Convolutional neural networks with dynamic convolution for time series classification

Krisztian Buza[0000−0002−7111−6452] and Margit Antal[0000−0003−3596−1365]

Department of Mathematics-Informatics
Sapientia Hungarian University of Transylvania
Targu Mures, Romania
buza@biointelligence.hu, manyi@ms.sapientia.ro

Abstract. Due to its prominent applications, time series classification is one of the most important fields of machine learning. Although there are various approaches for time series classification, dynamic time warping (DTW) is generally considered to be a well-suited distance measure for time series. Therefore, in the early 2000s, techniques based on DTW dominated this field. On the other hand, deep learning techniques, especially convolutional neural networks (CNN) were shown to be able to solve time series classification tasks accurately. Although CNNs are extraordinarily popular, the scalar product in convolution only allows for rigid pattern matching. In this paper, we aim at combining the advantages of DTW and CNN by proposing the *dynamic convolution* operation and *dynamic convolutional neural networks* (DCNNs). The main idea behind dynamic convolution is to replace the dot product in convolution by DTW. We perform experiments on 10 publicly available real-world time-series datasets and demonstrate that our proposal leads to statistically significant improvement in terms of classification accuracy in various applications. In order to promote the use of DCNN, we made our implementation publicly available at https://github.com/kr7/DCNN .

Keywords: time series classification, dynamic convolution, convolutional neural networks, dynamic time warping

1 Introduction

Time series classification is the common denominator of numerous recognition tasks in various domains ranging from biology, medicine and healthcare over astronomy and geology to industry and finance. Such tasks include signature verification, speech recognition, earthquake prediction or the diagnosis of heart diseases based on electrocardiograph signals. Due to the aforementioned applications, and many others, time series classification is one of the most prominent fields of machine learning.

In the last decades, various approaches have been introduced for time series classification, including methods based on neural networks, Bayesian networks, hidden Markov models, genetic algorithms, support vector machines, decision

trees, frequent pattern mining and hubness-aware classifiers, see e.g. [4], [6], [7], [11], [13], [14], [20] and [3], [17] for introductory surveys. However, one of the most surprising results states that the simple k-nearest neighbor classifier using dynamic time warping (DTW) as distance measure is competitive (if not superior) to many other classifiers. In particular, Xi et al. compared various time series classifiers and concluded that that 1-nearest neighbor with DTW "is an exceptionally competitive classifier" [19]. Although, this result dates back to 2006, the observation has been confirmed by many researchers who used DTW in their approach and achieved high accuracy, see e.g. [15], [17] and the references therein. The primary reason why DTW is appropriate for time series classification is that DTW is an elastic distance measure in the sense that it allows for shifts and elongations while matching two time series.

Although DTW is a well-suited distance measure for time series classification, meanwhile state-of-the-art solutions are based on deep learning techniques, see e.g. [8], [18], [20], [21]. Especially, recent convolutional neural networks (CNNs) perform well for time series classification tasks, in many cases they outperform the previous baseline of kNN-DTW, see [7] for a review on CNNs for time series classification. Convolution is intended to act as local pattern detector, however, convolution itself only allows for rigid pattern matching by design. Therefore, convolutional layers are usually followed by pooling layers in state-of-the-art CNNs. While these pooling layers may alleviate the aforementioned issue of rigidity in pattern matching to some extent, as we will explain in Section 3.1 in detail, this solution is inherently limited and the resulting operation is somewhat irregular in terms of its ability to account for translations of local patterns.

In this paper, we aim at exploiting the flexibility of DTW in convolutional neural networks. In particular, in order to allow for local shifts and elongations, we replace the dot product in the first convolutional layer by DTW and call the resulting operation *dynamic convolution*. We perform experiments on publicly available real-world time-series datasets and demonstrate that our proposal leads to statistically significant improvement in terms of classification accuracy in various applications.

The remainder of the paper is organized as follows. A short review of related work on time series classification with CNNs and the embedding of DTW into neural networks is given in Section 2. This is followed by the definition of dynamic convolution and its integration with CNNs in Section 3. In Section 4, we describe the datasets, experimental protocols, and compare the results obtained by CNNs using conventional and dynamic convolutional layers. The last section presents the conclusions.

2   Related work

Works that are most closely related to ours fall into two categories: (i) methods based on convolutional neural networks for time series classification and (ii) approaches that integrate DTW with neural networks.

As for the former, we refer to the recent survey of Fawaz et al. [7] and we point out that our approach is orthogonal to convolutional network architectures as it can be used with any convolutional network by replacing the (first) convolutional layer by our dynamic convolution.

Regarding the approaches that integrate DTW with neural networks, we point out the works of Iwana et al. [10] and Cai et al. [5] who used DTW to construct features. In contrast, Afrasiabi et al. [1] used neural networks to extract features and used DTW to compare the resulting sequences. Shulman [16] proposed "an approach similar to DTW" to allow for flexible matching in case of the dot product. In our current work, we propose to use DTW instead of modifying the dot product.

Most closely related to our work is probably the DTW-NN approach, in case of which the authors considered neural networks and replaced "the standard inner product of a node with DTW as a kernel-like method" [9]. However, they only considered multilayer perceptrons (MLP). In contrast, we focus on convolutional networks.

3 Our approach

In this section, we describe the proposed approach in detail. We begin this section by discussing the limitations of "usual" convolution and max pooling. This is followed by the definition of *dynamic convolution*. Subsequently, we will discuss how the "usual" convolution can be replaced by dynamic convolution and how the weights (parameters) of dynamic convolution can be learned.

3.1 Convolution and max pooling

In CNNs, convolutional layers act as local pattern detectors and they are often followed by max pooling layers. Max pooling layers allow for some flexibility in pattern matching by hiding the exact location of a pattern within the time series. In other words: even if the pattern is shifted by a few positions, depending on the window size used in the max pooling layer, the activation of the layer may remain unchanged. However, max pooling is only able to establish this robustness in pattern matching if the pattern is shifted within the max pooling window: if the pattern is located at the boundary of the max pooling window, even if it is shifted just by one position outside the max pooling window, the activation of the max pooling layer will change. This is illustrated in Fig. 1 where the same pattern has been shifted by one position to the left and right. In the former case, the activation of the max pooling layer remains unchanged, whereas in the later case, the activation of the max pooling layer changes to a non-negligible extent. We argue that this behavior is somewhat irregular as one would expect the same changes regardless whether the pattern is shifted to the left or to the right.

More importantly, while convolution with max pooling may account for minor translations (even if its behaviour is somewhat irregular), we point out that there may be other types of temporal distortions as well, such as elongations
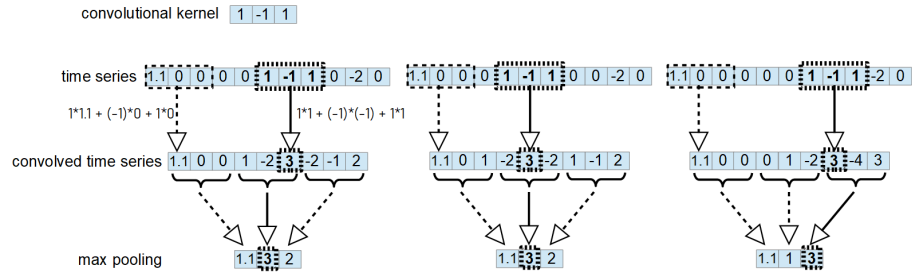
Fig. 1. Convolution with max pooling allows for a limited and, more importantly, irregular robustness against translations of local patterns. The convolutional kernel in the top is expected to detect 'V'-shaped local patterns. In the left, the pattern has been detected at one of the central positions in the time series which is reflected by the high activation of the max pooling layer at the second position (see the highlighted '3'). In the time series depicted in the center of the figure, the pattern has been translated by one position to the left. The activation of the max pooling layer remains unchanged indicating the robustness against small translations. On the other hand, in the time series in the right, the same pattern has been translated by one position to the right (compared to its original location in the time series in the left), and the activation of the max pooling layer changed.

within local patterns, that can not be taken into account by the dot product in convolution.

3.2   Dynamic convolution

The main idea behind dynamic convolution is to replace the calculation of dot products (or inner products) in convolution by the calculation of DTW distances between the kernel and time series segments. This is illustrated in Fig. 2. We omit the details of the calculation of DTW distances, as it has been described in various works, see e.g. [3] or [17].

Regarding dynamic convolution in neural networks, we propose to use dynamic convolution in the first hidden layer (i.e., directly after the input layer). We call the resulting model *dynamic convolutional neural network* or *DCNN* for short.

In order to determine the parameters of the dynamic convolutional layer, we propose to train an analogous neural network with "usual" convolution in the first hidden layer and use its learned kernel as kernel of the dynamic convolution. With *pre-training phase of DCNN*, we refer to the aforementioned process of training the analogous neural network with "usual" convolution in its first hidden layer. Once the pre-training is completed, the weights (parameters) of dynamic convolution are frozen (i.e., they do not change anymore), and other layers of DCNN can be trained by various optimization algorithms, such as stochastic gradient descent or Adam [12]. In order to implement training effi-
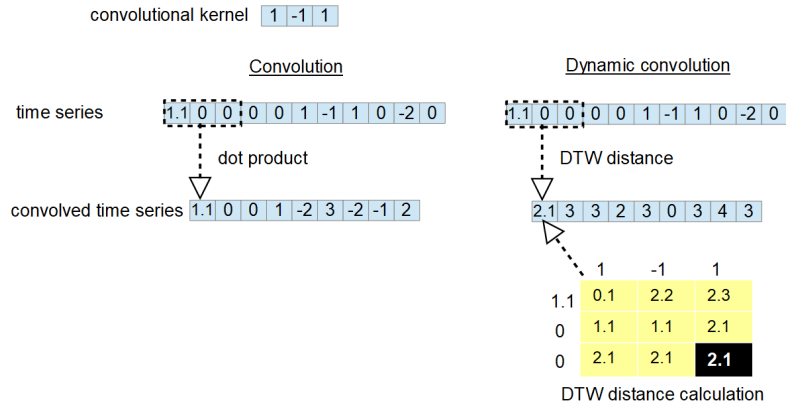
Fig. 2. Convolution (left) vs. dynamic convolution (right). In case of dynamic convolution, instead of the dot product (or inner product), DTW distances between the kernel and time series segments are calculated.

ciently, the activation of the dynamic convolutional layer may be pre-computed (as the weights of dynamic convolution do not change after pre-training).

4 Experimental evaluation

The goal of our experiments is to examine whether the proposed dynamic convolution improves the accuracy of neural networks in the context of time series classification.

Data: We performed experiments on real-world time series datasets that are publicly available in "The UEA & UCR Time Series Classification Repository" [2]. The datasets used in our experiments are listed in the first column on Tab. 1.

Experimental settings: In order to assess the contribution of dynamic convolution, we trained two versions of the same networks: *with* and *without* dynamic convolution, and compared the results. In the former case, the first hidden layer was a dynamic convolutional layer (with DTW calculations), whereas in the later case, we used the "usual" convolution (with dot product).

   To account for the fact that dynamic convolution may be used in various neural networks, we assessed the contribution of dynamic convolution in case of two different neural networks:

- *Net1* is a simple convolutional network, containing a single convolutional layer with 10 filters, followed by a max pooling layer with window size of 2, and a fully connected layer with 100 units.
- *Net2* contains a convolutional layer with 25 filters, followed by a max pooling layer with window size of 2, a second convolutional layer with 10 filters, a

Table 1. Average accuracy $\pm$ its standard deviation (calculated over 10 folds) for the neural networks with "usual" convolution (denoted as CNN) and dynamic convolution (denoted as DCNN). The best out of the both networks is denoted by bold font. In each case we also provide a symbol $\bullet$/$\circ$ denoting if the observed difference is statistically significant ($\bullet$) or not ($\circ$) according to paired t-test at significance level (p-value) of 0.05.

Dataset	Net1		Net2	
	CNN	DCNN	CNN	DCNN
Adiac	0.506±0.061	**0.575±0.046** $\bullet$	0.558± 0.052	**0.640±0.055** $\bullet$
ArrowHead	0.886±0.064	**0.896±0.083** $\circ$	**0.900±0.062**	0.887±0.082 $\circ$
Beef	0.733±0.170	**0.800±0.163** $\bullet$	0.700±0.180	**0.783±0.130** $\bullet$
EarthQuakes	0.725±0.042	**0.733±0.069** $\circ$	0.699±0.072	**0.731±0.063** $\circ$
ECG200	0.870±0.050	**0.890±0.044** $\circ$	0.865±0.084	**0.870±0.064** $\circ$
FiftyWords	0.702±0.033	**0.714±0.045** $\circ$	0.686±0.034	**0.715±0.027** $\bullet$
Plane	0.981±0.032	**0.990±0.029** $\circ$	0.976±0.032	**0.995±0.014** $\bullet$
SwedishLeaf	0.864±0.041	**0.883±0.027** $\bullet$	0.862±0.036	**0.881±0.033** $\circ$
WordSynonyms	0.682±0.031	**0.714±0.050** $\bullet$	0.681±0.049	**0.727±0.047** $\bullet$
Yoga	0.951±0.013	**0.960±0.012** $\bullet$	0.945±0.022	**0.959±0.008** $\circ$

second max pooling layer with window size of 2 and a fully connected layer with 100 units.

In both cases, the output layer contains as many units as the number of classes in the dataset. Both in case of Net1 and Net2, we experimented with the aforementioned two versions, i.e., with and without dynamic convolution in the first hidden layer. Although, there may be other neural networks that are better suited for a particular time series classification task, we observed that both Net1 and Net2 lead to accurate models in the examined tasks and we point out that the primary goal of our experiments was to assess the contribution of dynamic convolution. We implemented both Net1 and Net2 in pytorch. In order to calculate DTW distances quickly, we used a function that was implemented in Cython. We executed the experiments in Google Colab.[1]

We performed experiments according to the 10-fold cross-validation protocol and report average classification accuracy together with its standard deviation for both versions of Net1 and Net2 in Tab. 1. Additionally, we used paired t-test at significance level (p-value) of 0.05 in order to assess whether the observed differences are statistically significant (denoted by $\bullet$) or not (denoted by $\circ$).

Codes: In order to assist reproduction of the results, we published our code in our GitHub repository

https://github.com/kr7/DCNN

in form of IPython notebooks that can be directly executed in Google Colab.

---

[1] https://colab.research.google.com

Results: As one can see in Tab. 1, in half of the examined cases, DCNN significantly outperforms CNN. In the remaining cases, where the difference is statistically insignificant, DCNN still usually performs better than CNN, the only exception is the experiment with Net2 on the ArrowHead dataset. These results indicate that the proposed dynamic convolution may indeed improve the accuracy of convolutional neural networks.

Discussion: We point out that both Net1 and Net2 contain max pooling layers. Therefore, our results also demonstrate that "usual" convolution with max pooling was not able to account for all the temporal distortions that are present in the data. This is inline with our expectations based on our analysis in Section 3.1 where we discussed the rigidity of the dot product and pointed out that max pooling only has a limited ability to account for translations, however, it can not take elongations into account.

Training time and complexity: The method to train DCNN consists of two phases: in the first (pretrain) phase, a CNN is trained; whereas in the second phase, the weights of DCNN (except for the weights of the dynamic convolutional layer) are learned. Therefore, the overall training time is roughly twice of the training time of a single CNN. However, as pointed out in Section 3.2, the activation of the dynamic convolutional layer may be precomputed, i.e., it needs to be computed only once, even if the network is trained for many epochs in the second phase. Therefore, the second phase of training may actually be slightly faster than the training of an analogous CNN.

In the sense of complexity theory, under the assumptions that the number of convolutional filters in the dynamic convolutional layer is a constant, the sizes of each of them is a (small) constant and the length of time series is constant as well, the time required for computation of the activation of the dynamic convolutional layer is constant which does not change the complexity of training in the sense of algorithm theory. Also training two networks instead of one, is just a multiplication by a constant factor of 2 which again does not change the complexity in the sense of algorithm theory. Therefore, the theoretical complexity of training DCNN is the same as training a "usual" CNN.

5   Conclusions and outlook

In this paper we introduced dynamic convolution as an alternative to the "usual" convolution operation. Dynamic convolutional layers can be used within various neural networks. We performed experiments in context of time series classification on publicly available real-world datasets. The results are very promising: they show that dynamic convolution is indeed competitive with "usual" convolution, moreover, the neural networks using dynamic convolutional layers systematically, often statistically significantly, outperform analogous neural networks with "usual" convolutional layers. In principle, the proposed neural networks with dynamic convolution may be used for any time series classification tasks,

such as handwriting recognition, signature verification, ECG-based diagnosis of various heart diseases, etc. In order to assist reproduction of our work and to promote using dynamic convolutional layers in various applications, we published our codes.

As future work, we plan to perform experiments with additional neural network architectures, such as deeper neural network, or networks with residual connections. Moreover, in order to examine the generality of our results, we plan to experiment with further datasets as well.

References

1. Afrasiabi, M., Mansoorizadeh, M., et al.: Dtw-cnn: time series-based human interaction prediction in videos using cnn-extracted features. The Visual Computer pp. 1–13 (2019)
2. Bagnall, A., Lines, J., Vickers, W., Keogh, E.: The uea & ucr time series classification repository, www.timeseriesclassification.com
3. Buza, K.: Time series classification and its applications. In: Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics. pp. 1–4 (2018)
4. Buza, K.: Asterics: Projection-based classification of eeg with asymmetric loss linear regression and genetic algorithm. In: 2020 IEEE 14th International Symposium on Applied Computational Intelligence and Informatics (SACI). pp. 35–40. IEEE (2020)
5. Cai, X., Xu, T., Yi, J., Huang, J., Rajasekaran, S.: Dtwnet: a dynamic time warping network. Advances in neural information processing systems **32** (2019)
6. Esmael, B., Arnaout, A., Fruhwirth, R.K., Thonhauser, G.: Improving time series classification using hidden markov models. In: 2012 12th International Conference on Hybrid Intelligent Systems (HIS). pp. 502–507. IEEE (2012)
7. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Mining and Knowledge Discovery **33**(4), 917–963 (2019)
8. Guzy, F., Woźniak, M.: Employing dropout regularization to classify recurring drifted data streams. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–7. IEEE (2020)
9. Iwana, B.K., Frinken, V., Uchida, S.: Dtw-nn: A novel neural network for time series recognition using dynamic alignment between inputs and weights. Knowledge-Based Systems **188**, 104971 (2020)
10. Iwana, B.K., Uchida, S.: Time series classification using local distance-based features in multi-modal fusion networks. Pattern Recognition **97**, 107024 (2020)
11. Jankowski, D., Jackowski, K., Cyganek, B.: Learning decision trees from data streams with concept drift. Procedia Computer Science **80**, 1682–1691 (2016)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Lines, J., Davis, L.M., Hills, J., Bagnall, A.: A shapelet transform for time series classification. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 289–297 (2012)
14. Pavlovic, V., Frey, B.J., Huang, T.S.: Time-series classification using mixed-state dynamic bayesian networks. In: Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149). vol. 2, pp. 609–615. IEEE (1999)

15. Radovanović, M., Nanopoulos, A., Ivanović, M.: Time-series classification in many intrinsic dimensions. In: Proceedings of the 2010 SIAM International Conference on Data Mining. pp. 677–688. SIAM (2010)
16. Shulman, Y.: Dynamic time warp convolutional networks. arXiv preprint arXiv:1911.01944 (2019)
17. Tomašev, N., Buza, K., Marussy, K., Kis, P.B.: Hubness-aware classification, instance selection and feature construction: Survey and extensions to time-series. In: Feature selection for data and pattern recognition, pp. 231–262. Springer (2015)
18. 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)
19. Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: Proceedings of the 23rd international conference on Machine learning. pp. 1033–1040 (2006)
20. Zhao, B., Lu, H., Chen, S., Liu, J., Wu, D.: Convolutional neural networks for time series classification. Journal of Systems Engineering and Electronics **28**(1), 162–169 (2017)
21. Zheng, Y., Liu, Q., Chen, E., Ge, Y., Zhao, J.L.: Time series classification using multi-channels deep convolutional neural networks. In: International conference on web-age information management. pp. 298–310. Springer (2014)