Expert Systems With Applications 156 (2020) 113456

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—DeepBreath

Farzan Soleymani, Eric Paquet*

National Research Council, 1200 Montreal Road, Ottawa, ON K1K 2E1, Canada

ARTICLE INFO

Article history:
Received 3 October 2019
Revised 10 February 2020
Accepted 13 April 2020
Available online 18 April 2020

Keywords:
Portfolio management
Deep reinforcement learning
Restricted stacked autoencoder
Online leaning
Settlement risk
Blockchain

ABSTRACT

The process of continuously reallocating funds into financial assets, aiming to increase the expected return of investment and minimizing the risk, is known as portfolio management. In this paper, a portfolio management framework is developed based on a deep reinforcement learning framework called DeepBreath. The DeepBreath methodology combines a restricted stacked autoencoder and a convolutional neural network (CNN) into an integrated framework. The restricted stacked autoencoder is employed in order to conduct dimensionality reduction and features selection, thus ensuring that only the most informative abstract features are retained. The CNN is used to learn and enforce the investment policy which consists of reallocating the various assets in order to increase the expected return on investment. The framework consists of both offline and online learning strategies: the former is required to train the CNN while the latter handles concept drifts i.e. a change in the data distribution resulting from unforeseen circumstances. These are based on passive concept drift detection and online stochastic batching. Settlement risk may occur as a result of a delay in between the acquisition of an asset and its payment failing to deliver the terms of a contract. In order to tackle this challenging issue, a blockchain is employed. Finally, the performance of the DeepBreath framework is tested with four test sets over three distinct investment periods. The results show that the return of investment achieved by our approach outperforms current expert investment strategies while minimizing the market risk.

1. Introduction

A collection of financial assets, known as a portfolio, may consist of financial instruments such as stocks, bonds, currencies, and cryptocurrencies. The process of reallocating the various assets forming the portfolio, in order to increase the expected return on investment, is called portfolio management. The financial instruments managed in such a portfolio may be assimilated to time series, which often exhibit complex behaviors, due to external factors such as the global economy and the political climate. As a result, their behaviors are inherently nonlinear, uncertain and nonstationary (Malkiel, 2003; Tsai & Hsiao, 2010).

Generally, the erratic and complex behavior of the financial market is determined by endogenous and exogenous factors (Filimonov & Sornette, 2012). In recent years, due to the constant rise of social media, such as Twitter^{TM}and Facebook^{TM}, it has been implicitly assumed that market evolution essentially results from these exogenous factors. This assumption finds its origin in the efficient market hypothesis (Fama, 1991; Malkiel & Fama, 1970) in which the price of an asset fully reflects all available information. Nevertheless, various studies have demonstrated the falsehood of this hypothesis showing that these exogenous factors have a limited impact on markets where behavior is essentially driven by endogenous factors (Bouchaud, 2011; Cutler, Poterba, & Summers, 1988; Joulin, Lefevre, Grunberg, & Bouchaud, 2008). The extent to which the market is endogenous is determined by the spectral radius associated with the underlying state-dependent Hawkes process (Filimonov & Sornette, 2012).

When managing a portfolio, one must take into account the volatility and the correlation associated with the various financial instruments. Portfolio management is an investment strategy that aims at maximizing the expected return on investment (ROI) while minimizing the financial risk by continuously reallocating the portfolio assets i.e. by attributing the proper weight to each financial instrument. With recent advances in machine learning and deep learning, it has become possible to predict complex financial be-

* Corresponding author.
E-mail addresses: Farzan.Soleymani@nrc-cnrc.gc.ca (F. Soleymani), Eric.Paquet@nrc-cnrc.gc.ca (E. Paquet).

https://doi.org/10.1016/j.eswa.2020.113456

haviors and to automatize the decision process; at least up to a certain extent (Jangmin, Lee, Lee, & Zhang, 2006).

Stock market behavior prediction with an artificial neural network (ANN) was proposed for the first time by Atsalakis and Valavanis (2009). Artificial neural networks are data-driven machine learning techniques which do not require a complete prior model of the data generation mechanism. Nonetheless, traditional multilayer neural networks do not take advantage of the time series structure associated with financial data (Eakins & Stansell, 2003; Hussain, Knowles, Lisboa, & El-Deredy, 2008; Lam, 2004).

Moreover, the performance of an ANN is strongly dependent on the network architecture, the learning, and hyper-parameters, as well as the training, set just to mention a few. Therefore, other alternatives have been explored in order to improve the accuracy such as support vector machine (SVM), random forest (RF) and naïve Bayes with seminal work by Patel, Shah, Thakkar, and Kotecha (2015) with non-significant improvement. A major breakthrough came with the introduction of deep reinforcement learning (Jiang, Xu, & Liang, 2017) in which the policy, which determines the actions taken by the agent, is either learned with a convolutional neural network (CNN) or with some kind of recurrent neural network (RNN) such as the long-short-term memory network (LSTM). In the portfolio management context, the actions are the weights attributed to the various assets forming the portfolio.

Financial data are particularly suited for deep learning because of the very large amount of data available for training (Goodfellow, Bengio, & Courville, 2016). Nevertheless, deep learning algorithms performances are affected by multiple factors such as the activation and pooling functions, noise, network structure, which may consist of multiple sub-networks as well as the features selection mechanism. In the financial context, noise reduction has been achieved with wavelet transforms (Bao, Yue, & Rao, 2017) while more informative representations and dimensionality reduction were obtained with Boltzmann machines and autoencoders (Hinton & Salakhutdinov, 2006) among other techniques. For large portfolios, dimensionality reduction is desirable in order to optimize the learning process while avoiding the curse of dimensionality (Sorzano, Vargas, & Montano, 2014). Various types of encoders have been employed such as convolutional autoencoders (Cheng, Sun, Takeuchi, & Katto, 2018) and stacked Autoencoders (Bao et al., 2017) just to mention a few.

In this paper, we propose an online framework for portfolio management based on deep reinforcement learning and restricted stacked autoencoder for feature selection. High-level features are extracted from the input data which contain eleven (11) correlated features. The restricted stacked autoencoder extracts new informative features while reducing the dimensions down to three (3). Therefore, the training may be completed very rapidly making this approach suitable for online learning and online portfolio management. The investment policy is enforced using a CNN. The algorithm is initially trained offline which results in a pre-trained model from historical data. Then, the weights are updated following an online learning scheme in order to follow the evolution of the market. Both offline and online strategies are employed for learning: the former being based on passive concept drift detection (Gama, Indrė, Bifet, Pechenizkiy, & Bouchachia, 2014) and the latter being based on online stochastic batching. Using passive concept drift detection, which allows the system to learn new emerging patterns in the market while preserving the knowledge acquired during offline training with historical data. In order to facilitate the audit and to secure the settlement process, a blockchain is employed.

The paper is organized as follows. The underlying mathematical formalism associated with portfolio management is introduced in Section 2. Features normalization and selection are addressed in Section 3. Our deep reinforcement learning framework, named DeepBreath, is presented in Section 4. This framework integrates passive concept drift detection and online stochastic batching. This is followed, in Section 5, by a description of the blockchain employed for settlement. Our experimental results are presented in Section 6. A conclusion follows in Section 7.

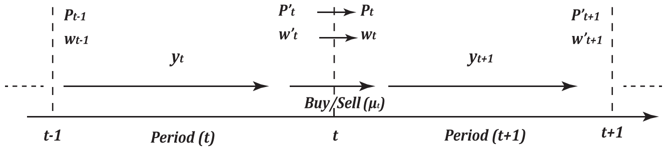

Fig. 1. Weight reallocation process. The portfolio weight vector $\mathbf{w}'_{t-1}$ and the portfolio value P'_{t-1} at the beginning of period t evolve to (t) and P'_t respectively. At that moment, assets are sold and bought in order to increase the expected return on investment. These operations involve commission fees which shrink the portfolio value to P_t which result in new weights $\boldsymbol{w}_t$.

2. Mathematical model for portfolio management

The trading period for financial instruments on the stock exchange is one calendar day. During this period, the price fluctuations associated with a financial instrument may be characterized by four metrics namely the price at the opening of the market, the lowest and the highest prices reached during the day and, the price at the closure of the market. Based on these characteristics, this section provides a mathematical framework for portfolio management. An approach similar to (Jiang et al., 2017), which was introduced earlier by Ormos and Urbán (2013), is followed.

The price vector $\mathbf{V}_t$ consists of the closing price of all assets forming the portfolio at the time in addition to the amount of cash available for trading (here in USD). The normalized price vector $\mathbf{Y}_t$ is defined as the element-wide division in between the price vector at the time t and $t-1$:

$$\mathbf{Y}_t := \mathbf{V}_t \oslash \mathbf{V}_{t-1} = \left[1, \frac{\mathbf{V}_{1,t}}{\mathbf{V}_{1,t-1}}, \frac{\mathbf{V}_{2,t}}{\mathbf{V}_{2,t-1}}, \ldots, \frac{\mathbf{V}_{m,t}}{\mathbf{V}_{m,t-1}}\right]^T. \tag{1}$$

Initially, the portfolio consists solely of cash which means that the weight associated with the cash is one (1), while the weights associated with the other financial instruments are zero (0). This means that no prior assumption is made about the portfolio composition: the weight allocation is entirely performed by our deep reinforcement learning algorithm. Therefore, no other prior assumption about the portfolio composition is made:

$$\mathbf{w}_0 = [1, 0, \ldots, 0]^T. \tag{2}$$

In order to buy or to sell an asset, one must pay a commission fee. As illustrated in Fig. 1, owing to prices fluctuations, at the end of period t, the value of the portfolio vector is determined by

$$\mathbf{w}'_t = \frac{\mathbf{Y}_t \odot \mathbf{w}_{t-1}}{\mathbf{Y}_t \cdot \mathbf{w}_{t-1}}, \tag{3}$$

where $\mathbf{w}_{t-1}$ is the portfolio weight vector at the beginning of period t. At that moment, the portfolio is reallocated by buying and selling assets in order to increase the expected return on investment; therefore, the portfolio weight vector evolves from $\mathbf{w}'_t$ to $\mathbf{w}_t$. The portfolio reallocation involves commission fees which shrink the portfolio value by a factor of $\mu \in (0, 1]$,

$$P_t = \mu_t P'_t. \tag{4}$$

The portfolio value at the end of period t is given by

$$P_t = \mu_t P_{t-1} \cdot \mathbf{Y}_t \cdot \mathbf{w}_{t-1}. \tag{5}$$

The return rate for period t and the logarithmic return rate are defined as

$$\rho_t := \frac{P_t}{P_{t-1}} - 1 = \frac{\mu_t \cdot P'_t}{P_{t-1}} - 1 = \mu_t \mathbf{Y}_t \cdot \mathbf{w}_{t-1} - 1, \tag{6a}$$

Table 1
Financial indicators employed as features.

Financial indicators	Definition
Average True Range	The average true range is an indicator that measures market volatility by decomposing the entire range of an asset price for that period.
Commodity Channel Index	The commodity channel index is a momentum-based oscillator used to help determine when an investment vehicle is reaching a condition of being overbought or oversold
Commodity Selection Index	The commodity selection index is a momentum indicator that attempts to identify which commodities are the most suitable for short-term trading.
Demand Index	The demand index is an indicator that uses price and volume to assess buying and selling pressure affecting a security.
Dynamic Momentum Index	The dynamic momentum index is a technical indicator used to determine if an asset is overbought or oversold.
Exponential Moving Average	An exponential moving average is a type of moving average that places a greater weight and significance of the most recent data points.
Hull Moving Average	The hull moving average solves the age old dilemma of making a moving average more responsive to current price activity whilst maintaining curve smoothness.
Momentum	The momentum is the rate of acceleration of a security's price or volume – that is, the speed at which the price is changing.

$$R_t := \ln \frac{P_t}{P_{t-1}} = \ln(\mu_t \mathbf{Y}_t \mathbf{w}_{t-1}) - 1. \tag{6b}$$

Therefore, the final value of the portfolio for a time horizon t_f is given by

$$P_f = P_0 \cdot \exp\left(\sum_{t=1}^{t_f+1} R_t\right) = P_0 \prod_{t=1}^{t_f+1} \mu_t \mathbf{Y}_t \mathbf{w}_{t-1}. \tag{7}$$

Consequently, the total amount of cash gained C_g by selling assets is

$$C_g = (1 - c_s) P'_t \sum_{i=1}^{m} \cdot ReLU\left(\mathbf{w}'_{t,i} - \mu_t \mathbf{w}_{t,i}\right), \tag{8}$$

where the selling commission rate is denoted by $0 < C_s < 1$. As a result, the cash reserve $P'_t \mathbf{w}'_{t,0}$ becomes $\mu_t P'_t \mathbf{w}_{t,0}$ which corresponds to the new amount of cash available for acquiring new assets. On the other hand, the amount of cash spent C_s is

$$C_s = (1 - c_b)\left[\mathbf{w}'_{t,0} + (1 - c_s) \sum_{i=1}^{m} ReLU(\mathbf{w}'_{t,i} - \mu_t \mathbf{w}_{t,i}) - \mu_t \mathbf{w}_{t,0}\right]$$
$$= \sum_{i=1}^{m} ReLU(\mu_t \mathbf{w}_{t,i} - \mathbf{w}'_{t,i}). \tag{9}$$

while the buying commission rate is given by $0 < C_b < 1$. With the help of $w'_{t,0} + \sum_{i=1}^{m} \mathbf{w}'_{t,1} = 1 = \mathbf{w}_{t,0} + \sum_{i=1}^{m} \mathbf{w}_{t,i}$ and $ReLU(x - y) - ReLU(y - x) = x - y$ the Eq. (9) yields:

$$\mu_t = \frac{1}{1 - c_b \mathbf{w}_{t,0}}$$
$$\times \left[1 - c_b \mathbf{w}'_{t,0} - (c_s + c_b - c_s c_b) \sum_{i=1}^{m} ReLU(\mathbf{w}'_{t,i} - \mu_t \mathbf{w}_{t,i})\right]. \tag{10}$$

The transaction factor μ_t is a function of the relative price $\mathbf{Y}$ as well as of the current and previous weight vectors:

$$\mu_t = \mu_t(\mathbf{w}_{t-1}, \mathbf{w}_t, \mathbf{Y}_t). \tag{11}$$

This implicit equation cannot be solved analytically. However, an approximation for the shrinking factor may be obtained through an iterative process. It has been demonstrated (Jiang et al., 2017) that the sequence generated by Eq. (10) always converge to the right solution.

Theorem 1. *Given an initial estimate μ_0 , the sequence $\left\{\tilde{\mu}_t^{(k)}\right\}$ defined by*

$$\left\{\tilde{\mu}_t^{(k)} \middle| \tilde{\mu}_t^{(0)} = \mu_0 \text{ and } \tilde{\mu}_t^{(k)} = f\left(\tilde{\mu}_t^{(k-1)}\right), k \in \mathbb{N}\right\}, \tag{12}$$

$$f(\mu) := \frac{1}{1 - c_b \mathbf{w}_{t,0}}$$
$$\times \left[1 - c_b \mathbf{w}'_{t,0} - (c_s + c_b - c_s c_b) \sum_{i=1}^{m} ReLU(\mathbf{w}'_{t,i} - \mu_t \mathbf{w}_{t,i})\right], \tag{13}$$

always converge to the solution of Eq. (10) for any $\mu_0 \in [0, 1]$.

The convergence rate depends on the initial value of the transaction factor μ_t. According to (Moody, Wu, Liao, & Saffell, 1998), a fast convergence rate may be obtained if the initial transaction factor μ_0 is chosen as

$$\mu_0 = c \sum_{i=1}^{m} |\mathbf{w}'_{t,i} - \mathbf{w}_{t,i}|. \tag{14}$$

The transaction factor μ_t is a function of the relative price $\mathbf{Y}_t$ as well as the current and previous weight vectors: In the present work, the following assumptions are made:

- it is always possible to buy or to sell an asset at any time.
- the transactions do not affect the values of the financial instruments involved. In other words, the number of assets traded by the agent is small compared to the overall market liquidity.

In order to optimize the learning process, data normalization and feature selection are addressed in the next section.

3. Features normalization and selection

As mentioned earlier, each financial instrument is characterized by four prices: the opening price, the lowest and the highest prices reached during the trading period as well as the price at the closing of the markets. In addition, financial instruments may be described by a plethora of financial indicators such as the average true range, the exponential moving average, the commodity channel index among others (Murphy, 1999): each index characterizing a specific behavior of the underlying asset. The financial indicators employed in our framework are described in Table 1; they are among the most commonly employed in the financial industry (Achelis, 2001).

Features normalization is addressed in the next subsection.

3.1. Features normalization

The portfolio is characterized by twelve (12) feature vectors. Each feature vector refers to a particular feature for all the assets forming the portfolio. The first four vectors contain the opening, low, high and closing price of all the assets respectively. The last eight vectors contain the financial indicators for all the assets as described in Table 1. In order for the agent to efficiently learn the policy, these features must be normalized (Ross, Mineiro, & Langford, 2013). Because both the extrema and the effective range are unknown a priori, it is not possible to normalize them with the min-max approach. Therefore, the low, high and closing prices are normalized with respect to the opening price:

$$\mathbf{V}_t^{(Lo)} = \left[\frac{Lo(t-n-1)}{Op(t-n-1)}, \ldots, \frac{Lo(t-1)}{Op(t-1)}\right]^T,$$
$$\mathbf{V}_t^{(Cl)} = \left[\frac{Cl(t-n-1)}{Op(t-n-1)}, \ldots, \frac{Cl(t-1)}{Op(t-1)}\right]^T,$$
$$\mathbf{V}_t^{(Hi)} = \left[\frac{Hi(t-n-1)}{Op(t-n-1)}, \ldots, \frac{Hi(t-1)}{Op(t-1)}\right]^T. \tag{15}$$

Such normalization is more informative than the min-max normalization scheme (Hussain et al., 2008) or the Z-score as it clearly identifies the market trend (up and down). A similar approach is followed for the financial indicators which are normalized with respect to their previous trading period:

$$\mathbf{V}_t^{(FI)} = \left[\frac{FI(t-n)}{FI(t-n-1)}, \ldots, \frac{FI(t)}{FI(t-1)}\right]^T. \tag{16}$$

As a result, the feature tensor consists of three (3) price vectors as defined in Eq. (15) as well as of eight(8) financial indicator vectors as described by Eq. (16).

Policies are more difficult to learn when the feature tensors have a large number of dimensions which in turn has a detrimental effect on the system performance (Choung, Lee, & Jeong, 2017). Such a situation may occur when the portfolio consists of a large number of assets; a commonplace situation in real life. In addition, some of these features may be highly correlated which may result in sub-optimal learning; not to mention that the training process may become computationally prohibitive. In order to alleviate these problems, a smaller number of non-correlated and more informative features must be extracted from the original ones. The next section describes our features selection approach which is based on an unsupervised stacked restricted autoencoder.

3.2. Features selection

Features selection is performed with an unsupervised stacked restricted autoencoder inspired by Wong and Luo (2018). The input feature vector consists of eleven (11) features namely the normalized low, high and closing prices as well as the eight normalized financial indicators; all taken at the same time *t*. The network is designed in order to extract low-dimensional uncorrelated features while optimizing the training process. Autoencoders are a type of unsupervised feedforward neural network that reconstructs the output from the input. It is formed of two parts namely the encoder and the decoder. The encoder reduces the dimensionality of the input data in the so-called latent of hidden-layer while the decoder reconstructs the input data from the hidden layer; such a network is said to be under-complete because of the restriction imposed by the latent layer (Bengio, Goodfellow, & Courville, 2015). Autoencoders are capable of learning nonlinear dimension reduction functions, removing redundancies and correlations while extracting highly informative features (Bengio et al., 2015).

As mentioned earlier, we employed a stacked restricted autoencoder. This network consists of seven (7) layers: one input layer, five (5) hidden layers and one output layer. The network structure is illustrated in Fig. 2. The network is asymmetrical: the first hidden layer has (10) neurons, the second (9) neurons while the next three consist of (3) neurons each. The feature extraction is performed at the level of the third hidden layer. The output layer also has only (3) neurons. The network is said to be restricted because the decoder performs a partial reconstruction of the input vector. Indeed, the three neurons of the output vector refer to the normalized low, high and closing prices. Therefore, the financial indicators are not reconstructed by the decoder. Consequently, our encoder is not only under-complete but also constrained: the constraint being the partial reconstruction at the level of the decoder. As later

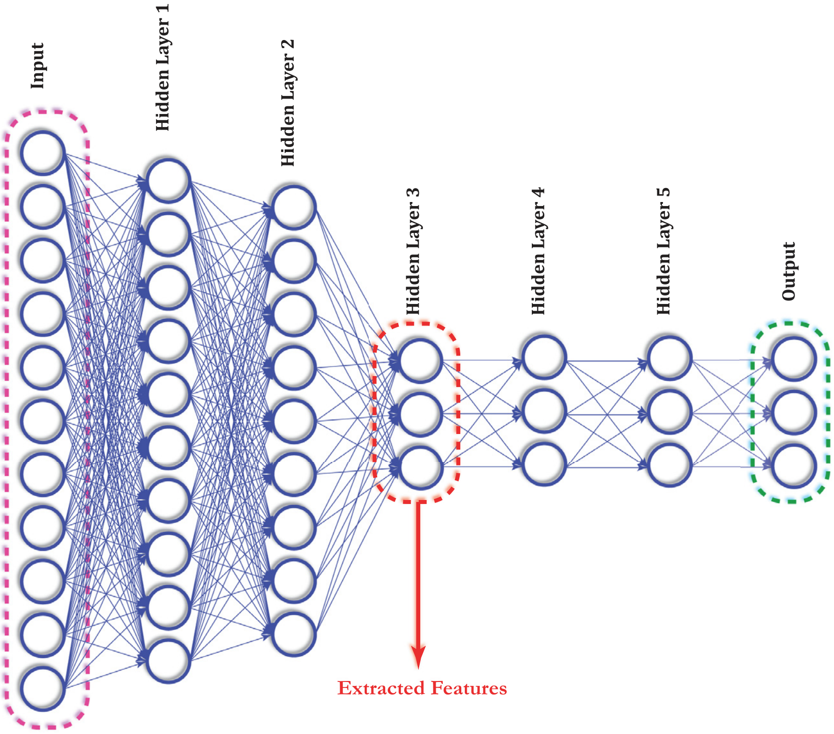

Fig. 2. Structure of the restricted stacked autoencoder.

demonstrated by the experimental results, the partial reconstruction constraint generates more informative feature vectors while putting the emphasis on what was foreseen as the most important features.

Our deep reinforcement learning algorithm is described in the next section.

4. Deep reinforcement learning

A deep reinforcement learning algorithm (Ivanov & D'yakonov, 2019) consists of an agent that takes the best actions in the environment given the state of the latter in order to maximize a reward r. Each action taken by the agent is associated with a reward. The best action is determined by a policy which is learned with deep learning techniques. The agent seeks to maximize the reward by taking proper actions. In our framework the action consists in determining the optimal weight vector for the portfolio in order to increase the expected return on investment:

$$\mathbf{a}_t = \mathbf{w}_t. \tag{17}$$

Due to the relation in between the return rate, Eq. (6a), and the transaction factor, Eq. (11), the current action is partially determined by the previous one. Therefore, the state vector consists of the feature tensor at time t as well as the weight vector at time $t-1$ (the state of the portfolio at this time):

$$\mathbf{s}_t = [\mathbf{X}_t, \mathbf{w}_{t-1}]^T. \tag{18}$$

The portfolio weight vector $\mathbf{w}_{t-1}$ and the features tensor $\mathbf{X}_t$ correspond to the internal (or latent) and external states respectively. As previously mentioned, the primary objective is to increase the portfolio return on investment which is determined by Eq. (7) over an investment horizon $t_f + 1$. It follows that the reward is associated with the logarithmic accumulated return $\mathcal{R}$, which may be inferred from Eqs. (6b), (7) and (11)):

$$\begin{aligned}\mathcal{R}(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) &= \frac{1}{t_f} \cdot \ln\left(\frac{P_f}{P_0}\right), \\ = \frac{1}{t_f} \cdot \sum_{t=1}^{t_f+1} \ln(\mu_t \cdot \mathbf{Y}_t \cdot \mathbf{w}_{t-1}) &= \frac{1}{t_f} \cdot \sum_{t=1}^{t_f+1} R_t.\end{aligned} \tag{19}$$

The algorithm employed to learn the policy is described in the next subsection.

4.1. SARSA

Portfolio management is considered a decision making process in which the agent constantly takes actions in order to increase the portfolio value. The SARSA algorithm (Sutton & Barto, 2018) is employed to train a CNN in order to learn the investment policy. This network is described in the next subsection. SARSA is an on-policy which means that the previous actions taken by the agent are taken into account in the learning process. This is in contrast with the Q-learning algorithm in which the optimal policy is learned irrespectively of the actions taken by the agent. Since on-policy methods are simpler and converge faster than their off-policy counterpart, they are more adapted to portfolio management (Sutton & Barto, 2018).

Algorithm 1 *SARSA* (ε-greedy).

```
Data: $\alpha \in [0, 1]$ Learning rate
Data: $\varepsilon \in [0, 1]$ epsilon value
Data: $\beta_d = 1 \times 10^{-2}$ epsilon decay rate
Data: $N_t, B_n, r$ : Number of iterations, Batch Number, Immediate
      Reward
Data: $\pi_\theta, Q$ : Policy, state-action value
Data: $\tilde{\mathbf{w}}_t$ : Selected action at (t+1)
Data: $\mathcal{A}, \mathcal{S}$ : Action space, State Space
Initialization:
$iteration = 0$
initial $\varepsilon \leftarrow 0.6 \leq \varepsilon \leq 0.85$
while $iteration \leq N_t$ do
    if $random\_number < \varepsilon$ then
        $\tilde{\mathbf{w}}_t \leftarrow random\_action \in \mathcal{A}$
    else
        $\tilde{\mathbf{w}}_t \leftarrow \pi_\theta(\mathbf{X}_{t+1}, \mathbf{w}_t)$
    end
    $\delta \leftarrow r_{t+1} + \gamma \cdot Q(\mathbf{X}_{t+1}, \mathbf{w}_t, \tilde{\mathbf{w}}_t)$
    for $(\mathbf{X}_{t+1}, \mathbf{w}_t, \tilde{\mathbf{w}}_t) \in \mathcal{S} \times \mathcal{A}$ do
        $Q(\mathbf{X}_t, \mathbf{w}_{t-1}, \mathbf{w}_t) \leftarrow Q(\mathbf{X}_t, \mathbf{w}_{t-1}, \mathbf{w}_t) + \alpha \cdot (\delta -$
        $Q(\mathbf{X}_t, \mathbf{w}_{t-1}, \mathbf{w}_t))$.
    end
    Update:
    $[\mathbf{X}_{t+1}, \mathbf{w}_t]^T \leftarrow [\mathbf{X}_t, \mathbf{w}_{t-1}]^T$
    $\mathbf{w}_{t+1} \leftarrow \tilde{\mathbf{w}}_t$
    Update:
    $\varepsilon \leftarrow \varepsilon_{min} + (\varepsilon_{max} - \varepsilon_{min}) \cdot \exp(-\beta_d \cdot B_n)$
end
Return: $Q$
```

Our implementation of SARSA is described in Algorithm 1. The normalized features tensor is fed to the staked restricted autoencoder for features selection and dimensionality reduction. These new features are employed to train the convolutional neural network i.e. to determine the weights associated with the kernel. The state tensor and the action vector are initialized based on the input tensor as well as on the previous and current weight vectors. The initial weight vector is given by Eq. (2) and consists only of cash which means that the CNN is entirely responsible for the reallocation of the assets. The discount rate γ determines the trade-off in between immediate (or myopic reward) and long term strategy. The Q value corresponds to the average logarithmic accumulated return. The Q value is defined as

$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha \cdot (r_{t+1} + \gamma \cdot Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - Q(\mathbf{s}_t, \mathbf{a}_t)), \tag{20}$$

where r_{t+1} is the immediate reward. Therefore, following a transition from state $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$, the Q value must be updated according to Eq. (20). In other words, the Q value is continuously estimated based on the current policy π_θ while the latter is repeatedly optimized in order to increase the Q value. The action and the state are given by Eqs. (17) and (18) respectively. Each action is taken according to an ε-greedy strategy: a random number is generated from a uniform distribution: if this number is smaller than a given threshold, a random action is taken; if not, the action is determined by the current policy. This strategy improves the exploration of the action space which results in turn in an optimal solution while avoiding over-fitting and addiction (Garcia & Fernández, 2012). As such, this approach is reminiscent of simulated annealing (Van Laarhoven & Aarts, 1987). The ε value is initially chosen close to one and is progressively reduced toward zero as the policy is learned. Thereafter, the policy is updated according to Eq. (20) which is essentially a gradient descent algorithm (Sutton, Barto et al., 1998).

The implementation of the gradient descent algorithm is described in the next section.

4.2. Online stochastic batching

The gradient descent algorithm is implemented with an online stochastic batching approach in order to obtain an unbiased and

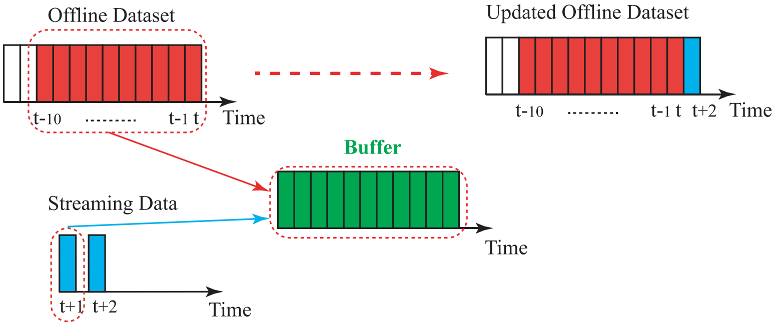

Fig. 3. Online learning batches generation.

optimal estimate of the gradient (Jiang et al., 2017). Financial data are sequential time series. Therefore, training batches must follow the same time ordering as the original data. In addition, different starting points generate completely distinctive batches. As a result, distinct mini-batches may be obtained by simply changing their starting point. For instance, $[t_B, t_B + B_s)$ and $[t_B + 1, t_B + B_s + 1)$ correspond to two distinct batches. In this paper, the online stochastic method, as proposed by Jiang et al. (2017), is employed in order to generate mini-batches. In this approach, it is assumed that the starting points are distributed according to a geometrical distribution:

$$P_\beta(t_B) = \beta(1-\beta)^{t-T_B-B_s}, \tag{21}$$

where the decaying rate is given by $\beta \in (0, 1)$ and where the relation in between the starting point and the mini-batch size is given by B_s. This distribution tends to favor more recent events; the extent of which is controlled by the decaying rate. In order to train the CNN, both offline learning and online learning are utilized.

Our learning strategy is described in the next subsection.

4.3. Offline and online learning

The CNN learns the policy offline which means that the policy is learned from historical data. Historical data is any data anterior to the current calendar time. The offline learning technique is based on online stochastic batching which has been described in the previous subsection. Nevertheless, the underlying distribution associated with a financial time series may vary over time. This phenomenon is called concept drift (Gama et al., 2014). In the case of financial instruments, concept drift is usually triggered by unforeseen exogenous factors such as political turmoil, unexpected loss or panic reaction (Cavalcante & Oliveira, 2015). These factors may affect the efficacy of the actions taken by the agent. Essentially, concept drift may be handled in two different ways which are known as the active and the passive approach (Rodríguez & Kuncheva, 2008). In the first case, a concept drift detector attempts to detect the occurrence of a concept drift. This is usually achieved by determining if there is a noteworthy change in the underlying data distribution (Elwell & Polikar, 2009). Once a concept drift is detected, the agent is trained afresh from the subsequent events (Gama, Medas, Castillo, & Rodrigues, 2004). In the second approach, the agent is initially trained from historical data and then, continuously updated as new data become available.

In our framework, we employ the passive approach which bestows us with feature drift adaptation (Gama et al., 2014). This is mainly motivated by two reasons: firstly, in trading, behaviors or events that occurred in the past are likely to repeat themselves in the future (Hu et al., 2015). By contrast, the active approach does not take advantage of this prior knowledge that exists in a latent in the historical data. Secondly, if the market is unstable, concept drift may occur repeatedly, and the agent might tend to employ short-term investment strategies which may have a detrimental effect on long-term investments (Hu et al., 2015). Nonetheless, passive concept drift detection tends to forget patterns that do not repeat themselves after a certain amount of time (Gama et al., 2004). These so-called anomalies have a relatively small impact on long-term investment strategies.

The online learning is performed for both the restricted stacked autoencoder and the CNN. The batches for online learning are generated as outlined in Fig. 3. A buffer is designed to store the last ten (10) days of offline data along with one time-step (one business day) of streaming data. At the end of each day, the offline dataset is expended by adding the current data point to the offline dataset.

In the next subsection, the convolutional neural network responsible for learning the policy is described

4.4. Convolutional neural network

In our framework, the policy is enforced with a convolutional neural network which is illustrated in Fig. 4. Indeed, convolutional neural networks have demonstrated their ability to learn and implement efficiently complex policies in a deep reinforcement learning framework (Jiang et al., 2017). The policy is initially learned offline from historical data with SARSA in conjunction with on-

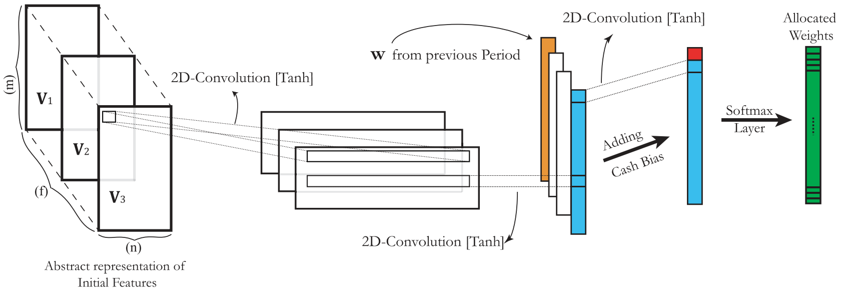

Fig. 4. Structure of the CNN employed to learn the policy.

line stochastic batching. Once the offline training is completed, the policy is continuously updated online as new information becomes available.

The architecture of our CNN is described in Algorithm 2. The CNN is fed with the features learned by the restricted stacked autoencoder. These features form a tensor which consists of the three (3) channels. Each channel forms a matrix which is associated with a particular abstract feature. Each matrix contains the values of the abstract feature for the assets (rows) forming the portfolio taken for the last n days (columns). Then, convolution is performed on each channel with a single kernel of size 1×3 followed by a *Tanh* activation function. Once more, a convolution with *Tanh* activation is performed on each of the three channels obtained from the previous convolution with a single kernel of size $1 \times n$ in order to obtain three vectors of size m. Thereafter, a fourth layer channel is added which consists of the weights (action) obtained from the previous iteration. A third and last convolution with activation is performed with a 1×1 kernel (the depth is equal to the number of channels) in order to obtain a single vector of size m. The last two steps constitute an implementation of Eq. (18) which states that the current action is determined by the current state as well as the previous action (weights). Finally, the previous vector is normalized with a function in order to obtain the portfolio new weights i.e. the current action.

Algorithm 2 Implementation of convolutional neural network.

```
Data: $f, m, n$ : Number of Features, Number of Assets, Trading Window Length
Data: $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3$ : Extracted Features
Initialization:
Input: $\mathbf{X}_t = [\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3]^T$
while $i \leq n$ do
    1. First Convolution:
       Activation Function: Tanh
       Kernel size → (1,3)
       Kernel depth → $f$
    2. Second Convolution:
       Activation Function: Tanh
       Kernel size → (1,n)
       Kernel depth → $f$
    3. Third Convolution:
       Concatenate weights for previous and current time steps (Weight Matrix).
       $\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_i]^T$
       Activation Function: Tanh
       Kernel size → (1,1)
       Kernel depth → $f + 1$
    4. Adding Cash:
       Concatenate Cash Bias and computed Weight Matrix
    5. Softmax Layer:
       Return: Distributed Portfolio Weights ← $\mathbf{W}_{ptf} = [\mathbf{w}_C, \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]^T$
end
```

The settlement process is addressed in the next section.

5. Settlement with blockchain

So far, we have implicitly assumed that stock exchange transactions may be finalized in real-time. Unfortunately, this is not the case. Indeed, although a sale or a purchase may be performed in a fraction of a second, a transaction settlement may take days, even weeks (Iansiti & Lakhani, 2017). This is at the origin of a major problem called settlement risk which results from a failure, either on the part of the seller or on the part of the buyer, to deliver a sold asset to or to pay for the latter.

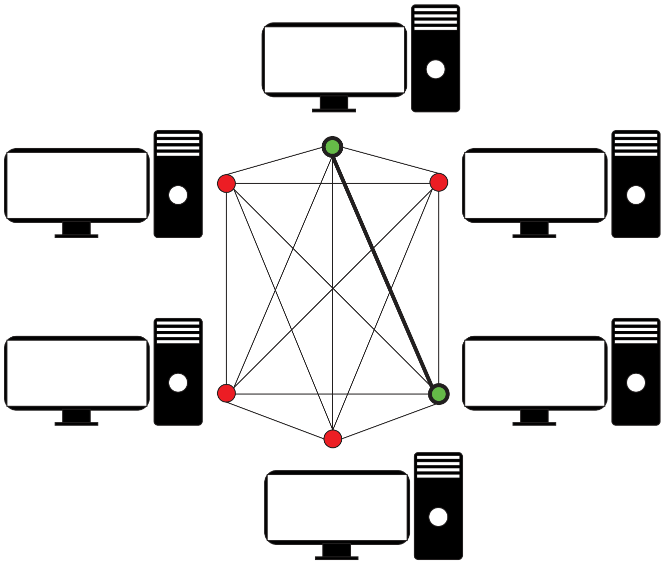

Fig. 5. Distributed network associated with the settlement blockchain.

Generally speaking, a specialized third-party coordinates legal ownership of securities versus payment: the so-called Central Securities Depository (Schmiedel, Malkamäki, & Tarkka, 2006). In addition, intermediaries such as brokers, custodians and payment agents are involved in the process (Pacces, 2000). Consequently, a typical settlement process is both time, and cost consuming (Benos, Garratt, & Gurrola-Perez, 2017). In order to address this problem, we have implemented a settlement system on a Bitcoin blockchain (Nakamoto et al., 2008). It is assumed that the assets may be exchanged on a digital market: the extent of which is determined by legal considerations rather than by technical ones. The cash and assets associated with the portfolio are stored in a digital wallet (Dwyer, 2015).

Each time the reinforcement learning algorithm sells or buys an asset, a transaction is initiated. Each transaction corresponds to a block in the blockchain. A transaction consists of two primary components; the input object, which contains the information about the original sender (timestamp, balance, signature, sender's public key) and the output object, which contains the nature and the amount of the transactions as well as the address of the receiver. The transaction is not validated by a third party but by a distributed consensus mechanism known as proof-of-work (Nakamoto et al., 2008).

In a consensus mechanism, all the nodes forming the network, such as the one illustrated in Fig. 5, compete to solve a cryptographic problem which is computationally prohibitive to solve but whose solution is easily validated. The nodes or miners may only solve this problem on a trial and error basis which means that it is almost impossible to obtain rapidly the solution. Consequently, it is virtually impossible to predict which miner shall validate the transaction.

The cryptographic problem consists of finding a number which is lower than the hash (Sobti & Geetha, 2012) associated with the new block (target hash) associated with the transaction. Not only this number must be lower than the target hash, but it must be determined within a certain tolerance given by the nonce. The lower the tolerance, the more difficult the problem becomes. The miner that first solves the problem adds the new validated block to the blockchain and receives a reward in terms of either digital or cryptocurrencies as an incentive. The validation consists of determining

Table 2
Composition of our Portfolio for the period in between January 2, 2002 and July 31, 2018.

Sector	Name
Technology	Apple Inc. (AAPL)
	Adobe Inc. (ADBE)
	Cisco Systems, Inc. (CSCO)
	Intel Corporation (INTC)
	International Business Machines Corporation (IBM)
	Microsoft Corporation (MSFT)
	NVIDIA Corporation (NVDA)
	Oracle Corporation (ORCL)
Financial Services	Bank of America Corporation (BAC)
	HSBC Holdings plc (HSBC)
	Royal Bank of Canada (RY)
	The Toronto-Dominion Bank (TD)
Consumer Cyclical	Amazon.com, Inc. (AMZN)
	The Walt Disney Company (DIS)
	The Home Depot, Inc. (HD)
Consumer Defensive	Walmart Inc. (WMT)
	The Coca-Cola Company (KO)
Industrials	Boeing Co (BA)
Healthcare	Johnson & Johnson (JNJ)
	Merck & Co., Inc. (MRK)
	Novo Nordisk A/S (NVO)
	Novartis AG (NVS)
Communication Services	Verizon Communications Inc. (VZ)

Table 4
Training and test sets.

ID	Training set	Test set
Test 1	2002-01-02 to 2008-08-15	2009-06-15 to 2009-10-21
Test 2	2002-01-02 to 2011-12-06	2012-10-03 to 2013-02-14
Test 3	2002-01-02 to 2015-04-06	2016-02-01 to 2016-06-09
Test 4	2002-01-02 to 2017-05-19	2018-03-19 to 2018-07-27

Table 5
Maximum drawdown (MDD) and Sharpe ratio ($\mathbf{S}_r$) as obtained with our system for the four test sets.

ID	Investment duration	MDD (%) algorithm	$\mathbf{S}_r$ Algorithm
Test Set 1	30 Days	1.71	−1.82
	60 Days	4.87	1.57
	90 Days	7.49	2.44
Test Set 2	30 Days	2.43	−1.014
	60 Days	6.91	2.22
	90 Days	8.23	2.20
Test Set 3	30 Days	4.62	2.9
	60 Days	9.44	2.34
	90 Days	17.46	3.47
Test Set 4	30 Days	3.13	3.13
	60 Days	12.52	3.09
	90 Days	12.52	2.26

Table 6
Return on investment for our approach and for the Dow Jones Industrial (Bloomberg, 2019).

ID	Investment duration	RoI (%) algorithm	RoI(%) DJI
Test Set 1	30 Days	0.24	5.62
	60 Days	3.67	10.85
	90 Days	5.74	15.52
Test Set 2	30 Days	0.5	−6.85
	60 Days	4.83	−4.12
	90 Days	6.91	3.88
Test Set 3	30 Days	2.43	4.88
	60 Days	7.4	9.68
	90 Days	18.09	9.33
Test Set 4	30 Days	2.83	−2.07
	60 Days	11.14	2.4
	90 Days	11.93	3.72

if both the money and the asset are available and then, to finalize the transaction. As a result, the settlement problem is solved without any recourse to an intermediary (Wall & Malm, 2016). In addition, such a distributed system may overcome a cyber-attack in which 50% or less of its nodes are overtaken (Sayeed & Marco-Gisbert, 2019).

It is important to point out that the blockchain is not an essential part of our framework in the sense that it does not affect the deep reinforcement learning investment strategy. The role of the blockchain is to mitigate the settlement risk; a risk that is also present in traditional trading. Furthermore, our approach is not limited to digital assets as it may be applied to any financial instrument traded on a stock exchange.

Our experimental results are presented in the next section.

6. Experiment and results

The composition of our portfolio, in terms of stocks, appears in Table 3. The times series starts on January 2, 2002, and ends on July 31, 2018.

In order to evaluate the performance of our system in terms of return on investment, two metrics were employed: namely the Sharpe ratio and the maximum drawdown (MDD). These metrics are among the most commonly employed in portfolio performance evaluation (Bessler, Opfer, & Wolff, 2017). The Sharpe ratio was introduced by Sharpe (1994) and is essentially a ratio in between the expected return on investment and the risk:

$$S_r = \frac{E\left[R_f - R_b\right]}{\sigma_d}. \tag{22}$$

where the numerator is the expectation on the differential return and where σ_d is the standard deviation associated with the asset excess return (or volatility); the higher the Shape ratio the better. It is generally accepted that a Sharpe ratio below one is sub-optimal; a ratio in between one and two is acceptable while a ratio in between two and three is considered very good. Any ratio over three is regarded as excellent (Maverick, 2019). The maximum draw down metric was introduced by Magdon-Ismail and

Table 3
Hyper-parameters.

Hyper-parameters	Size	Description
Portfolio Size	23	Number of assets considered to create a portfolio
Trading Window	30	Number of trading days in which agents trained to increase the portfolio value (investment return)
Trading Period	1 Day	Length of time in which price movements occur
Time Interval	4172	Number of Days from 02/01/2002 to 27/07/2018
Offline Learning Rate	10^{-4}	Parameter α of the Adam optimization (Kingma & Ba, 2015) for offline training.
Online Learning Rate	10^{-1}	Learning rate for online training session.
Sample Bias	10^{-5}	Parameter of geometric distribution when selecting online training sample batches. (The parameter β in Eq. (21) of Section 4.2)
Commission Rate	1.5%	Rate of commission fee applied to each transaction
Risk Free Rate	1%	The theoretical investment return rate with zero risk
Regularization Coefficient	10^{-8}	The L2 regularization coefficient.

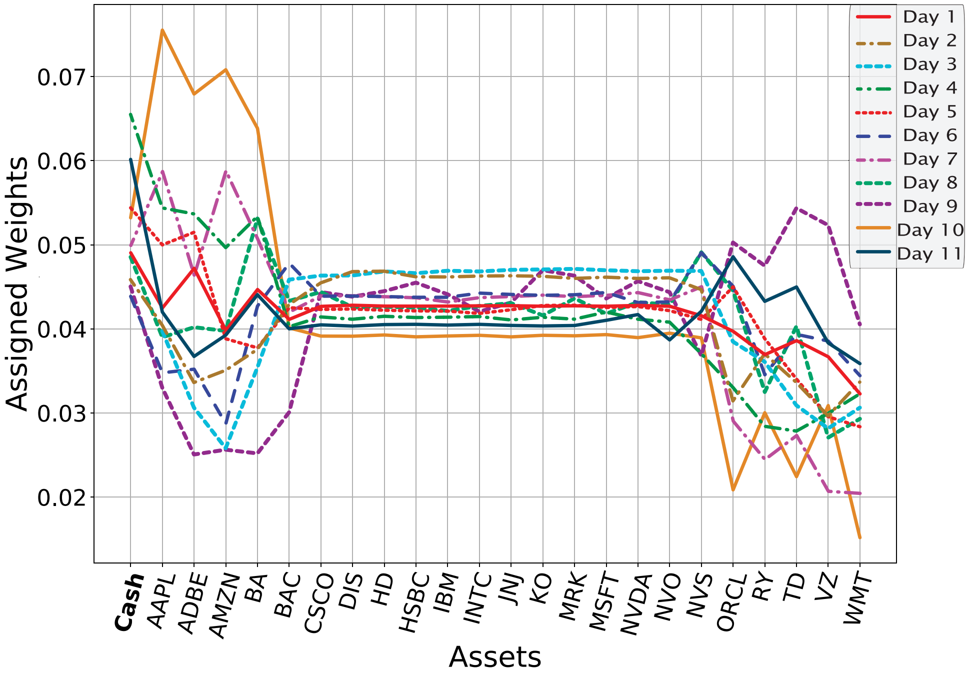

Fig. 6. Evolution of the portfolio weights during the first 11 days of training for the online learning process.

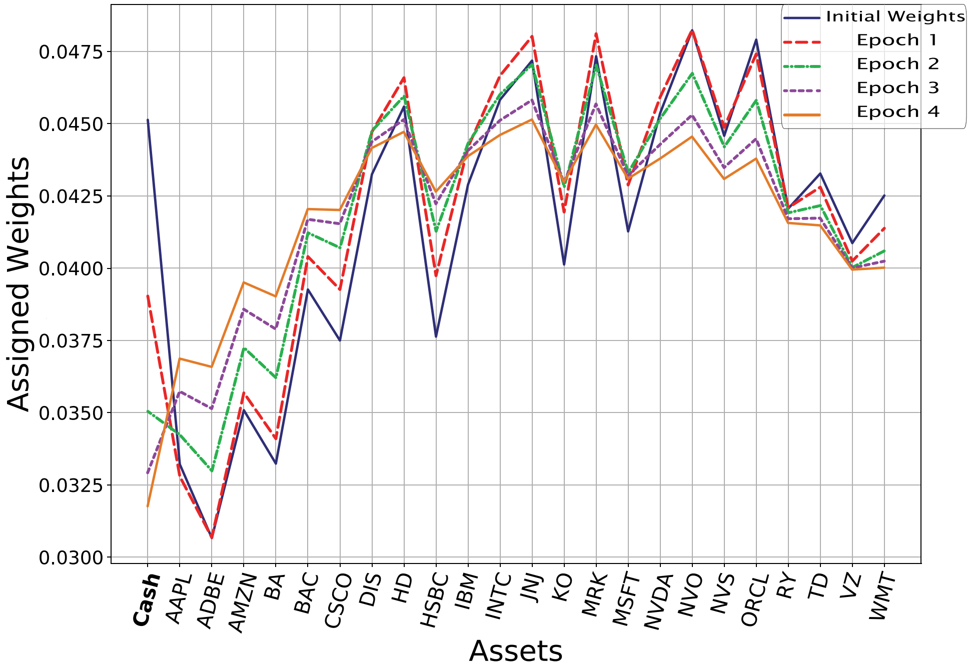

Fig. 7. Final portfolio weights distributions for the first training set.

Atiya (2004). This ratio is a function of the maximum P and minimum value L reached by the portfolio throughout a certain period of time; performant portfolios are characterized by a high MDD.

$$MDD = \frac{P - L}{P}. \tag{23}$$

In order to validate our approach, four training sets were generated. As mentioned earlier, the assets forming the portfolio are described in Table 2. Each training set covers a different period or window: all periods are initiated on January 2, 2002, while ending on August 15, 2008; December 6, 2011; April 6, 2015, and May 19, 2017 respectively. To these four training sets correspond four test sets of 90 days each starting the day following the end of their respective training sets. The training sets and their corresponding test sets are described in Table 4. Initially, the portfolio consists solely of cash. For each training set, the agent learns the weights associated with the assets in order to increase the expected return on investment. The hyper-parameters associated with the agent appear in Table 3.

The final weights distributions, for the first training set, during training episodes, are shown in Fig. 7. The final weights distributions of the second training set, for the four training episodes, are shown in Fig. 10. As it may be noticed from the figures, the final distribution is affected by the duration of the training period. Indeed, longer periods encompass states or events that do not occur

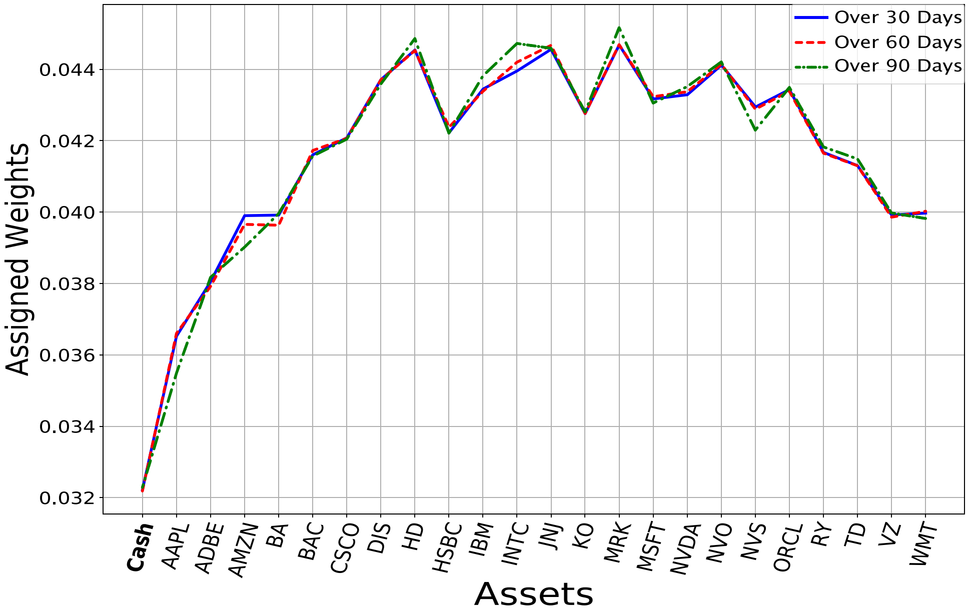

Fig. 8. Final portfolio weights distributions for the first test set over a period of 30, 60 and 90 days.

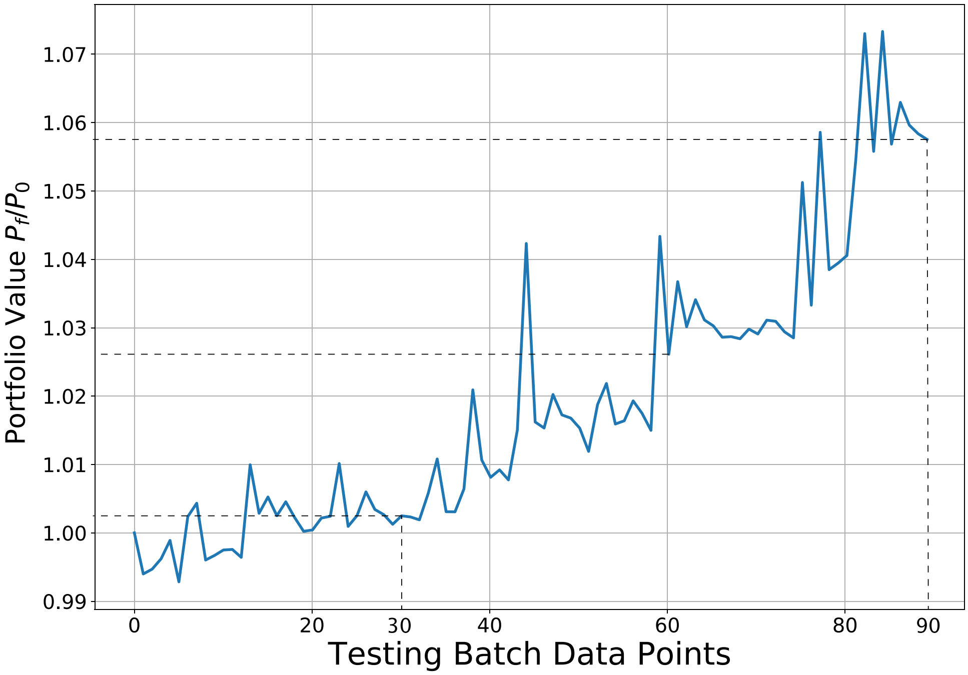

Fig. 9. Relative return on investment for the first testing set over a period of 30, 60 and 90 days.

in shorter periods which in turn affect the decisions taken by the agent. As illustrated by Fig. 6, weights fluctuate during the online learning process. Nevertheless, some weights are subjected to important variations, such as Apple and Amazon, while others, such as HSBC and IBM, are not. This behavior may be explained by the fact that the assets associated with the most stable weights are utilized in order to mitigate the risk (portfolio hedging) while the most volatile are employed for their leverage effect which explains the high variability of their weights.

Fig. 8 shows the final weights distributions, for the first test set, after thirty, sixty and ninety days respectively. Although sometimes subtle, the corrections have nevertheless a large impact on the expected return on investment. Indeed, as illustrated by Fig. 9, the relative return on investment, with respect to the initial amount of cash, steadily increases over this ninety days period with a relative again of almost 7%. Despite some fluctuations, there is a continued upward trend.

Likewise, Figs. 10–18 show the final training weights distributions for the last three training sets, the final weights distributions for the last three testing sets as well as their respective relative return on investments. In each set, a clear upward trend for the expected return on investment may be observed. The longer the training period, the more discontinuous is the evolution of the expected return on investment. The origin of the behavior is to be found in the fact that a longer training period allows the agent to learn more complex actions which in turn result in more drastic decisions. Nonetheless, the process remains very efficient as the fi-

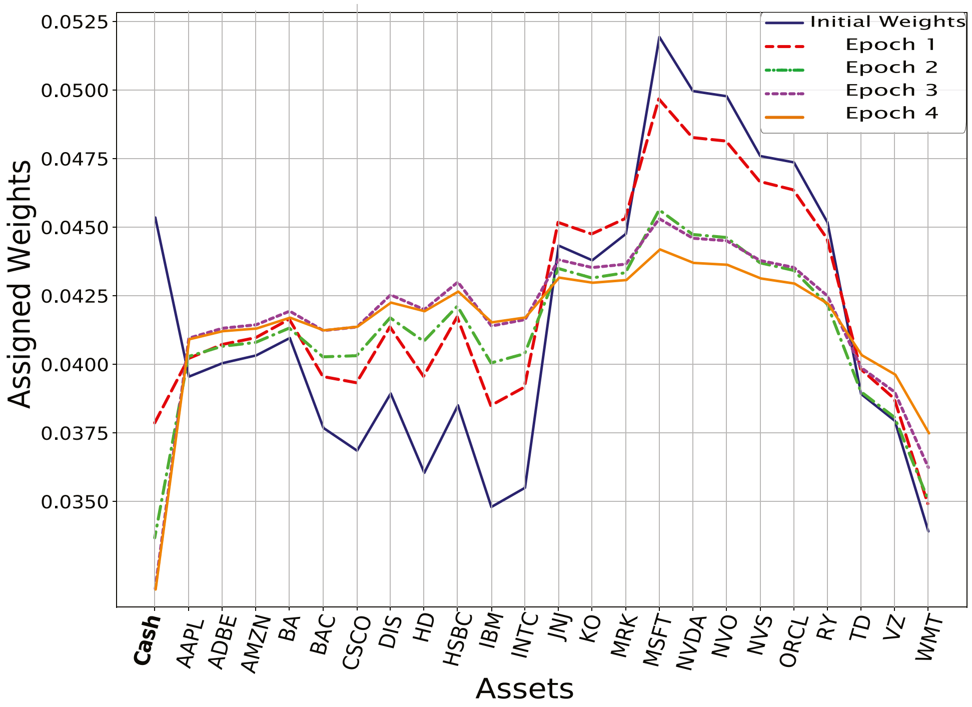

Fig. 10. Final portfolio weights distributions for the second training set.

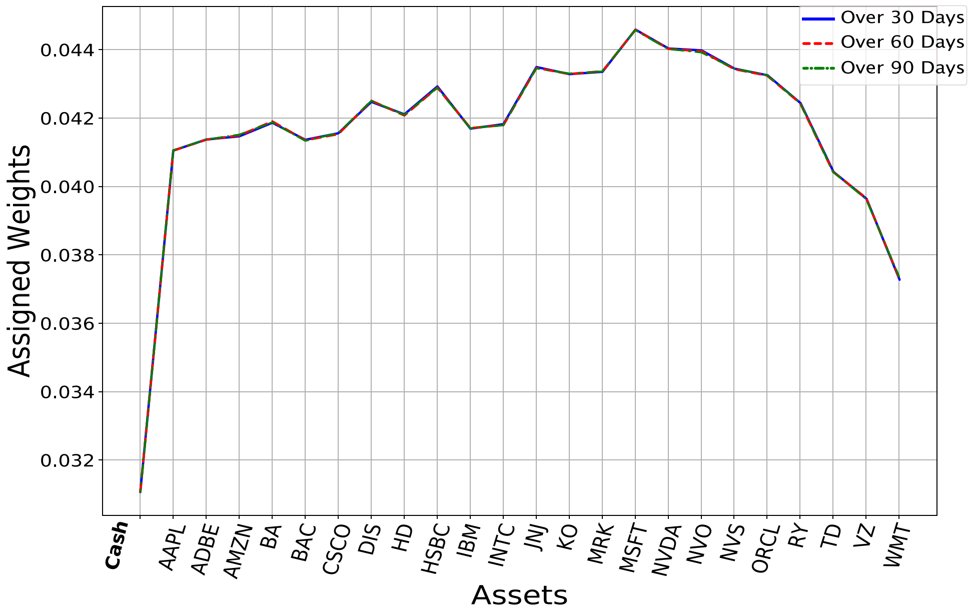

Fig. 11. Final portfolio weights distributions for the second test set.

nal relative returns on investment are approximately 7%, 18% and 11% for the last three test sets respectively.

Our results are summarized in Table 5. From the table, it should be noticed that the Sharpe ratios associated with our approach are either very good or excellent for nine out of the twelve test sets; the latter being described in Table 4. The test set 1 has the shortest training window while Test set 4 as the longest. Therefore, Table 5. clearly demonstrates that the performances of our system are improved if the CNN is trained over a large historical dataset such as Test set 4. In other words, a large amount of historical data is required in order to learn efficiently the policy. Indeed, for Test set 4, the Sharpe ratio is excellent for a time horizon of thirty and sixty days while it is very good for a long-term investment made over ninety days. These results are corroborated by the maximum drawdown metric.

In order to further ascertain the performances of our system, we compare the expected return on investment of our approach with the expected return on investment associated with the Dow Jones Industrial (DJI); the latter being considered a benchmark in the finance industry (Zhang, Wang, Li, & Shen, 2018). We restrict our discussion to Test set 4 as we have established that the agent must be trained over a large historical dataset in order to properly learn the policy. Our results are reported in Table 6. With our approach, the returns on investment (RoI) are 2.83%, 11.14% and 11.93% over periods of 30, 60 and 90 days respectively while the RoI, for the DJI, is 2.07% (a loss!), 2.4% and 3.72% over the same

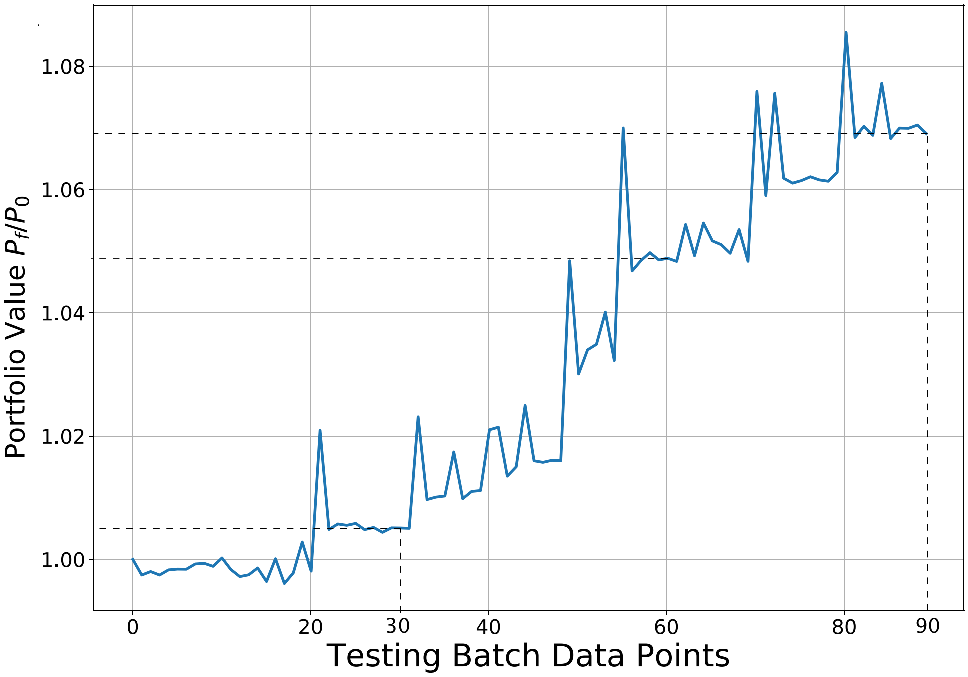

Fig. 12. Relative return on investment for the second testing set over a period of 30, 60 and 90 days.

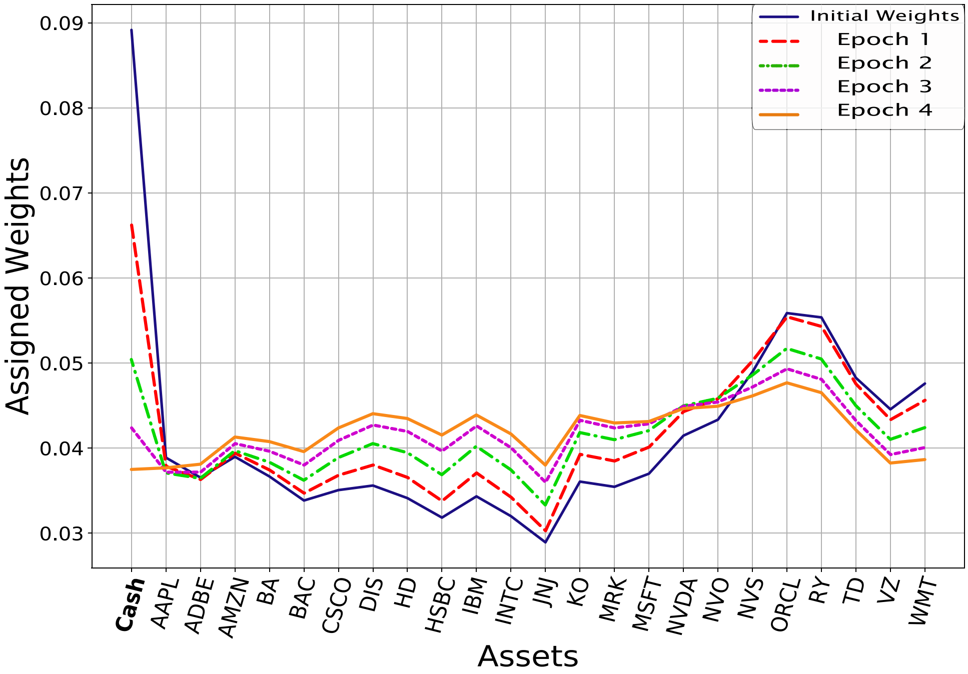

Fig. 13. Final portfolio weights distributions for the third training set.

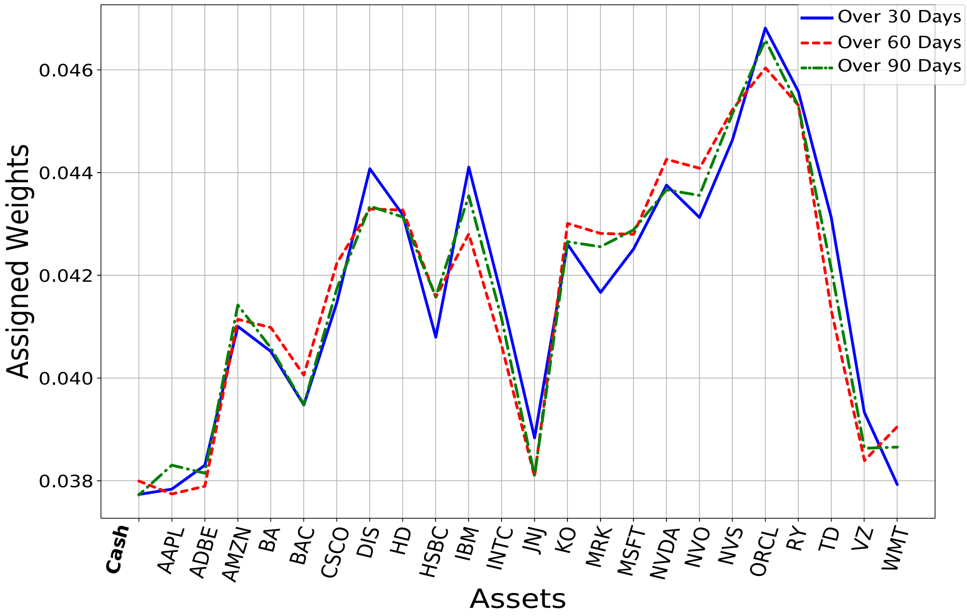

Fig. 14. Final portfolio weights distributions for the third test set.

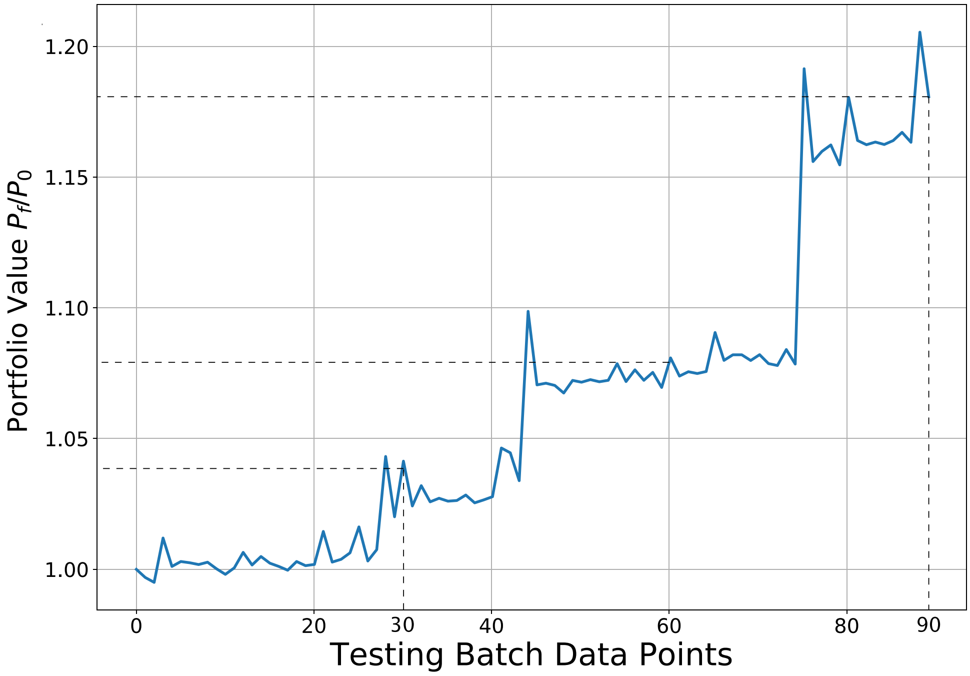

Fig. 15. Relative return on investment for the third testing set over a period of 30, 60 and 90 days.

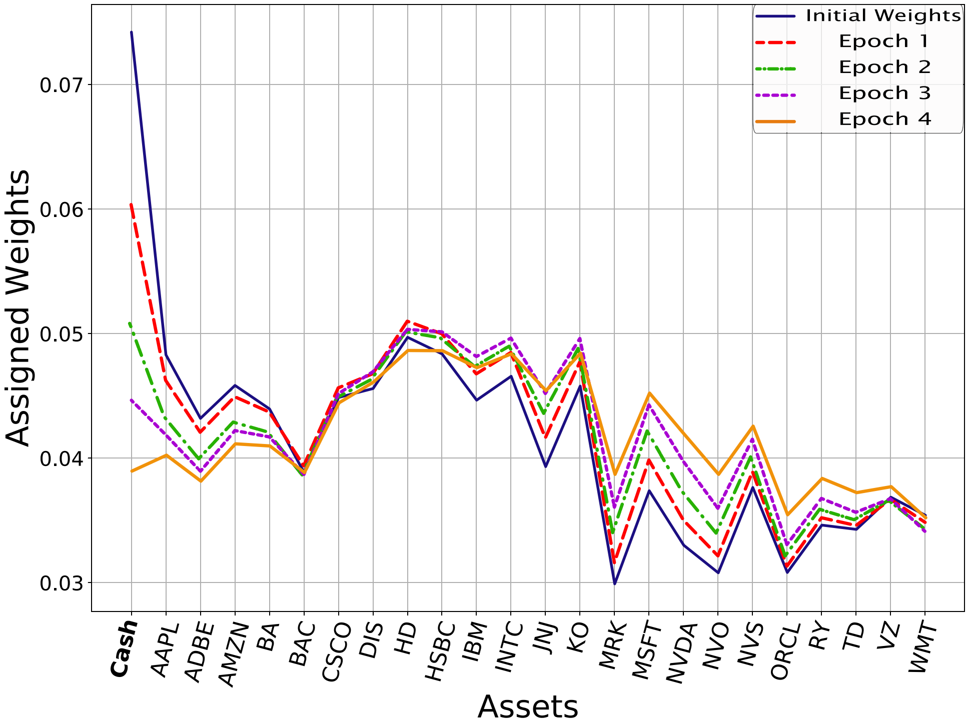

Fig. 16. Final portfolio weights distributions for the fourth training set.

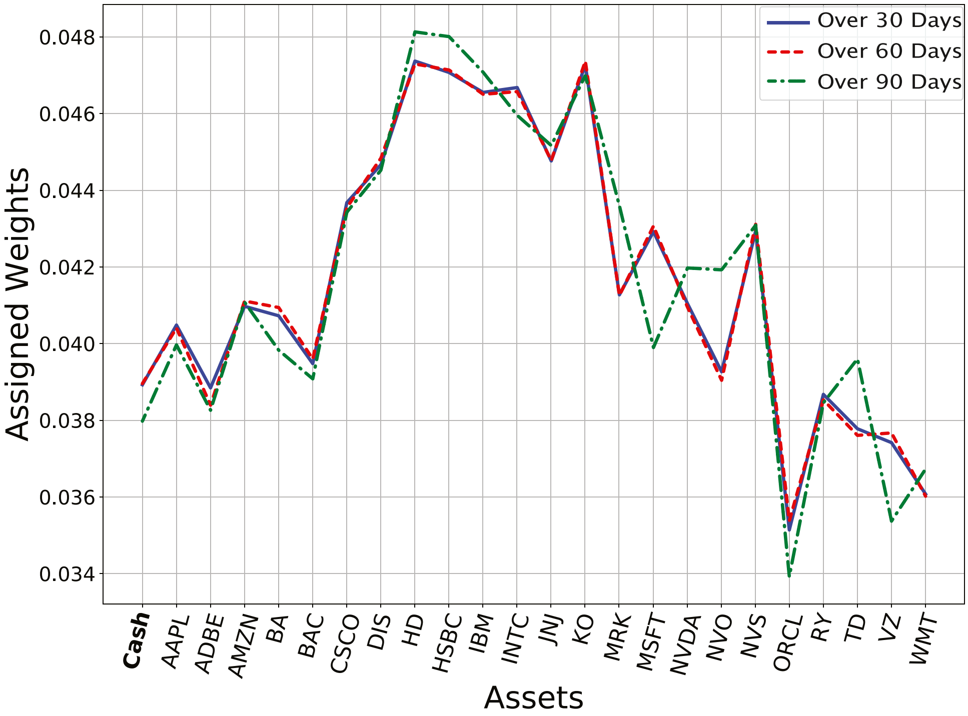

Fig. 17. Final portfolio weights distributions for the fourth test set.

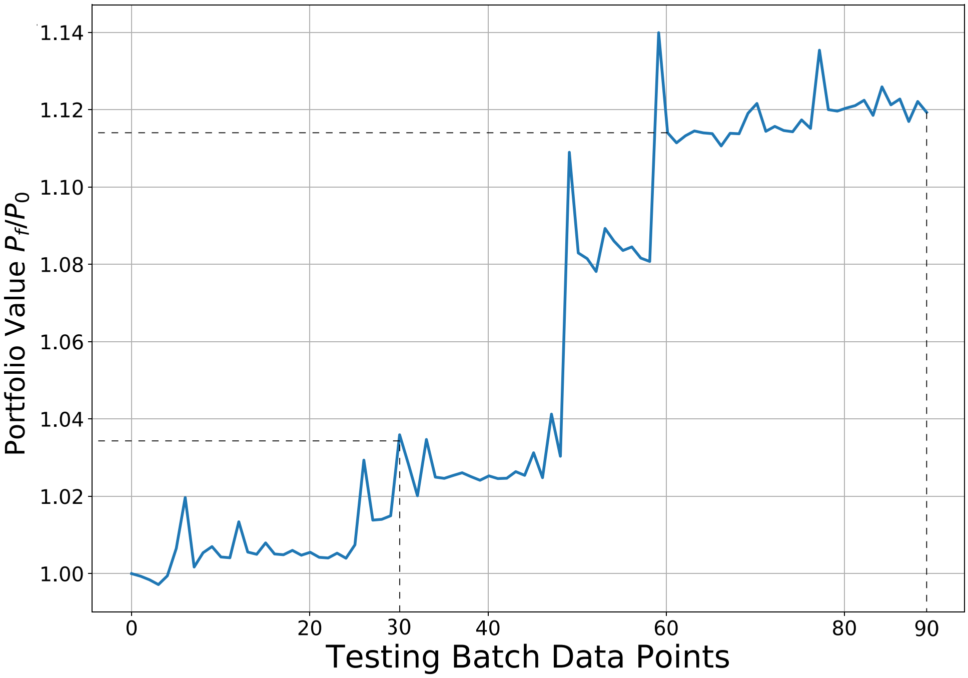

Fig. 18. Relative return on investment for the fourth testing set over a period of 30, 60 and 90 days.

time horizons. These results clearly demonstrate the effectiveness of our system.

7. Conclusion

This paper proposed a framework for portfolio management and optimization based on deep reinforcement learning called DeepBreath. A portfolio contains multiple assets that are traded simultaneously to automatically increase the expected return on investment while minimizing the risk. The investment policy is implemented using CNN; as a result, financial instruments are reallocated by buying and selling shares on the stock market. The neural network, as well as its hyper-parameters, is common to all the assets which means that the computational complexity only increases linearly with the number of financial instruments. In order to reduce the computational complexity associated with the size of the training dataset, a restricted stacked autoencoder was employed in order to obtain non-correlated and highly informative features. The investment policy was learned with the SARSA algorithm, both offline and online, using online stochastic batching and passive online learning respectively. The passive online learning approach handled concept drift resulting from exogenous factors. The settlement risk problem was mitigated with a blockchain. Our experimental results demonstrated the efficiency of our system. In future work, exogenous factors, such as social media, will be integrated into the framework in order to take into account their effect on price fluctuations. In addition, the theoretical model will be improved in order to take into account transactions in which a large volume of assets are traded i.e. when the actions (buying or selling) of the agent may have a large impact on the environment (large variation in prices and market liquidity). Finally, we plan to explore deep reinforcement learning architectures which are more suitable for long-term investment strategies.

Declaration of Competing Interest

All authors have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version.

This manuscript has not been submitted to, nor is under review at, another journal or other publishing venue.

The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript

Credit authorship contribution statement

Farzan Soleymani: Conceptualization, Methodology, Writing - original draft, Software, Visualization, Validation. **Eric Paquet:** Supervision, Writing - review & editing, Data curation.

References

Achelis, S. B. (2001). *Technical analysis from a to z*. McGraw Hill New York.

Atsalakis, G. S., & Valavanis, K. P. (2009). Surveying stock market forecasting techniques–part II: Soft computing methods. *Expert Systems with Applications, 36*(3), 5932–5941.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS One, 12*(7), e0180944.

Bengio, Y., Goodfellow, I. J., & Courville, A. (2015). Deep learning. *Nature, 521*(7553), 436–444.

Benos E, Garratt R, Gurrola-Perez P. The economics of distributed ledger technology for securities settlement. Available at SSRN 3023779. 2017 Aug 22.

Bessler, W., Opfer, H., & Wolff, D. (2017). Multi-asset portfolio optimization and out--of-sample performance: An evaluation of black–litterman, mean-variance, and naïve diversification approaches. *The European Journal of Finance, 23*(1), 1–30.

Bloomberg (2019). Dow Jones Industrial (DJI), stock quote. https://www.bloomberg.com/quote/INDU:IND. Accessed: 2019-04-15.

Bouchaud, J.-P. (2011). The endogenous dynamics of markets: Prce impact, feedback loops and instabilities. *Lessons from the credit crisis*. Risk Publications.

Cavalcante, R. C., & Oliveira, A. L. I. (2015). An approach to handle concept drift in financial time series based on extreme learning machines and explicit drift detection. In *2015 International joint conference on neural networks (IJCNN)* (pp. 1–8). IEEE.

Cheng, Z., Sun, H., Takeuchi, M., & Katto, J. (2018). Deep convolutional autoencoder-based lossy image compression. In *2018 Picture coding symposium (PCS)* (pp. 253–257). IEEE.

Choung, O.-h., Lee, S. W., & Jeong, Y. (2017). Exploring feature dimensions to learn a new policy in an uninformed reinforcement learning task. *Scientific Reports, 7*(1), 17676.

Cutler, DM, Poterba, JM, & Summers, LH (1988). What moves stock prices? *National Bureau of Economic Research, 17* Mar 1.

Dwyer, G. P. (2015). The economics of bitcoin and similar private digital currencies. *Journal of Financial Stability, 17*, 81–91.

Eakins, S. G., & Stansell, S. R. (2003). Can value-based stock selection criteria yield superior risk-adjusted returns: An application of neural networks. *International Review of Financial Analysis, 12*(1), 83–97.

Elwell, R., & Polikar, R. (2009). Incremental learning of variable rate concept drift. In *International workshop on multiple classifier systems* (pp. 142–151). Springer.

Fama, E. F. (1991). Efficient capital markets: II. *The Journal of Finance, 46*(5), 1575–1617.

Filimonov, V., & Sornette, D. (2012). Quantifying reflexivity in financial markets: Toward a prediction of flash crashes. *Physical Review E, 85*(5), 056108.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Brazilian symposium on artificial intelligence* (pp. 286–295). Springer.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR), 46*(4), 44.

Garcia, J., & Fernández, F. (2012). Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research, 45*, 515–564.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504–507.

Hu, Y., Liu, K., Zhang, X., Xie, K., Chen, W., Zeng, Y., & Liu, M. (2015). Concept drift mining of portfolio selection factors in stock market. *Electronic Commerce Research and Applications, 14*(6), 444–455.

Hussain, A. J., Knowles, A., Lisboa, P. J. G., & El-Deredy, W. (2008). Financial time series prediction using polynomial pipelined neural networks. *Expert Systems with Applications, 35*(3), 1186–1199.

Iansiti, M., & Lakhani, K. R. (2017). The truth about blockchain. *Harvard Business Review, 95*(1), 118–127.

Ivanov, S., & D'yakonov, A. (2019). Modern deep reinforcement learning algorithms. arXiv preprint arXiv:1906.10025.

Jangmin, O., Lee, J., Lee, J. W., & Zhang, B.-T. (2006). Adaptive stock trading with dynamic asset allocation using reinforcement learning. *Information Sciences, 176*(15), 2121–2147.

Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. arXiv:1706.10059.

Joulin, A., Lefevre, A., Grunberg, D., & Bouchaud, J.-P. (2008). *Stock price jumps: News and volume play a minor role* arXiv:0803.1769.

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization (2014). arXiv:1412.6980. 15.

Lam, M. (2004). Neural network techniques for financial performance prediction: Integrating fundamental and technical analysis. *Decision Support Systems, 37*(4), 567–581.

Magdon-Ismail, M., & Atiya, A. F. (2004). Maximum drawdown. *Risk Magazine, 17*(10), 99–102.

Malkiel, B. G. (2003). The efficient market hypothesis and its critics. *Journal of Economic Perspectives, 17*(1), 59–82.

Malkiel, B. G., & Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance, 25*(2), 383–417.

Maverick, J. B. (2019). What is a good sharpe ratio?https://www.investopedia.com/ask/answers/010815/what-good-sharpe-ratio.asp.

Moody, J., Wu, L., Liao, Y., & Saffell, M. (1998). Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting, 17*(5–6), 441–470.

Murphy, J. J. (1999). *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin.

Nakamoto S, Bitcoin A. A peer-to-peer electronic cash system. Bitcoin.URL: https://bitcoin. org/bitcoin. pdf. 2008.

Ormos, M., & Urbán, A. (2013). Performance analysis of log-optimal portfolio strategies with transaction costs. *Quantitative Finance, 13*(10), 1587–1597.

Pacces, A. M. (2000). Financial intermediation in the securities markets law and economics of conduct of business regulation. *International Review of Law and Economics, 20*(4), 479–510.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications, 42*(1), 259–268.

Rodríguez, J. J., & Kuncheva, L. I. (2008). Combining online classification approaches for changing environments. In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)* (pp. 520–529). Springer.

Ross, S., Mineiro, P., & Langford, J. (2013). *Normalized online learning* arXiv:1305.6646.

Sayeed, S., & Marco-Gisbert, H. (2019). Assessing blockchain consensus and security mechanisms against the 51% attack. *Applied Sciences, 9*(9), 1788.

Schmiedel, H., Malkamäki, M., & Tarkka, J. (2006). Economies of scale and technological development in securities depository and settlement systems. *Journal of Banking and Finance, 30*(6), 1783–1806.

Sharpe, W. F. (1994). The sharpe ratio. *Journal of Portfolio Management, 21*(1), 49–58.

Sobti, R., & Geetha, G. (2012). Cryptographic hash functions: A review. *International Journal of Computer Science Issues (IJCSI), 9*(2), 461.

Sorzano, C. O. S., Vargas, J., & Montano, A. P. (2014). *A survey of dimensionality reduction techniques* arXiv:1403.2877.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*: vol. 2. MIT Press Cambridge.

Tsai, C.-F., & Hsiao, Y.-C. (2010). Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems, 50*(1), 258–269.

Van Laarhoven, P. J. M., & Aarts, E. H. L. (1987). Simulated annealing. In *Simulated annealing: Theory and applications* (pp. 7–15). Springer.

Wall, E., & Malm, G. (2016). Using blockchain technology and smart contracts to create a distributed securities depository.

Wong, T., & Luo, Z. (2018). Recurrent auto-encoder model for large-scale industrial sensor signal analysis. In *Engineering applications of neural networks* (pp. 203–216). Springer International Publishing.

Zhang, W., Wang, P., Li, X., & Shen, D. (2018). The inefficiency of cryptocurrency and its cross-correlation with dow jones industrial average. *Physica A: Statistical Mechanics and its Applications, 510*, 658–670.