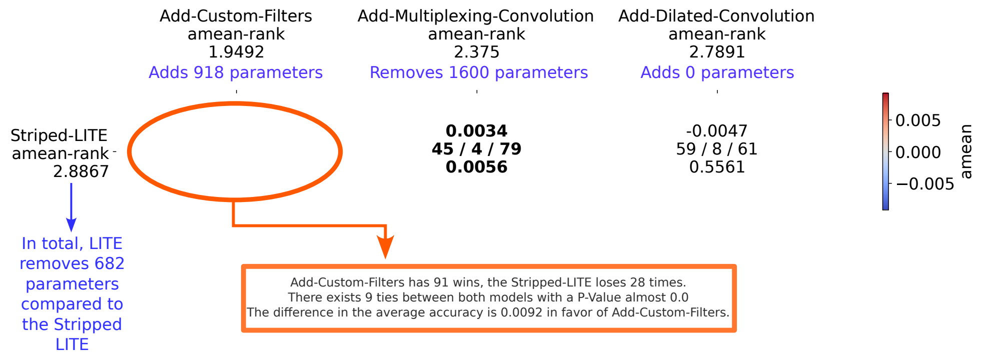

Fig. 11. The Heat Map shows the one-vs-one comparison between the Striped-LITE and the three variants: (1) Add-Custom-Filters, (2) Add-Multiplexing-Convolution and (3) Add-Dilated-Convolution. The colors of the Heat Map follow the value of the first line in each cell. This value is the difference between the value of the first line (average accuracy when winning/losing). The second line represents the Win/Tie/Loss count between the models in question (wins for the column model). The last line is the statistical P-Value between the two classifier using the Wilcoxon Signed Rank Test.

significantly boosts the performance. The colors of the Heat Map indicates that adding the custom filters has also a positive impact on the average accuracy though it can add some parameters. This is not the case for the multiplexing convolutions. We believe that this small average impact (0.34% overall) is not as important as the positive one. This is due to the fact that multiplexing reduces the number of parameters and wins over the majority of the datasets significantly. The addition of the dilated convolution is shown to not have a statistical significance on the performance (P-Value > 0.05). However, the average difference in accuracy shows that most of the times, using the dilated convolutions can improve the results. Given that dilation does not add more parameters and on average it boosts the performance, we keep it in the LITE architecture. This is due to the fact that dilation increases the receptive field, so for large datasets this can be a boosting feature. The reason why sometimes Dilation can have a negative effect is because some of the datasets in the UCR archive do not require a large receptive field. Altogether, the LITE model (with the boosting techniques) will have less parameters compared to the striped LITE while preserving performance compared to state-of-the-art models. The decrease in number of parameters when using all the boosting techniques together comes from the fact that multiplexing removes more parameters than the custom filters adds. Lastly, Figure 11 shows the average rank of the models, such as in the CD Diagram explained in Section IV-C2. The average rank of the Add-Custom-Filters is the lowest, while the Striped-LITE has the highest rank. Therefore, the worst model between the four presented in the Heat Map is the LITE without any boosting techniques.

2) Impact of DWSC: To show the effect of DWSC as well, we replace them by standard convolutions followed by a BottleNeck. To get a non-noisy comparison, we used ensembles. Note that the usage of the ensemble technique is necessary in this case given that by removing the DWSC, the difference in number of parameters becomes very high (LITE has almost 11% of the compared model's number of parameters (the compared model has around $85,000$ parameters). In Figure 12, the one-vs-one comparison between LITETime and LITETime with Standard Convolutions is presented. Results show that the usage of DWSC does not have an effect on the performance because the P-Value is high 0.4556. This means that the difference in performance is not statistically significantly different with less parameters mainly due to the usage of the DWSC.

V. Conclusions

In this paper, we addressed the Time Series Classification problem by reducing the number of parameters compared to existing deep learning approaches for Time Series Classification, while preserving performance of InceptionTime. We presented a new architecture for Time Series Classification, LITE, and evaluated its performance on the UCR archive. LITE has only 2.34% of InceptionTime's number of parameters. This model is faster than the state-of-the-art in training and inference time. It consumes as well less CO2 and power, which is a topic we believe to be very important nowadays. Results have illustrated that the usage of LITE allows us to achieve state-of-the-art results on the UCR archive. Furthermore, the presented ablation study demonstrated the importance of the techniques used in LITE. We believe this work can be the start of optimizing deep learning architectures in the time series domain. We believe that this study can address clustering, representation learning and generative models as well. In future work, we aim to tackle these others domains given the impressiveness in performance of LITE compared to ResNet and InceptionTime.

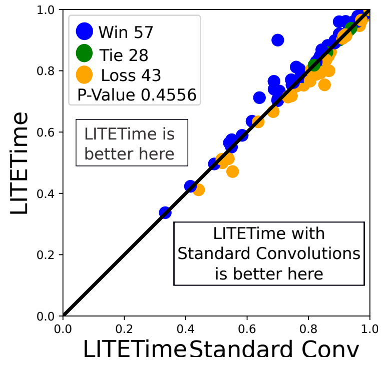

Fig. 12. One-vs-one comparison between LITETime and LITETime with Standard convolutions over the 128 datasets of the UCR archive.

Acknowledgment

This work was supported by the ANR DELEGATION project (grant ANR-21-CE23-0014) of the French Agence Nationale de la Recherche. The authors would like to acknowledge the High Performance Computing Center of the University of Strasbourg for supporting this work by providing scientific sup- port 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. The authors would also like to thank the creators and providers of the UCR Archive.

References

[1] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Evaluating surgical skills from kinematic data using convolutional neural networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 214–221.

[2] L. Tao, E. Elhamifar, S. Khudanpur, G. D. Hager, and R. Vidal, "Sparse hidden markov models for surgical gesture classification and skill evaluation," in *International conference on information processing in computer-assisted interventions*. Springer, 2012, pp. 167–177.

[3] G. Forestier, F. Lalys, L. Riffaud, B. Trelhu, and P. Jannin, "Classification of surgical processes using dynamic time warping," *Journal of biomedical informatics*, vol. 45, no. 2, pp. 255–264, 2012.

[4] M. Devanne, H. Wannous, S. Berretti, P. Pala, M. Daoudi, and A. Del Bimbo, "3-d human action recognition by shape analysis of motion trajectories on riemannian manifold," *IEEE transactions on cybernetics*, vol. 45, no. 7, pp. 1340–1352, 2014.

[5] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[6] J. P. Pinto, A. Pimenta, and P. Novais, "Deep learning and multivariate time series for cheat detection in video games," *Machine Learning*, vol. 110, no. 11-12, pp. 3037–3057, 2021.

[7] R. Younis, S. Zerr, and Z. Ahmadi, "Multivariate time series analysis: An interpretable cnn-based model," in *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2022, pp. 1–10.

[8] F. Madrid, S. Singh, Q. Chesnais, K. Mauck, and E. Keogh, "Matrix profile xvi: efficient and effective labeling of massive time series archives," in *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2019, pp. 463–472.

[9] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The ucr time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.

[10] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[11] 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.

[12] 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.

[13] A. Ismail-Fawaz, M. Devanne, J. Weber, and G. Forestier, "Deep learning for time series classification using new hand-crafted convolution filters," in *2022 IEEE International Conference on Big Data (IEEE BigData 2022)*. IEEE, 2022, pp. 1–8.

[14] G. Pialla, M. Devanne, J. Weber, L. Idoumghar, and G. Forestier, "Data augmentation for time series classification with deep learning models," in *Advanced Analytics and Learning on Temporal Data (AALTD)*, 2022.

[15] B. Lafabregue, J. Weber, P. Gançarski, and G. Forestier, "End-to-end deep representation learning for time series clustering: a comparative study," *Data Mining and Knowledge Discovery*, vol. 36, no. 1, pp. 29–81, 2022.

[16] T. Terefe, M. Devanne, J. Weber, D. Hailemariam, and G. Forestier, "Time series averaging using multi-tasking autoencoder," in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2020, pp. 1065–1072.

[17] A. Ismail-Fawaz, M. Devanne, J. Weber, and G. Forestier, "Enhancing time series classification with self-supervised learning," in *15th International Conference on Agents and Artificial Intelligence: ICAART 2023*, INSTICC. SciTePress, 2023.

[18] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "Representation learning of multivariate time series using a transformer framework," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2021.

[19] K. Wickstrøm, M. Kampffmeyer, K. Ø. Mikalsen, and R. Jenssen, "Mixing up contrastive learning: Self-supervised representation learning for time series," *Pattern Recognition Letters*, vol. 155, pp. 54–61, 2022.

[20] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Adversarial attacks on deep neural networks for time series classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[21] G. Pialla, H. I. Fawaz, M. Devanne, J. Weber, L. Idoumghar, P.-A. Muller, C. Bergmeir, D. Schmidt, G. Webb, and G. Forestier, "Smooth perturbations for time series adversarial attacks," in *Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, May 16–19, 2022, Proceedings, Part I*. Springer, 2022, pp. 485–496.

[22] 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.

[23] A. Dempster, D. F. Schmidt, and G. I. Webb, "Minirocket: A very fast (almost) deterministic transform for time series classification," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 248–257.

[24] C. W. Tan, A. Dempster, C. Bergmeir, and G. I. Webb, "Multirocket: Multiple pooling operators and transformations for fast and effective time series classification," *Data Mining and Knowledge Discovery*, pp. 1–24, 2022.

[25] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[27] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[28] T. Schick and H. Schütze, "It's not just size that matters: Small language models are also few-shot learners," *arXiv preprint arXiv:2009.07118*, 2020.

[29] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *arXiv preprint arXiv:1603.06995*, 2016.

[30] 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.

[31] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, and G. I. Webb, "Proximity forest: an effective and scalable distance-based classifier for time series," *Data Mining and Knowledge Discovery*, vol. 33, no. 3, pp. 607–635, 2019.

[32] J. Zhao and L. Itti, "shapedtw: Shape dynamic time warping," *Pattern Recognition*, vol. 74, pp. 171–184, 2018.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[34] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks for multivariate time series classification," *Frontiers of Computer Science*, vol. 10, no. 1, pp. 96–112, 2016.

[35] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[36] E. Ay, M. Devanne, J. Weber, and G. Forestier, "A study of knowledge distillation in fully convolutional network for time series classification," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[38] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep neural network ensembles for time series classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–6.

[39] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, "Hive-cote 2.0: a new meta ensemble for time series classification," *Machine Learning*, vol. 110, no. 11-12, pp. 3211–3243, 2021.

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

[41] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[42] A. Benavoli, G. Corani, and F. Mangili, "Should we really use post-hoc tests based on mean-ranks?" *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 152–161, 2016.

[43] A. Ismail-Fawaz, A. Dempster, C. W. Tan, M. Herrmann, L. Miller, D. F. Schmidt, S. Berretti, J. Weber, M. Devanne, G. Forestier *et al.*, "An approach to multiple comparison benchmark evaluations that is stable under manipulation of the comparate set," *arXiv preprint arXiv:2305.11921*, 2023.