Data streams classification using deep learning under different speeds and drifts

PEDRO LARA-BENÍTEZ*, *Division of Computer Science, University of Sevilla, ES-41012 Seville, Spain.*

MANUEL CARRANZA-GARCÍA, *Division of Computer Science, University of Sevilla, ES-41012 Seville, Spain.*

DAVID GUTIÉRREZ-AVILÉS, *Division of Computer Science, University of Sevilla, ES-41012 Seville, Spain.*

JOSÉ C. RIQUELME, *Division of Computer Science, University of Sevilla, ES-41012 Seville, Spain.*

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, much effort has been put into the adaption of complex deep learning (DL) models to streaming tasks by reducing the processing time. The design of the asynchronous dual-pipeline DL framework allows making predictions of incoming instances and updating the model simultaneously, using two separate layers. The aim of this work is to assess the performance of different types of DL 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 at different speeds. In addition, we evaluate how the different architectures react to concept drifts typically found in evolving data streams. The obtained results indicate that convolutional architectures achieve a higher performance in terms of accuracy and efficiency, but are also the most sensitive to concept drifts.

Keywords: Deep learning, data streaming, online learning, time series, classification.

1 Introduction

Learning from data arriving at high speed is one of the main challenges in machine learning. Over the past decades, different models have been developed to 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 must detect possible changes in the incoming data distribution, which is known as concept drift [1].

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

*E-mail: plbenitez@us.es

https://doi.org/10.1093/jigpal/jzac033

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, a DL framework for data streaming classification that uses an asynchronous dual-pipeline architecture (ADLStream) was introduced in [15]. In this framework, training and classification can be done simultaneously in two different processes. This separation allows using 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 [15], 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 [6] that have been simulated as streams. For this reason, we have designed DL models that are suitable for data having an inner temporal structure [17]. 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. Furthermore, 13 synthetic datasets have been generated to evaluate the performance of these models under evolving streams with different types of concept drift.

The rest of the paper is organized 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. Finally, Section 5 presents the conclusions and future work.

2 Related work

One of the most popular approaches is devoted to develop incremental or online algorithms based on decision trees, for instance, the Hoeffding Adaptive Trees [3]. These models build trees incrementally based on the Hoeffding principle, which 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 [10]. 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 are few studies exploring the use of DL for data stream mining. There have been proposals using simple networks such as the MLP [9, 19]. For more complex networks, a DL framework for data streaming that uses a dual-pipeline architecture was developed in [15]. A more detailed description of this framework, which was the first using complex DL networks for data streaming, is provided in the next section.

3 Materials and methods

This section provides a description of the materials and methods used in this work. Thus, Section 3.1 introduces the ADLStream framework. Section 3.2 describes the datasets used as benchmark. Section 3.3 describes the experimental study.

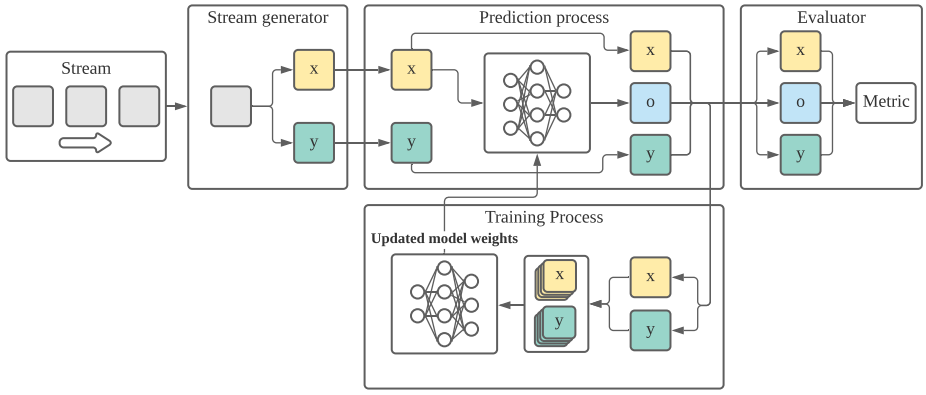


FIGURE 1. Asynchronous dual-pipeline DL framework.

3.1 ADLStream framework

In this study, we use the ADLStream framework for data streaming presented in [15]. As can be seen in Figure 1, the proposed system has two asynchronous 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 be adjusted to the evolution of the stream. This framework allows using 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 the ADLStream Python library can be found in [13] and the implementation of the experiments can be found here [14].

3.2 Datasets

3.2.1 Time series classification 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 and the number of classes of each dataset. For the experimental study, each dataset is simulated with three different stream speed: 20, 10 and 5 instances per seconds.

3.2.2 Concept drift datasets We have generated 15 synthetic datasets that simulate a wide variety of concept drift. More specifically, we can distinguish between three different types of concept drift: abrupt, which represents a significant sudden change on the data distribution; gradual, when the data stream oscillates between two distribution before completely drifting; and incremental, when the distribution experiences barely noticeable change at each time step. These streams have been simulated with different algorithms (RBF, LED, RandomTree, Agrawal, SEA) found in River library [16]. Table 2 provides a detailed description for the number of attributes, number of classes, imbalance rate (IR) and type of drift 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 describe the details of the evaluation method used for the data streaming classification task.

3.3.1 DL 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: MLP, LSTM, CNN and 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 [12], 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

TABLE 2. Datasets used for drift analysis.

Dataset	Attributes	Classes	IR	Drift type
RTGa	20	3	4 to 2	
RTGa3	20	3	4 to 30	
RTGa6	20	3	4 to 60	Abrupt
ARGWa-F1F4	9	2	2 to 1	
ARGWa-F2F5F8	9	2	1 to 50	
SEAa-F2F4	3	2	1	
RTGg	20	3	4 to 2	
RTGg3	20	3	4 to 30	
RTGg6	20	3	4 to 60	Gradual
ARGWg-F1F4	9	2	2 to 1	
ARGWg-F2F5F8	9	2	1 to 50	
SEAg-F2F4	3	2	1	
RBFi-slow	20	3	3	
RBFi-fast	20	3	3	Incremental
LED-4	24	10	1	

TABLE 3. MLP 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 4. CNN 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 5. LSTM 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 6. TCN 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	$f \times 4096 + 372096 + c \times 32$	

the short-term patterns using special gates [8]. 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 [11]. Moreover, they have far less trainable parameters than recurrent networks due to their weight sharing scheme [4]. More recently, TCNs have emerged as a specialized 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 [18]. Tables 3–6 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 3) 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 4), 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 6), 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, which incrementally updates the accuracy by testing the model with unseen examples [7]. 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 P_α at moment i can be formulated as follows, where o and y are the real and expected output respectively and n_i is the number of examples used to compute the loss function L_i.

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

with

$$S_\alpha(i) = L(y_i, o_i) + \alpha \times S_\alpha(i-1), \qquad B_\alpha(i) = n_i + \alpha \times B_\alpha(i-1)$$

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 i9-10900X and two NVIDIA GeForce RTX 2080 Ti 12GB GPU.

4.1 Prequential Kappa

Table 7 presents the prequential Kappa accuracy results obtained with the different models for each dataset at a stream speed of 5 instances per second. 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 DL architecture chosen.

4.2 Statistical analysis

The ranking of the accuracy of the models obtained with the Friedman test is presented in Table 8. 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 less than the significance level ($\alpha = 0.05$).

In Bergmann–Hommel's post-hoc analysis, we perform pair-wise comparisons for all models. Table 9 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

TABLE 7. Prequential Kappa accuracy results

#	Dataset	MLP	LSTM	CNN	TCN
1	TwoPatterns	0.741	**0.964**	0.960	**0.964**
2	CinCECGtorso	0.381	0.875	**0.930**	0.805
3	TwoLeadECG	0.536	0.816	**0.954**	0.886
4	Wafer	0.926	0.964	**0.965**	0.954
5	pendigits	0.965	0.964	**0.974**	0.969
6	FacesUCR	0.716	0.842	**0.850**	0.819
7	Mallat	0.899	0.904	**0.968**	0.921
8	FaceAll	0.704	**0.868**	0.842	0.820
9	Symbols	0.857	0.876	**0.919**	0.895
10	ItalyPowerDemand	**0.919**	0.866	0.916	0.915
11	ECG5000	0.810	**0.873**	0.872	0.867
12	MoteStrain	0.742	0.768	**0.794**	0.767
13	NonInvasiveFetalECGThorax1	0.050	0.620	**0.735**	0.597
14	NonInvasiveFetalECGThorax2	0.038	0.618	**0.809**	0.706
15	SwedishLeaf	0.634	0.674	**0.752**	0.701
16	FordA	0.003	0.397	0.526	**0.722**
17	Yoga	0.051	**0.686**	0.671	0.671
18	UWaveGestureLibraryX	0.589	**0.680**	0.654	0.652
19	FordB	0.006	0.351	0.476	**0.640**
20	ElectricDevices	0.438	**0.667**	0.598	0.516
21	UWaveGestureLibraryY	0.574	**0.593**	0.566	0.545
22	UWaveGestureLibraryZ	0.551	**0.614**	0.570	0.574
23	HandOutlines	0.006	0.500	**0.624**	0.230
24	InsectWingbeatSound	0.552	**0.572**	0.533	0.504
25	ShapesAll	0.425	**0.478**	0.449	0.463
26	MedicalImages	0.403	0.417	**0.432**	0.399
27	PhalangesOutlinesCorrect	0.140	0.427	**0.501**	0.430
28	ChlorineConcentration	0.002	0.316	**0.676**	0.587
29	Phoneme	0.002	0.041	**0.079**	0.029
	Average kappa	**0.471**	**0.663**	**0.710**	**0.674**
	Average time per instance (ms)	**20.595**	**43.832**	**22.075**	**27.820**

of the CNN and the other architectures considered. Nevertheless, there are no significant differences between the accuracy of LSTM and the TCN.

4.3 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 7. Obviously, 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. Due to the properties of parameter sharing, the CNN is able to process instances two times faster than the LSTM. The TCN is a more complex

TABLE 8. Friedman Test Ranking

Friedman	**Test Ranking**
CNN	1.517
TCN	2.368
LSTM	2.546
MLP	3.569

TABLE 9. Bergmann-Hommel's analysis

	PostHoc Analysis		
Comparison	p	z	Conclusion
MLP - CNN	<0.001	10.5	!=
MLP - TCN	<0.001	6.14	!=
LSTM - CNN	<0.001	5.26.1	!=
MLP - LSTM	<0.001	5.23	!=
TCN - CNN	<0.001	4.35	!=
LSTM - TCN	0.363	0.91	==

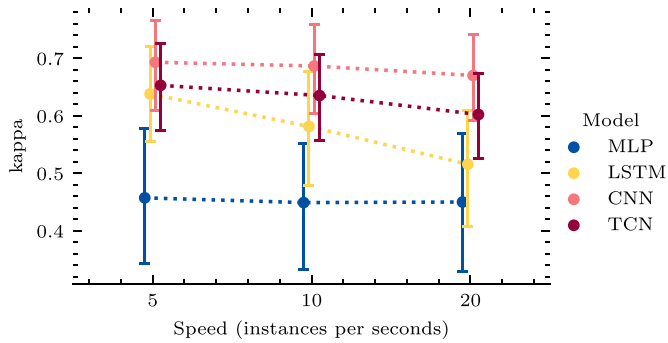


FIGURE 2. Kappa accuracy results for each model architecture at different stream speed. Points represent the mean average and extend vertically showing the 90% confidence interval.

version of the CNN, with more convolutions, resulting in a processing speed considerably slower than the CNN but still faster than the LSTM.

Due to the dual-pipeline architecture of ADLStream, the training time does not influence the system response time. However, it may have an impact on the accuracy performance of the model, as a faster model will be trained more often. Analogously, the speed of the input stream may affect the accuracy of the model. Figure 2 shows a comparison of the performance of the different DL architecture at different rates. We can appreciate how the slowest models (LSTM) are more sensitive to the speed of the stream. However, faster models, such as CNN or TCN, can achieve good results at high speed, and therefore there is almost no significant improvement when slowing down, or even no improvement at all in the case of MLP.

4.4 Concept drift analysis

Adaptability to concept drift is a key feature required for models that learn from evolving data streams. ADLStream framework benefits from the incremental nature of DL training procedure to handle concept drifts. However, unlike concept-drift detection methods, ADLStream's ability to adapt to change is highly dependent on the neural network used. For this reason, we consider it essential to study how different architectures deal with evolving data streams.

Recurrent and convolutional architectures are not optimal for this datasets as, unlike time series, these data do not have a sequential nature. Therefore, the aim of this analysis is not to determine which model achieves the best performance in terms of accuracy, but to understand how the different DL architectures behaves under the conceptual drifts.

Figure 3 shows the evolution of the prequential Kappa obtained by the different architectures over the concept-drift streams. In general, all models can be adapted to the new concept reasonably fast, achieving similar results. Comparing the performance of the different models, it is worth mentioning that, although all of them achieve similar accuracy at the beginning of the stream, CNN struggles more to adapt to the new distribution of the data stream. This behaviour is common for the different types of concept drift, and it is clearly illustrated in the evolution of the prequential Kappa of the most challenging dataset, such as RTGa3, RTGg and RBFi-fast.

From the abrupt datasets (Figure 3a), we can highlight that, except for the CNN, all models recover considerably fast from the first drifts. However, as we can see in RTGa6, the models struggle after undergoing multiple consecutive concept drifts. For ARGW and SEA data generator, all models perform exceptionally well, achieving high prediction accuracy and recovering very quickly from the concept drifts, or almost instantaneously in the case of SEA-F2F4.

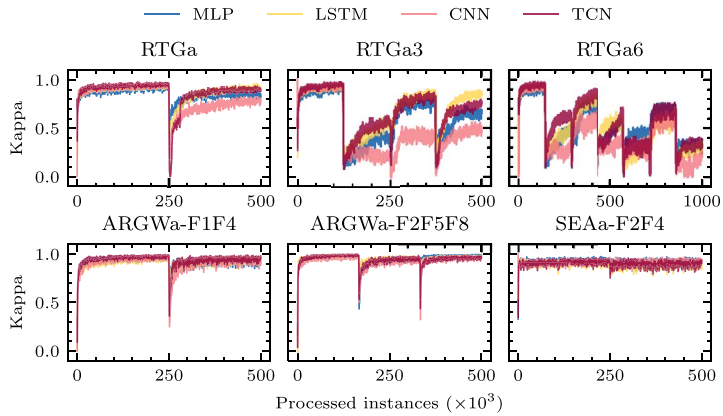
As Figure 3b illustrates, the models behave similarly to the abrupt version. However, as during the drift the data oscillates between the two distribution, the models do not recover until the drift is finished. However, when the drift is subtler like in RTGg, the performance drop is less severe than its abrupt version (RTGa). In addition, we must highlight the great performance of TCN, compared to other architectures, when recovering from multiple concept drift like in RTGg6 dataset. Moreover, we can observe that in the gradual version of ARGW and SEA generators, all models have similar behaviour. The models' accuracy bottoms out when the data oscillates evenly between the two distributions and starts to improve reaching its best performance immediately after the concept drift ends.

Regarding incremental drift datasets (Figure 3c), as we can see in RBFi-slow and LED-4, models can be adapted to changes, maintaining or even increasing the accuracy of their predictions. However, when the change in distribution is stronger, like in RBFi-fast, the models' performance starts to gradually decay, especially the CNN.

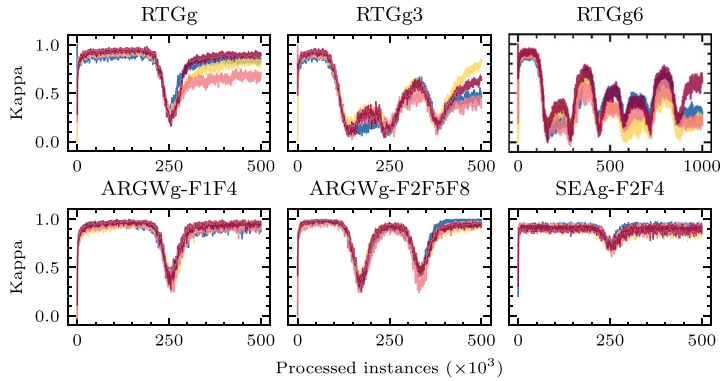
5 Conclusions

In this paper, the performance of several DL 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 MLP, recurrent and CNNs.

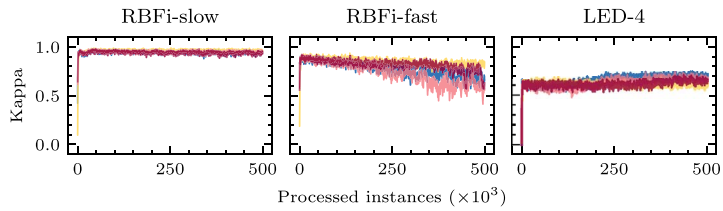
The results obtained, using 29 time series classification data streams, provide evidence that CNNs outperform the other models in terms of accuracy, with a very high processing rate. These characteristics present convolutional networks as a great alternative for processing data arriving at high speed. However, when studying the behaviour of the models under a concept drift, convolutional networks are the ones struggling the most to adapt to change. The other deep models, such as LSTM



(a) **Abrupt drift.**



(b) **Gradual drift.**



(c) **Incremental drift.**

FIGURE 3. Evolution of the prequential Kappa of each model over the concept-drift datasets.

or TCN, were not able to achieve such performance for time series classification in streaming; their processing rate was slower, but they show a better ability to adapt to concept drift.

Future works should carry out a more in-depth sensitivity study to evaluate how the speed of the stream affects the performance of the different DL models. Furthermore, a parameter optimization process could provide more specific architectures for the models and improve the performance. Future studies should also consider other DL models such as Echo State Networks or Stochastic TCNs, as well as traditional models such as Leveraging Bag, ARF and KUE for comparison.

Acknowledgements

This research has been funded by FEDER/Ministerio de Ciencia, Innovación y Universidades - Agencia Estatal de Investigación MICINN PID2020-117954RB-C22 and by the Andalusian Regional Government under the projects BIDASGRI (Big Data Technologies for Smart Grids; US-1263341) and adaptive hybrid models to predict solar and wind renewable energy production (P18-RT-2778). We are grateful to NVIDIA for their GPU Grant Program that has provided us high-quality GPU devices for carrying out the study.

References

[1] R. Anderson, Y. S. Koh, G. Dobbie and A. Bifet. Recurring concept meta-learning for evolving data streams. *Expert Systems with Applications*, **138**, 112832, 2019. https://doi.org/10.1016/j.eswa.2019.112832.

[2] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes and B. Pfahringer. Efficient online evaluation of big data stream classifiers. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'15*, pp. 59–68. ACM, New York, NY, USA, 2015.

[3] A. Bifet and R. Gavaldà. Adaptive learning from evolving data streams. In *Advances in Intelligent Data Analysis VIII*, N. M. Adams, C. Robardet, A. Siebes and J.-F. Boulicaut., eds, pp. 249–260. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[4] A. Borovykh, S. Bohte and C. W. Oosterlee. Dilated convolutional neural networks for time series forecasting. *Journal of Computational Finance*, **22**, 73–101, 2019.

[5] A. Cano and B. Krawczyk. Kappa updated ensemble for drifting data stream mining. *Machine Learning*, **109**, 109–218, 2019.

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

[7] J. Gama, R. Sebastião and P. P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, **90**, 317–346, 2013.

[8] F. A. Gers, J. Schmidhuber and F. Cummins. Learning to forget: continual prediction with LSTM. *Neural Computation*, **12**, 2451–2471, 2000.

[9] A. Ghazikhani, R. Monsefi and H. Sadoghi Yazdi. Online neural network model for non-stationary and imbalanced data stream classification. *International Journal of Machine Learning and Cybernetics*, **5**, 51–62, 2014.

[10] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes and T. Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, **106**, 1469–1495, 2017.

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

[12] P. Lara-Benitez, M. Carranza-Garcia and J. C. Riquelme. An experimental review on deep learning architectures for time series forecasting. *International Journal of Neural Systems*, **31**, 2130001, 2021.

[13] P. Lara-Benítez and M. Carranza-García. ADLStream: asynchronous dual-pipeline deep learning framework for online data stream mining. Available online: https://github.com/pedrolarben/ADLStream. (access date: 15 March 2021).

[14] P. Lara-Benítez and M. Carranza-García. Time series classification with deep learning in streaming. Available online: https://github.com/pedrolarben/deep-learning-online-classification. (access date: 15 March 2021).

[15] P. Lara-Benítez, M. Carranza-García, J. García-Gutiérrez and J. Riquelme. Asynchronous dual-pipeline deep learning framework for online data stream classification. *Integrated Computer-Aided Engineering*, **27**, 1–19, 01 2020.

[16] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem and A. Bifet. River: machine learning for streaming data in python. *Journal Of Machine Learning Research*, **22**, 1–8, 2021.

[17] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez and A. Troncoso. Deep learning for time series forecasting: a survey. *Big Data*, **9**, 3–21, 2021.

[18] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *The 4th International Conference on Learning Representations, ICLR 2016*. Conference Track Proceedings, San Juan, Puerto Rico, May 2–4, 2016, 2016.

[19] Y. Zhang, Y. Jiaxin, W. Liu and K. Ota. Ensemble classification for skewed data streams based on neural network. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **26**, 839–853, 2018.

Received 20 February 2021