Data Augmentation for Time Series Classification with Deep Learning models

Gautier Pialla(✉), Maxime Devanne, Jonathan Weber, Lhassane Idoumghar, and Germain Forestier

IRIMAS, Université de Haute-Alsace, Mulhouse, France
firstname.lastname@uha.fr

Abstract. Deep Learning models for time series classification are benchmarked on the UCR Archive. This archive contains 128 datasets. Unfortunately only 5 datasets contain more than 1000 training samples. For most deep learning models, this lead to over-fitting. One way to address this issue and improve the generalization of the models is data augmentation. Although it has been extensively studied and is widely used for images, fewer works have been done on time series. InceptionTime is an ensemble of 5 Inception classifiers and is still regarded as the state-of-the-art deep learning model for time series classification. However, most of the work on data augmentation were not done on the Inception classifier. In this paper we solve this issue by studying 4 different data augmentation methods through 4 experiments on the Inception model. We studied trainings with one or several augmentations at the same time and with or without generating new samples at each epoch. We also conducted experiments with ensembling and benchmarked our results on the UCR Archive. We showed that using a combination of both the scaling and window warping data augmentation methods, we can significantly improve the accuracy of Inception and InceptionTime models.

Keywords: data augmentation · time series · scaling · window warping · Inception

1 Introduction

As deep learning models get deeper and deeper, training them has become a real challenge. The training part is based on statistics. A distribution of the data is learned on a training set. The aim is to generalize on a test set, unseen by the model. *Generalization* is the main challenge for the training as we expect the model to perform as well on unknown data as on the training set. If a model performs better on the training set than on the test set, we say that it *over-fits*. For this reason, in addition to an important computing power, the training needs quality and quantity of data. This raises the question "For a given task how much data do we need ?". In the literature, this problem is called *sample complexity* as the answer depends on both the complexity of the problem and that of the chosen model. In computer vision, state-of-the art models are benchmarked on huge

This is the author's version of an article published at ECML/PKDD Workshop AALTD 2022.

datasets like ImageNet[3], MS-COCO[14] or Open Image[12]. These datasets contain thousands of samples for each different class. Regarding time series, most of the models are benchmarked on the UCR Archive [1]. The latest version of this archive contains 128 datasets regrouped in 7 categories such as Sensor, ECG, or Devices. The downside is their few number of training samples, and the lack of validation sets. As a matter of fact only 5 of them contain more than a 1000 training instances. Thus, it is hard for the models to generalize well because of the over-fitting. In order to improve further generalization and reduce over-fitting, different strategies can be carried out in machine learning. Among them, one can reduce the complexity of a model by simply removing layers or reducing the number of neurons. *Regularization* consists of adding a penalty term to the loss function. Usually, the chosen penalty is the L1 or the L2 norm. Regarding the weigths, the L1 norm minimizes their absolute value while the L2 norm minimizes their squared magnitude. *Early stopping* is also a form of regularization. It entails stopping the training before the appearance of over-fitting. It is widely used for training deep learning models. Differently, *Ensembling* combines several weak independent models in order to obtain a stronger model. Ensembles can have different size, and be composed of different models. It exists several ways to create ensembles. An easy and effective one is to average to predictions of the different models. Another regularization method consists of using more data. One can either collect new data, or increase artificially the amount of training data by creating of modifying existing samples. This is called *data augmentation* (DA). Using synthetic data is particularly useful when it is too difficult to obtain new data. Such data can be created in a realistic way using algorithms, or deep models like Variational Encoders or Generative Adversarial Networks. Most of the time, the augmented data is simply a copy of the training set to which has been applied random transformations. Commonly, for images these transformations are color changes, rotations, zooming, blurring or cropping parts of the images. Increasing the dataset in such a way, helps to provide more context. InceptionTime [8], introduced in 2019, is considered as the state-of-the-art deep learning time series classifier. It is an ensemble of 5 Inception networks. To reduce over-fitting, beside creating an ensemble, Fawaz et al. used early stopping during the training of each separate network. Despite that, InceptionTime still over-fits most of the UCR datasets. To overcome this issue, data augmentation appears like an interesting solution. Unlike in computer vision, where this is widely used, few time series classifiers are using it during training. Iwana et al. [9] benchmarked 12 data augmentation methods over 6 time series classifiers. Some of these methods were adapted from computer vision, but other are specific for time series. As this survey was not conducted on Inception networks and InceptionTime ensembles, we propose to investigate the use of data augmentation on this state-of-the-art architecture. After having selected 4 promising data augmentation methods from [9], our main contributions are:

– We benchmarked them on both Inception and InceptionTime over the entire UCR Archive.

– We went further than existing papers by testing the methods independently but also combined together.
– We created ensembles of Inceptions networks trained with the same data augmentation method but also with different ones.
– Finally, we showed how data augmentation can significantly improve the accuracy of both architectures.

2 Related Work

The aim of using data augmentation is to reduce over-fitting, by improving the generalization of a deep learning model, thus also improving it's accuracy. Many well-known networks have used it to boost their performances and become the state-of-the-art. Among them, AlexNet [11] in 2012 used cropping, mirroring, zooming, blurring and rotation. VGG [16] in 2014 used scaling and cropping. The inception networks [17] introduced in 2014 used cropping and mirroring. ResNet [7] has used cropping and mirroring.

Hence, data augmentation is not a new technique and has been extensively studied for computer vision. Most of the previously cited architectures have been adapted for time series classification. They have given FCN [20], ResNet [20] and InceptionTime [8]. These architectures have all achieved state-of-the-art performances, but unlike their counterparts in computer vision, they have not used data augmentation.

Regarding time series, data augmentation has not been explored in depth until recently. Indeed, two surveys benchmarked most of the existing methods. The first one from Iwana et al. [9] benchmarked 12 data augmentation methods over 6 different classifiers. Similarly, the second study by Weng et al. [21] compared data augmentation methods, but considered not only classification but also anomaly detection and prediction.

Out of the benchmarked methods, most of them are adaptations from the ones used in computer vision. *Jittering* consists of adding to each time step of a time series a random Gaussian noise. *Window slicing* [13] is the process of extracting sub-parts or windows from a time series. Each part is then classified. The majority class obtained within the patches is then assigned to the original series. *Flipping* inverts a time series. Some of these techniques can only be used for specific datasets. Indeed, it does not make sense to apply flipping for every kind of time series, like ECGs. Similarly, for Window slicing, if the discriminative parts are not present in each slice, it can result in false predictions. For this reason, some methods were specifically designed for time series. Guided Warping techniques like *Random Guided Warping* and *Discriminative Guided Warping* [10] use DTW to warp patterns from one time series to another. Forestier et al. [6] introduced weighted DBA which averages a set of time series to create a new one.

Instead of using and modifying existing time series, other methods focus on creating synthetic data. Generative Adversarial Networks (GANs) and Variational Auto-Encoders (VAEs) are popular ways to do so. Regarding GANs,

TimeGAN [22], Recurrent Conditional GAN [5] or Continuous recurrent neural networks [22] are adapted for time series. Regarding auto-encoders we can cite TimeVAE [4], the use of masked autoencoders [23], or the averaging of time series using auto-encoders [18]. Differently, Pialla et al. [15] designed a smooth adversarial attack and showed through adversarial training how adversarial samples can also be used to improve the robustness of deep models.

Although these generative methods can be used as some kind of data augmentation, they require to train new models and often to fine tune them for each dataset. Thus, they do not represent a general and quick approach to implement.

3 Proposed approach

3.1 Data Augmentation Methods used

In this paper, we took 4 methods from [9]. We selected the methods that were the most recommended across all deep learning models to asses if they also generalize on the Inception models.

RGW Random Guided Warping (RGW) has been introduced in [10]. For a given class, two different patterns are selected. Then, these patterns are randomly switched between the samples. The warping between the samples is computed using the DTW algorithm.

DGW Discriminative Guided Warping (DGW) was also introduced in [10] and is similar to RGW. This time, only the most discriminative pattern is selected inside a batch.

Scaling The scaling method multiplies a time series with a random scalar α. It can be taken from a random Gaussian distribution $\alpha \sim \mathcal{N}(1, \sigma^2)$. We use the same parameters as in [19]: $\mu = 1$ and $\sigma = 0.1$. This method modifies the magnitude of the time series. An example of a scaled time series is displayed in Figure 1.

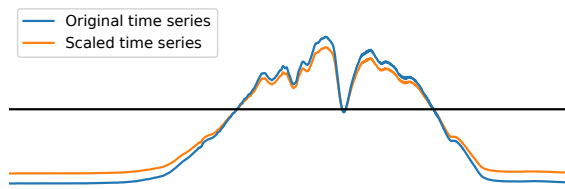

Fig. 1. Example of scaling a time series from the EthanolLevel dataset.

Window Warping Window warping as been introduced in [13]. It consists of selecting a random window. The window is then sped up by a factor 2 or slowed down by a factor 0.5.

We used a window size equal to 10% of the TS length. An example of Window Warping can be seen below in Figure 2.

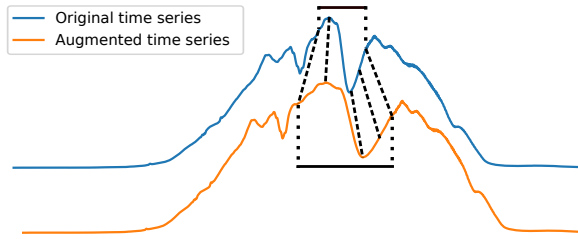

Fig. 2. Example of Window Warping. The part of the original time series in blue has been increased two times.

3.2 Models used

The Inception network was first introduced by Fawaz et al. in [8]. It is an adaptation for time series from the famous Inception architecture in computer vision. The model is composed of 6 inception modules, a Global Average Pooling (GAP) and a Dense layer. The architecture is represented in Figure 3 and the inception modules in Figure 4.

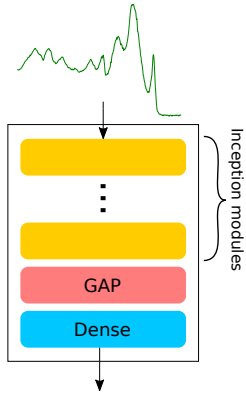

Fig. 3. Inception network.

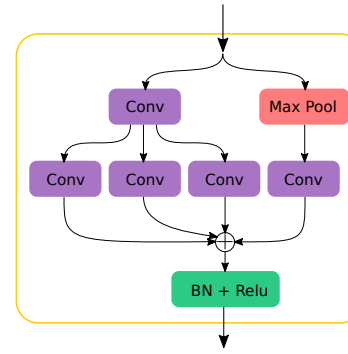

Fig. 4. Inception module.

Each inception module, is composed of several convolutions with different kernel sizes. All the convolutions outputs are concatenated and followed by a Batch Normalization layer and a Relu activation function. The intuition behind the inception module is to make the network wider instead of deeper and prone to over-fitting.

3.3 Ensembling of Inception networks

InceptionTime [8] is an ensemble of 5 Inception classifiers. Today, it is still regarded as the state-of-the-art deep learning model for time series classification. The ensemble is done by averaging the softmax predictions made by the 5 Inception networks. Thus, for a given input $\mathbf{x}$, the final prediction of the ensemble of n models is:

$$\hat{y} = \arg\max \left(\frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x}) \right) \tag{1}$$

In this paper, we use InceptionTime to benchmark our ensembling results. We did several kinds of ensemblings: ensemblings of Inception models trained with the same data augmentation method, ensemblings of Inception models trained with different data augmentation methods and finally ensemblings of Inception models trained with several data augmentation methods.

3.4 ROCKET

ROCKET [2], is a non-deep model for time series classification. It is composed of random, unlearnt convolutions. Using theses convolutions, features are extracted using the percentage of positive values (PPV). Finally, the classification is done using a linear classifier. This model is really famous because of its accurate classifications and its exceptionally fast training time, hence the name. As ROCKET is a state-of-the-art time series classifier, we proposed to use it as a benchmark for our best models.

4 Experimental Setup

4.1 Datasets

All our experiments where conducted on the UCR Archive[1]. This archive is commonly used by the time series community to benchmark methods or deep learning classifiers. The 2018 version of the UCR Archive contains 128 different time series datasets. Instead of using the whole archive, many research papers only use a subset of 112 datasets. The reason is that 15 datasets are unequal in length and one, Fungi, has a single instance per class in the training set. Regarding the datasets with unequal length, we decided to use a padded version provided by the same authors. For the Fungi dataset, we could not used it, as the methods RGW and DGW require several training samples of the same class. Thus, in total, we used 127 datasets.

4.2 Implementation

For this work we have reused the code and data from several sources. Regarding the data augmentation, we used the implementation provided by [10]. We used the code and results of Inception and InceptionTime as presented in [8] and the results of ROCKET [2]. As we used several open source codes, we also made freely available all the code and results in our companion GitHub repository[1].

4.3 Protocol and parameters

If not specified otherwise, we use the same following parameters for each experiment. All trainings where realized using the Adam optimizer during 900 epochs. At each epoch, the training set was randomly shuffled. The objective was to mix the original training samples and the augmented ones. Regarding the training data, we use a mix of original samples and augmented ones. Each original sample has its augmented counterpart generated by each data augmentation method. Thus, if we use 1 data augmentation method, the training set will be twice the size of the original dataset. If we use 2 methods, the amount of training data used is 3 times the size of the training set, and so on. Each training was conducted 5 times. Having several iterations helps to have more consistent results. Indeed the random initialization of the models and the stochasticity of some data augmentation method can have an impact on the trainings. The results presented in section 5, are always an average over the 5 runs. In order to compare the data augmentation with InceptionTime, we created ensembles out of the 5 runs. Theses ensembles average the predictions made by each model composing it. It is the same process used by InceptionTime. These ensembles are unique, and thus, the results presented for those are not an average over several runs.

5 Experiments

5.1 Experiment 1

In the first experiment, we aim to compare the vanilla Inception model, as presented in [8], to Inception models trained with data augmentation.

Results Figure 5 represents a critical diagram. Such diagrams are useful when comparing multiple methods over several datasets, as it is the case here. The methods are ordered given their average rank over the datasets. A thick horizontal line links a set of classifiers that are not significantly different, according to the Wilcoxon-Holm analysis. With the Figure 5, we learn that for the Inception classifier, the data augmentations Scaling and Window Warping are both significantly better than DGW and RGW. DGW is the only method significantly worst than the vanilla Inception classifier. Both RGW and Window Warping are not significantly different than the vanilla classifier. Only Scaling is significantly

[1] https://github.com/Gpialla/DataAugForTSC

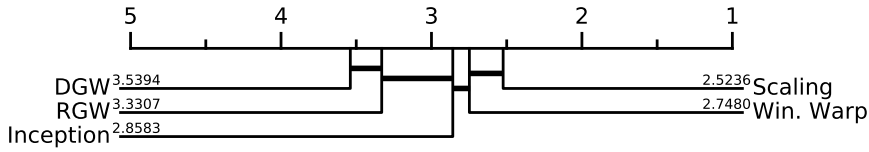

Fig. 5. CD diagram of experiment 1

better than the vanilla classifier. Globally, the Guided Warping methods seems in average to degrade the performances. However these methods are not useless. Indeed, for respectively 43 and 55 datasets, DGW and RGW are still better than the vanilla classifier.

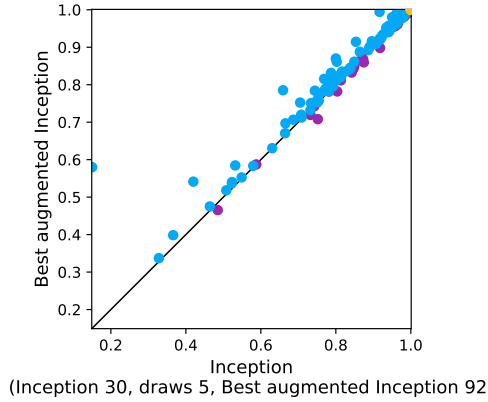

Fig. 6. Pairwise diagram: Inception vs best Inception with augmentation for each dataset. Each dot represent a single dataset. Blue dots represent datasets whose accuracy is improved by at least one data augmentation. Gold dots represent datasets whose accuracies have not changed (same as Inception).

The Figure 6 compares for each dataset, the vanilla Inception model to the best augmented Inception model. We can observe that for 92 datasets, using one of the four data augmentation methods improves the accuracy of the Inception model. The vanilla Inception is the best method for only 30 datasets. This shows the relevance of data augmentation for time series. Like for image classification, most of the time series datasets can leverage the use of data augmentation. Using the best data augmentation method improves the average accuracy by +1.40%. The best improvement regards the DodgerLoopDay dataset with an improvement of +43.00% using the RGW data augmentation. For this specific

dataset, all data augmentation methods, significantly improve the accuracy, with an average of +41.75%.

Results with ensembling Following the previous results, we propose to analyze the impact of data augmentation over ensembling. For each method, we created ensembles in the same way as InceptionTime, by averaging the predictions made by the 5 iterations.

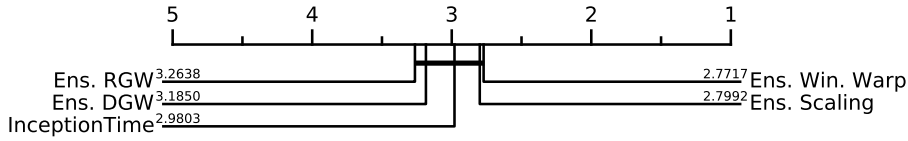

Fig. 7. CD diagram of experiment 1 with ensembling

In Figure 7 we can see that the methods that were performing the best individually, Scaling and Window Warping, are the ones that provide the best ensembles in term of rank. After ensembling, the rank of the data augmentation methods is conserved. However, all methods are not significantly different and none is significantly better than InceptionTime. Creating an ensemble will smooth and improve the individual performance of the models that compose it. Thus, it is harder to observe any difference between the augmentation methods.

Results with ensembling of several data augmentation methods Previously, we created ensembles using the five trainings of each data augmentation method. Another way to make ensembles is to use different methods. Here we created five ensembles. Within each ensemble, we used four models, each one trained using a different data augmentation method. In the following results, the accuracy of the five ensembles is averaged.

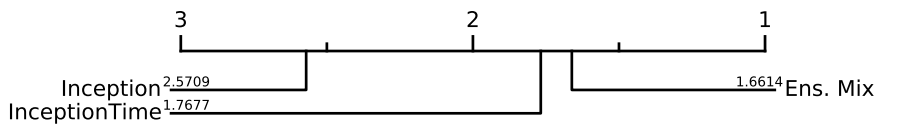

Fig. 8. CD diagram of experiment 1 with ensembling of several data augmentation methods

We called this ensemble Ens. Mix. We can see in Figure 8 that it significantly outperforms InceptionTime. It seems that using different methods inside the

same ensemble leads to better results. Even if the gap between these ensembles and InceptionTime is not huge, this result is important. Before, we should have had to benchmark and select a single data augmentation method, but now we can simply use them all within a single ensemble.

Table 1. Average accuracy

Method	Avg. Accuracy
DGW	83.55
RGW	83.51
Scaling	83.99
Win. Warp.	**84.06**
Inception	83.48
Ens. DGW	84.47
Ens. RGW	84.53
Ens. Scaling	84.73
Ens. Win. Warp.	**85.13**
Ens. Mix	84.86
InceptionTime	84.24

In Table 1 is recorded the average accuracy for each method presented so far. We observe that all data augmentation method improve the accuracy over their competitor Inception or InceptionTime. Regarding this metric, Window Warping is provide the best results with and without ensembling.

5.2 Experiment 2

This second experiment aims to put in light the importance of the original training set. As all data augmentation methods improve the accuracy of the Inception model, is it efficient to train a model using only augmented data? We reproduced the experiment 1, with the same parameters, but only using augmented data.

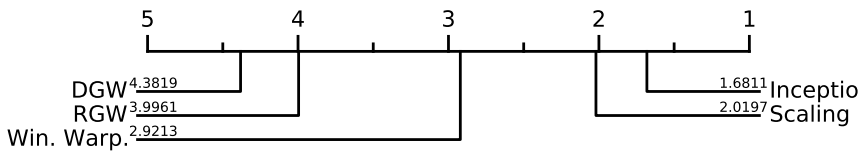

Fig. 9. CD diagram of experiment 2

Results Figure 9 shows that all methods trained only on augmented data, performs worse than before. The vanilla Inception is significantly better than all

of them. The models trained with DGW and RGW are the ones which suffer the most, with a loss in accuracy of respectively -9.63% and -7.89%.

These results underline the importance of the original data. Augmented data can not be used as a substitution of the original data but should be used as a complement.

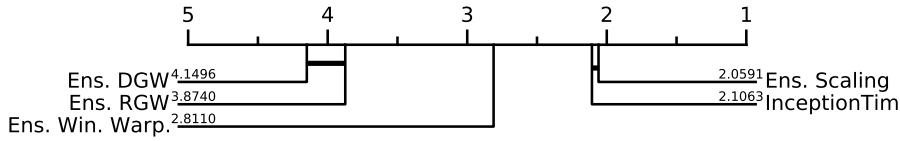

Fig. 10. CD diagram of experiment 2 with ensembling

Results with ensembling The previous results are also reflected with ensembling. Only the Scaling ensemble, despite not having been trained on the original data, remains competitive with InceptionTime. The intuition behind this is that Scaling produces augmented samples that are close to the original data. Thus, even without the original data, the model can still generalize over the test set.

5.3 Experiment 3

In image classification, data augmentation is randomly applied at each epoch. As the model never sees several times the same augmented sample, it improves the generalization power of the model. In this experiment we aim to apply this computer vision trick to time series by generating new data augmentation at each epoch. Except for the Discriminative Guided Warping, all presented data augmentation methods use randomness to generate the augmented sample. Thus, we can create almost an infinity of different augmented samples. We only consider the methods Scaling and the Window Warping. As explained before, DGW is not random and both Guided Warping methods are slow at runtime. Indeed, they compute warping paths using the DTW algorithm which is time consuming. Generating augmented samples once, prior to the training does not slow it much, but using them at each epoch would have taken too much time.

Results Figure 11 compares the results from the experiment 1, with their counterpart trained with data augmentation randomly generated at Each Epoch (EE). None of the results are significantly different from Inception. Window Warping seems to be the method that benefits the most from it as *Win. Warp. EE* has a better rank than *Win. Warp.*. For the Scaling method, EE data augmentation provides a slightly lower rank.

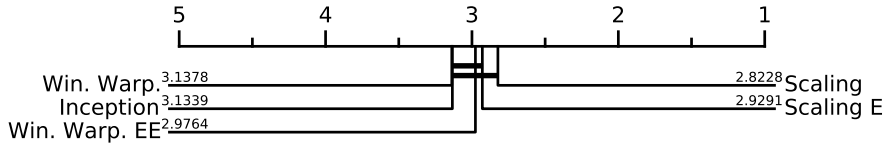

Fig. 11. CD diagram of experiment 3

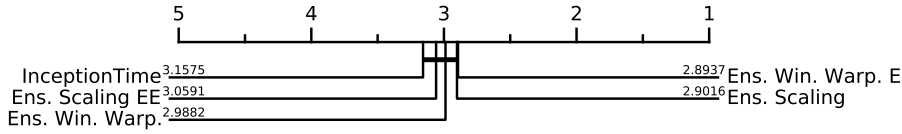

Fig. 12. CD diagram of experiment 3 with ensembling

Results with ensembling After ensembling, we notice that once more, Window Warping is the method that benefits the most. Once again the results with EE data augmentation do not outperform the baseline.

Table 2. Detailed results of experiment 3

Method	Avg. Accuracy
Scaling EE	83.84
Win. Warp. EE	**83.89**
Inception	83.48
Ens. Scaling EE	84.66
Ens. Win. Warp. EE	**85.05**
InceptionTime	84.24

When comparing Table 1 and Table 2 we notice that using data augmentation at each epoch results in a slightly lower average accuracy. EE data augmentation should improve the generalization power of the model if used correctly. However, our experiments did not result in significant improvements of the performances, neither in the rank or the accuracy. This shows that this trick, while being efficient in computer vision, does not work well on the UCR archive for time series.

5.4 Experiment 4

Experiment Until now, we used the different data augmentation methods individually, only one for each training. In experiment 1, we showed that some data

augmentation methods are complementary when ensembled together. This can be referred as late fusion.

In this experiment we aim to assess early fusion by training our models with multiple data augmentation methods. As Scaling and Window Warping proved to be the best methods, we did a training using them both. Finally, we did a second training using all four methods.

As we use more training samples, we can reduce the number of epochs. We decided to fix it to 300 epochs.

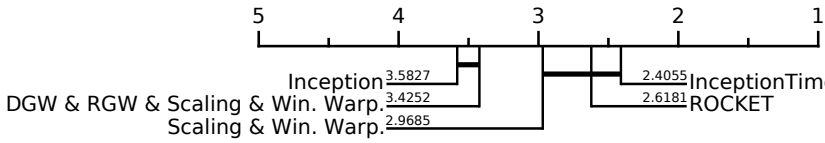

Fig. 13. CD diagram of experiment 4

Results Figure 13 represents the results of the experiment 4. We can notice that the ensemble *Scaling&Window Warping* is significantly better than the other two methods. *DGW&RGW&Scaling&Win. Warp.* is equivalent to Inception.

This shows that the combined use of several methods can be better than the use of them independently. However the choice of the methods is important. If we use our two best methods this lead to even better results, but using all of them provides the same results as the baseline.

With this early fusion strategy, it is the first time we managed to obtain a single Inception model not significantly worse than the InceptionTime model. As InceptionTime is composed of 5 Inception models, using this training method, we can reduce the training time and the inference time by a factor 5, with similar performances.

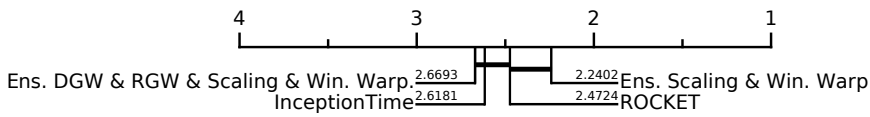

Fig. 14. CD diagram of experiment 4 with ensembling

Results with ensembling After ensembling, Figure 14 shows that *Scaling&Window Warping* is still significantly better than InceptionTime. As this ensemble rep-

resent our best result, we also compared it with ROCKET. According to the CD diagram and the Wilcoxon-Holms test, we can not say that it is significantly better than ROCKET but it represents a serious competitor.

Table 3. Detailed results of experiment 4

Method	Avg. Accuracy
DGW & RGW & Scaling & Win. Warp.	83.16
Scaling & Win. Warp.	**84.32**
Inception	83.48
Ens. DGW & RGW & Scaling & Win. Warp.	84.18
Ens. Scaling & Win. Warp.	**85.34**
InceptionTime	84.24
ROCKET	84.68

In Table 3 we can observe than the combined use of *Scaling* and *Window Warping* provides the best results across all our experiments, for both with and without ensembling.

6 Conclusion

In this paper, we have shown the relevance of data augmentation for time series classification through four different experiments, each time, with and without ensembling and on the entire UCR Archive.

First, we trained an Inception classifier using the original train sets along with the augmented ones. Then, we repeated the training using only augmented data. These experiments showed the importance of the original training set. Without it, it is much harder for the model to generalize on the test sets which do not contain any augmented samples. This also highlighted the methods Scaling and Window Warping, as the most efficient ones of our benchmark.

As the data augmentation methods are stochastic, for our third experiment, we tried to generate new data augmentation at each epoch. This experience lead to unsatisfactory results, close to the previous ones.

Finally, we did trainings using several data augmentation methods at the same time. Using all four data augmentation methods was not conclusive but using only Scaling and Window Warping lead to our best results. With this method, we manage to train a single Inception model obtaining similar performances to InceptionTime. As InceptionTime is an ensemble of five Inception models, our method requires 5 times less training, and also reduces the inference time by a factor five. When ensembled like InceptionTime, this method becomes significantly better than InceptionTime. Although not significantly better than ROCKET, this method obtain a better average accuracy.

We think that data augmentation will become commonly used for time series classification but still need further improvements. As our future work, we would

like to apply data augmentation to Inception and InceptionTime, but for multivariate time series. Moreover, in order to create smarter ensembles, we would like to use weighted ensembles in order to automatically choose the best combination of data augmented models.

Acknowledgment

This work was funded by ArtIC project "Artificial Intelligence for Care" (grant ANR-20-THIA-0006-01) and co-funded by Région Grand Est, Inria Nancy - Grand Est, IHU of Strasbourg, University of Strasbourg and University of Haute-Alsace. The authors would like to thank the providers of the UCR archive as well as the Mésocentre of Strasbourg for providing access to the GPU cluster.

References

1. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The ucr time series archive. IEEE/CAA Journal of Automatica Sinica **6**(6), 1293–1305 (2019)
2. Dempster, A., Petitjean, F., Webb, G.I.: Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery **34**(5), 1454–1495 (2020)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
4. Desai, A., Freeman, C., Wang, Z., Beaver, I.: Timevae: A variational auto-encoder for multivariate time series generation. arXiv preprint arXiv:2111.08095 (2021)
5. Esteban, C., Hyland, S.L., Rätsch, G.: Real-valued (medical) time series generation with recurrent conditional gans. arXiv preprint arXiv:1706.02633 (2017)
6. Forestier, G., Petitjean, F., Dau, H.A., Webb, G.I., Keogh, E.: Generating synthetic time series to augment sparse datasets. In: 2017 IEEE international conference on data mining (ICDM). pp. 865–870. IEEE (2017)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
8. Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P.A., Petitjean, F.: Inceptiontime: Finding alexnet for time series classification. Data Mining and Knowledge Discovery **34**(6), 1936–1962 (2020)
9. Iwana, B.K., Uchida, S.: An empirical survey of data augmentation for time series classification with neural networks. Plos one **16**(7), e0254841 (2021)
10. Iwana, B.K., Uchida, S.: Time series data augmentation for neural networks by time warping with a discriminative teacher. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 3558–3565. IEEE (2021)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)

12. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., et al.: The open images dataset v4. International Journal of Computer Vision **128**(7), 1956–1981 (2020)
13. Le Guennec, A., Malinowski, S., Tavenard, R.: Data augmentation for time series classification using convolutional neural networks. In: ECML/PKDD workshop on advanced analytics and learning on temporal data (2016)
14. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
15. Pialla, G., Fawaz, H.I., Devanne, M., Weber, J., Idoumghar, L., Muller, P.A., Bergmeir, C., Schmidt, D., Webb, G., Forestier, G.: 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. p. 485–496. Springer-Verlag, Berlin, Heidelberg (2022)
16. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
17. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
18. Terefe, T., Devanne, M., Weber, J., Hailemariam, D., Forestier, G.: Time series averaging using multi-tasking autoencoder. In: 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI). pp. 1065–1072. IEEE (2020)
19. Um, T.T., Pfister, F.M., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U., Kulić, D.: Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In: Proceedings of the 19th ACM international conference on multimodal interaction. pp. 216–220 (2017)
20. Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International joint conference on neural networks (IJCNN). pp. 1578–1585. IEEE (2017)
21. Wen, Q., Sun, L., Yang, F., Song, X., Gao, J., Wang, X., Xu, H.: Time series data augmentation for deep learning: A survey. arXiv preprint arXiv:2002.12478 (2020)
22. Yoon, J., Jarrett, D., Van der Schaar, M.: Time-series generative adversarial networks. Advances in Neural Information Processing Systems **32** (2019)
23. Zha, M.: Time series generation with masked autoencoder. arXiv preprint arXiv:2201.07006 (2022)