A deep reinforcement learning approach for early classification of time series

C. Martinez
Data Analytics
bioMérieux
Marcy l'Etoile, France

G. Perrin
Data Analytics
bioMérieux
Marcy l'Etoile, France

E. Ramasso
FEMTO-ST Institute
Univ. Bourgogne Franche-Comté
Besançon, France

M. Rombaut
Univ. Grenoble Alpes, CNRS
Grenoble INP, GIPSA-Lab*
Grenoble, France

***Abstract*—In many real-world applications, ranging from predictive maintenance to personalized medicine, early classification of time series data is of paramount importance for supporting decision makers. In this article, we address this challenging task with a novel approach based on reinforcement learning. We introduce an early classifier agent, an end-to-end reinforcement learning agent (deep Q-network, DQN) [1] able to perform early classification in an efficient way. We formulate the early classification problem in a reinforcement learning framework: we introduce a suitable set of states and actions but we also define a specific reward function which aims at finding a compromise between earliness and classification accuracy. While most of the existing solutions do not explicitly take time into account in the final decision, this solution allows the user to set this trade-off in a more flexible way. In particular, we show experimentally on datasets from the UCR time series archive [2] that this agent is able to continually adapt its behavior without human intervention and progressively learn to compromise between accurate and fast predictions.**

***Index Terms*—time series, early classification, reinforcement learning, Deep Q-Network, time sensitive applications**

I. Introduction

Surrounded by information in our environment, we often collect data over time and make decisions based on these sequences of information. A physician will, for example, prescribe a new treatment to his patient as a response to the underlying identification of particular patterns in his medical exam history. A production chain manager will anticipate a breakdown after having identified atypical behaviours in the machines' signals. In time sensitive applications such as medical diagnosis, disaster prediction, intrusion detection and process control, particular situations should be recognized as soon as possible and decisions taken quickly in order to take the best possible actions. In this paper, we present a solution capable of analyzing time series, and more generally, streaming data, and predicting their class label early in time, using as little data as possible. Such a solution is referred to as an "early classifier" in the scientific literature. Just like humans do while making strategic decisions, these early classifiers should balance the two contradictory costs of classification accuracy and earliness. An early classifier should find an optimal critical point between acquiring more information to gather enough evidence and providing early results.

Emails:
Coralie Martinez (corresp. author): martinezcoralie.mc@gmail.com
Guillaume Perrin: guillaume.perrin@biomerieux.com
Emmanuel Ramasso: emmanuel.ramasso@femto-st.fr
Michèle Rombaut: michele.rombaut@gipsa-lab.grenoble-inp.fr
* Institute of Engineering Univ. Grenoble Alpes

A. Related work

The topic of early prediction first appears in [3] with a method capable of making predictions on prefixes of time series by combining simple literals through an adaptation of boosting algorithm. However, this method does not seek to optimize earliness. Later, several authors propose dictionary based methods and identify discriminant sequence patterns for classification [4] [5]. The trade-off between earliness and accuracy of classification is first mentioned in [4]. They identify patterns in symbolic sequences that are frequent, early and distinctive and classify a sequence as soon as it matches with a set of learned patterns. In [5], the authors seeks for local shapelets, sub-sequences in numerical time series relevant for classification but also early. A distance based approach is proposed in [6] which aims at adapting the highly competitive nearest neighbours approach on sequence classification to the early prediction problem. In [7], the authors address the early classification task as a problem of classification with confidence from incomplete information. Their solution is based on probability forecasting and on linear and quadratic discriminant functions as classifiers. In [8], the authors combine a set of probabilistic classifiers and an optimized stopping rule. The authors in [9] propose a "non myopic" framework which forecasts the future earliest time from which classification should be made through clustering and a set of classifiers.

B. Our contributions

In this article, we address the problem of early classification of time series as a decision making process. The main contribution holds in developing a novel approach based on reinforcement learning which is traditionally used in robotics and games environment [1] and has recently exhibited state-of-the-art results in these challenging tasks. Here we introduce an early classifier agent, an end-to-end reinforcement learning agent able to perform early classification in an efficient way. We formulate the early classification problem in a reinforcement learning framework: we introduce a suitable set of states

ISBN 978-90-827970-1-5 © EURASIP 2018

and actions but we also define a specific reward function which aims at finding a compromise between earliness and classification accuracy. The early classifier agent is trained with the popular Deep Q-Network (DQN) algorithm from [1]. In experiments, we particularly show on UCR time series benchmark [2] that this agent is able to continually adapt its behavior without human intervention. This makes this approach of particular interest compared to standard approaches which are generally based on a set of decision rules (such as if-then rules).

The advantages of the proposed solution are the following:

- We introduce a reward function balancing the competing costs of accuracy and earliness with a trade-off parameter. As a consequence, the user can directly affect the degree of importance of time in the final prediction.
- During its training, the early classifier agent continually learns to compromise between accurate and fast predictions. The user can then choose the agent's behaviour that performed best according to his trade-off criteria.
- It can be plugged with a large number of reward functions and, for example, deal with multi-criteria evaluation and imbalanced misclassification costs.
- It benefits from an end-to-end learning with no use of human intervention in the design of the decision rules and in the features extraction on the time series. It autonomously and simultaneously learns optimal patterns of interest in the time series for classification and optimal strategic decisions for time of predictions.
- It is suitable for diverse types of temporal sequences, such as univariate and multivariate time series and symbolic sequences.

In section II, we introduce notations and define the problem of early classification. In section III, we solve the problem with the definition of an early classifier agent in a reinforcement learning context. In section IV, we evaluate our method on datasets from the UCR time series archive.

II. Problem description

A. Notation

Let $\mathbf{S} \in \mathbb{R}^{p*T}$ be a time series (eq. 1) with length $T \in \mathbb{N}^+$ and dimension $p \in \mathbb{N}^+$. Our work both applies to univariate time series ($p = 1$) and multivariate time series ($p > 1$). For the sake of simplicity, we suppose that each dimension has equal length but the proposed solution can be applied to time series with variable length and with unequal dimension's length. Let $\mathbf{S}_{:t}$ be the prefix of the time series $\mathbf{S}$ acquired until time t (eq. 2). Given our notations, we have $\mathbf{S} = \mathbf{S}_{:T}$.

$$\mathbf{S} = \begin{pmatrix} s_1^1 & \cdots & s_1^p \\ \vdots & \ddots & \vdots \\ s_T^1 & \cdots & s_T^p \end{pmatrix} \quad (1) \qquad \mathbf{S}_{:t} = \begin{pmatrix} s_1^1 & \cdots & s_1^p \\ \vdots & \ddots & \vdots \\ s_t^1 & \cdots & s_t^p \end{pmatrix}_{t \leq T} \quad (2)$$

Let $\mathcal{S}$ be the set of complete time series $\mathbf{S}$, such that $\mathcal{S} = \{\mathbf{S}\}$. $\mathcal{S}_{:t}$ refers to the set of truncated time series, such that $\mathcal{S}_{:t} = \{\mathbf{S}_{:t}\}$. Each time series in $\mathcal{S}$ is associated to a class label $l \in \mathcal{L}$, with $\mathcal{L}$ the set of labels.

B. Time series classification

Given a learning dataset $\mathcal{D}$ composed of time series associated to their class label (eq. 3), a classification model C is a mathematical function modelling the relationship between data evolution over time and associated label. It maps between a time series $\mathbf{S}^i \in \mathcal{S}$ and a label $l^i \in \mathcal{L}$, such that $C : \mathcal{S} \rightarrow \mathcal{L}$.

$$\mathcal{D} = \{(\mathbf{S}^i, l^i)_{i=1..n} \mid \mathbf{S}^i \in \mathcal{S}, l^i \in \mathcal{L}\} \quad (3)$$

with n being the number of examples in the dataset $\mathcal{D}$. To evaluate the performance of a classification model C on a test dataset $\mathcal{D}'$, we measure the classifier accuracy: the ratio of correctly labelled examples among all predictions (eq. 4).

$$Accuracy(C) = \frac{\#\{\hat{l^i} = l^i \mid \hat{l^i} = C(\mathbf{S}^i), (\mathbf{S}^i, l^i) \in \mathcal{D}'\}}{\#\mathcal{D}'} \quad (4)$$

with $\#\mathcal{D}'$ being the number of elements in the test dataset and $\hat{l^i}$ the prediction of the classifier.

C. Early classification of time series

We define an early classifier as a mathematical model capable of analyzing time series and identifying their class label early in time. It should compromise between accuracy and earliness of classification under some criteria accepted by the user's application. An early classifier C has to meet two requirements:

- Analyze prefixes of temporal sequences and be ready to classify this partial information at any time (eq. 5).

$$\forall \mathbf{S} \in \mathcal{S}, \forall t \in [1, T], C : \mathcal{S}_{:t} \rightarrow \mathcal{L} \quad (5)$$

- Make a decision about when (at which time step $t_{pred} \in [1, T]$) to classify.

$\forall (\mathbf{S}^i, l^i) \in \mathcal{D}$, an early classifier C outputs a time of prediction and a classification result: $(C(\mathbf{S}_{:t_{pred}}), t_{pred})$.

III. Early classification in a reinforcement learning framework

In this article, we propose an original use of reinforcement learning to train an end-to-end agent able to perform early classification in an efficient way. The contribution holds in the definition of an early classifier agent. It is based on the introduction of a suitable set of states and actions for reinforcement learning and on the definition of a specific reward function which aims at finding a compromise between earliness and accuracy. We propose to train the early classifier agent with the standard Deep-Q-Network algorithm from [1] which aims at approximating the agent's behaviour function with a deep neural network combined with Q-learning [10].

A. Background

1) Reinforcement learning: Reinforcement learning is an active field of research for sequential decision making. It is used to train agents interacting with an environment such as artificial intelligence playing video games. Agents are trained while playing what's called "episodes". At the beginning of an episode, the agent receives an observation of its initial state

ISBN 978-90-827970-1-5 © EURASIP 2018

$\mathbf{O}_1$. At each time step t, after it receives an observation $\mathbf{O}_t$, the agent picks an action a_t. In response, the environment gives the agent a feedback: a reward r_t and a new observation $\mathbf{O}_{t+1}$. These interactions go on until the agent reaches a terminal state, leading to the end of the current episode and the start of a new one. The agent is characterized by his policy π modelling how it picks its actions. The policy π is a behaviour function (eq. 6) which returns the probability of picking an action a given a particular state $\mathbf{O}$.

$$\pi(a|\mathbf{O}) = \mathbb{P}[a_t = a|\mathbf{O}_t = \mathbf{O}] \tag{6}$$

The agent's goal is to receive the maximum total discounted reward g_t which represents the immediate reward plus the future discounted rewards (eq. 7).

$$g_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \text{ with } \gamma \in [0,1] \text{ the discount factor} \tag{7}$$

In reinforcement learning, the agent has to learn an optimal behaviour π^* from its interactions $< \mathbf{O}_t, a_t, r_t, \mathbf{O}_{t+1} >$ with the environment during episodes of training. It must find the best possible action to take in every circumstance. This type of learning is close to human behaviour where one takes some decisions, observes results and consequences, and progressively learns from his errors.

Depending on its policy π, the agent can evaluate the expected return g_t it can get by starting the game from state $\mathbf{O}$, picking a particular action a and then following its policy π. This is the action value function, also called the Q-function (eq. 8).

$$Q_\pi(\mathbf{O}, a) = \mathbb{E}_\pi[g_t|\mathbf{O}_t = \mathbf{O}, a_t = a] \tag{8}$$

$$Q_\pi(\mathbf{O}, a) = \mathbb{E}_\pi[r_t + \gamma Q_\pi(\mathbf{O}_{t+1}, a_{t+1})|\mathbf{O}_t = \mathbf{O}, a_t = a] \tag{9}$$

with $a_{t+1} \sim \pi(.|\mathbf{O}_{t+1})$. The Q-function indicates, for a given policy π, if selecting an action a in a particular state $\mathbf{O}$ is likely to have good repercussions in the following steps by getting large rewards or not. For two actions a_1 and a_2, the Q-function indicates which action is the optimal choice to make. If $Q_\pi(\mathbf{O}, a_1) > Q_\pi(\mathbf{O}, a_2)$, then action a_1 is a better choice to make when the agent is in state $\mathbf{O}$ than action a_2.

The goal in reinforcement learning problems is to find the optimal policy π^* the agent should follow in order to get the maximum rewards g_t. One way to achieve an optimal policy π^* is to find the optimal Q-function Q^*. If the optimal Q-function is known then the optimal policy can naturally be inferred from it by acting greedily, that is to say by picking the action maximizing Q^* (eq. 10).

$$\pi^*(a|\mathbf{O}) = \begin{cases} 1 & \text{if } a = \arg\max_a Q^*(\mathbf{O}, a) \\ 0 & \text{else} \end{cases} \tag{10}$$

2) Deep Q-Network algorithm: The DQN algorithm from [1] aims at approximating the optimal Q-function Q^* from interactions $< \mathbf{O}_t, a_t, r_t, \mathbf{O}_{t+1} >$ between an agent and its environment. It approximates the optimal Q-function Q^* by a deep neural network $Q(\mathbf{O}, a, \mathbf{\Theta})$ with parameters $\mathbf{\Theta}$, also called the Deep-Q-Network. Parameters update is detailed in [1]. In the following, we briefly give an overview of the DQN algorithm.

During episodes of training, starting with random parameters $\mathbf{\Theta}_1$, the agent modifies the parameters $\mathbf{\Theta}$ of its function approximator $Q(\mathbf{O}, a, \mathbf{\Theta})$ in order to get to an accurate approximation of the true Q-function. Consequently, it modifies its policy $\pi_{\mathbf{\Theta}}$ by acting greedily over its approximated Q-function (eq. 10). After each interaction $< \mathbf{O}_t, a_t, r_t, \mathbf{O}_{t+1} >$, the following scheme is repeated. The agent stores its interaction in its replay memory D, allowing to reuse this experience later. It randomly samples a mini-batch of transitions $< \mathbf{O}, a, r, \mathbf{O'} >$ from D and performs a gradient descent step on the Deep-Q-Network with respect to the loss function (eq. 11).

$$L_t(\mathbf{\Theta}_t) = (r + \gamma \max_{a'} Q(\mathbf{O}', a', \mathbf{\Theta}_t^-) - Q(\mathbf{O}, a, \mathbf{\Theta}_t))^2 \tag{11}$$

with $\mathbf{\Theta}_t^-$ an old fixed version of the Deep-Q-Network. By doing so, the agent progressively estimates the relationships between being in a state $\mathbf{O}_t$, choosing an action a_t and getting future rewards g_t.

B. Early classification as a decision making process

We consider an early classifier as an algorithm allowed to make decisions. While receiving streaming data, the algorithm analyzes the sequence and can make an action. It can *predict a class label* based on the assumption that it collected enough information to identify discriminant patterns of interest. Or it can *wait* for future additional information. We propose to train an early classifier as an agent evolving in a reinforcement learning framework where:

- The observations $\mathbf{O}_t$ given to the agent at time t are partial time series acquired until time t.

$$\mathbf{O}_t = \mathbf{S}_{:t} \tag{12}$$

- The actions a_t the agent can take are either to *wait* for additional observations or to *predict a class label* in $\mathcal{L}$.

$$a_t \in \mathcal{A}, \text{ with } \mathcal{A} = \{wait, \bigcup_{k \in \mathcal{L}} predict\ label\ k\} \tag{13}$$

- The agent picks an action a_t based on its policy $\pi_{\mathbf{\Theta}}$ with parameters $\mathbf{\Theta}$, such that $a_t = \pi_{\mathbf{\Theta}}(\mathbf{O}_t)$.

$$\pi_{\mathbf{\Theta}} : \{\mathbf{S}_{:t}, \forall t \in [1, T]\} \rightarrow \mathcal{A} \tag{14}$$

  If $a_t = wait$, the episode goes on and the next observation is the partial time series with an additional point: $\mathbf{O}_{t+1} = \mathbf{S}_{:t+1}$. If $a_t \in \{ \bigcup_{k \in \mathcal{L}} predict\ label\ k\}$, the episode terminates.
- An episode ends when the partial sequence is fully completed, $\mathbf{O}_t = \mathbf{S}$, or when the agent predicts a class label, $a_t \in \{ \bigcup_{k \in \mathcal{L}} predict\ label\ k\}$.
- Rewards r_t are a major component of an agent's training. They encode the agent's task and will highly affect its behaviour. Rewards should reflect an early classifier goal: to give fast predictions while maintaining an acceptable accuracy.

ISBN 978-90-827970-1-5 © EURASIP 2018

C. Reward function for early classification

When deep reinforcement learning interacts with video games, the reward traditionally relates to the score of the game. Here, we propose a solution for the attribution of rewards to an early classifier agent in a reinforcement learning context. The reward attribution will be drawn from a reward function $R : \mathcal{A}, \mathcal{D} \rightarrow \mathbb{R}$ that will analyze the relevance of an action a_t given a particular observation $\mathbf{O}_t = \mathbf{S}_{:t}$, corresponding to the underlying time series $\mathbf{S}$ with true label l, $(\mathbf{S}, l) \in \mathcal{D}$.

$$r_t = R(a_t, \mathbf{S}_{:t}, l) \tag{15}$$

We propose a reward function balancing the two competing costs of a prediction accuracy and earliness (eq. 16).

$$R(a_t, \mathbf{S}_{:t}, l) = \begin{cases} 1 & \text{if } a_t = predict\ label\ l \\ -1 & \text{if } a_t \in \bigcup\limits_{k \in \mathcal{L} \setminus l} predict\ label\ k \\ -\lambda t^p & \text{if } a_t = wait \end{cases} \tag{16}$$

with $p \geq 0$. $\lambda \in \mathbb{R}^+$ is the trade-off parameter between earliness and accuracy. Under the proposed formulation of rewards (eq. 16), the agent gets positive reward when it correctly classifies the time series. An incorrect prediction leads to a penalty of one "point". Delaying the prediction costs the agent a variable amount of "points". With parameter p set to 0, the agent will receive the same penalties for delaying the prediction independently of time. With $p > 0$, the penalty becomes time-dependant. At the beginning of the episode, when the partial time series has limited information, the agent is less penalized for delaying the prediction than later in the episode when it received more inputs from the sequence.

The trade-off parameter λ allows the user to control the compromise he is willing to make between speed and accuracy. The larger λ, the faster the agent will be encouraged to make its predictions.

D. Deep Q-Network algorithm for early classification

The proposed definition of states, actions and rewards allow us to train an early classifier agent with reinforcement learning. To find the optimal policy π^* (eq. 10) that will lead to maximum rewards g_t, we adapt the DQN algorithm [1] and look for the optimal Q-function parametrized using a deep neural network with parameters $\mathbf{\Theta}$.

At training time, the agent will train and modify its behaviour by playing successive episodes. In each episode, the agent will train on a time series from the learning dataset: $(\mathbf{S}, l) \in \mathcal{D}$. At time step t of the episode, the agent receives the partial time series $\mathbf{S}_{:t}$ and has to decide between gathering more information ($a_t = wait$) or making a prediction ($a_t \in \bigcup\limits_{k \in \mathcal{L}} predict\ label\ k$). As a feedback, the agent receives a reward $R(a_t, \mathbf{S}_{:t}, l)$. Depending on its choice of action, the episode either terminates or the agent receives a new observation $\mathbf{S}_{:_{t+1}}$. Q is then re-evaluated and $\mathbf{\Theta}$ udpated.

At test time, the policy $\pi_{\mathbf{\Theta}^*}$ with best parameters $\mathbf{\Theta}^*$ will be used to early classify new time series.

The use of a deep neural network as a function approximator of the Q-function leads to a end-to-end learning of both features of interest in the time series and of decision rules about when and which class label should be predicted.

IV. Experimental results

We run experimental tests on the UCR archive [2] which is widely used as benchmark for classification and clustering of time series. We evaluate the proposed solution on Gun-Point, Wafer and ECG datasets. These datasets have various amount of training data and allow to evaluate the suitability of the solution when few training samples are available.

A. Agent training

Each dataset is originally split into a training and a testing set in the UCR archive. We use time series from the training set as episodes of training for the agent. We run the DQN algorithm for 100,000 iterations (updates of the deep neural network's parameters $\mathbf{\Theta}$ (eq 11)). Samples from the training set can be re-used several times over training. The neural networks used to approximate the Q-function are composed of convolution filters to learn time dependencies in the sequences and dropout to prevent over-fitting. The amount of layers, filters and dropout vary from a dataset to another depending on its complexity and amount of training data. Architectures of neural networks were tuned through the different experiences.

Figure 1 shows the evolution of the agent behaviour over training when rewarded with parameters $p = 1/3$ and $\lambda = 0.001$. It shows that the agent is able to continually adapt its behaviour without human intervention. At the beginning of its training (blue dots), the agent gave its classification results at $t < 10$ in average leading to a poor accuracy (eq. 4) of 70%. After 100,000 training iterations (yellow dots), it learned to slow its prediction down to $t = 35$ and reached an accuracy superior to 95% on the training set.

During experiments, we noticed that the choice of hyper-parameters λ and p for the reward function can cause the agent to learn a sub-optimal policy. Too small values of λ encouraged the agent to wait until the end of a sequence before predicting a class label while large values of λ encouraged it to give immediate predictions at the expense of accuracy. To tune these hyper-parameters, we conducted a grid search and selected those with best performance on the training set. Optimal parameters λ and p vary from a dataset to another depending on the maximal length of the sequences and on their complexity.

One advantage of the proposed solution is that it gives the user the freedom to evaluate the agent performance regularly over training and then pick the agent's policy that performed best according to his criteria and will to compromise.

B. Testing

From the experiment illustrated in figure 1, we chose to keep as the output of the learning process the agent's policy that performed best during training: among the most accurate policies, we selected the fastest one (policy surrounded by a

ISBN 978-90-827970-1-5 © EURASIP 2018

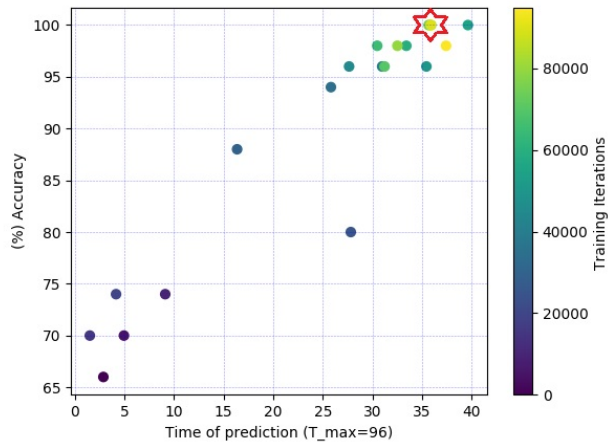

Fig. 1. Evolution of the early classifier agent behaviour on Gun-Point dataset. The scatter plot shows the relationship between accuracy (in percentage) and average time of prediction of the agent over training. We evaluate the agent on the whole training set every 5,000 iterations. Each evaluation corresponds to one dot. Dot points are coloured according to iterations of training: blue dots correspond to early training while yellow dots correspond to the agent's performance after 100,000 iterations of training. We evaluate the agent's policy surrounded by the red star on the testing set and we report its performance in table I. In this experiment, the agent learned to slow its predictions down and improved its accuracy over training.

TABLE I
RESULTS ON THREE DATASETS FROM UCR TIME SERIES ARCHIVE

Dataset		Early classifier agent	ECTS	EDSC	1NN Full
Gun-Point: 2 classes, T = 150 50 train. samples 150 test. samples	Accuracy	96%	86.67%	94.67%	91.33%
	Ave. Len.	32.47	70.39	69.3	150
	Coverage	100%	100%	100%	100%
Wafer: 2 classes, T = 152 1000 train. samples 6174 test. samples	Accuracy	99.32%	99.08%	98.87%	99.55%
	Ave. Len.	5.73	67.39	38.97	152
	Coverage	100%	100%	100%	100%
ECG: 2 classes, T = 96 100 train. samples 100 test. samples	Accuracy	89%	89%	88%	88%
	Ave. Len.	16.09	57.71	30.93	96
	Coverage	100%	100%	100%	100%

Accuracy is defined in eq. 4. T is the maximal length of the time series. Ave Len is the average length of time series used before classification, i.e. the average time of prediction. Coverage is the percentage of classified time series in the dataset.

red star in figure 1). In table I, we report the performance of this policy on Gun-Point testing set. We also trained early classifier agents on Wafer and ECG training sets. We selected policies that performed best during training and reported their performance on Wafer and ECG testing sets in table I.

As a comparison, we indicate the performance of Early Classification on Time Series (ECTS) [6] and Early Distinctive Shapelet Classification (EDSC) [5] methods. We did not reproduce their experiments but simply reported results mentioned in the original papers. We also indicate the performance of a 1 Nearest Neighbor (1NN) classifier on full time series with results provided in the UCR archive.

On Gun-Point dataset, the agent predicted once it received 22% of the full sequence in average and reached an accuracy superior to that of full length 1NN method.

On Wafer dataset, our solution has an accuracy slightly inferior to that obtained by the full length 1NN method and an average time of prediction of 4% of the full length. This speed in prediction is due to the identification of a very early pattern in time series from class 2.

On ECG dataset, the agent gave fast predictions (17% of full length) with an accuracy comparable to that of other methods. With all three datasets, we experienced over-fitting when the neural network architecture was not appropriately sized.

From these experiments, we showed that the proposed solution can achieve early classification and can retain an accuracy comparable to that of the full length 1NN classifier.

V. CONCLUSION

To tackle the problem of early classification, we propose an original use of reinforcement learning in order to train an end-to-end early classifier agent with a simultaneous learning of both features in the time series and decision rules. Our experimental results show that the early classifier agent can achieve effective early classification with fast and accurate predictions. In this work we suggest a static setting of the reward function but more efforts could be put onto the identification of optimal parameters. As future work, we plan to improve the proposed approach with a dynamic adjustment of the reward function parameters over training based on the user trade-off criteria. We will also propose a new management of the agent's replay memory which could be more suitable for the problem of early classification.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[2] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

[3] J. J. Rodríguez, C. J. Alonso, and H. Boström, "Boosting interval based literals," *Intelligent Data Analysis*, vol. 5, no. 3, pp. 245–262, 2001.

[4] Z. Xing, J. Pei, G. Dong, and P. S. Yu, "Mining sequence classifiers for early prediction," in *Proceedings of the 2008 SIAM International Conference on Data Mining*, pp. 644–655, SIAM, 2008.

[5] Z. Xing, J. Pei, P. S. Yu, and K. Wang, "Extracting interpretable features for early classification on time series," in *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 247–258, SIAM, 2011.

[6] Z. Xing, J. Pei, and S. Y. Philip, "Early prediction on time series: A nearest neighbor approach.," in *IJCAI*, pp. 1297–1302, 2009.

[7] N. Parrish, H. S. Anderson, M. R. Gupta, and D. Y. Hsiao, "Classifying with confidence from incomplete information," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 3561–3589, 2013.

[8] U. Mori, A. Mendiburu, S. Dasgupta, and J. A. Lozano, "Early classification of time series from a cost minimization point of view," in *Proceedings of the NIPS Time Series Workshop*, 2015.

[9] A. Dachraoui, A. Bondu, and A. Cornuéjols, "Early classification of time series as a non myopic sequential decision making problem," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 433–447, Springer, 2015.

[10] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

ISBN 978-90-827970-1-5 © EURASIP 2018