A study of Knowledge Distillation in Fully Convolutional Network for Time Series Classification

Emel Ay, Maxime Devanne, Jonathan Weber, and Germain Forestier
IRIMAS
Université de Haute-Alsace, Mulhouse, France
firstname.lastname@uha.fr

***Abstract*—In recent years, deep learning revolutionized the field of machine learning. While many applications of deep learning are observed in computer vision, other domains like natural language processing (NLP) or speech recognition also benefited from advances in deep learning research. More recently, the field of time series analysis and more especially time series classification (TSC) also witnessed the emergence of deep neural networks providing competitive results. Through the years, the proposed network architectures became deeper and deeper pushing the performance higher. While these very deep models achieve impressive accuracy, their training and deployment became challenging. Indeed, a large number of GPUs is often required to train state-of-the-art networks and obtain high performances. While the requirements needed for the training step can be acceptable, deploying very deep neural networks can be difficult especially in embedded systems (e.g. robots) or devices with limited resources (e.g. web browsers, smartphones). In this context, knowledge distillation is a machine learning task consisting in transferring knowledge from a large model to a smaller one with fewer parameters. The goal is to create a lighter model mimicking the predictions of a larger one in order to obtain similar performances with a fraction of the computational cost. In this paper, we introduce and explore the concept of knowledge distillation for the specific task of TSC. We also present a first experimental study showing promising results on several datasets of the UCR time series archive. As current state-of-the-art models for TSC are deep and sometimes ensemble of models, we believe that knowledge distillation could become an important research area in the coming years.**

***Index Terms*—Times Series Classification, Knowledge Distillation, Fully Convolutional Network**

I. Introduction

The creation of increasingly deep neural networks (DNNs) now makes it possible to perform tasks that were still unthinkable a few years ago. High performance has been achieved in various areas such as computer vision tasks [1] and natural language processing [2]. The great success of DNNs in such domains can in part be explained by the huge amount of available data to train complex models. Following the trend, the last few years have also seen an explosion in the amount of time series data of various modalities such as electrocardiogram [3], power consumption [4], human motion [5] and satellite images [6] among others. Due to its wide range of applications, time series analysis has attracted researchers who developed deep learning-based models for time series clustering [7], averaging [8], forecasting [9] and classification [10]. In this paper, we focus on the task of time series classification (TSC) for which a recent study has shown that DNNs based on 1D temporal convolutions are achieving great performances [10].

Similarly to other fields, the network architectures have gotten deeper and deeper over the years for TSC, always pushing performance up. However, it is difficult to deploy these large deep models into devices with limited resources, such as mobile phones and integrated devices, not only due to the high complexity of the calculations, but also the large storage requirements. Thus, to achieve faster speeds while maintaining good performance, recent work in computer vision focuses on knowledge distillation techniques. The distillation process begins with separately training a bulky model (*teacher*), followed by learning a smaller model (*student*) that seeks to mimic the *teacher*'s predictions and / or its feature representations. This strategy has first been developed in [11] and later extended by Hinton et al. [12] and Romero et al. [13] for image classification. Findings in these papers convincingly demonstrate that a shallower model can achieve competitive or even superior performance in comparison to a very deep model.

With the advent of newer and deeper neural networks architectures, the efficiency of knowledge distillation has been validated not only in computer vision but also in document retrieval [14] or speech recognition [15]. Surprisingly, out of our knowledge, no study considering knowledge distillation has been undertaken in the area of TSC. Hence, in this work, we propose to investigate the use of knowledge distillation for the task of TSC. Our aim is to analyze the impact of some hyper-parameters on the performances of smaller *student* models leveraging the knowledge of deeper pre-trained *teacher* models. The use of knowledge distillation for TSC is evaluated on the UCR Archive 2018 [16]. Experiments show that knowledge distillation can be very beneficial for intermediate deep *student* models.

II. Related Work

A. Knowledge Distillation

In parallel with the race for performance of DNNs developing deeper and deeper neural networks, Hinton et al. [12] proposed to investigate a different idea on building shallower models mimicking behavior of more complex neural networks.

Authors evaluated this process called knowledge distillation for image classification where a *student* model is trained to output soft probabilities similar to a pre-trained ensemble of *teacher* models. Results demonstrated that by leveraging the *teacher* ensemble's knowledge, a single *student* can obtain very competitive results. An extended approach has then been proposed in [13] by not only considering the outputs but also intermediate feature maps as hints for improving the *student* performance. Moreover, intermediate features have also been considered in [17]. Additional extensions of the original approach have been proposed like in [18] where metric learning is introduced through a triplet loss. Differently, in [19] a teacher assistant is proposed as an intermediate model to fill the gap between a very deep *teacher* and a shallower *student*. While knowledge has mainly been used in computer vision [20], its promising results also motivated research to employ it in various fields like document analysis [14], speech processing [15] and natural language processing [21]. For time series analysis, only a few works investigated the use of knowledge distillation for regression tasks [22] and forecasting [23]. In this work we propose to study the use of knowledge distillation for the classification of time series.

B. Time Series Classification

Time series analysis and in particular TSC has attracted many researchers in the last decade. The use of the nearest neighbor (NN) classifier coupled to a distance function has been the subject of numerous studies. In particular, Dynamic Time Warping (DTW) distance [24] showed excellent performances compared to other distance measures. Moreover, instead of using a single model for classifying time series, the combination of several models was also considered and led to very satisfactory results. For instance, a set of decision trees, i.e. a random forest, was employed in [25], while the ensemble of different classifiers, i.e. Hive Cote 2.0 proposed in [26], became the state-of-the-art when evaluated on the UCR archive. However, these models have the huge disadvantage of being too large and are associated with a high computational time-consuming training process.

In parallel, with the rapid development of deep learning, TSC did not make the exception. A first approach considering Convolutional Neural Network (CNN) in a multi-scale paradigm was proposed in [27]. Moreover, a Fully Convolutional Network (FCN) demonstrated very good results in [28]. These architectures have been later compared with other deep learning-based approaches in a review paper proposed by Ismail Fawaz et al. [10]. Comparative results demonstrated that the Residual Network (ResNet) obtained the best performances on the UCR archive. A more recent approach adapting the famous Inception architecture for TSC, namely InceptionTime [29], demonstrated that considering various sizes of convolutional filters results in better classification accuracies. Finally, inspired by the success of convolutional filters for TSC and to overcome the huge training time of deep learning-based approaches, a large amount of random convolutions kernels was employed as feature extractor combined with a Ridge regression-based classifier [30]. This approach called RandOm Convolutional KErnel Transform (ROCKET) is currently the state-of-the-art for TSC. In this paper, our aim is not to develop a state-of-the-art model but instead to study the impact of knowledge distillation in convolution-based models. For that, we consider the classical FCN architecture.

III. Knowledge Distillation for Time Series Classification

A. Background on Deep Learning for TSC

A time series, or chronological series, is an ordered series of numerical values representing the evolution of a specific quantity over time. Let t_1, t_2, ..., t_T be successive timestamps, a time series can be defined as $\mathbf{X} = [x_{t_1}, x_{t_2}, \ldots, x_{t_T}], \mathbf{X} \in \mathbb{R}^T$. In this work, we are considering univariate time series, i.e. $x_{t_i} \in \mathbb{R}, i = 1, \ldots, T$.

An univariate time series dataset $\mathbf{D}$, including M time series, is a collection of pairs $(\mathbf{X}_i, \mathbf{Y}_i), i = 1, \ldots M$, where $\mathbf{Y}_i \in \mathbb{R}^C$ is a one-hot vector representing the class label $c \in [1, C]$ associated to $\mathbf{X}_i$.

The time series classification (TSC) problem consists of assigning label vector $\mathbf{Y}$ to a time series $\mathbf{X}$. Hence the TSC task consists of training a classifier $\mathcal{F}$ mapping inputs $\mathbf{X}_i$ to their corresponding label vectors $\mathbf{Y}_i$.

In deep learning, such function $\mathcal{F}$ corresponds to a Deep Neural Network (DNN). During training, the true labels for each time series are available, thus the goal is to optimize DNN parameters so that predicted labels $\hat{\mathbf{Y}}$ match the true labels $\mathbf{Y}$ as much as possible. For that, a loss function $\mathcal{L}$ measuring the difference between predictions and true labels is employed. In most DNNs used for TSC, the categorical cross-entropy loss $\mathcal{L}_{CE}$ is considered. Hence the training objective is, given a training dataset $\mathbf{D}_{train}$ find a DNN $\mathcal{F}*$ minimizing $\mathcal{L}_{CE}$.

As shown in [10], deep models performing better for TSC are the one using convolutional layers computing 1D temporal convolutions on time series. A convolution operation can be seen as applying and sliding a filter ω of size l along the time series $\mathbf{X}$ of size T. For a time step t, a convolution operation C_t in a convolutional layer is defined as:

$$C_t = f(\omega * X_{t-l/2:t+l/2} + b) \mid \forall t \in [1, T], \tag{1}$$

where f is a nonlinear function and b is the biais.

B. Knowledge Distillation for TSC

The core idea of knowledge distillation implies two neural networks, a *teacher* (generally a deep model) and a *student* (generally a shallow model). The knowledge distillation process is illustrated in Figure 1. The goal is to train a *student* model on a time series dataset $\mathbf{D}$ by leveraging the knowledge acquired by a pre-trained *teacher* model. The knowledge generally refers to the last layer's output of the *teacher* model. Hence, during training the *student* model is optimized to mimic *teacher* final predictions.

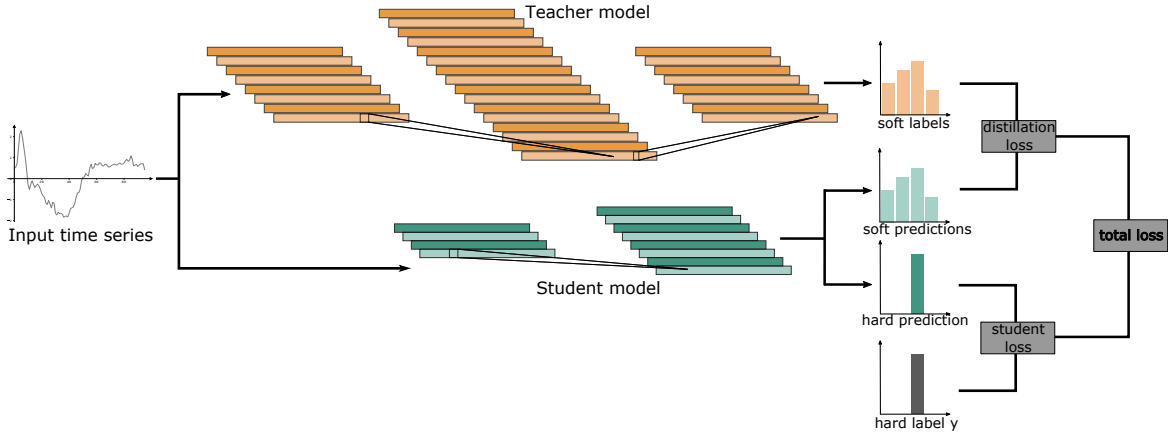


Fig. 1: Overview of our knowledge distillation architecture for time series classification

For a TSC task, neural networks generally produce class probabilities, i.e. predicted labels $\hat{\mathbf{Y}}_i$ using the Softmax function in the last layer applied on the output logits $z_i, i = 1, \dots C$, as defined in equation 2.

$$\hat{\mathbf{Y}}_i = Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{2}$$

As stated before, the aim of knowledge distillation is to make the *student* model mimics predictions of the pre-trained *teacher* model. For relaxing the problem, soft labels are considered by incorporating a temperature hyper-parameter τ in the Softmax function, as defined in equation 3.

$$\hat{\mathbf{Y}}_i^{\tau} = Softmax(z_i) = \frac{e^{z_i/\tau}}{\sum_{j=1}^{C} e^{z_j/\tau}} \tag{3}$$

The temperature hyper-parameter τ allows to control the smoothness of predicted probabilities. The more τ is higher than 1, smoother the probabilities are (the uncertainty among classes is considered). Let $\hat{\mathbf{Y}}_T^{\tau}$ and $\hat{\mathbf{Y}}_S^{\tau}$ be the predictions of the *teacher* model and the *student* model, respectively. The distillation loss measuring the similarity between the two probability distributions is the Kullback-Leibler divergence $\mathcal{L}_{KL}(\hat{\mathbf{Y}}_T^{\tau}, \hat{\mathbf{Y}}_S^{\tau})$.

Finally during training, the *student* model is optimized to both maximize classification of training time series and to leverage the knowledge of the pre-trained *teacher* model. Thus, the final knowledge distillation loss $\mathcal{L}_{KD}$ is defined as:

$$\mathcal{L}_{KD} = \lambda \times \mathcal{L}_{CE}(\mathbf{Y}, \hat{\mathbf{Y}}_S) + (1-\lambda) \times \tau^2 \times \mathcal{L}_{KL}(\hat{\mathbf{Y}}_T^{\tau}, \hat{\mathbf{Y}}_S^{\tau}), \tag{4}$$

where λ controls the weight of both distillation loss $\mathcal{L}_{KL}$ and student loss $\mathcal{L}_{CE}$. The student loss $\mathcal{L}_{CE}$ corresponds to the classification loss defined as the cross-entropy between student predictions $\hat{\mathbf{Y}}_S$ and true labels $\mathbf{Y}$.

C. Teacher model

Inspired by the great success of Convolutional Neural Networks for TSC [10], we choose a Fully Convolutional Neural Network (FCN) [28] as our *teacher* model. As our main goal is to assess the impact of reducing the number of convolutional layers and convolutional filters, we keep the *teacher* architecture simple instead of considering deeper and more complex architectures like ResNet and InceptionTime despite their demonstrated performances for TSC. Hence, our *teacher* FCN model includes three convolutional blocks where each block contains three operations. First, 1D convolutions are applied on input time series, followed by a Batch Normalization. Then, the result is fed to a ReLU activation. In all blocks, 1D convolutions have a stride equal to 1 with a zero padding preserving lengths of time series. The first convolutional block contains 128 filters with a kernel size equal to 8. The second block employs 256 filters of size 5. The last convolutional block is composed of 128 filters with a filter length of 3. The output of this third block is averaged along the temporal dimension using a Global Average Pooling (GAP) and finally fed to the Softmax classifier. The architecture of the *teacher*, including 267 018 parameters, is depicted in Figure 1 and summarized in Table I.

D. Student models

For our *student* models, we followed the same idea of FCN but with different hyper-parameters to evaluate their impacts. For that, we build different versions of our *teacher* models as summarized in Table I. For evaluating the impact of the filters, we first consider smaller *student* models, denoted as Model_F#, with three layers but with fewer convolutional filters. For assessing the impact of the number of layers, two *student* models (Model_L#) are then built with one layer and two layers, respectively but with a similar number of filters than the *teacher* model. Finally, inspired by what is done in computer vision, we also investigate the use of depthwise separable convolutions [31] instead of traditional convolutions in the two *student* models (Model_DSC). In depthwise separable convolution, the convolution operation is factorized into two steps. In the first step, a depthwise convolution is applied to each input channel to capture temporal correlations. Then, the second step applies a 1×1 pointwise convolution to combine

TABLE I: Configurations of our *teacher* and *student* FCN architectures

Models		Filters # layer 1	Filters # layer 2	Filters # layer 3	Total param. #
teacher		128	256	128	267 018
student	Model_F64	64	128	64	67 978
	Model_F32	32	64	32	17 610
	Model_F28	28	56	28	13 618
	Model_F24	24	48	24	10 138
	Model_F20	20	40	20	7 170
	Model_F16	16	32	16	4 714
	Model_F12	12	24	12	2 770
	Model_F8	8	16	8	1 338
	Model_F4	4	8	4	418
	Model_L2	128	256	-	169 354
	Model_L1	128	-	-	2 954
	Model_DSC	128	256	128	70 930

the output of depthwise convolution and capture channel-wise correlations.

IV. Experimental evaluation

In this section, we propose to quantitatively evaluate the use of knowledge distillation for TSC. After describing the employed experimental setup, we analyze the impact of both number of layers and number of filters in fully convolutional network architectures, as well as the replacement of traditional convolutions by depthwise separable convolutions.

A. Experimental setup

1) Data: For all our experiments, in order to generally assess the impact of knowledge distillation for TSC, we employ the UCR Archive 2018 [16]. The archive consists of 128 univariate time series datasets coming from various domains and sensors and with very different characteristics (time series length, number of samples, etc.). Among these datasets, the number of classes varies from 2 to 60. For a fair evaluation analogous to state-of-the-art, we use the original train/test split. Moreover, all time series are z-normalized. Following the state-of-the-art for a fair comparison, we discard datasets containing series of unequal length or missing values, as well as the Fungi dataset, which only provides a single train case for each class label. As a result, we consider 112 univariate time series datasets.

2) Experimental protocol: For each dataset, we first train a FCN *teacher* model five times and keep the best model for distilling knowledge to the *student*. We note that to select such a best model, we consider the best loss value obtained on the training set to keep experiments reproducible in a real world context. We then use the best *teacher* model to train a smaller *student* model following the procedure described in section III-B. Different *student* models are trained according to the hyper-parameter that we modify (number of filters, number of layers or use of depthwise separable convolutions), as detailed in Table I. For a given *student* model configuration, we also train in parallel a similar model from scratch, without considering knowledge distillation from the *teacher* model. We denote this model *studentAlone* to differentiate from the *student* leveraging knowledge distillation. In order to be less dependent on random initialization of FCN models, we train both *student* and *studentAlone* models five times. Finally, for all our models, average accuracy and standard deviation among the five runs are considered for comparing the performances.

3) Comparison protocol and metrics: For each *student* architecture configuration, we compare the performance on the whole UCR Archive with first both the *teacher* and *studentAlone* models, and second with only the *studentAlone* model. For that, we consider the number of datasets where a particular model wins, i.e. obtains a higher classification accuracy than other models. Similarly, the number of ties and losses are also counted. In addition, as we are also interested in assessing the robustness of models to random initialization, we do the same win/tie/loss computation by considering accuracy's standard deviation among the five runs of each model. Conversely to classification accuracy, a target model wins if its corresponding standard deviation is lower than other models.

4) Implementation details: All our models are built using Keras framework. Except for the number of layers and the number of filters that are varying according to the architectures, we are following the same configuration as [10] in all our FCN models. Kernel sizes are set to 8, 5 and 3 for the first layer, second layer (if present) and third layer (if present), respectively. As optimizer, we adopt the Adam algorithm with a reducing factor of 0.5, a patience of 50 and an initial learning rate of 0.0001. During training, we use a batch size of 16. For knowledge distillation, we set the temperature τ to 10 and weight factor λ to 0.1. All experiments were conducted on a Desktop with Ubuntu 20.04 OS, a AMD Ryzen 9 5950X 16-Core Processor with 64 GB of RAM, and a Nvidia RTX 3090 GPU. Our code is publicly available[1].

B. Impact of fewer convolutional filters

We first propose to analyze the impact of the number convolutional filters in the *student* FCN architectures. Comparative results considering the number of wins, ties and losses based on the classification accuracies are reported in Table II. Moreover, the number of wins for each model is depicted in Figure 2. We can first observe that if we include the *teacher* model in the comparison, it not surprisingly obtains better results on average than all configurations of both student models. This is further depicted in Figure 2a emphasizing the number of wins for each model with various number of filters. However, we can also notice that, except when compared to very small student models, the *teacher* model wins on less than half of the datasets. This shows the large variability in the UCR archive where it is difficult to identify a particular architecture with a fixed number of filters performing well on all datasets. As seen in the third and fourth group of columns of Table II, a *student* model with less convolutional filters is able to win against a larger *teacher* model for about a third of datasets. This can be explained by the fact that smaller models are less prone to overfit.

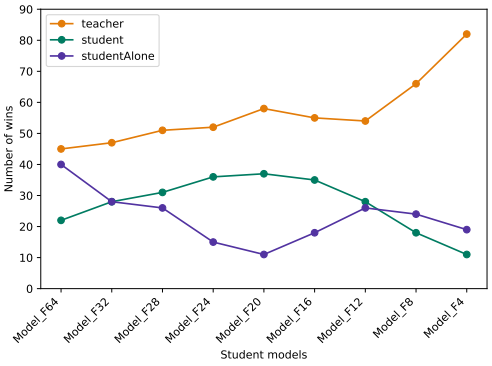
Then, we focus on the comparison of only *student* and *studentAlone* models for evaluating the impact of knowledge

[1] https://github.com/maxwell1503/kdFCN-4-tsc

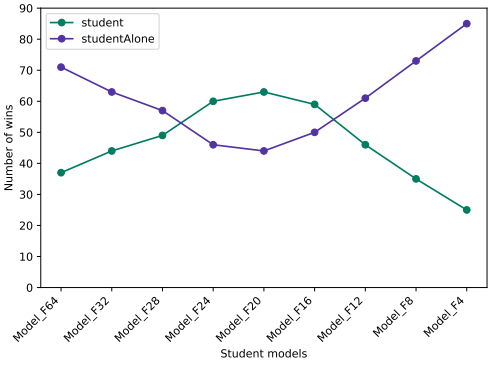
Models	Comparison of three models									Comparison of two *student* models only					
	teacher			*student*			*studentAlone*			*student*			*studentAlone*		
	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss
Model_F64	**45**	4	**63**	22	4	86	40	4	68	37	4	71	**71**	4	**37**
Model_F32	**47**	7	**58**	28	6	78	28	8	76	44	5	63	**63**	5	**44**
Model_F28	**51**	3	**58**	31	4	77	26	4	82	49	6	57	**57**	6	**49**
Model_F24	**52**	6	**54**	36	5	71	15	9	88	**60**	6	**46**	46	6	60
Model_F20	**58**	4	**50**	37	5	70	11	6	95	**63**	5	**44**	44	5	63
Model_F16	**55**	3	**54**	35	4	73	18	3	91	**59**	3	**50**	50	3	59
Model_F12	**54**	3	**55**	28	4	80	26	4	82	46	5	61	**61**	5	**46**
Model_F8	**66**	3	**43**	18	4	90	24	3	85	35	4	73	**73**	4	**35**
Model_F4	**82**	0	**30**	11	0	101	19	0	93	25	2	85	**85**	2	**25**

TABLE II: Win/Tie/Loss comparison of *teacher*, *student* and *studentAlone* models with various numbers of filters.

distillation. By analyzing the fifth and sixth group of columns of Table II and Figure 2b, we can clearly identify three main distinct parts. First, for a larger number of filters, we can see that a *studentAlone* model without any knowledge distillation is performing better than a distilled *student* model. This can be explained by the fact that as models still have a large number of filters, they are able to capture discriminant features through convolution operations and thus obtain quite competitive results with the *teacher*. Moreover, these results also show that constraining the distilled *student* model to mimic the *teacher* model does not allow learning different features to improve the performance. Second, for an intermediate number of filters, we can see that the distilled *student* model is able



(a) Comparison of three models



(b) Comparison of two *student* models only

Fig. 2: Number of wins considering classification accuracy according to various numbers of filters of student models.

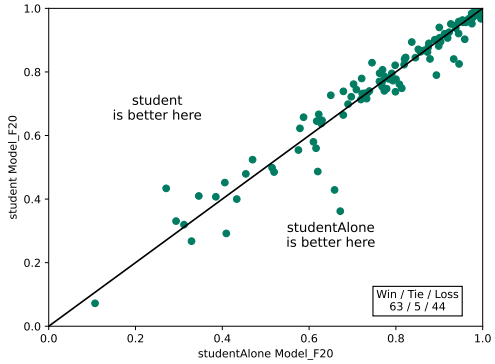


Fig. 3: Accuracy plot showing how the *student* Model_F20 is performing better than *studentAlone* Model_F20.

to leverage *teacher*'s knowledge to obtain better performances than the *studentAlone* model. This shows that in the case of intermediate models, knowledge distillation allows to reduce the performance loss due to the models complexity decrease. Hence, the best *student* configuration leveraging knowledge distillation is the architecture including 20 filters in the first and third layers and 40 filters in the second layer. For this particular configuration, Figure 3 depicts the accuracy plot of the *student* model against the *studentAlone* model for each of the 112 UCR datasets. Finally, when the number of filters is low, we can notice that the *student* model is not complex enough to capture the *teacher*'s knowledge. Thus, it results in very poor results compared to the *studentAlone* model.

In addition, we propose to consider the standard deviation among different runs for each *student* model. Figure 4 illustrates the number of wins considering standard deviation for both *student* models with different numbers of filters. We can observe that except for the larger and smaller models where performances are competitive, the distilled *student* models result in lower standard deviation for the majority of datasets. This shows that knowledge distillation allows to make *student* models much more robust to random initialisation.

C. Impact of fewer convolutional layers

We then propose to assess the impact of the number of convolutional layers in our knowledge distillation framework. Table III includes the number of wins, ties, and losses considering classification accuracy for the *teacher* model and *student*

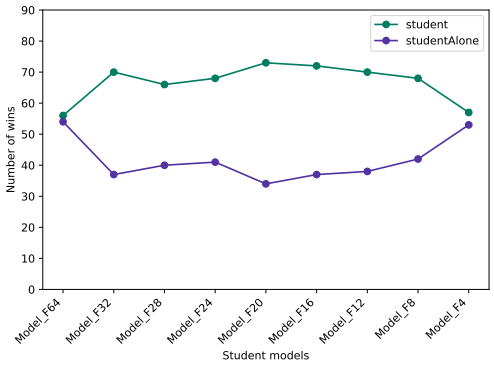


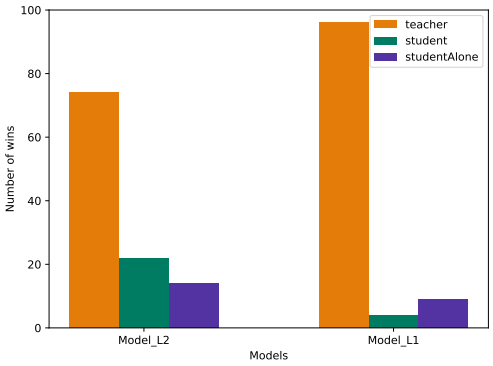
Fig. 4: Number of wins considering standard deviation according to various numbers of filters of *student* models.

models with 1 and 2 layers. Moreover, the number of wins is depicted in Figure 5. Similarly to the number of filters we consider both cases where the *teacher* is included or not in the comparison. As expected, we can observe that the *teacher* model with three layers performs largely better than both student models with fewer layers with a number of wins equal to 74 and 96 against *student* models with two layers and one layer, respectively. By comparing only *student* models (fifth and sixth group of columns in Table III), we can notice that the distilled *student* model with two layers performs slightly better than the *studentAlone* model, with a total number of wins equal to 58. However, if only one layer is included in the model the classification performance of the distilled *student* model drops significantly. This is inline with the observations made in Section IV-B. A model with an intermediate complexity can leverage the knowledge of a larger *teacher* model. However, if the model complexity is too low, it is not able to mimic the *teacher*'s behavior.

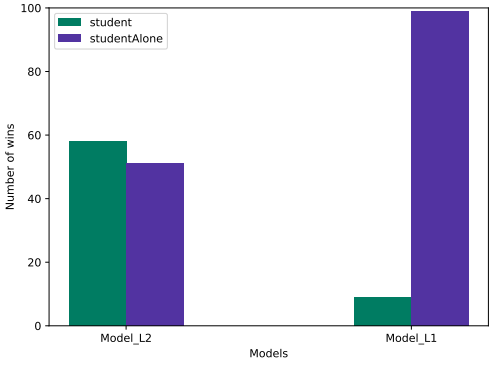
In addition, similarly to the number of filters, we assess the impact of varying the number of layers on the robustness of *student* models to random initialization. Number of wins considering standard deviation of both *student* models are depicted in Figure 6. We can notice that the robustness of the distilled model is much higher compared to the *studentAlone* for both configurations with one and two convolutional layers. This confirms the intuitions drawn in the previous section that knowledge distillation improves the robustness of *student* models to random initialization.

D. Impact of depthwise separable convolutions

Finally, we analyze the impact of using depth separable convolutions instead of traditional convolutions in *student* architectures in our knowledge distillation scheme. Comparative Win/Tie/Loss results are reported in Table IV and number of wins are depicted in Figure 7. By comparing with the *teacher* model, we can see that while the replacement of traditional convolutions by depth separable convolutions significantly affects the performances, its impact is mitigated by the use of knowledge distillation. This is further emphasized by analyzing the fifth and sixth group of columns of Table IV



(a) Comparison of three models



(b) Comparison of two *student* models only

Fig. 5: Number of wins considering classification accuracy according to various numbers of layers of *student* models.

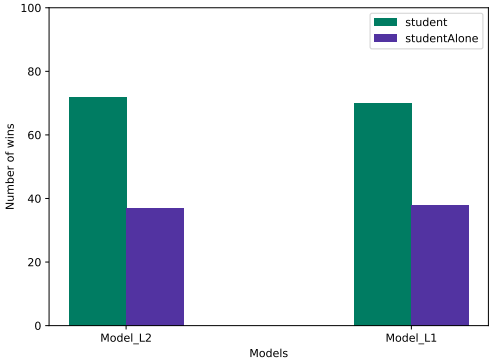


Fig. 6: Number of wins considering standard deviation according to various numbers of layers of *student* models.

where we can see that by only comparing the performances of *studentAlone* and *student* models, the latter obtains better classification performances on 62 datasets among 112 datasets from the UCR Archive. In addition, like other experiments, we also compare the robustness of both architectures by comparing their standard deviations among multiple runs' performances. Like other experiments, the *student* model is more robust with a total number 65 datasets where it obtains lower standard deviation than the *studentAlone* model.

Models	Comparison of three models									Comparison of two *student* models only					
	teacher			*student*			*studentAlone*			*student*			*studentAlone*		
	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss
Model_L2	**74**	2	**36**	22	2	88	14	2	96	**58**	3	**51**	51	3	58
Model_L1	**96**	2	**14**	4	1	107	9	3	100	9	4	99	**99**	4	**9**

TABLE III: Win/Tie/Loss comparison of *teacher*, *student* and *studentAlone* models considering classification accuracies according to various number of layers in both *student* and *studentAlone* models.

Models	With *teacher*									Without *teacher*					
	teacher			*student*			*studentAlone*			*student*			*studentAlone*		
	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss
Model_DSC	**71**	3	**38**	21	2	89	17	2	93	**62**	3	**47**	47	3	62

TABLE IV: Win/Tie/Loss comparison of *teacher*, *student* and *studentAlone* models considering classification accuracies when depth separable convolutions are employed in convolutional layers of both *student* and *studentAlone* models.

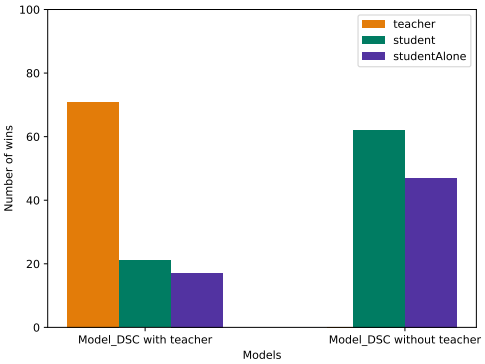


Fig. 7: Number of wins considering classification accuracy when using depth separable convolutions in student models.

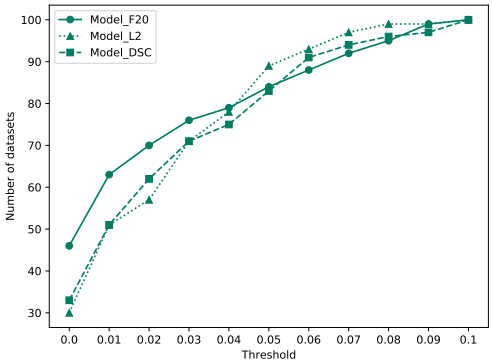


Fig. 8: Number of datasets for which the performance difference between the *student* and the *teacher* is positive or superior to a threshold

E. Impact of reducing models complexity

Previous experiments suggest that employing knowledge distillation on smaller *student* models with fewer parameters is beneficial in the majority of cases. However, they also show that even with knowledge distillation, reducing the number of parameters of *student* models results in lower classification performances on most of the time series datasets from the UCR Archive. Here we propose to assess the impact of reducing the number of parameters in the *student* model on its performance. For that, we consider the following *student* architectures: Model_F20, Model_L2 and Model_DSC. We then compare their performance with respect to the *teacher* architecture by counting the number of datasets for which the performance difference between the *student* and the *teacher* is positive or superior to a threshold. For instance, for a threshold of 0.02, every dataset for which the *student* performs better than the *teacher* or the deviation between both models' performances is lower than 2% is considered. Results for a threshold varying from 0.0 to 0.1 are depicted in Figure 8. A threshold of 0.0 means that only cases where the *student* obtains better results than the *teacher* are counted. Obviously, the more we relax the threshold, the more cases are considered. We can observe that for high values of the threshold, performances of the three compared *student* models are quite similar while for lower values, Model_F20 seems to be more efficient, despite its lower complexity. Moreover, we can see that if we accept a maximum performance loss of 4%, the use of knowledge distillation is beneficial for about two thirds of the datasets for all the three compared models. This seems to be a good trade-off between the model's performance and complexity.

V. Conclusion

In this paper, we have proposed a first study of the use of knowledge distillation in Fully Convolutional Network for TSC. We assessed the impact of reducing the number of parameters in *student* models while leveraging *teacher* performance through knowledge distillation. In particular, we investigate the reduction of the number of convolutional filters and convolutional layers, as well as the use of depthwise separable convolutions instead of traditional convolutions. The experimental evaluation carried out on the UCR archive 2018 suggests that intermediately complex *student* architectures can benefit from a deeper *teacher*'s knowledge. This is in line with the observations made in [19] for image classification. Among the three assessed hyper-parameters, reducing the number of convolutional filters seems to be the most suitable when combined with knowledge distillation. In particular, experiments showed that the Model_20 *student* architecture allows to significantly reduce the total number of parameters

by a factor of about 38 while preserving relatively good performances in comparison to a more complex *teacher* model. However, conversely to image classification where knowledge distillation is evaluated on few datasets, we considered 112 time series datasets showing that finding a suitable architecture for every type of time series is not trivial. Nevertheless, we note that our experiments showed that knowledge distillation has a major impact on *student* models robustness to random initialisation among different runs. This is particularly crucial when models are deployed in real-world scenarios.

Finally, as we consider a FCN model which is not a very complex model, we believe that the impact of knowledge distillation may be even more interesting for larger models like ResNet and InceptionTime. This is particularly true for the latter as it includes an ensemble of multiple complex models. Hence, considering a single smaller *student* model leveraging the knowledge of an ensemble is part of our future work.

ACKNOWLEDGMENT

This work was supported by the ANR TIMES project (grant ANR-17- CE23-0015) 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 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. The authors would also like to thank the creators and providers of the UCR archive.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84 – 90, 2012.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding." in *NAACL-HLT (1)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186.

[3] J. Huang, C. Huang, L. Yang, and Q. Zhang, "A novel ecg signal classification algorithm based on common and specific components separation," in *Int. Conference on Pattern Recognition and Artificial Intelligence*. Springer, 2020, pp. 583–595.

[4] J. Priesmann, L. Nolting, C. Kockel, and A. Praktiknjo, "Time series of useful energy consumption patterns for energy system modeling," *Scientific Data*, vol. 8, no. 1, pp. 1–12, 2021.

[5] M. Devanne *et al.*, "Multi-level motion analysis for physical exercises assessment in kinaesthetic rehabilitation," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 529–534.

[6] M. Chelali, C. Kurtz, A. Puissant, and N. Vincent, "Spatio-temporal stability analysis in satellite image times series," in *International Conference on Pattern Recognition and Artificial Intelligence*. Springer, 2020, pp. 484–499.

[7] 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*, pp. 1–53, 2021.

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

[9] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of AAAI*, 2021.

[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, p. 917–963, Mar 2019.

[11] L. J. Ba and R. Caruana, "Do deep nets really need to be deep?" *arXiv preprint arXiv:1312.6184*, 2013.

[12] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[13] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[14] S. Shakeri, A. Sethy, and C. Cheng, "Knowledge distillation in document retrieval," *arXiv preprint arXiv:1911.11065*, 2019.

[15] R. Takashima, S. Li, and H. Kawai, "An investigation of a knowledge distillation method for ctc acoustic models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5809–5813.

[16] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML, "The ucr time series classification archive," October 2018.

[17] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4133–4141.

[18] H. Oki, M. Abe, J. Miyao, and T. Kurita, "Triplet loss for knowledge distillation," in *2020 International Joint Conference on Neural Networks*. IEEE, 2020, pp. 1–7.

[19] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5191–5198.

[20] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[21] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for bert model compression," *arXiv preprint arXiv:1908.09355*, 2019.

[22] Q. Xu, Z. Chen, M. Ragab, C. Wang, M. Wu, and X. Li, "Contrastive adversarial knowledge distillation for deep model compression in time-series regression tasks," *Neurocomputing*, 2021.

[23] Z. M. Ibrahim, D. Bean, T. Searle, H. Wu, A. Shek, Z. Kraljevic, J. Galloway, S. Norton, J. T. Teo, and R. J. Dobson, "A knowledge distillation ensemble framework for predicting short and long-term hospitalisation outcomes from electronic health records data," *arXiv preprint arXiv:2011.09361*, 2020.

[24] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 565–592, 2015.

[25] M. Baydogan, G. Runger, and E. Tuv, "A bag-of-features framework to classify time series," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2796–2802, 2013.

[26] 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," *arXiv preprint arXiv:2104.07551*, 2021.

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

[28] 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*. IEEE, 2017, pp. 1578–1585.

[29] H. I. 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.

[30] 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.

[31] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.