Does catastrophic forgetting negatively affect financial predictions?

Alberto Zurli[1,2], Alessia Bertugli[3], and Jacopo Credi[2]

[1] Università di Modena e Reggio Emilia, Modena, Italy, name.surname@unimore.it
[2] Axyon AI, Modena, Italy, name.surname@axyon.ai
[3] Università di Trento, Trento, Italy, name.surname@unitn.it

Abstract. Nowadays, financial markets produce a large amount of data, in the form of historical time series, which quantitative researchers have recently attempted at predicting with deep learning models. These models are constantly updated with new incoming data in an online fashion. However, artificial neural networks tend to exhibit poor adaptability, fitting the last seen trends, without keeping the information from the previous ones. Continual learning studies this problem, called catastrophic forgetting, to preserve the knowledge acquired in the past and exploiting it for learning new trends. This paper evaluates and highlights continual learning techniques applied to financial historical time series in a context of binary classification (upward or downward trend). The main state-of-the-art algorithms have been evaluated with data derived from a practical scenario, highlighting how the application of continual learning techniques allows for better performance in the financial field against conventional online approaches. [1]

1 Introduction

The financial market is a worldwide virtual place meant for the exchange of financial instruments, such as shares, contracts, stocks, or commodities. In the past, investments were made in person by a small circle of domain experts, basing their decisions on experience and a small amount of data. Nowadays, the actors and the dynamics involved have radically changed. The strong availability of real-time data and the great computational capabilities of computers brought that most of the investments are not made only by human traders. Therefore, they are assisted by an ever-increasing number of equipped "intelligent machines" able to understand the best timing for carrying out financial transactions. Consequently, the concept of algorithmic trading has taken hold. The algorithms are characterized by a set of rules defined to perform certain actions based on the state of the market. These rules are written by human traders who study particular techniques for market choices. The natural evolution of these algorithms is the use of cutting-edge machine learning and deep learning techniques to develop predictive models. This way, human intervention

[1] Code is available at https://github.com/albertozurli/cl_timeseries.

is no longer required to define steady rules, and decisions support tools can rely only on data and their patterns over time, through the use of tailored neural networks. The problem of using machine and deep learning techniques with time series is the need to periodically retrain the model to allow for the updating and acquisition of knowledge of the most recent data. This leads neural networks to suffer from catastrophic forgetting, meaning that their weights are overwritten in favor of last seen data, losing their predictive power over the older data. It has been shown that market trends over time are qualitatively similar, and therefore learning the behavior of the time series in the past could be useful for predicting its future trend. Continual learning is a deep learning technique developed to face the catastrophic forgetting problem, and that has recently shown promising and significant results even in image classification areas. The study and application of these techniques could be relevant within the financial market, due to its cyclical nature.

Exploring the potential of continual learning is a novel topic in the financial world and therefore, to the best of our knowledge, this is the first work that analyzes and evaluates the potential of continual learning in-depth. In particular, we analyzed whether catastrophic forgetting could lead to a bad predictive performance in financial time series classification, comparing the classical online learning paradigm with continual learning techniques designed to overcome catastrophic forgetting. Specifically, a problem of binary classification of time series has been studied, where each time series consists of a vector of consecutive daily samplings. The goal is to predict whether this will have an increasing or decreasing trend in the future, leading to a "buy" or "sell" decision. We define continual learning tasks as periods, which may have different lengths, but specific behavior. For example, the first task can be represented by a stationary prices trend, the second one by a sudden increase of prices, the third one by a slow drop. Each task is time bounded by a change of these regimes. Approaching the problem this way, we place it in a Domain-Incremental Learning (Domain-IL) scenario [20], where the distribution of the classes remains unchanged while the change of task is defined by a variation of the distribution of input data. Several state-of-the-art continual learning methods have been developed and deeply analyzed. Eventually, the experimental results achieved suggest that CL techniques, alleviating the forgetting phenomenon, exhibit better performance than online learning.

2 Related work

2.1 Financial market predictions

Time series research has experienced strong growth in several sectors in the last few years: from the prediction of pedestrian and car trajectories for video surveillance [13,1] to the prediction of machinery failures in industries. Among them, financial market predictions have been deeply investigated leading to the development of increasingly sophisticated algorithms capable of predicting the trend of the financial market. Machine learning and deep learning techniques have been

applied to financial time series, to make these algorithms as automatic as possible to facilitate the traders' decisions. However, due to the unpredictability of the market is still hard to design machine learning algorithms that can properly work on financial time series [14]. In fact, deep learning models, like 1-D convolutional neural networks, multi-layer perceptrons, temporal transformers [19]), that achieve the State Of The Art on other tasks, do not always exhibit satisfactory performance on financial problems. For this reason, deep learning for finance is at the cutting-edge of research and in the last few years, several methods have been designed or adapted specifically to financial time series [10,6,15].

2.2 Continual learning techniques

Real-world computational systems are exposed to continuous data flow, and they have to learn and remember multiple tasks. The traditional optimization technique applied to machine and deep learning models is not suited to tackle continuous data flow due to their nature to forget and overwrite the previously learned knowledge. Neural networks tend to overwrite the previously acquired knowledge by updating the network parameters when trained with data from a new task or distribution. This phenomenon is known as catastrophic forgetting [12]. It typically leads to a sudden drop in performance or, in the worst case, the total overwriting of the old task against the new one. If you train on sequential tasks, the performance of traditional neural networks decreases in past tasks as the number of tasks increases. Continual learning techniques try to solve this issue, through specifically designed algorithms that alleviate catastrophic forgetting. Several continual learning methods have been studied and presented in the literature in the last few years. However, no consensus has been reached about a globally valid continual learning algorithm. We can therefore distinguish three categories: replay-based, regularization-based and parameter isolation methods. *Replay-based methods* [16,3,2], also known as rehearsal, save examples from previous tasks. These examples are sampled from a memory buffer and used as input when training the current task. These previous task samples contribute to the loss function to prevent forgetting and interference. The rehearsal methods use only a subset typically limited by the size of the memory used to save the examples. In the absence of previous task data, pseudo-rehearsal techniques collect information regarding the distribution of data and, through a generative model, generate fictitious data but similar to the distribution of the original dataset. These methods generally scale very well but, for a large number of tasks, they can run into problems of saturation of the available memory. On the other hand, *regularization-based methods* [21,9] act on the parameters of the model, discouraging the updating of neurons or layers deemed relevant for the individual tasks. The various methods differ on the type of penalty to be applied and how this is calculated. This allows consolidating the knowledge of the previous tasks, leaving the model the possibility to observe and learn data coming from the current task. The regularization methods, with rare exceptions, are not particularly computational expansive but they lack scalability. To preserve past knowledge, the neural network weights are "frozen". So, when the number

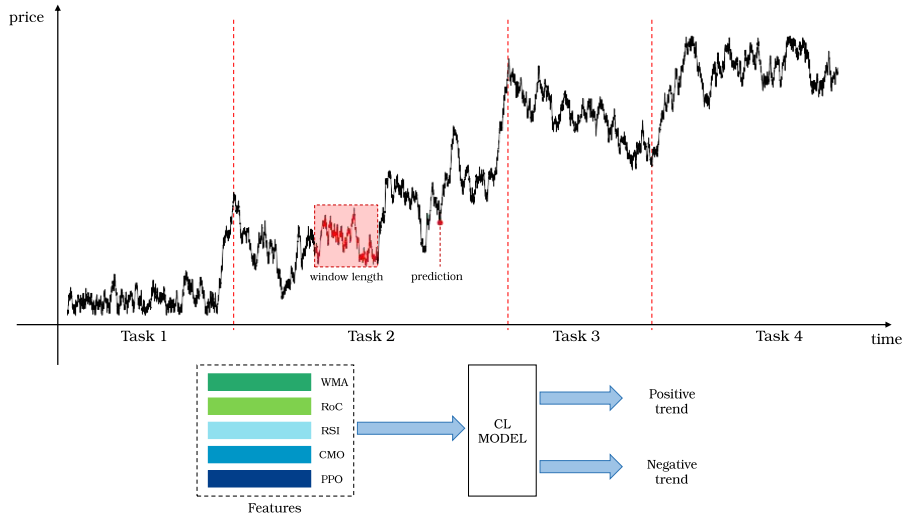

Fig. 1. Data flow of a financial time series in our setting. The prices time series is subdivided into tasks by a change-point detector (dashed red lines). Raw prices, relative to a certain period (determined by a window length), are turned into financial indicators, that become the inputs of the neural network. The problem consists of a binary classification to predict prices trend (positive or negative) at N time step later.

of tasks increases over time, the neural network tends to saturate, bringing to low performance on the last tasks. Finally, *parameter isolation or architectural methods* [17] dedicate different parts of the model or make a different copy of if to each task exclusively. In the absence of constraints on the architecture, a new model's branch, with exclusive parameters, is instantiated for each new task. The final layers of the model involved in the classification can be exclusive or shared. The architectural methods are extremely expensive from the point of view of the memory and computational power required since there are N models for N tasks, but they have potentially infinite scalability.

3 Problem formulation

A great advantage of trading using algorithms is the ability to process a large amount of data in real-time to extract as much information as possible. It is necessary to provide the model with data that best reflects the state of the market in each timestep to make the most appropriate investment choice. In this section, we present the data, their structure, and the features generated by the models.

3.1 Financial time series and indicators

The input provided to the model consist of fixed-length sequences representing the trend of financial time series (daily close values) over a period of one month (i.e. 20 business days) of observations. Sequences are constructed by taking all possible consecutive 20-day windows in the available data. Each series is used to obtain the features and time sequences by building a different dataset, giving rise to an analysis of multivariate historical series for each financial series. Even if a neural network can extrapolate the information it deems relevant in a completely autonomous way, using only time series does not carry out any learning despite copying the label of the previous example in output. This fact demonstrates a dependence between time series even where they have no provable relationships. Consequently, it was necessary to manipulate the series by carrying out various engineering operations to obtain features. The first step is carried out to eliminate copying between outputs, a phenomenon that is not uncommon in time series and which is usually solved by creating a series obtained as the difference between data points of two consecutive instants. The input has doubled its dimensionality from a vector to a matrix. The increase in dimensionality is an aspect to be taken into account in a problem of this type: if, on the one hand, only the raw series is insufficient, on the other hand, using too many features could push the model to focus on unnecessary or worse misleading information. This aspect was taken into account in choosing which financial indicators to use as features that could provide helpful information. Thanks to moving averages, we can catch the real market trends, removing irrelevant fluctuations in the original series. Using *Weighted Moving Average*, we put a specific weight which decades overtime to any timestep, giving a higher impact on more recent timesteps. *Rate of Change* provides a first qualitative analysis regarding possible changepoints computing the percentual difference between the current timestep and another n step in the past. A minor variation means that current data do not differ from previous ones, while a bigger one indicates a possible change in the distribution. To measure variation speed and magnitude in time series, we used *Relative Strength Index*. This momentum oscillator reports the strength of the current time series trend, highlighting periods of excessive overconfidence and underconfidence of stocks. *Chande Momentum Oscillator* provides more fine-grain values in respect to RSI, enabling the model to confirm or deny results of the previous indicator while at the same time finding out new changepoints. Finally, we can keep track of market reversals with the *Percentage Price Oscillator*, another momentum oscillator able to compare moving averages with different temporal horizons. An exciting feature of the chosen indicators is that they collect information about the past and mediate it with the present. This allows forming features that represent the current state and a set of past timesteps used to constitute the feature itself. Many indicators are characterized by an observation period overtime in the series. Those listed below have been generated for 5 to 20 days to extract informative content both in the short and long term. Shorter periods could capture information that is too little specific for the sequence,

just as long periods would provide information of a greater window than the observation window of the model itself.

3.2 Definition of domain regimes

The classic problems of continual learning do not deal with time series, but rather images. Consequently, it is necessary to understand which algorithms could be applied, taking into account that input data of a different nature lead to different assumptions. The first is the temporal aspect of the financial series, where the time series itself defines the order of tasks and examples. Namely, the only way to learn from these series is respecting the temporal order, hence excluding data shuffling or any kind of offline learning. However, this assumption does not indicate a limitation in the study of the problem since the training of the tasks is sequential. Continual learning is defined for different scenarios (task, domain, and class-IL) and the next step was to identify in which of these the problem treated is found. Since all the tasks come from the same time series, the data distribution of each task will also be the same; therefore, we can discard the class-IL scenario. Considering the problem of predicting the market by classification, for each task, the model receives as input a sequence equal to one month of data updated every day at the close of the markets, for a sequence length equivalent to 20 days. The reason for this choice is dictated above all by the opening and closing of the world market. The second reason for choosing a window of this size is given by the assumption that in a month of daily observations it is possible to collect a sufficient amount of data to make a reliable analysis. There is also a label associated with each sequence. There will be $y = 1$ if the value of the time series 20 days after the end of the sequence will be greater than the last data of the same, zero otherwise. This corresponds to one month of observation and prediction of the following month. The labels predicted by the model indicate how to act: a positive prediction suggests an increase in market value, and therefore it is generally advisable to buy. In contrast, a negative prediction invites to sell. For each task, the possible outputs will belong to the same domain and consequently, we are faced with a domain-IL scenario. In this scenario, there is no information regarding a task identifier, leading to discard architectural or parameter isolation methods. Figure 1 shows the data flow in our setting. Dashed red lines indicate a change of regime (task in the continual learning problem) in the raw time series; the network takes as input a series of financial indicators, derived from the raw prices series, relative to a period, defined by a window on the timeline; the neural networks operate a binary classification defining if the series exhibits a positive or a negative trend.

3.3 Continual learning techniques

In this section, all the examined algorithms will be explored, describing the tricks we sometimes employed to adapt these methods to the problem. In fact, not all of the state-of-the-art methods are suitable for the problem of classifying financial time series or do not offer the desired performance.

Gradient Episodic Memory The early learning problems of multiple sequential tasks were based on Empirical Risk Minimization (ERM) [18], which defines the theoretical limits of the learning algorithm performance. This limitation comes from the inability to know the data distribution on which the algorithm will work. Gradient Episodic Memory (GEM) [11] was proposed as a learning method disconnected from data distribution and focused on an example by example observation. In particular, the classic example-label pair (x, y) is abandoned in favor of a triplet (x, y, t), where t is a task descriptor. Applied to financial time series, the task descriptor can be the task-id, as done in this work, or a complex structure describing the distribution to which the data belong. The main feature of this method is the episodic memory M_t capable of saving a subset of each task t. With T tasks and total available memory M, each task will have an exclusive memory equal to M/T. If the total number of tasks is not known a priori, it is possible to gradually reduce the number of examples for each task as the number of tasks increases. The goal of this method is to sequentially train a model on T tasks, preventing overwriting in future tasks with the constraint that training a task should not lead to worse performance in previous tasks. Given a triplet (x, y, t), the optimization problem to be solved is the following:

$$\begin{aligned} &\text{minimize}_\theta \quad \ell(f_\theta(x,t),y) \\ &\text{subject to} \quad \ell(f_\theta,\mathcal{M}_k) \leq \ell(f_\theta^{t-1},\mathcal{M}_k) \text{ for all } k<t, \end{aligned} \tag{1}$$

The problem is complex to solve with this formulation, but it is possible to make two observations. The first one concerns the conservation of the parameters of each task: if the constraint on the loss is maintained, it is no longer necessary to save the state of the network at the end of each task. The second and, more important, allow us to represent the variation of the loss between two updates through the angle between the two gradient vectors if the function is locally linear, assumption valid between two gradient steps. The second observation allows us to rewrite the optimization constraints:

$$\langle g, g_k \rangle := \left\langle \frac{\partial \ell(f_\theta(x,t),y)}{\partial \theta}, \frac{\partial \ell(f_\theta,\mathcal{M}_k)}{\partial \theta} \right\rangle \geq 0, \text{ for all } k<t. \tag{2}$$

For every training step, there is, therefore, a system of k inequalities to resolve. It is easy to guess that this operation becomes more onerous as the number of tasks increases. In case of at least one constraint is not met, a gradient step in a new direction is required. This makes the optimization problem a QP-complete problem for that specific training step. Only approximations of the optimal solution are valid, and the authors of the method proposed a valid one using the dual problem. Buffer size plays a relevant role in the performance evaluation: if we use more memory, we could expect better performances. But, in time series problems, this becomes tricky. A too big memory should allow saving too much data, going to a pseudo-parallel training of more tasks, and, in a scenario where temporal order is fundamental, this aspect must be avoided.

Averaged Gradient Episodic Memory Averaged Gradient Episodic Memory (A-GEM) [5] has been proposed as an optimization of the forerunner. To alleviate the weight of the computation, the authors opted for a relaxation of the constraints, going from loss reduction on examples of each of past tasks to an average on all the episodic memory. While the objective function to minimize remains the same, the constraints collapse to a single one valid for all past tasks.

As before, we can reformulate the objective function and the constraints regarding previous observation of the loss variation and gradient vectors:

$$\text{minimize}_{\tilde{g}} \quad \frac{1}{2}||g - \tilde{g}||_2^2 \quad \text{s.t.} \quad \tilde{g}^\top g_{ref} \geq 0 \tag{3}$$

where g_{ref} indicates gradient computed from a random batch obtained by the episodic memory from all previous tasks; in other words, A-GEM replaces $t-1$ inequalities with only one. However, it remains possible that the unique constraint is not met. In this scenario, there is no particular problem or approximation to compute but the solution is given by the projected gradient method:

$$\tilde{g} = g - \frac{g^\top g_{ref}}{g_{ref}^\top g_{ref}} g_{ref} \tag{4}$$

As GEM, this method exploits different algorithms to fill the buffer. Since in A-GEM, we got a single batch sampled from the whole buffer, the data distribution of the batch could not reflect the original distribution between tasks. We opted for a different strategy to fill the buffer to maintain the correct distribution, equal as far as possible to the stream one, using the reservoir sampling algorithm.

Synaptic Intelligence A significant limitation in the development of neural networks capable of learning multiple sequential tasks lies in the one-dimensional structure of the neuron, leading a network to catastrophic forgetting. Defining which neurons are most responsible for learning is necessary to consolidate acquired knowledge on a task. The best way to assess how significant a neuron is for a task is to calculate its contribution to the global loss of the current task. In this way, at the end of each task, it will be possible to determine which neurons contribute most to the current task's learning and prevent their update in the future, maintaining knowledge of past tasks, thus avoiding forgetting. Synaptic Intelligence (SI) [21] does not require external memories or architecture variation. Still, it acts only on network parameters, defining an additional loss related to the state of the neurons themselves. When training on a new task, changes to important parameters are penalized to prevent old memories from being overwritten. We can compute weight importance ω_k^μ for each neuron θ_k related to task μ. During the training of a task, a learning trajectory $\theta(t)$ is described in the network parameter space. This trajectory will come as close as possible to a minimum for the loss function for each task. We can now consider an update $\delta(t)$ at time t, leading to a variation on the loss of the current task. This variation can be approximated by the gradient g_k and in such case the relation

$$\ell(\theta(t) + \delta(t)) - \ell(\theta(t)) \approx \sum_k g_k(t) \cdot \delta_k(t) \tag{5}$$

can be considered valid. The variation $\delta_k(t) = \theta'_k(t) = \frac{\partial \theta_k}{\partial t}$ therefore contributes to the variation of the global loss. If we want to compute the variation over the entire trajectory, we must sum up all small updates. This amounts to computing the path integral of the gradient vector along the parameter trajectory. Since the gradient is a conservative field, the value of the integral is equal to the difference in loss between the end point t_{end} and start point t_{start}. In addition, the integral can be decomposed as the sum of the impact of the importance ω_k^μ on loss variation. In practice, ω_k^μ is the online approximation of the running sum of the product of the gradient $g_k(t) = \frac{\partial L}{\partial \theta_k}$ with the update θ'_k. In a sequential tasks scenario, the model will have only a loss $\mathcal{L}_\mu$ available on the current task μ. Catastrophic forgetting occurs when minimizing $\mathcal{L}_\mu$ there is a significant increase of the loss $\mathcal{L}_\upsilon$ of past tasks $\upsilon < \mu$. In this context, the importance of parameters θ_k is determined by: 1) how much the parameter contributes to a loss drop and 2) the difference $\theta_k(t^\mu) - \theta_k(t^{\mu-1})$. To avoid a significant variation in these parameters, a modified loss has been proposed:

$$\widetilde{\mathcal{L}_\mu} = \mathcal{L}_\mu + c \sum_k \Omega_k^\mu (\widetilde{\theta_k} - \theta_k)^2 \tag{6}$$

with $\widetilde{\theta_k} = \theta_k(t^{\mu-1})$ and parameter c to mange regularization. Coefficient Ω_k^μ determines the regularization strenght of each parameter:

$$\Omega_k^\mu = \sum_{\upsilon<\mu} \frac{\omega_k^\upsilon}{(\Delta_k^\upsilon)^2 + \epsilon} \tag{7}$$

with $\Delta_k^\upsilon = \theta_k(t^\upsilon) - \theta_k(t^{\upsilon-1})$.

Elastic Weight Consolidation Like the previous one, this method presented by Kirkpatrick *et al.* [9] is based on the possibility of determining a coefficient of importance for each neuron to be used in the computing of the global loss. Given task A, multiple valid network weights configurations θ_A ensure the same performances. In this way, when task B occurs, the model will maintain the performance binding model parameters in a solution space with low error for the previous task while maximizing performance on the new task B. Elastic Weight Consolidation (EWC) does not aim to find the optimal solution for each task but focuses on finding an intersection of low error solution space. To find which parameters are the most significant for a task, the authors addressed the problem from a probabilistic point of view. Optimizing the parameters of a network given the training set $\mathcal{D}$ is equal to probability $p(\theta|\mathcal{D})$. In presence of two independent tasks A($\mathcal{D}_\mathcal{A}$) and B($\mathcal{D}_\mathcal{B}$), with the Bayes' law we can write:

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}_B|\theta) + \log p(\theta|\mathcal{D}_A) - \log p(\mathcal{D}_B) \tag{8}$$

where the right-hand side depends only on the loss on task B $\log p(\mathcal{D}_B|\theta)$. Simultaneously, all the information regarding task A $\log p(\theta|\mathcal{D}_A)$ is soaked up by the posterior probability. Suppose the true posterior probability cannot be

computed. In that case, we can obtain a good approximation from a Gaussian distribution with mean given by parameters θ_A and angular precision from the diagonal of the Fisher Information Matrix F. This matrix has three key properties: 1) it is equivalent to the second derivative of the loss near a minimum, 2) it can be computed from first-order derivates, and 3) it is guaranteed to be positive semi-definite. Given this approximation, the loss function $\mathcal{L}$ to minimize:

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2 \tag{9}$$

where $\mathcal{L}_B(\theta)$ is the loss for task B, λ sets the importance of old task compared to the new one. When moving to a third task C, EWC will try to keep the model parameters close to the learned parameters on both task A and B, where this can be enforced either with two separate penalties.

Experience Replay The consolidation of acquired knowledge can occur in several ways in the human brain. One consists of periodic observation to consolidate the knowledge acquired but potentially overwritten over time. Experience Replay (ER) [16] uses an external memory buffer to save data from previous tasks, as example-label couple (x, y), without any reference about the task that the data belong to. During each training step, in addition to the batch of examples of the current task, another batch composed of examples from past tasks is sampled from the buffer, and the loss is computed on both of its, driven by hyperparameters α and β:

$$\mathcal{L} = \alpha \cdot \mathbb{E}_{(x,y)\sim D_t}[l(y, f(x))] + \beta \cdot \mathbb{E}_{(x,y)\sim M}[l(y, f(x))] \tag{10}$$

where $\mathcal{D}_t$ denotes training set of the current task and $\mathcal{M}$ the external memory.

Dark Experience Replay Several CL algorithms have been proposed as improvements of ER. Dark Experience Replay (DER) [3] is one of these, and it relies on *dark knwowledge* for distilling past experiences, sampled over the entire training trajectory. Differently from the other rehearsal-based methods, this method retains the network's logits $z \triangleq h_{\theta_t}(x)$, instead of the ground truth labels y. This stratagem allows avoiding the loss of information due to the compression made by the final activation function. The corresponding loss function results:

$$\mathbb{E}_{(x,y)\sim D_t}[l(y, f(x))] + \alpha E_{(x,z)\sim \mathcal{M}}\left[\, \|z - h_\theta(x)\|_2^2 \,\right] \tag{11}$$

This approach is related to *Knowledge Distillation* [8], a paradigm that allows the transfer of knowledge from a teacher to a student model. In particular, DER exploits a variant of this, known as self-distillation [7], in which transfer occurs between the same architecture. In this scenario, by saving logits of previous task examples, the model transfers knowledge to a version of itself in the future. Moreover, *Dark Experience Replay ++* has been proposed that equips equation 11 with an additional term on buffer datapoints, promoting higher conditional likelihood concerning their ground labels.

4 Experiments

4.1 Architecture

All the methods under examination were evaluated with the same architecture, using a fully-connected network composed of three hidden layers with respectively 100, 50, and 25 neurons each and with LeakyReLu as an activation function. All methods were tested using Stochastic Gradient Descent (SGD) with momentum as an optimizer for a total of 480000 training steps for each task. For rehearsal methods, we set the buffer size to 500 samples.

4.2 Evaluation metrics

To properly assess learning quality at training time, it is mandatory to consider both single tasks as the whole training process. In other words, a CL algorithm should be evaluated both on the *past* and the *present* tasks to reflect in its behavior on the *future* unseen tasks. It is crucial to assess the ability to transfer knowledge across tasks to achieve this, along with average accuracy (ACC). More specifically, we would like to measure *Forward Transfer* (FWT) and *Backward Transfer* (BWT) [11] (*Forgetting* (FRG) [4] has been omitted because it is equal to BT except for the sign). The first one assesses the influence that learning a task t has on the performance on a future task $k > t$, whereas the second and third ones measure the performance degradation in subsequent tasks. FWT is computed as the difference between the accuracy before starting training on a given task and the random-initialized network, then averaged across all tasks. FRG and BWT compute the difference between the current accuracy and its best value for each task, presumably at the end of the training of the task itself. Except for FRG, the larger these metrics, the better the model. If two models have similar ACC, the preferable one is the one with larger BWT and FWT. While BWT measures the influence of a task on the previous ones and FWT the influence on the following ones, it is meaningless to discuss backward for the first task or forward for the last one.

4.3 Dataset

For the experimental analysis, two datasets have been employed. We used Brent Oil dataset [2], the historical series of the oil prices on a daily basis. In particular, we used 9282 time steps, collected between 02/01/1986 and 31/07/2021 [3]. We also employed the copper dataset [2], consisting in 8500 time steps, taken from 02/01/1989 and 31/07/2021 [3]. An example contains twenty consecutive daily observations. The next example is obtained by shifting the time window by one timestep. To provide more refined information to the model, it was decided to proceed towards an engineering of the features using some of the most famous financial and statistical indicators, as explained in section 3.1. Finally, for the definition of the various tasks within the time series, Bayesian Online

[2] https://datahub.io/core/oil-prices, https://help.yahoo.com/kb/SLN2311.html
[3] Dates are reported as DD/MM/YY.

	Online	SI	EWC	ER	GEM	A-GEM	DER	DER++
Accuracy	65.04	70.34	71.64	72.29	65.25	65.47	72.59	**73.37**
Backward Transfer	-	-6.06	-4.15	**-3.72**	-11.29	-5.74	-4.96	-4.11
Forward Transfer	-	25.28	26.02	24.05	12.21	25.97	23.39	**27.24**

Table 1. Results of tested methods on Brent Oil dataset. For accuracy, backward and forward transfer bigger is better.

	Online	SI	EWC	ER	GEM	A-GEM	DER	DER++
Accuracy	58.28	65.46	68.01	68.00	54.23	58.86	64.09	**68.77**
Backward Transfer	-	-5.08	-3.28	**-0.85**	-8.94	-6.74	-4.22	-3.11
Forward Transfer	-	14.45	11.64	17.92	7.02	13.39	**14.32**	12.28

Table 2. Results of tested methods on copper dataset. For accuracy, backward and forward transfer bigger is better.

Changepoint Detection (BOCD) was used, an online algorithm for the detection of changepoints, i.e. moments in which a significant change occurs in the data distribution. To verify the validity of this technique, we asked a financial expert to manually find out the change-points on the time series. The results almost completely coincide with those found by the algorithm, with an occasional variation of maximum 1 or 2 time steps. This allows the whole continual learning process to work without the need for further human intervention to detect regime changes, allowing us to use public datasets without further processing. Within each task, we split the data into two different sets: train and evaluation set. Between the two sets, we leave a gap excluding any sample whose evaluation time is posterior to the earliest prediction time in the validation set. This ensures that predictions on the validation set are free of look-ahead bias.

4.4 Quantitative results

This section will discuss the results of each Continual Learning method, comparing them with the sequential training of each task without any continual technique, called *online learning*. Each method is evaluated not only with accuracy across all tasks but also with ad hoc Continual Learning metrics: *forward transfer* and *backward transfer*. In Table 1 and Table 2 are reported performances measured at the end of the whole training, respectively on the Brent oil dataset and on the copper dataset. To obtain less noisy performance estimates, values are reported as averages of three runs with different runs with different initialization. In the regularization methods, SI and EWC, we found an attenuation of the average forgetting across all tasks, even if the task nature heavily influences the accuracy of past tasks. This aspect is emphasized in SI, while EWC demonstrates good stability. Figure 2(a) shows the evolution of the accuracy of these methods during the training, compared to online learning on

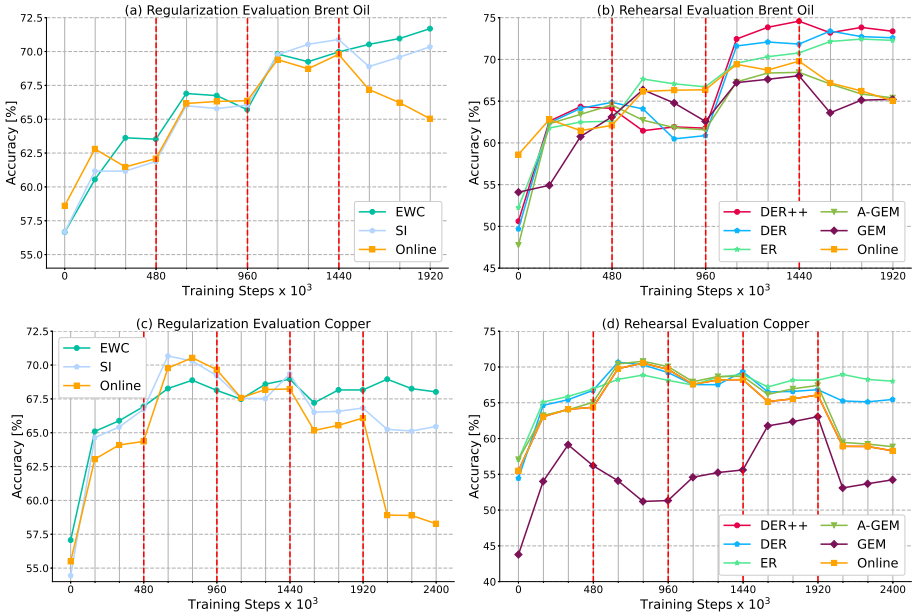

Fig. 2. Performance results. Dashed lines indicate task change: regularization methods performance on Brent Oil dataset (a), rehearsal method performance on Brent Oil dataset (b), regularization methods performance on the copper dataset (c), rehearsal method performance on the copper dataset (d).

Brent oil dataset, while Figure 2(c) on the copper dataset. SI and EWC experience similar accuracy on both datasets, while in the continual metrics, EWC performs slightly better. In replay-based methods (Figure 2(b) on the Bernt oil dataset and Figure 2(d) on the copper dataset), Gem and A-GEM are complex and hard to evaluate algorithms. GEM experiences the worst performance across all continual methods tested, especially on the copper dataset; probably due to the constraints violations introduced by the technique, event more and more frequently as the number of tasks increases. Although A-GEM significantly relaxes the constraints, accuracy remains aligned with baseline online learning while continual metrics are comparable to other methods. Vice versa, ER, DER and, DER++ experience the best performances. DER++ combines the features of the other two methods taking advantage of its: replaying past data from the buffer as ER, it uses logits information context as done in DER to improve further the predictions, bringing it to be the most performing method. In Figure 3(a) and (b) are reported the training times of each method, respectively on Brent oil dataset and on the copper dataset. Again, the behavior is similar on both datasets, demonstrating the robustness of these algorithms on financial time series. SI demonstrates to be the quickest to be trained thanks to online estimate models weights. Simultaneously, the other regularization method, EWC, is the second most time-consuming algorithm due to the necessity to compute

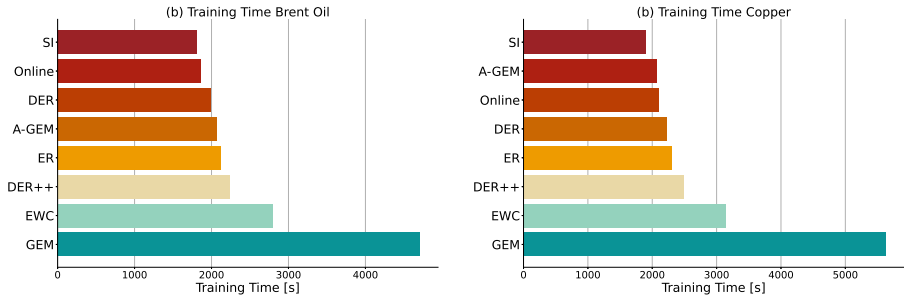

Fig. 3. Training time comparison on Brent Oil dataset (a) on the left and the copper dataset (b).

the Fisher Information Matrix at the end of each task. GEM constraints, finally, do not make up a complexity element only from a computational perspective, but they also make this method the most time-consuming for the model training.

5 Conclusion

In this paper, an experimental analysis of continual learning algorithms on market predictions has been conducted, highlighting their significant contribution in the field of artificial intelligence applied to finance. A deep investigation of the most promising state-of-the-art continual learning algorithms has been made, discovering that not all of them are suitable to financial time series. Furthermore, we found that other factors such as training time, computational complexity, and memory requirements can be decisive in defining the scenario and choosing the most appropriate algorithm to apply. The formulation adopted represents only an exemplifying model of more complex dynamics, but the development of this tool could be a concrete help for professional traders in the future. As future work, a specific continual learning algorithm for financial time series could be designed, taking into account the variety and complexity of the markets.

Acknowledgement

Work supported by FF4EuroHPC: HPC Innovation for European SMEs, Project Call 1. FF4EuroHPC has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951745.

References

1. Bertugli, A., Calderara, S., Coscia, P., Ballan, L., Cucchiara, R.: Ac-vrnn: Attentive conditional-vrnn for multi-future trajectory prediction. Computer Vision and Image Understanding **210** (2021)
2. Bertugli, A., Vincenzi, S., Calderara, S., Passerini, A.: Few-shot unsupervised continual learning through meta-examples. Neural Information Processing Systems Workshops (2020)

3. Buzzega, P., Boschini, M., Porrello, A., Abati, D., Calderara, S.: Dark experience for general continual learning: a strong, simple baseline. In: Neural Information Processing Systems (2020)
4. Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 532–547 (2018)
5. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with A-GEM. In: International Conference on Learning Representations (2019)
6. Dixon, M.F., Halperin, I., Bilokon, P.: Machine Learning in Finance. Springer, Cham, 1st edn. (2020)
7. Furlanello, T., Lipton, Z., Tschannen, M., Itti, L., Anandkumar, A.: Born again neural networks. In: International Conference on Machine Learning. pp. 1607–1616. PMLR (2018)
8. Hinton, G., Vinyals, O., Dean, J., et al.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 **2**(7) (2015)
9. Kirkpatrick, J.N., Pascanu, R., Rabinowitz, N.C., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. In Proceedings of the National Academy of Sciences of the United States of America **114 13**, 3521–3526 (2016)
10. Liu, X.Y., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., Wang, C.D.: Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance. In: Neural Information Processing Systems Workshops (2020)
11. Lopez-Paz, D., Ranzato, M.A.: Gradient episodic memory for continual learning. In: Advances in Neural Information Processing Systems (2017)
12. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of Learning and Motivation, vol. 24, pp. 109–165. Academic Press (1989)
13. Monti, A., Bertugli, A., Calderara, S., Cucchiara, R.: Dag-net: Double attentive graph neural network for trajectory forecasting. In: International Conference on Pattern Recognition (2020)
14. de Prado, M.L.: Advances in Financial Machine Learning. Wiley Publishing, 1st edn. (2018)
15. López de Prado, M.M.: Machine Learning for Asset Managers. Cambridge University Press (2020)
16. Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G.: Experience replay for continual learning. In: Neural Information Processing Systems (2019)
17. Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R., Hadsell, R.: Progress & compress: A scalable framework for continual learning. In: International Conference on Machine Learning (2018)
18. Vapnik, V.: Principles of risk minimization for learning theory. In: Advances in Neural Information Processing Systems (1991)
19. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (2017)
20. van de Ven, G.M., Tolias, A.S.: Three scenarios for continual learning. In: Neural Information Processing Systems Workshops (2018)
21. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International Conference on Machine Learning (2017)