On the performance of deep learning models for time series classification in streaming

Pedro Lara-Benítez[1], Manuel Carranza-García[1],
Francisco Martínez-Álvarez[2], and José C. Riquelme[1]

[1] Division of Computer Science, University of Sevilla, ES-41012 Seville, Spain
[2] Data Science & Big Data Lab, Pablo de Olavide University, ES-41013 Seville, Spain
plbenitez@us.es

Abstract. Processing data streams arriving at high speed requires the development of models that can provide fast and accurate predictions. Although deep neural networks are the state-of-the-art for many machine learning tasks, their performance in real-time data streaming scenarios is a research area that has not yet been fully addressed. Nevertheless, there have been recent efforts to adapt complex deep learning models for streaming tasks by reducing their processing rate. The design of the asynchronous dual-pipeline deep learning framework allows to predict over incoming instances and update the model simultaneously using two separate layers. The aim of this work is to assess the performance of different types of deep architectures for data streaming classification using this framework. We evaluate models such as multi-layer perceptrons, recurrent, convolutional and temporal convolutional neural networks over several time-series datasets that are simulated as streams. The obtained results indicate that convolutional architectures achieve a higher performance in terms of accuracy and efficiency.

Keywords: Classification · Data Streaming · Deep Learning · Time Series.

1 Introduction

Learning from data arriving at high speed is one of the main challenges in machine learning. Over the last decades, there have been several efforts to develop models that deal with the specific requirements of data streaming. Traditional batch-learning models are not suitable for this purpose given the high rate of arrival of instances. In data streaming, incoming data has to be rapidly classified and discarded after using it for updating the model. Predicting and training have to be done as fast as possible in order to maintain a processing rate close to real-time. Furthermore, the models have to be able to detect possible changes in the incoming data distribution, which is known as concept drift.

Despite the incremental learning nature of neural networks, there is little research involving deep learning (DL) models in the data streaming literature. Neural networks can adapt to changes in data by updating their weights with

incoming instances. However, the high training time of deep networks presents challenges to adapt them to a streaming scenario. Very recently, our research group developed a deep learning framework for data streaming classification that uses an asynchronous dual-pipeline architecture (ADLStream) [14]. In this framework, training and classification can be done simultaneously in two different processes. This separation allows to use DL networks for data arriving at high speed while maintaining a high predictive performance.

The aim of this study is to evaluate how different DL architectures perform on the data streaming classification task using the ADLStream framework. Despite the promising results presented in [14], the experiments only considered convolutional neural networks, hence the suitability and efficiency of other types of deep networks is an area that has yet to be studied. In this work, we focus the experimental study on time-series data obtained from the UCR repository that have been simulated as streams. For this reason, we have designed DL models that are suitable for data having an inner temporal structure. The basic Multi-Layer Perceptron (MLP) is set as the baseline model and compared with other three architectures: Long-Short Term Memory network (LSTM), Convolutional Neural Network (CNN), and Temporal Convolutional Network (TCN). These models are evaluated in terms of accuracy and computational efficiency.

The rest of the paper is organised as follows: Section 2 presents a review on related work; Section 3 describes the materials used and the methodology; in Section 4 the experimental results obtained are reported; Section 5 presents the conclusions and future work.

2   Related work

Over the last decades, there have been several efforts to develop models that deal with the specific requirements of data streaming. Traditional batch-learning models are not suitable for this purpose given the high rate of arrival of instances. In data streaming, incoming data has to be rapidly classified and then discarded after using it for updating the learning model. Predicting and training have to be done as fast as possible in order to maintain a processing rate close to real-time. Furthermore, the models have to be able to detect possible changes in the incoming data distribution, which is known as concept drift[1].

One of the most popular approaches has been to develop incremental or online algorithms based on decision trees, for instance, the Hoeffding Adaptive Trees (HAT) [3]. These models build trees incrementally based on the Hoeffding principle, that splits a node only when there is statistical significance between the current best attribute and the others. Later, ensemble techniques have been successfully applied to data stream classification, enhancing the predictive performance of single classifiers. ADWIN bagging used adaptive windows to control the adaptation of ensemble members to the evolution of the stream [3]. More recently, researchers have focused on building ensemble models that can deal effectively with concept drifts. The Adaptive Random Forest (ARF) algorithm proposes better resampling methods for updating classifiers over drifting data

streams [11]. In [5], the authors proposed the Kappa Updated Ensemble (KUE) that uses weighted voting from a pool of classifiers with possible abstentions.

Despite the incremental learning nature of neural networks, there is little research involving DL models in the data streaming literature. Neural networks can adapt to changes in data by updating their weights with incoming instances. However, the high training time of deep networks presents challenges to adapt them to a streaming scenario in real-time. There have been proposals using simple networks such as the Multi-Layer Perceptron [10,16]. A deep learning framework for data streaming that uses a dual-pipeline architecture was developed in [14]. A more detailed description of the framework, which was the first using complex DL networks for data streaming, is provided in the next section.

3   Materials and methods

3.1   ADLStream framework

In this study, we use the asynchronous dual-pipeline deep learning framework (ADLStream) for data streaming presented in [14]. As can be seen in Figure 1, the proposed system has two separated layers for training and predicting. This improves the processing rate of incoming data since instances are classified as soon as they arrive using a recently trained model. In the other layer, the weights of the network are constantly being updated in order to adjust to the evolution of the stream. This framework allows to use complex DL model, such as recurrent or convolutional, that would not be possible to use in a data streaming scenario if they are trained sequentially. The source code of ADLStream framework can be found at [13].
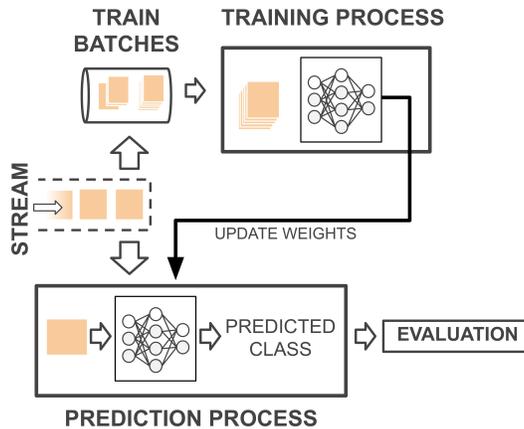

Fig. 1. Asynchronous dual-pipeline deep learning framework

3.2   Datasets

For the experimental study, 29 one-dimensional time series datasets from the UCR repository have been simulated as streams [6]. The selected datasets have different characteristics and are categorized into six different domains. Table 1 presents a detailed description of the number of instances, length of the time series instances, and the number of classes of each dataset.

Table 1. Datasets used for the study.

#	Dataset	Instances	Length	Classes	Type
1	TwoPatterns	5000	128	4	SIMULATED
2	CinCECGtorso	1420	1639	4	ECG
3	TwoLeadECG	1162	82	2	ECG
4	Wafer	7164	152	2	SENSOR
5	Pendigits	10992	16	10	MOTION
6	FacesUCR	2250	131	14	IMAGE
7	Mallat	2400	1024	8	SIMULATED
8	FaceAll	2250	131	14	IMAGE
9	Symbols	1020	398	6	IMAGE
10	ItalyPowerDemand	1096	24	2	SENSOR
11	ECG5000	5000	140	5	ECG
12	MoteStrain	1272	84	2	SENSOR
13	NonInvasiveFetalECGThorax1	3765	750	42	ECG
14	NonInvasiveFetalECGThorax2	3765	750	42	ECG
15	SwedishLeaf	1125	128	15	IMAGE
16	FordA	4921	500	2	SENSOR
17	Yoga	3300	426	2	IMAGE
18	UWaveGestureLibraryX	4478	315	8	MOTION
19	FordB	4446	500	2	SENSOR
20	ElectricDevices	16637	96	7	DEVICE
21	UWaveGestureLibraryY	4478	315	8	MOTION
22	UWaveGestureLibraryZ	4478	315	8	MOTION
23	HandOutlines	1370	2709	2	IMAGE
24	InsectWingbeatSound	2200	256	11	SENSOR
25	ShapesAll	1200	512	60	IMAGE
26	MedicalImages	1141	99	10	IMAGE
27	PhalangesOutlinesCorrect	2658	80	2	IMAGE
28	ChlorineConcentration	4307	166	3	SIMULATED
29	Phoneme	2110	1024	39	SENSOR

3.3   Experimental study

In this section, we present the design of the different types of DL models selected for the experimental study. Furthermore, we also describe the details of the evaluation method used for the data streaming classification task.

3.3.1 Deep learning models

Our aim in this study is to evaluate the performance of different DL architectures within the ADLStream framework. Four different families of architectures are considered in the experiments: the Multi-layer Perceptron (MLP) which will serve as the baseline, recurrent networks using Long Short-Term Memory cells (LSTM), Convolutional Neural Networks (CNN), and Temporal Convolutional Networks (TCN). While the MLPs is unable to model the time relationships within the input data, the last three architectures are particularly indicated for dealing with data that has a temporal or spatial grid-like structure, such as the selected datasets. LSTM networks are one of the most popular types of recurrent neural networks. They connect each time step with the previous ones in order to model the long temporal dependencies of the data without forgetting the short-term patterns using special gates [9]. On the other hand, CNNs are networks based on the convolution operation, which creates features maps using sliding filters. They are also suitable for one-dimensional time series data since they are able to automatically capture repeated patterns at different scales [12]. Moreover, they have far less trainable parameters than recurrent

Table 2. Multi-Layer Perceptron architecture

MLP		
Layer	**Type**	**Neurons**
0	Input	f
1	Dense	32
2	Dense	64
3	Dense	128
4	Softmax	c
Params $f \times 32 + 10240 + c \times 128$		

Table 3. Convolutional Neural Network architecture. k indicates the kernel size

CNN		
Layer	**Type**	**Neurons**
0	Input	f
1	Conv. ($k = 7$)	$f \times 64$ maps
2	Max-Pool ($k = 2$)	$f/2 \times 64$ maps
3	Conv. ($k = 5$)	$f/2 \times 128$ maps
4	Max-Pool ($k = 2$)	$f/4 \times 128$ maps
5	Dense	64
6	Dense	32
7	Softmax	c
Params	$f \times 2048 + 43648 + c \times 32$	

Table 4. Long Short-Term Memory Network architecture

LSTM		
Layer	**Type**	**Neurons**
0	Input	f
1	LSTM	$f \times 64$ units
2	LSTM	$f \times 128$ units
3	Dense	64
4	Dense	32
5	Softmax	c
Params $f \times 8192 + 117760 + c \times 32$		

Table 5. Temporal Convolutional Network architecture. k indicates the kernel size

TCN		
Layer	**Type**	**Neurons**
0	Input	f
1	TCN ($k = 5$)	$f \times 64$ maps
2	Dense	64
3	Dense	32
4	Softmax	c
Params $f \times 4096 + 372096 + c \times 32$		

networks due to their weight sharing scheme [4]. More recently, TCNs have emerged as a specialised architecture that can capture long-term dependencies more effectively by using dilated causal convolutions. With this operation, the receptive field of neurons is increased without the need for pooling operations, hence there is no loss of resolution [15]. Tables 2-5 provide a detailed description of the layers composing the four DL models considered. In these tables, the values of f and c are the number of features of the instances and the number of classes respectively. The baseline MLP model (Table 2) is composed of three dense layers with an increasing number of neurons. As can be seen, the other three models have a similar architecture since the convolutional or recurrent layers have the same number of maps or units and are followed by fully-connected layers with the same number of neurons. In the CNN (Table 3), two convolutional blocks with decreasing kernel size and max-pooling of stride 2 are applied before the dense layers. In the LSTM and TCN layers, the complete sequences are returned and connected to the next layers in order to use the information of all patterns extracted at different scales. In the TCN (Table 5), only one stack of residual blocks is used, and the dilated convolution is used with kernel ($k = 5$) and dilations ($d = \{1, 2, 4, 8, 16, 32, 64\}$). Another important element to consider is the use of a dropout with rate 0.2 on all dense layers in all models, with the aim of reducing over-fitting issues. The number of trainable parameters illustrates the computational cost of each model. The TCN has the highest number, which can be 37 times greater than the MLP model.

3.3.2   Evaluation

For evaluating the results we use the prequential method with decaying factors, that incrementally updates the accuracy by testing the model with unseen examples [8]. The decaying factors are used as a forgetting mechanism to give more importance to recent instances for estimating the error, given the evolving nature of the stream. In our study, we use a decaying factor of $\alpha = 0.99$. The process of calculating the prequential accuracy can be formulated as follows, where L is the loss function and o and y are the real and expected output respectively.

$$P_\alpha(i) = \frac{\sum_{k=1}^{i} \alpha^{i-k} L(y_k, o_k)}{\sum_{k=1}^{i} \alpha^{i-k}} = = L(y_i, o_i) + \frac{1}{\alpha} P_\alpha(i-1) \tag{1}$$

The metric selected is the Kappa statistic, that is more suitable than standard accuracy in data streaming due to the frequent changes in the class distribution of incoming instances [2]. The Kappa value can be computed as shown in the following equation, where p_0 is the prequential accuracy and p_c is the hypothetical probability of chance agreement.

$$k = \frac{p_0 - p_c}{1 - p_c} \tag{2}$$

4   Experimental results

This section presents the Kappa accuracy results and the statistical analysis. The experiments have been carried out with an Intel Core i7-770K and two NVIDIA GeForce GTX 1080 8GB GPU. The Apache Kafka server is used to reproduce the streaming scenario since it is the most efficient tool available [7].

4.1   Prequential Kappa

Table 6 presents the prequential kappa accuracy results obtained with the different models for each dataset. As can be seen, the CNN achieves the best performance for almost all the datasets considered, obtaining the highest average kappa accuracy value. The second model on average is the TCN, but closely followed by the LSTM that shows a similar performance. In general, the results prove that the ADLStream framework is able to achieve reliable results regardless of the deep learning architecture chosen.

4.2   Computation time analysis

In a data streaming environment, it is fundamental to analyse the efficiency of the architectures considered. The average processing rate of each model (average time to process each incoming instance) is provided at the end of Table 6. Logically, the MLP is the fastest model given its simple architecture. The second fastest model is the CNN, which has a significantly smaller number of parameters than the other two DL architectures. Thanks to the properties of parameter sharing, the CNN is able to process instances three times faster than the LSTM. The TCN is a more complex model with more convolutions which results in a processing rate of almost 8 times slower than the CNN.

4.3   Statistical analysis

The ranking of the accuracy of the models obtained with the Friedman test is presented in Table 7. The CNN model leads the ranking, with a high difference in score with respect to the rest of the models. The TCN and LSTM obtain a similar score, while the MLP offers the worst performance. The null hypothesis is rejected since the p-value obtained (<0.001) is below the significance level ($\alpha = 0.05$).

In Bergmann-Hommel's post-hoc analysis, we perform pair-wise comparisons between all models. Table 8 reports the p-values and conclusions obtained. As can be seen, for the CNN all null hypothesis can be rejected since the p-values are always below the significance level. Therefore, it can be concluded that there is a statistical significance in the differences between the performance of the CNN and the other architectures considered. Nevertheless, there are no significant differences between the accuracy of LSTM and the TCN.

Table 6. Prequential kappa accuracy results

#	Dataset	MLP	LSTM	CNN	TCN
1	TwoPatterns	0.818	0.999	**1.000**	0.999
2	CinCECGtorso	0.348	**0.990**	0.994	0.933
3	TwoLeadECG	0.947	0.941	**0.991**	0.987
4	Wafer	0.581	0.995	**0.996**	0.710
5	pendigits	0.728	0.987	**0.992**	0.953
6	FacesUCR	0.834	0.952	**0.974**	0.952
7	Mallat	0.963	0.920	**0.986**	0.978
8	FaceAll	0.841	0.953	**0.962**	0.948
9	Symbols	0.877	0.900	**0.949**	0.914
10	ItalyPowerDemand	**0.942**	0.921	0.935	0.934
11	ECG5000	0.881	**0.891**	0.888	0.890
12	MoteStrain	0.778	0.843	**0.878**	0.851
13	NonInvasiveFetalECGThorax1	0.851	0.862	**0.881**	0.873
14	NonInvasiveFetalECGThorax2	0.894	0.893	**0.901**	0.900
15	SwedishLeaf	0.679	0.775	**0.874**	0.844
16	FordA	-0.021	**0.691**	0.644	0.632
17	Yoga	0.213	0.659	**0.737**	0.689
18	UWaveGestureLibraryX	0.560	0.748	**0.761**	0.732
19	FordB	0.009	**0.654**	0.626	0.219
20	ElectricDevices	0.353	**0.803**	0.801	0.755
21	UWaveGestureLibraryY	0.506	0.641	**0.648**	0.608
22	UWaveGestureLibraryZ	0.496	0.646	**0.658**	0.613
23	HandOutlines	0.643	0.674	**0.721**	0.714
24	InsectWingbeatSound	0.605	0.602	**0.613**	0.598
25	ShapesAll	0.542	0.598	**0.606**	0.596
26	MedicalImages	0.304	0.566	**0.580**	0.503
27	PhalangesOutlinesCorrect	0.153	0.156	**0.474**	0.451
28	ChlorineConcentration	0.242	0.157	**0.900**	0.891
29	Phoneme	0.032	0.117	**0.182**	0.138
	Average kappa	**0.572**	**0.743**	**0.798**	**0.752**
	Average time per instance (ms)	**4.993**	**22.09**	**7.347**	**47.34**

Table 8. Bergmann-Hommel's analysis

Table 7. Friedman Test Ranking

Friedman Test Ranking	
CNN	1.200
TCN	2.533
LSTM	2.566
MLP	3.700

PostHoc Analysis			
Comparison	p	z	Conclusion
MLP - CNN	<0.001	7.5	!=
LSTM - CNN	<0.001	4.1	!=
TCN - CNN	<0.001	4	!=
MLP - TCN	0.001	3.49	!=
MLP - LSTM	0.001	3.39	!=
LSTM - TCN	0.920	0.09	==

5 Conclusions

In this paper, the performance of several deep learning architectures for data streaming classification is compared using the ADLStream framework. An extensive study over a large number of time-series dataset was conducted using multi-layer perceptron, recurrent, and convolutional neural networks.

The research carried out for this study provided evidence that convolutional neural networks are currently the most suitable model for time series classification in streaming. Convolutional neural networks obtained the best results in terms of accuracy, with a very high processing rate. These characteristics present convolutional networks as the best alternative for processing data arriving at high speed. The other deep models, such as Long Short-Term Memory or Temporal Convolutional networks were not able to achieve such performance and their processing rate was slower.

Future work should study the behaviour of different deep learning models over concept drifts and their capacity to adapt to changes in the data distribution. Furthermore, a parameter optimization process could provide more specific architectures for the models and improve the performance. Future studies should also consider other less known models such as Echo State Networks, Stochastic Temporal Convolutional Networks or Gated Recurrent Units Networks.

Acknowledgements

References

1. Anderson, R., Koh, Y., Dobbie, G., Bifet, A.: Recurring concept meta-learning for evolving data streams. Expert Systems with Applications **138** (2019). https://doi.org/10.1016/j.eswa.2019.112832
2. Bifet, A., de Francisci Morales, G., Read, J., Holmes, G., Pfahringer, B.: Efficient online evaluation of big data stream classifiers. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 59–68. KDD '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2783258.2783372
3. Bifet, A., Gavaldà, R.: Adaptive learning from evolving data streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.F. (eds.) Advances in Intelligent Data Analysis VIII. pp. 249–260. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
4. Borovykh, A., Bohte, S., Oosterlee, C.: Dilated convolutional neural networks for time series forecasting. Journal of Computational Finance **22**(4), 73–101 (2019). https://doi.org/10.21314/JCF.2018.358

5. Cano, A., Krawczyk, B.: Kappa updated ensemble for drifting data stream mining. Machine Learning (2019). https://doi.org/10.1007/s10994-019-05840-z

6. Dau, H.A., Bagnall, A.J., Kamgar, K., Yeh, C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.J.: The UCR time series archive. CoRR **abs/1810.07758** (2018)

7. Fernández-Rodríguez, J.Y., Álvarez García, J.A., Fisteus, J.A., Luaces, M.R., Magaña, V.C.: Benchmarking real-time vehicle data streaming models for a smart city. Information Systems **72**, 62 – 76 (2017). https://doi.org/10.1016/j.is.2017.09.002

8. Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Machine Learning **90**(3), 317–346 (Mar 2013). https://doi.org/10.1007/s10994-012-5320-9

9. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with LSTM. Neural Computation **12**(10), 2451–2471 (2000). https://doi.org/10.1162/089976600300015015

10. Ghazikhani, A., Monsefi, R., Sadoghi Yazdi, H.: Online neural network model for non-stationary and imbalanced data stream classification. International Journal of Machine Learning and Cybernetics **5**(1), 51–62 (2014). https://doi.org/10.1007/s13042-013-0180-6

11. Gomes, H.M., Bifet, A., Read, J., Barddal, J.P., Enembreck, F., Pfharinger, B., Holmes, G., Abdessalem, T.: Adaptive random forests for evolving data stream classification. Machine Learning **106**(9), 1469–1495 (2017). https://doi.org/10.1007/s10994-017-5642-8

12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. pp. 1097–1105. NIPS'12, Curran Associates Inc., USA (2012)

13. Lara-Benítez, P., Carranza-García, M.: ADLStream: Asynchronous dual-pipeline deep learning framework for online data stream mining. Available online: `https://github.com/pedrolarben/ADLStream`, (Accessed: 01-04-2020)

14. Lara-Benítez, P., Carranza-García, M., García-Gutiérrez, J., Riquelme, J.: Asynchronous dual-pipeline deep learning framework for online data stream classification. Integrated Computer-Aided Engineering pp. 1–19 (01 2020). https://doi.org/10.3233/ICA-200617

15. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), `http://arxiv.org/abs/1511.07122`

16. Zhang, Y., Yu, J., Liu, W., Ota, K.: Ensemble classification for skewed data streams based on neural network. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **26**(05), 839–853 (2018). https://doi.org/10.1142/S021848851850037X