Received: 23 May 2023 | Revised: 10 August 2024 | Accepted: 28 August 2024

DOI: 10.1111/exsy.13725

ORIGINAL ARTICLE

Expert Systems WILEY

Prediction of stock prices with automated reinforced learning algorithms

Said Yasin[1] | Adrian Paschke[1,2] | Jamal Al Qundus[1,2,3]

[1]Institute of Computer Science, Freie Universitaet Berlin, Berlin, Germany

[2]Data Analytics Center, Fraunhofer FOKUS, Berlin, Germany

[3]Business Intelligence and Data Analytics, German Jordanian University, Amman, Jordan

Correspondence
Said Yasin, Institute of Computer Science, Freie Universitaet Berlin, Berlin 14195, Germany.
Email: said_yasin1234@hotmail.com

[Correction added on 15 October, after first online publication: Author Biography section is added in this version.]

Abstract

Predicting stock price movements remains a major challenge in time series analysis. Despite extensive research on various machine learning techniques, few models have consistently achieved success in automated stock trading. One of the main challenges in stock price forecasting is that the optimal model changes over time due to market dynamics. This paper aims to predict stock prices using automated reinforcement learning algorithms and to analyse their efficiency compared with conventional methods. We automate DQN models and their variants, known for their adaptability, by continuously retraining them using recent data to capture market dynamics. We demonstrate that our dynamic models improve the accuracy of predicting the directions of various DAX stocks from 50.00% to approximately 60.00%, compared with conventional methods. Additionally, we conclude that dynamic models should be updated in response to shifts rather than at fixed intervals.

KEYWORDS
automation, deep Q-learning, reinforcement learning, stock price prediction

1 | INTRODUCTION

Predicting stock price movements is one of the greatest challenges in time series data analysis (Gu et al., 2020). Despite extensive research exploring various machine learning techniques (Jiang, 2021), only a few models have achieved success in automated stock trading (Gu et al., 2020). A major problem in stock price forecasting is the non-stationary and chaotic nature of stock data (Gandhmal & Kumar, 2019). Non-stationarity means that the distribution of input data shifts over time (Raza et al., 2015), requiring corresponding changes in the model (Chollet, 2018). This phenomenon can be illustrated graphically in Figure 1.

In this context, existing market dynamics indicate that static models are inadequate for capturing long-term market changes. Nevertheless, the modelling of time series from a distributional perspective remains unexplored (Du et al., 2021).

Against this background, this paper addresses the problem of dynamic data changes over time using automated machine learning models. Our solution utilizes deep Q-networks (DQNs) due to their adaptability and high performance demonstrated in previous studies (Gao, 2024; Thakkar & Chaudhari, 2021; Zhang & Lei, 2022). Consequently, automated reinforcement learning can adapt more effectively to market fluctuations, while static models trained on non-stationary data become unusable over time (Ditzler et al., 2015).

So far, supervised learning algorithms have been preferred for stock price forecasts (Shi et al., 2021). Therefore, reinforcement learning algorithms have received less attention, particularly in combination with automated implementation.

This study is unique in exploring how DQN models can be automated to adapt to the dynamic characteristics of non-stationary price data. The practical importance of this study is to present, to the best of our knowledge, the first experimental results on how automated DQN models

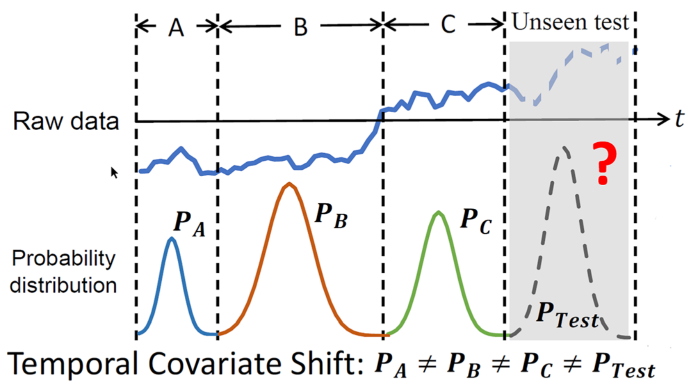

FIGURE 1 Covariate shift. *Source*: Du et al. (2021).

adapt to different market phases for stock price prediction, while its theoretical significance lies in highlighting the advantages of reinforcement learning algorithms through their dynamic implementation.

The paper aims to present the first insight into how automated deep reinforcement learning algorithms can be used to develop models that can better adapt to market fluctuations and learn to make optimal decisions in uncertain environments. The following research question is the subject of the paper:

To what extent can automated DQN models improve forecast accuracy?

This paper makes three significant contributions:

1. We introduce a novel approach using automated DQN models for stock price prediction.
2. We propose an automated approach based on extensive evaluations of different DAX stocks and updating strategies.
3. We conduct a comprehensive analysis of the performance of different DQN variants and architectures.

The remainder of the paper is organized as follows: Section 2 provides an overview of related work, focusing on the challenges and automated machine learning models for stock price prediction. Section 3 presents the theoretical background on DQN models. In Section 4, we explain the research methodology, including the data used and the updating process. Section 5 presents our findings, which are discussed in detail in Section 6. Finally, Section 7 summarizes our conclusions and outlines directions for future research.

2 | RELATED WORK

The prediction of stock price developments has been a significant research topic for many years (Jiang, 2021), leading to several comprehensive literature reviews (Gandhmal & Kumar, 2019; Kurani et al., 2021; Sezer et al., 2020; Thakkar & Chaudhari, 2021). For instance, the literature review by Kurani et al. (2021) documents investigations that date back to the 1970s. These reviews reveal a diverse array of models employed in stock price prediction, including multilayer perceptron networks (MLPs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), long short-term memory networks (LSTMs), generative adversarial networks (GANs), support vector machines (SVMs), ARIMA models, Prophet models, and various hybrid models combining these approaches. Jiang (2021) and Sezer et al. (2020) highlight that recurrent models are the most commonly used.

Furthermore, the previously mentioned literature studies demonstrate that researchers achieved test data accuracies of up to 99.9%, depending on the specific stocks, indices, and time periods analysed. The accuracy of 99.9% was achieved by Selvamuthu et al. (2019) using feed-forward neural networks for Reliance Private Limited stock tick data measured from 30.11.2017 to 11.01.2018.

Singh et al. (2022) review advancements in deep learning and reinforcement learning for financial applications such as forecasting, text mining, portfolio management, and credit risk analysis. They compare traditional methods with various reinforcement learning techniques, highlighting both value-based and policy-based methods. The study finds that reinforcement learning more effectively predicts market trends and outperforms traditional algorithms due to superior feature extraction and problem-solving capabilities.

Other researchers, such as Li et al. (2020), have also demonstrated the feasibility of deep reinforcement learning in financial markets, showing its credibility and utility for strategic decision-making through experiments with U.S. stocks. However, they point out that several challenges remain, including the exploration-exploitation balance problem, slow convergence rates, and the risk of space catastrophe, among others.

Zhang et al. (2023) aim to improve prediction accuracy through a fundamentally different approach. They introduce the first causal discovery algorithm that integrates the LiNGAM algorithm with LSTMs to identify causal relationships and predict time series data. The authors argue that the predicted stock prices depend on these causal relationships rather than merely on correlations.

Despite the versatility of various models, few have succeeded in automated stock trading (Gu et al., 2020). Kurani et al. (2021) point out that many studies in recent decades have neglected external factors, such as current investor sentiment, a concern also raised by Jin et al. (2019). Additionally, Jiang (2021) points out that some evaluations of trading strategies have focused solely on prediction results without considering overall portfolio performance, including transaction costs. Furthermore, modelling time series from a distributional perspective remains unexplored (Du et al., 2021), even though it is known that stock price data are non-stationary (Gandhmal & Kumar, 2019). Researchers such as Guo et al. (2018) have demonstrated this non-stationarity through experiments showing that stock price data can exhibit different distributions over time.

Krawczyk et al. (2020) and Souza et al. (2020) note that this issue can render historical data irrelevant or even harmful to forecasting models. To address this challenge, Chollet (2018) suggests continuously retraining models with new data.

Related work that aims to enhance stock price prediction accuracy through continuous retraining is presented in Nguyen et al. (2019). They compared the prediction accuracies of dynamic and static LSTMs using four stocks listed on NASDAQ, with daily price measurements. In the dynamic LSTM, the model was updated after each prediction by adding the current prediction to the training set, whereas the static LSTM did not adapt. After approximately 7 days of computation, the authors found that automating the LSTMs significantly improved prediction accuracy, reducing the mean square error by 45.9% for the stock period from 2014 to 2019. They also noted that dynamic LSTMs were less dependent on window size compared to static LSTMs.

Guo et al. (2018) explored the potential for automating the support vector machine regression (SVR) learning algorithm to improve prediction accuracy. The study analysed stock price data from five randomly selected stocks on the Shanghai Stock Exchange, covering the period from 2015 to 2017. They automated the SVR algorithm using particle swarm optimization (PSO) and updated the parameters only when the model error exceeded a specified threshold. The adaptive SVR algorithm demonstrated better adaptability and prediction accuracy compared to traditional SVR algorithms and backpropagation neural networks, reducing prediction error by 8%–41%. Additionally, their study showed that the predictive accuracy of static models deteriorated over time.

Liu et al. (2022) tackled the issue of changing data distributions by enabling LSTMs to dynamically fine-tune and adapt to the latest stock price data distributions using model-agnostic meta-learning. They demonstrated that their method significantly enhances prediction accuracy for Chinese and American stocks. In contrast, Wang et al. (2023) addressed the problem of changing distributions with a different approach, proposing a deep sequence model based on Koopman theory. This model captures shared characteristics through a global operator and adapts to local dynamics with a local operator, and demonstrating state-of-the-art performance on highly non-stationary datasets, such as cryptocurrency forecasting.

Other related works, such as that of de Lima e Silva et al. (2020), apply non-stationary fuzzy time series (NSFTS) which can be used to dynamically adjust fuzzy sets to take into account the changes in the stochastic process. Similarly, there is work such as that of Du et al. (2021), which tries to characterize the different data distributions in a time series and then use RNNs to reduce the distribution divergences and improve the predictions. However, to the best of our knowledge, no other study has investigated the automatic adjustment of DQNs to the changing characteristics of non-stationary stock price data. Nonetheless, several studies have shown that static DQNs effectively predict stock prices and outperform traditional methods.

For instance, in the study by Thakkar and Chaudhari (2021), DQNs achieved an accuracy of nearly 100%, while many other models, including LSTMs, RNNs, and CNNs, only reached an approximate accuracy of 50%.

Similarly, Bajpai (2021) evaluated the performance of DQNs using Indian stocks, suggesting that due to the highly stochastic and rapidly changing nature of stock markets, DQN models outperform conventional methods because of their rapid adaptability. Zhang and Lei (2022) observed similar results with U.S. stocks, noting that DQN models excel during turbulent market periods due to their adaptability. Analogously, Gao et al. (2020) found that DQNs outperform 10 other traditional strategies in the context of stock market portfolio management with U.S. stocks. Furthermore, Gao (2024) compared the performance of DQNs and Double DQNs in predicting WorldCall stock prices and concluded that these models are better at capturing nonlinear features and dynamic changes in the stock market, with Double DQNs being slightly superior. Similarly, Shi et al. (2021) analysed Double DQNs and found that they outperformed LSTMs and SVMs across various metrics using several American and Chinese stock indices.

Building on the success of DQNs in single-agent settings, Li and Hai (2024) explored their use in a multi-agent scenario. Their Collaborative Multi-Agent Reinforcement Learning-based Stock Portfolio Management System employs three agents:

- **Stock Market Agent**: Captures short-term stock trends.
- **Financial Index Agent**: Recognizes long-term market changes.
- **Stock Correlation Agent**: Analyses stock interdependencies.

Their outputs are integrated using a self-attention network, resulting in superior performance and higher cumulative returns compared to state-of-the-art models.

Finally, it is important to point out the weaknesses of our literature review. Since the approach of this paper is highly specific, there are no papers that show how well or poorly this approach worked for them. Either papers were cited that show how well automated ML algorithms work, or static DQNs, but not automated DQNs. Another significant weakness of the aforementioned work on automated ML models is that the very good results were achieved with small samples, such as in Guo et al. (2018) with five stocks or Nguyen et al. (2019) with four stocks, which are not necessarily representative of all stocks.

3 | BACKGROUND

3.1 | Reinforcement learning

Reinforcement learning algorithms are able to perceive an environment based on feature vectors s_i and accordingly execute actions a_i according to the perceived state (Burkov, 2019). Different actions can result in different rewards R_i (Alpaydin, 2022). This process is illustrated in Figure 2. The goal of reinforcement learning is to identify the actions that maximize expected rewards over time (Géron, 2019). A strategy is sought, similar to a supervised learning algorithm, in the form of a function $\pi(s)$ that outputs an optimal action given a perceiving feature vector representing the state (Burkov, 2019).

3.2 | Markov decision process

The reinforcement learning algorithms are based on the formalism of Markov decision processes, which in turn are based on ordinary Markov processes (Lapan, 2020). A Markov process is a stochastic process, that is, a family of random variables $\{X(t), t \in T\}$ whose probability distribution functions have the Markov property. Accordingly, Markov processes can be defined as continuous-time or discrete-time Markov processes. Formally, a discrete-time Markov process $\{X_n,\ n=0,1,2,\dots\}$ is a stochastic process that satisfies the following Markov property for any finite set of states $\{x_n,\ n=0,1,2,\dots\}$:

$$\begin{aligned} P(X_{n+1}=x_{n+1}|X_n=x_n, X_{n-1}=x_{n-1},\dots,X_0=x_0) \\ = P(X_{n+1}=x_{n+1}|X_n=x_n) \end{aligned} \tag{1}$$

Simply put, the Markov property means that the transition probabilities depend solely on the presence (Stewart, 2009).

Markov decision processes form the theoretical basis for reinforcement learning algorithms and were defined by Richard Bellman in the 1950s and differ from Markov processes by the following three additional characteristics (Géron, 2019):

1. At each step, there is the possibility of selecting multiple actions.
2. Transition probabilities depend on the action selected.
3. State transitions can return a positive or negative reward.

3.3 | Q-learning

Watkins (1989) introduced the Q-learning algorithm for situations where transition probabilities and the reward function are unknown. Q-learning involves an agent trying different actions in various states and learning from the immediate rewards or punishments. Over time, the

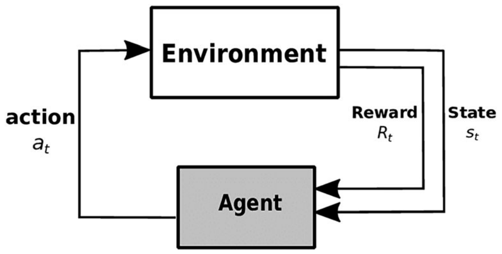

FIGURE 2 Agent and environment. *Source*: Amiri et al. (2018).

agent identifies the best actions by considering long-term rewards. The convergence theorem is detailed and proven in Watkins and Dayan (1992). The Q-learning algorithm is defined as follows:

$$\mathrm{Q}(\mathrm{s},\mathrm{a}) = (1-\alpha)\mathrm{Q}(\mathrm{s},\mathrm{a}) + \alpha[\mathrm{r} + \gamma \max \mathrm{Q}(\mathrm{s}',\mathrm{a}')] \tag{2}$$

In this equation, s is the current state, a is the chosen action, s' is the subsequent state, r is the received reward, α is the learning rate, γ is the discount factor and $\max \mathrm{Q}(\mathrm{s}',\mathrm{a}')$ is the best the agent thinks it can do from state s'. In other words, the Q-value is the expected discounted reward for taking action a in state s and following policy π thereafter.

3.4 | DQN

The described Q-learning algorithm is not scalable for Markov decision processes with many states and actions (Karunakaran et al., 2020). To this extent, for action and state spaces that are too large, it is no longer possible to estimate the Q-value of each state-action pair (s,a). To solve this problem, DeepMind published a paper in 2013 in which they proposed to learn a function $Q_\theta(s,a)$ that approximates the Q-value for each state-action pair (s,a) using feedforward neural networks (Mnih et al., 2013). This model idea was named deep Q-network (DQN) in the paper. The target values of the neural network, with which the network error is measured, are to be determined by the Q-learning rule (see formula 2).

3.5 | Double DQN

In 2015, DeepMind enhanced its DQN algorithm by introducing the Double DQN (van Hasselt et al., 2015). DeepMind researchers found that their 2013 version overestimated Q-values, which negatively impacted training performance. The reason for this is believed to be the max operation (see formula 2), as the approximated Q-values are often noisy, and using the max operation leads to an overestimation of the Q-values. To solve this problem, the DeepMind researchers proposed to split the max operation into action selection and action evaluation. They implemented this as follows:

$$y = r + \gamma Q_{\theta'}\left(s', \max_{\alpha'} Q_\theta(s',a')\right) \tag{3}$$

A first neural network Q_θ, called the online network, evaluates the greedy strategy, that is, the action with the highest Q value is selected, and a second neural network $Q_{\theta'}$, called the target network, calculates the Q value based on the action selected by the online network. The target network is a periodic copy of the online network.

3.6 | Duelling DQN

The Duelling DQN algorithm is also an updated version of the DQN algorithm also presented by DeepMind in 2015 (Wang et al., 2015). They describe that their update provides the main advantage to generalize learning across actions without the need to change the underlying reinforcement learning algorithm. The duelling DQN algorithm is based on the fact that the Q-value of a state-action pair (s,a) can be expressed, as it were, as follows:

$$Q(s,a) = V(s) + A(s,a) \tag{4}$$

$V(s)$ is a measure of how good it is to stay in a certain state s and $A(s,a)$ is a relative measure of the importance of the individual actions. For the Q-value of the best action $Q(s,a^*)$ it applies that $A(s,a) = 0$. The duelling DQN calculates the state-action value $Q(s,a)$ by combining the value $V(s)$ and the advantage function $A(s,a)$ with a single deep model.

3.7 | Deep recurrent Q-network

The DRQN is a DQN that uses a recurrent neural network to approximate the function $Q_\theta(s,a)$, and was also first introduced in 2015 by Hausknecht and Stone (2017). The authors described that in real environments it is rather rare that the complete state of the system can be provided to the agent or even determined, so that in real environments the Markov property (see formula 1) rather rarely holds. They describe that a partially observable Markov decision process (POMDP) better reflects the dynamics of many real environments. Therefore, Hausknecht and Stone (2017) defined DRQNs and hypothesized that they can better handle POMDPs by leveraging advances in recurrent neural networks.

3.8 | Covariate shift

The so-called covariate shift describes the phenomenon that the data distribution changes over time (Raza et al., 2015). In mathematical terms, a covariate shift occurs when the distribution $P(X)$ of the input changes, but the conditional distribution $P(Y|X)$ of the output remains the same (Huyen, 2022).

3.9 | Mean directional accuracy

$$MDA = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\left[sgn(y_i - y_{i-1}) = sgn\left(\widehat{y}_i - y_{i-1}\right)\right]} \tag{5}$$

The mean directional accuracy (MDA) (see formula 5) is a measure of the probability of how accurately a model can predict the actual direction (upward or downward) of a time series (Dauphin et al., 2022).

3.10 | Root mean square error

$$RMSE = \sqrt{\left(\frac{1}{N}\right)\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2} \tag{6}$$

The root mean square error (RMSE) (see formula 6) is a widely used metric for evaluating models and is optimal for gaussian errors (Hodson, 2022).

4 | RESEARCH METHODOLOGY

We begin by examining the data, with an emphasis on its distribution and significance (see section 5.1). This is followed by the main study, comprising two preliminary investigations and a primary investigation.

The first preliminary investigation identifies the optimal combinations of the four DQN variants with various network architectures (see section 5.2).

In the second preliminary investigation, the best models from the first preliminary investigation are trained with an extended dataset to ensure that any poor results are not due to unrepresentative training data (see section 5.3).

After that, the main investigation begins, where the best models from the first preliminary investigation are automated (see section 5.4). Five different strategies are tried for the automation process, which are explained in more detail in section 4.6.

The study is then extended by examining how different stock price intervals and window sizes affect automation results.

Finally, supervised learning algorithms are also trained with the same data to provide further benchmarks (see section 5.5), and a statistical test is performed to verify the improvements of the automated models (see section 5.6).

Before we start this research process in section 5, we first explain the conditions on which our study is based, such as our data structure, the agent environment, network architectures, hyperparameters, DQN algorithm and update strategies.

4.1 | Data

As use case, the DAX stocks Adidas, BASF and Allianz are examined. We consider opening, high, low, and closing prices, trading volume, DAX closing prices, and Google Trends data using the search queries ‘crash’ and the respective stock names (Adidas, BASF, or Allianz). The DAX closing prices are included in the study to have a rough overview of the other DAX participants and the Google Trends data is used to have a sentiment indicator included in the prediction. The stock data is downloaded using the Python library yfinance[1] and the Google Trends data is downloaded using the Python library pytrends.[2] The training data for the individual stocks is created using stock price data collected from 01.01.2017 to 31.12.2019.

For daily measured data, this corresponds to 738 data records (3 years $\times$ 52 weeks $\times$ 7 days) without the stock-free days. For the validation data, the period from 01.01.2020 to 31.12.2020 is considered, and for the test data, the period from 01.01.2021 to 31.12.2021 is considered. This corresponds to 248 data records (1 year $\times$ 52 weeks $\times$ 7 days) without the stock-free days. Together with the seven selected characteristics (five historical stock data characteristics and two Google Trends data), this results in three data sets with the dimensions 738 $\times$ 7 and two times 248 $\times$ 7. The data set size would be reduced by a factor of around 5 for weekly stock data and increased by a factor of around 11–12 for hourly stock data.

4.2 | Agent environment

One challenge of reinforcement learning is to provide an environment (see Figure 2) for an agent to develop a strategy for the underlying Markov decision process (Géron, 2019). This challenge was addressed in this paper using the OpenAI Gym API[3] by defining a custom agent environment, which is illustrated in the following figure:

In the initial state s_0, the agent has information about the opening, high, low, closing prices, trading volume, DAX closing prices and Google Trends data of a particular stock from time t_0 to t_2 (see Figure 4). Based on this perceived state, the agent must choose an action a_0 to determine whether the closing price will rise or fall in the next time unit t_3. If the price direction was predicted correctly, the agent receives a reward r_0 of 1, otherwise a reward r_0 of -1. Then the agent enters state s_1 and perceives the stock's data from time t_1 to t_3, and this process repeats until the agent is in the final state s_n and must choose the last action a_n in order to predict the price direction for the last time t_n. The DQN agents trained in this environment were implemented using Tensorforce[4] because it is very user-friendly.

4.3 | Neural network architectures

For the first preliminary investigation, the following neural network architectures are compared for the different DQN models (Table 1):

Model 1 is the default network architecture of Tensorforce and model 5 represents the DRQN. In addition, after the convolution layer, all models have a poling layer with filter size = 2 followed by a flattening layer that transforms the result of the pooling layer into a 1D array. The convolutions and pooling operations are performed with stride = 1 and padding = 'same' (input size = output size). Furthermore, the model 2 also has 5 batch normalization layer.

As you can see, we have used larger network architectures with multiple layers (see model 4) and smaller ones with fewer layers (see model 1). We have varied this so that we do not suffer from underfitting due to too few parameters and overfitting due to too many parameters.

The number of dropout layers (which randomly deactivate a certain number of neurons) in our case was dependent on the number of hidden layers, as we added one of these layers after each hidden layer (see Table 2). In this way, we wanted to make it as difficult as possible for irrelevant patterns to be memorized by individual layers. We determined the dropout rate of 0.3 experimentally.

At the same time, taking into account the dropout rates, we have set the number of hidden neurons in each layer slightly higher to 64, so that we do not have too few neurons after the temporary dropout of neurons and consequently suffer from underfitting. We also do not set the number of hidden neurons to 128, as we have observed that with this setting the validation accuracy decreased while the training accuracy increased (overfitting).

We only used one LSTM layer for the DRQN (see model 5) because we realized that the computing time for additional LSTM layers would increase drastically and we could no longer make good use of this network architecture due to the computing time required for automation, and so forth.

In our CNNs and in some hidden layers, we used the ReLU (rectified linear unit) activation function, defined by $f(x) = max(0,x)$. We chose ReLU due to its minimal computational demands and its ability to accelerate the convergence of the network (Lin & Shen, 2018). For models 3 and 4, we incorporated alternative activation functions in response to recent research suggesting superior alternatives. The Exponential Linear Unit, introduced by Clevert et al. (2015), integrates an exponential component to accommodate negative inputs, enhancing learning dynamics and accuracy. Additionally, the Scaled Exponential Linear Unit, further developed by Klambauer et al. (2017), not only processes negative inputs but also normalizes outputs automatically, significantly boosting model stability and performance.

We set the number of filters in models 2, 3, and 4 in ascending order (see Table 1), which is also recommended in literatures such as Géron (2019), so that the CNN gradually learns to recognize increasingly complex patterns, and we selected their kernel sizes experimentally. For the

TABLE 1 Overview of the convolutional neural networks (CNNs) of the models.

	First 1D convolutional layer			Second 1D convolutional layer		
Name of model	Filters	Kernel size	Activation function	Filters	Kernel size	Activation function
Model 1	64	3	ReLU	64	3	ReLU
Model 2	32	7	ReLU	64	7	ReLU
Model 3	32	7	ReLU	64	7	ReLU
Model 4	32	7	ReLU	64	7	ReLU
Model 5	64	3	ReLU	64	3	ReLU

TABLE 2 Overview of the neural networks of the models after the convolutional neural networks.

	Hidden layers			LSTM layer			Dropout layers	
Name of model	No. of layers	No. of neurons	Activation function	No. of layers	Units	Activation function	No. of layers	Rate
Model 1	1	64	ReLU	-	-	-	-	-
Model 2	5	64	ReLU	-	-	-	5	0.3
Model 3	5	64	ELU	-	-	-	5	0.3
Model 4	8	64	SELU	-	-	-	8	0.3
Model 5	3	64	ReLU	1	64	Tan*h*	3	0.3

DRQN (see model 5), we have defined the number of filters and the kernel sizes on the basis of the default network architecture of Tensorforce (see model 1).

We have visualized the default network architecture of Tensorforce (see model 1) in Figure 5. The task of the CNN at the beginning is to recognize individual sequential patterns depending on the kernel size during training. Then the forward network prepares after the flatten layer the output of the CNN with one hidden layer with 64 neurons and outputs the Q-value estimations $Q(s,a)$ for the prediction of the stock price direction (upwards or downwards).

4.4 | Hyperparameters

The following hyperparameters are used for the models:

We have chosen the parameters by testing different constellations and using the ones that produced the best results, while also taking into account what other researchers such as Bajpai (2021) have selected in the literature.

The unusual batch size of 50 (see Table 3) is derived from various experiments and has also been used by other researchers, such as Shah et al. (2018), in their stock price predictions. However, parameters such as a discount factor of 0.99 are also common practice in reinforcement learning (Franceschetti et al., 2022).

We chose the Adam algorithm introduced by Kingma and Ba (2015) as the learning algorithm because it is described as suitable for non-stationary targets and, according to researchers such as Camacho et al. (2019), is even the best choice for solving problems with non-stationary targets and very noisy gradients. Furthermore, researchers such as Zhang et al. (2022) describe the Adam algorithm as a theoretically justified algorithm that practitioners can use confidently.

4.5 | DQN algorithm

To better understand how the DQN by Mnih et al. (2013) works, its general steps are described below:

1. Initialize the weights of the neural network $Q_\theta(s,a)$ randomly and the maximum capacity N of the empty replay buffer D
2. Execute each step $t=0,\ldots,T$ for M episodes as follows:
   a. Select an action a_t using the epsilon-greedy method. Consequently, choose the action a_t with a probability of ϵ at random or with a probability of $1-\epsilon$ according to $arg\ max_a\ Q_\theta(s,a)$.
   b. Perform the selected action a_t, change to the next state s' and receive the reward r.

TABLE 3 Hyperparameters of the models.

Hyperparameter	Value
Window size	10
Batch size	50
Episodes	1000
Discount factor	0.99
Memory size	624
Learning algorithm	Adam

TABLE 4 Update strategies.

Strategy	Description
1	After each update, the agent is reset to the initial state. For each update, only the data of the last *n* data points are considered.
2	The agent's parameters are continuously adjusted in the dynamic process. For each update, only the data of the last *n* data points are considered.
3	After each update, the agent is reset to the initial state. For each update, all data known up to the time of the update is used.
4	The agent's parameters are continuously adjusted in the dynamic process. For each update, all data known up to the time of the update is used.
5	The agent is updated only if a covariate shift is suspected. For each update, only the data of the last *n* data points are considered.

c. Save the transition information (s,a,r,s') in the replay buffer D.
d. Extract a random mini-batch with K transition information from the replay buffer.
e. Calculate the target values $y_i = r_i + \gamma \cdot \max_{\alpha'} Q_\theta(s_{i'}, \alpha')$. When the final state a_T is reached, the target values $y_i = r_i$.
f. Calculate the network error $L(\theta) = \frac{1}{K}\sum_{i=1}^{K}(y_i - Q_\theta(s_i,a_i))^2$.
g. Update the network parameters θ of the function $Q_\theta(s_i,a_i)$ using a gradient descent method.

4.6 | Update strategies

The best performing models from the first preliminary investigation are updated at n fixed time intervals $t_i = t_0 + i \cdot n$.

Before the model is updated, at time t_{i-1} the stock price directions for the next n data points are predicted. The update is done by re-training the model with an updated training data set. The updated training data set takes into account the stock price data that was not known at the time of prediction. In addition to the described scheme, five following strategies are tried with respect to the agent state and the training dataset:

The purpose of resetting the agent after each update in strategies 1 and 3 (see Table 4) is to assess the utility of the pre-trained model. The purpose of distinguishing how much historical stock price data should be considered for the update is to investigate whether the most recent n information (instead of all known) are sufficient for the prediction. In this way, computation time can be saved and unnecessary price data can be omitted. The idea behind strategy 5 is to make only the necessary updates. The assumption of whether a covariate shift has occurred is evaluated using the open-source framework Arangopipe from ArangoDB.[5] Arangopipe determines a data shift value by developing a classifier that measures the accuracy of distinguishing between two datasets.

5 | RESULTS

With the study's foundational conditions established, we proceed to present the results of our research workflow (see Figure 3).

5.1 | Data exploration

In order to get an idea about the distribution of how the price direction changes for the next day, the following figures show histograms of the differences between the closing prices of adjacent days t_n and t_{n+1} for each stock over the period from 01.01.2017 to 31.12.2021. Since it is widely

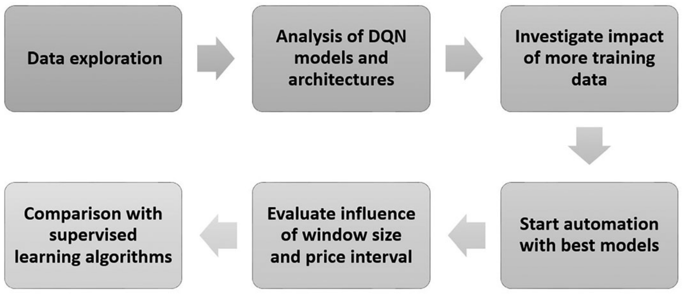
Data exploration
Analysis of DQN models and architectures
Investigate impact of more training data
Start automation with best models
Evaluate influence of window size and price interval
Comparison with supervised learning algorithms

FIGURE 3 Research workflow.

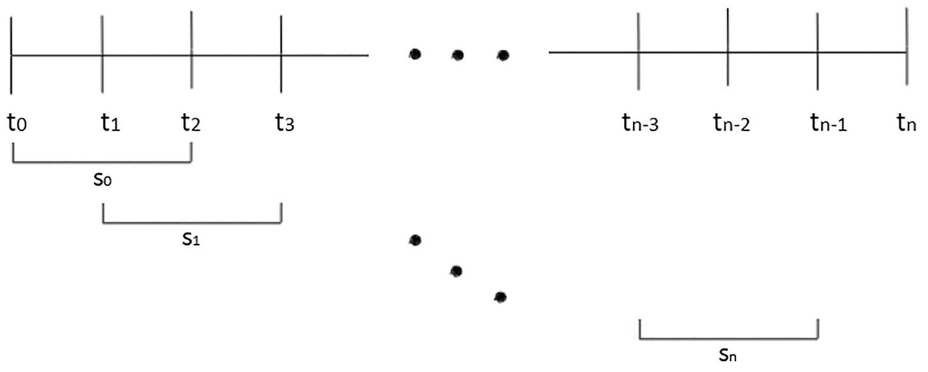

FIGURE 4 Example for self-defined environment with window size = 3.

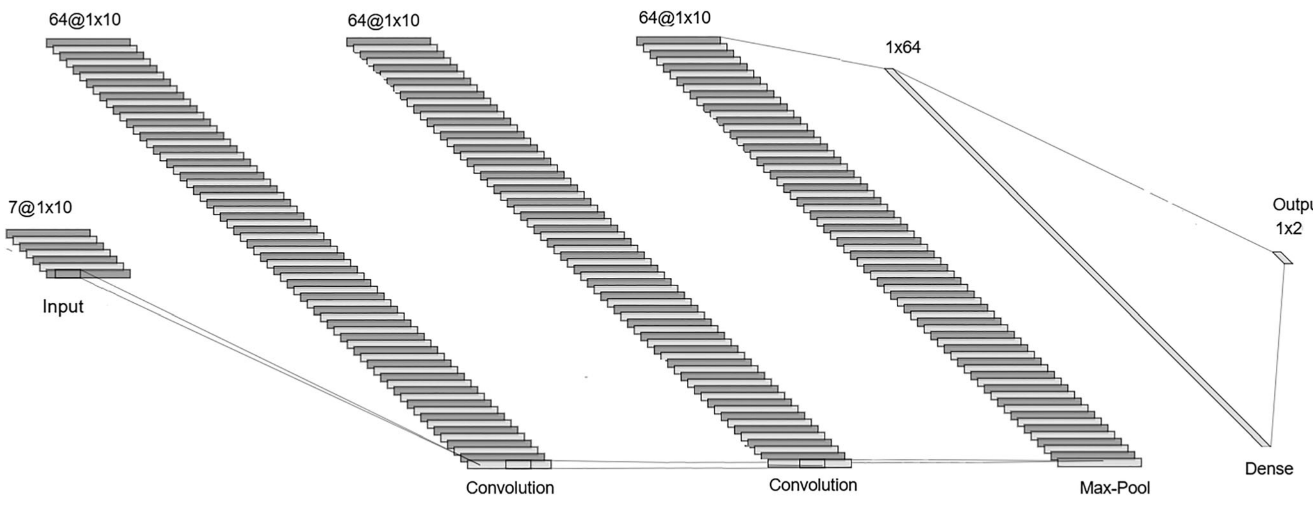

FIGURE 5 Model 1 with window size = 10.

believed that stock returns follow a normal distribution (Li, 2023), a probability density function of a normal distribution $N(\mu, \sigma^2)$ with the estimated sample mean $\bar{x}$ and variance s^2 as parameters is also plotted for each histogram.

From a mathematical point of view, it is intuitive to assume that stock returns follow a normal distribution since they fulfil the conditions of the central limit theorem (the sample size is large enough and is influenced by various factors), but the actual distribution of stock returns often has characteristics such as high peaks and fat tails that make them appear non-normally distributed (Li, 2023).

This can also be observed in our histograms. The samples appear to be normally distributed as most of them are approximately symmetrical around the mean, which is close to 0 for all histograms (see e.g., Figures 11 and 14). However, these histograms also exhibit a pronounced peak at this point, indicating a deviation from normality. Due to these observations, researchers recommend using distributions for stock returns that resemble a normal distribution while incorporating additional characteristics, such as high peaks and fat tails (Li, 2023). One example of such a

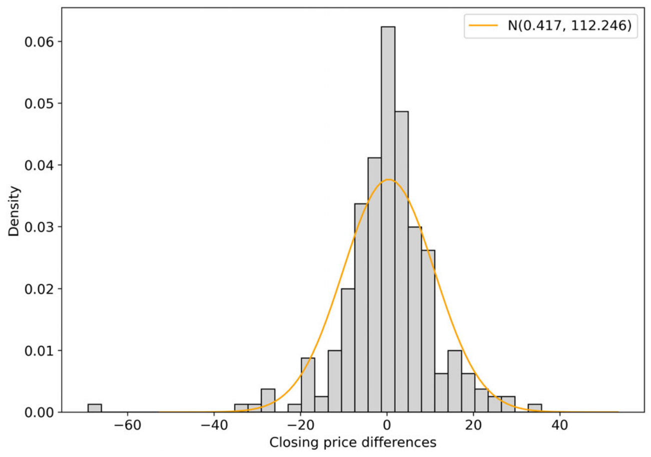

FIGURE 6 Histogram of weekly closing price differences for Adidas with $N(0.417,112.246)$.

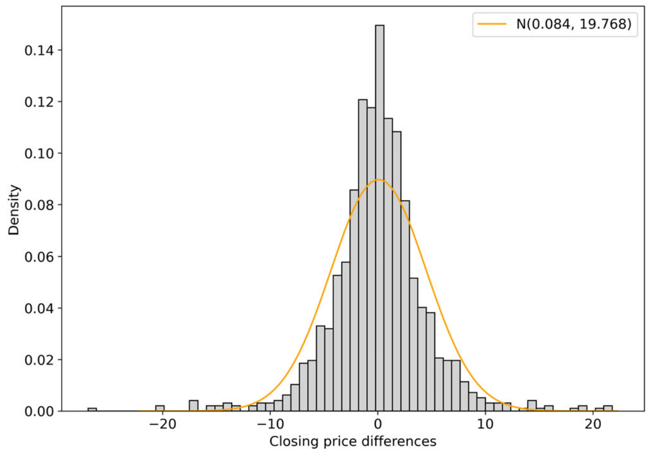

FIGURE 7 Histogram of daily closing price differences for Adidas with $N(0.084,19.786)$.

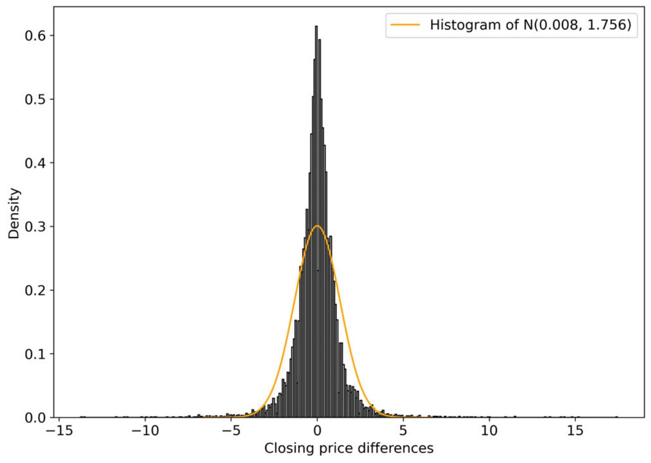

FIGURE 8 Histogram of hourly closing price differences for Adidas with $N(0.008,1.756)$.

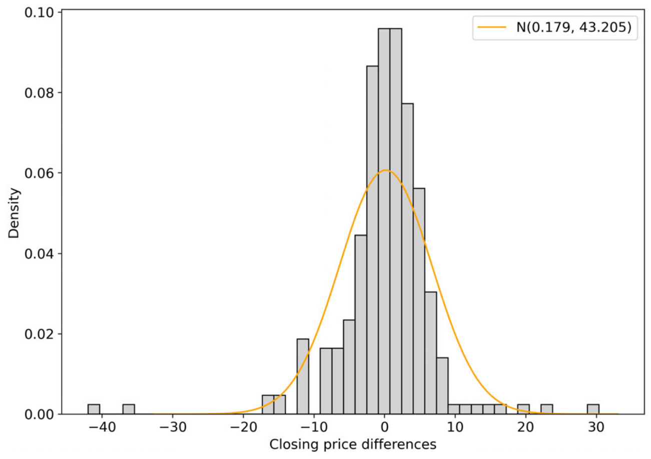

FIGURE 9 Histogram of weekly closing price differences for Allianz with $N(0.179,43.205)$.

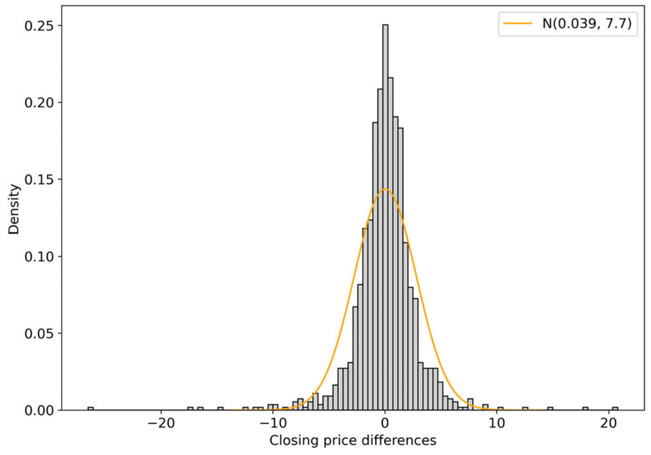

FIGURE 10 Histogram of daily closing price differences for Allianz with $N(0.039,7.7)$.

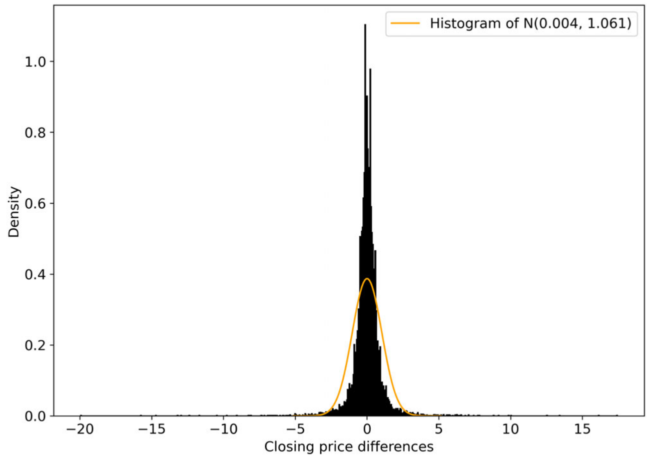

FIGURE 11 Histogram of hourly closing price differences for Allianz with $N(0.004,1.061)$.

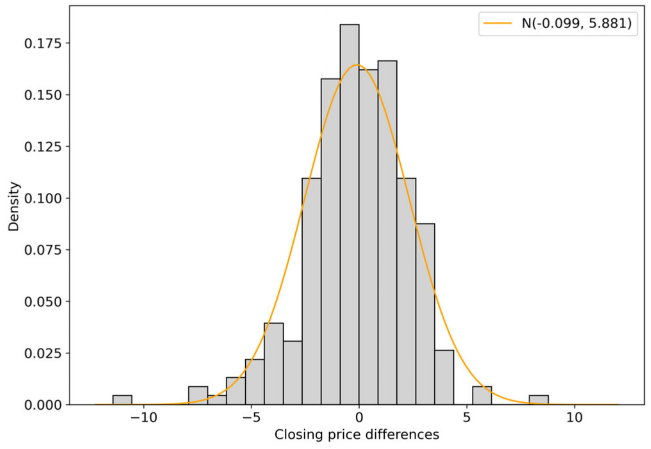

FIGURE 12 Histogram of weekly closing price differences for BASF with $N(-0.099,5.881)$.

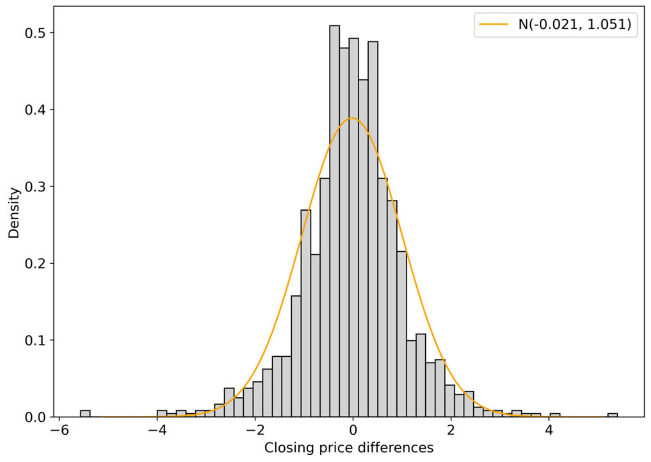

FIGURE 13 Histogram of daily closing price differences for BASF with $N(-0.021,1.051)$.

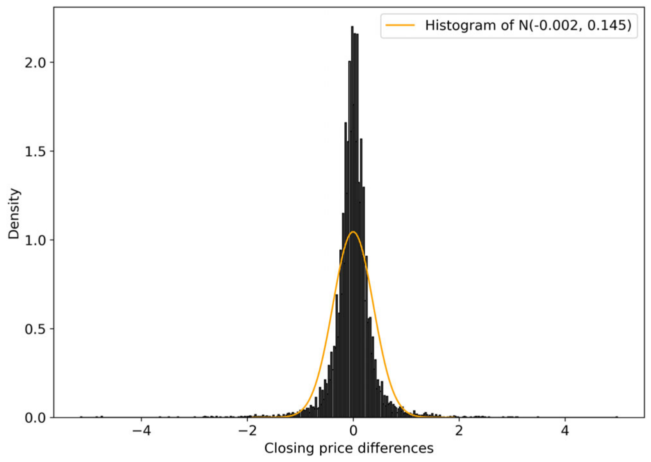

FIGURE 14 Histogram of hourly closing price differences for BASF with $N(-0.002,0.145)$.

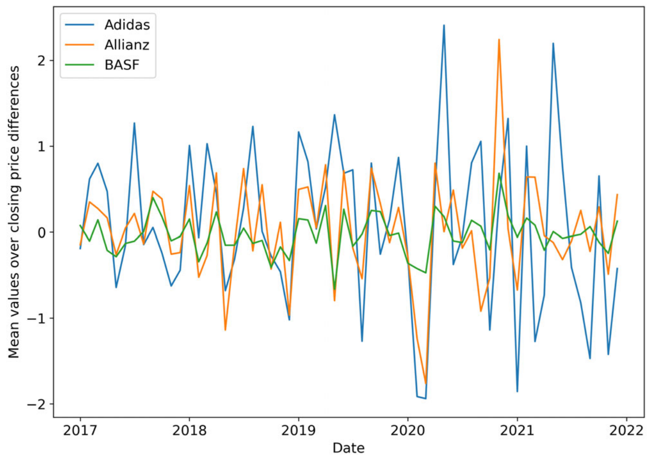

FIGURE 15 Monthly mean values of daily closing price differences.

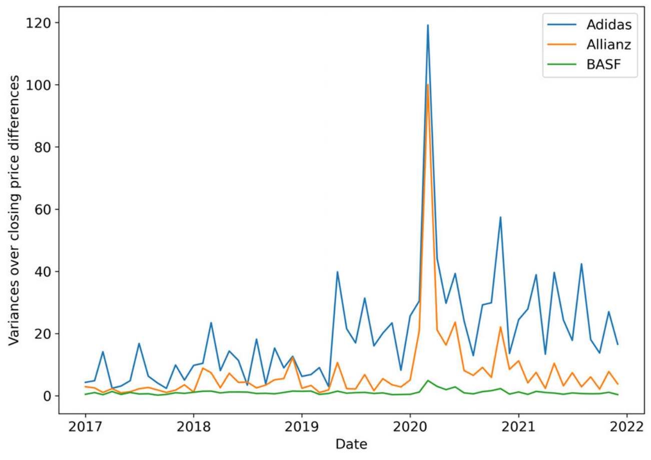

FIGURE 16 Monthly variances of daily closing price differences.

distribution is the exponential power distribution, which better captures these features and thus provides a more realistic modelling of stock returns (Li, 2023).

We hypothesize that by testing a sufficient number of parameters and distributions that are similar to a normal distribution, it is possible to identify a distribution that matches our samples. However, a more critical and challenging task lies in recognizing and modelling the evolving distributions over time. To illustrate the difficulty of this task, the sample means and variances of the daily closing price differences between adjacent days t_n and t_{n+1} are plotted for each month in Figures 15 and 16.

It can be observed that the monthly means fluctuate between the values of -2 and 2, and that the stocks Adidas and Allianz exhibit greater volatility than BASF (see Figure 15). It can also be observed that the mean values over the entire 5-year period (see Figures 7, 10, and 13) are close to 0; however, they can vary significantly from month to month. Similar to the mean values, the variances show that the fluctuations were much greater for the stocks Adidas and Allianz than for BASF (see Figure 16). We can also clearly see that the stocks Adidas and Allianz both suddenly had a very high variance and negative mean value in March 2020, probably due to the announcement by World Health Organization (2020) on 11 March 2020, that the coronavirus has been declared to a global pandemic. This graph shows why it is important to always update the predictive model, as a static model trained before March 2020, for example, would not take this drastic change in distribution into account.

The histograms show that the sample variance decreases with finer price intervals, as demonstrated by comparing Figures 6, 9, and 12 with Figures 8, 11, and 14. This means that the probability of a larger gain or loss increases with the length of time a stock remains in the portfolio. Due to the symmetry of the mean value of the normal distribution-like samples (see e.g., Figures 8 and 10), it can also be concluded that the probability of a positive direction of the price change is approximately the same as that of a negative direction. To verify this, one can look at the proportion of positive and negative price direction changes from the entire data set (Table 5).

It can be seen that the positive proportion of directional changes is close to 50% for almost all stocks and price intervals.

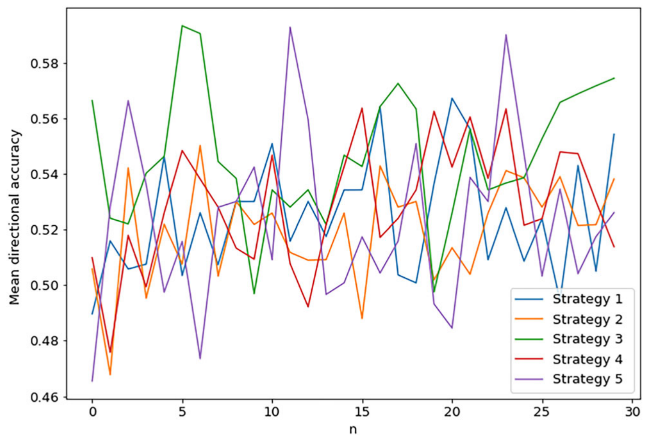

FIGURE 17 Mean directional accuracy regarding the stock Adidas and different update strategies.

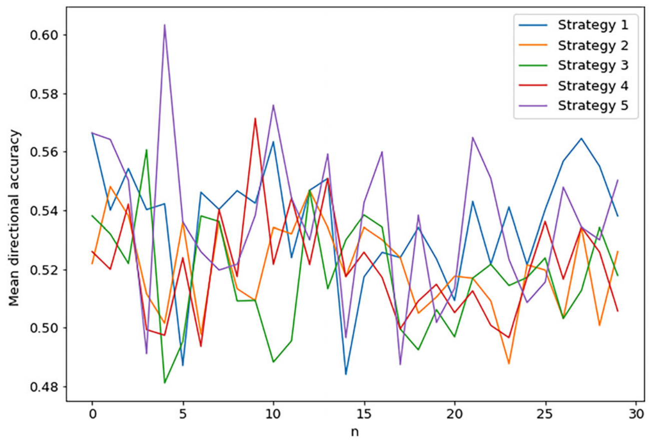

FIGURE 18 Mean directional accuracy regarding the stock Allianz and different update strategies.

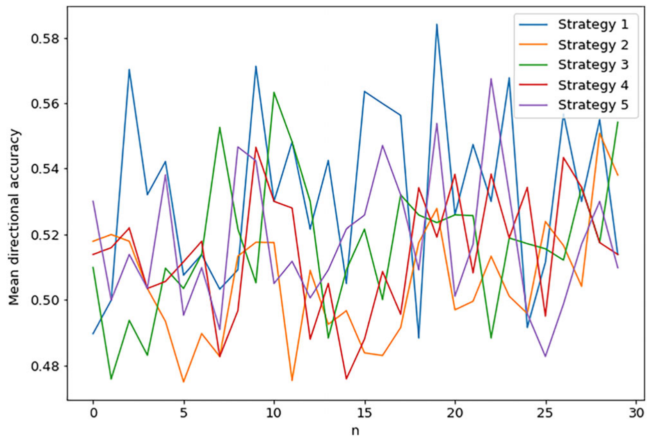

FIGURE 19 Mean directional accuracy regarding the stock BASF and different update strategies.

However, there are also cases, such as the Adidas stock in the weekly interval, where the positive share deviates by around 3%. We suspect that this is due to the amount of data and that the positive proportion of directional changes can fluctuate depending on which smaller slice of a data series is selected. The next step is to investigate how much and if at all our selected Google Trends data can add value to our stock prediction. The following Table 6 shows the Pearson correlation coefficients and their *p*-values between our Google Trends data and the closing prices for all price intervals.

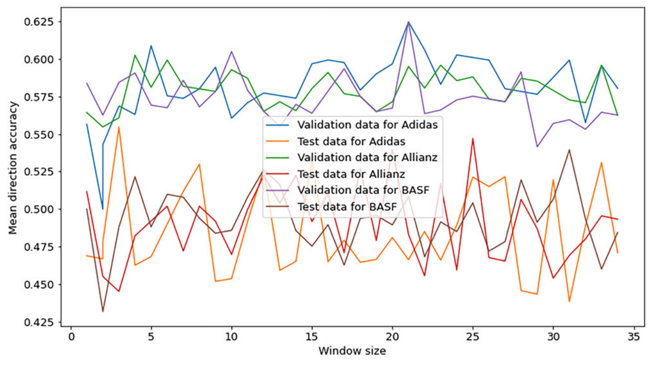

FIGURE 20 Mean directional accuracy regarding different window sizes and daily measured stock data.

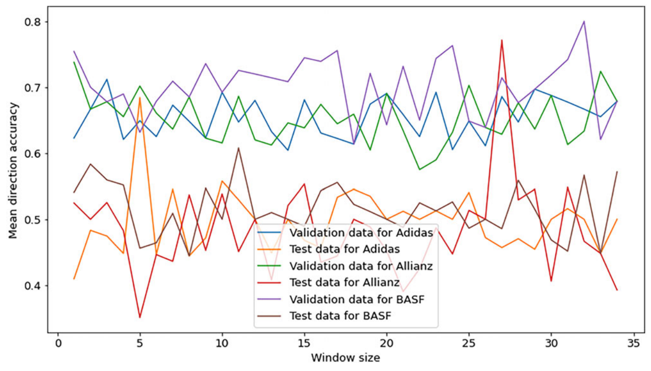

FIGURE 21 Mean directional accuracy regarding different window sizes and weekly measured stock data.

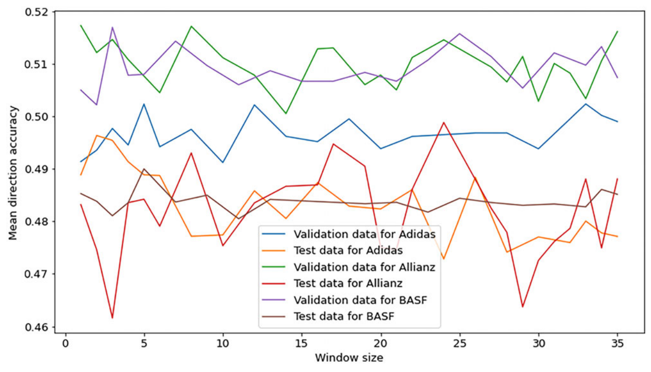

FIGURE 22 Mean directional accuracy regarding different window sizes and hourly measured stock data.

It can be seen that for the daily price data, all Pearson correlation coefficients are significant with a value greater than 0. However, with a significance level of $\alpha = 0.01$ for the weekly and hourly price data, there are Google Trends data that does not correlate significantly with our closing price. The highest correlation can be seen for all price intervals with the keyword 'crash'. It is interesting to see that the price interval of the stock can also influence the direction of the relationship (e.g., for hourly and daily BASF stock data with the keyword 'crash').

The lowest p-values are observed in the hourly data, likely due to the higher volume of data during this period. Overall, these statistics suggest that the Google Trends data, especially for the keyword 'crash', will likely help us increase the explainable variance.

5.2 | First preliminary investigation

This investigation aimed to identify the most effective network architectures for different DQN variants in predicting stock prices. For all possible combinations of network architectures (see Tables 1 and 2) and DQN variants, the following models achieved the best results (see Table 7).

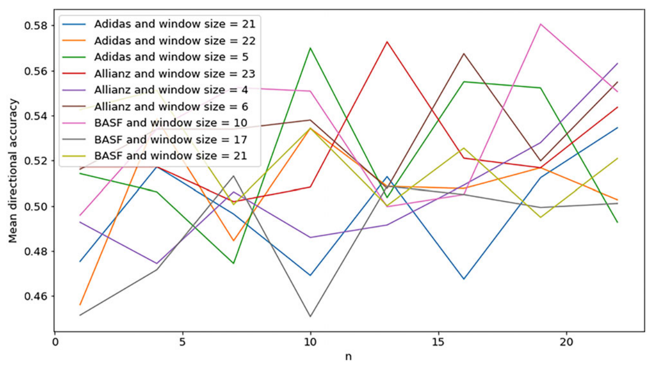

FIGURE 23 Mean directional accuracy of automated agents regarding daily measured stock data and different window sizes.

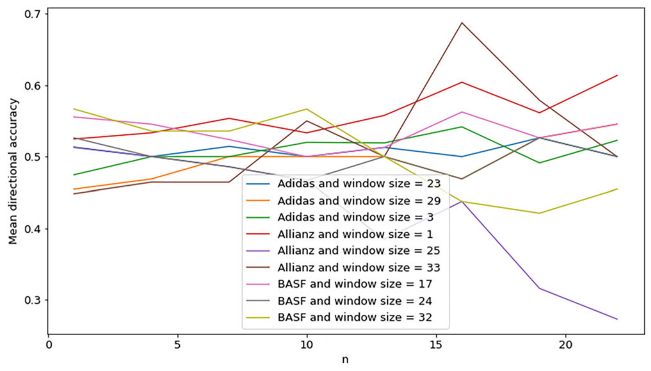

FIGURE 24 Mean directional accuracy of automated agents regarding weekly measured stock data and different window sizes.

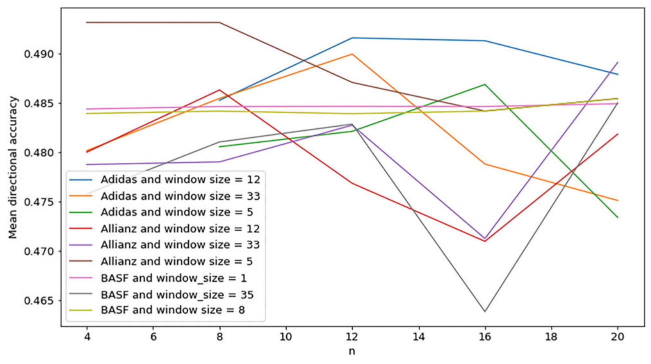

FIGURE 25 Mean directional accuracy of automated agents regarding hourly measured stock data and different window sizes.

The performances of all other model combinations are detailed in Table A1 in the Appendix A. This appendix also contains illustrations showing the training process of the DRQN (see Figures A1 and A2).

5.3 | Second preliminary investigation

This preliminary investigation aims to enhance the results of the best models from the first preliminary investigation by utilizing extended training data. For training data, instead of the period from 01.01.2017 to 31.12.2019, the period from 01.01.2010 to 31.12.2019 (1778 more data records) is now considered. The best models from the first preliminary study were able to achieve the following accuracies using the extended training data set (see Table 8).

The training process from this preliminary investigation can be seen in Appendix B (see Figures B1 and B2).

5.4 | Main investigation

In the main study, agents are first automated using daily stock price data and different strategies. Figures 17–19 thus show the accuracies of the automated agents per strategy and stock at different time steps *n* (time interval per update). The automated agents are evaluated based on the test data used in the preliminary experiments. This allows a comparison of whether the test accuracy, which ranged from 48.59% to 51.41% for the best agents (see Table 7), has improved with the automated implementation. Afterwards, the investigation is extended by examining window sizes from 1 to 35 (see Figures 20–22). Daily and weekly measured price data are now also examined. Then, based on the three best window sizes for each stock and price interval, the agents are automated again by strategy 1 (see Figures 23–25). On the following pages you can see these nine figures, for which a total of several weeks of computing time was required:

5.5 | Comparison with supervised learning algorithms

To compare the effectiveness of automatic agents against conventional methods, we also trained and tested three LSTMs, SVMs, and logistic regression models using the same data split (see section 4.1) and hyperparameters (see Table 3). We selected SVM and logistic regression models

TABLE 5 Overview of the proportion of positive and negative price direction changes.

Stock	Interval	Positive directions	Negative directions	Positive ratio (%)
Adidas	Weekly	122	138	46.9
Allianz	Weekly	129	131	49.6
BASF	Weekly	125	135	48.0
Adidas	Daily	588	646	47.6
Allianz	Daily	605	629	49.0
BASF	Daily	599	635	48.5
Adidas	Hourly	6551	6392	50.6
Allianz	Hourly	6600	6324	51.0
BASF	Hourly	6603	6527	50.2

TABLE 6 Correlation between Google Trends data and stock closing prices.

Stock	Interval	Keyword	Correlation	*p*-Value
Adidas	Weekly	'Adidas'	0.128	0.03846
Allianz	Weekly	'Allianz'	−0.068	0.26930
BASF	Weekly	'BASF'	0.175	0.00442
Adidas	Weekly	'Crash'	−0.177	0.00395
Allianz	Weekly	'Crash'	−0.211	0.00059
BASF	Weekly	'Crash'	0.162	0.00869
Adidas	Daily	'Adidas'	0.113	6.005e-05
Allianz	Daily	'Allianz'	−0.108	1.351–04
BASF	Daily	'BASF'	0.114	5.220e-05
Adidas	Daily	'Crash'	−0.173	8.733e-10
Allianz	Daily	'Crash'	−0.190	1.416e-11
BASF	Daily	'Crash'	0.152	7.834e-08
Adidas	Hourly	'Adidas'	0.101	3.148e-32
Allianz	Hourly	'Allianz'	−0.069	6.765e-16
BASF	Hourly	'BASF'	−0.120	2.871e-44
Adidas	Hourly	'Crash'	0.144	1.032e-63
Allianz	Hourly	'Crash'	−0.018	0.0304
BASF	Hourly	'Crash'	−0.219	2.880e-146

because our target variable is binary (price rises/price falls), and these algorithms are particularly well-suited for binary classification tasks. The LSTM models were initialized with ten LSTM layers, each containing 64 LSTM units and a dropout layer with a dropout rate of 0.4. The supervised learning algorithms achieved the following results:

5.6 | Statistical analysis of the improvement

In order to verify the improved accuracy provided by the automation, the non-parametric test from Pesaran and Timmermann (1992) is applied. This test was chosen because its aim of examining the prediction's ability to forecast the direction of change fits perfectly with our target criterion of maximizing the MDA of our agent environment (see section 4.2). We also do not need a distribution assumption. Our null hypothesis H_0 here is that our model does not have the ability to predict directions of change, whereby the Pesaran-Timmermann test statistic S follows a standard normal distribution. As the test is a right-tailed hypothesis test, the null hypothesis is rejected with a significance level of $\alpha = 0.01$ for S-values >2.3263. Table 11 shows the S-values and p-values for the best static (see Table 7) and automated DQN models (see Table 9).

The p-values clearly show that at a significance level of $\alpha = 0.01$, H_0 can be rejected for the automated models and H_0 can be assumed for the static models. This shows that our methodology improved the results.

5.7 | Comparison with previous systems

In comparison to other studies that have automated their models, we also achieved significant enhancements. Guo et al. (2018) reduced the RMSE by 8%–41% with their adaptive SVR learning algorithm, and Nguyen et al. (2019) achieved a 45.9% reduction in MSE with their dynamic LSTM. In our study, we accomplished an RMSE reduction for our stocks ranging from approximately 8%–12.5% (compare Table 7 with Table 9).

Although our results are not as substantial as those of the other researchers, it is important to note that Guo et al. (2018) and Nguyen et al. (2019) focused on predicting actual stock prices, while we predicted the direction of price movements. Predicting price directions generally offers less potential for significant RMSE improvements compared to predicting continuous price values.

Therefore, it is also relevant to consider studies that aimed to predict price direction without automating their models, such as the study by Nair et al. (2010), which proposed a hybrid decision tree-neuro-fuzzy system. This system uses technical analysis for feature extraction and decision trees for feature selection, followed by dimensionality reduction. The refined features are then applied to an adaptive neuro-fuzzy system for next-day trend prediction. Tested on the BSE-SENSEX, FTSE 100, NASDAQ 100, and NIKKEI 225 indices from January 2003 to March 2010, it achieved a prediction accuracy ranging from 82% to 91.2%.

Compared to our work, their approach achieved significantly higher accuracy, likely due to effective feature selection and reduction. However, our stock data, influenced by the financial crisis induced by the coronavirus, may be more challenging to predict.

TABLE 7 The best models of the performance comparison.

	Best combination		MDA			RMSE		
Stock	Variant	Model	Training (%)	Validation (%)	Test (%)	Training	Validation	Test
Adidas	Duelling DQN	4	86.34	61.24	51.14	0.37	0.62	0.70
Allianz	Duelling DQN	4	91.12	59.68	48.59	0.30	0.63	0.72
BASF	DQN	3	88.94	60.07	48.99	0.33	0.63	0.71

Abbreviations: MDA, mean directional accuracy; RMSE, root mean square error.

TABLE 8 Performance based on the extended training data set.

	MDA			RMSE		
Stock	Training (%)	Validation (%)	Test (%)	Training	Validation	Test
Adidas	50.37	56.29	53.68	0.70	0.66	0.68
Allianz	50.49	55.90	48.62	0.70	0.66	0.72
BASF	56.36	57.48	46.27	0.66	0.65	0.73

Abbreviations: MDA, mean directional accuracy; RMSE, root mean square error.

TABLE 9 Best strategies and update intervals for Figures 17–19.

Stock	Strategy	*n*	Test MDA (%)	Test RMSE
Adidas	3	7	59.32	0.64
Allianz	5	4	60.31	0.63
BASF	1	20	58.41	0.64

Abbreviations: MDA, mean directional accuracy; RMSE, root mean square error.

TABLE 10 Performance of the supervised learning algorithms.

	Test MDA			Test RMSE		
Stock	LSTM (%)	Logistic regression (%)	SVM (%)	LSTM	Logistic regression	SVM
Adidas	50.22	50.39	50.02	0.71	0.70	0.71
Allianz	50.17	51.18	50.01	0.71	0.70	0.71
BASF	49.92	48.81	48.81	0.71	0.72	0.72

Abbreviations: MDA, mean directional accuracy; RMSE, root mean square error.

TABLE 11 Static verification of the improved accuracy.

	Static model			Automated model		
Stock	MDA (%)	*S*-value	*p*-Value	MDA (%)	*S*-value	*p*-Value
Adidas	51.14	0.376	0.353	59.32	3.146	0.0008
Allianz	48.59	−0.413	0.660	60.31	3.261	0.0006
BASF	48.99	−0.317	0.625	58.41	2.595	0.0047

Abbreviation: MDA, mean directional accuracy.

A system specifically tested for financial crises, like the one introduced by Atsalakis et al. (2016), may provide better insights for comparison. This neuro-fuzzy-based system consists of two adaptive neuro-fuzzy inference systems: a controller managing the process model and a process model forecasting the stock price trend for the next day. Tested on stocks listed on the NYSE during four major financial crises, it achieved an average accuracy of 66.54%, with a range of 64.23%–68.5%. Although this system's accuracy is significantly lower than that of Nair et al. (2010), it still surpasses our results, albeit not by a large margin.

Zhong and Enke (2017) aimed to predict the daily direction of the SPDR S&P 500 ETF by utilizing 60 financial and economic factors. To achieve this, they combined three dimensionality reduction techniques, including principal component analysis (PCA) with artificial neural networks (ANNs). Their dataset encompassed the period from June 2003 to May 2013 and was divided into 70% training data (1762 records) and 15% each for test and validation data (378 records each). This study achieved MDAs ranging from 58.1% to 59.2% for the test data. While their results were similar to ours and did not involve automation, their main advantage was the use of significantly more input variables (60 compared to our 7).

In contrast, Kara et al. (2011) employed 10 technical indicators to predict the daily direction of the Istanbul Stock Exchange National 100 Index. The analysis included data from January 1997 to December 2007, which was split into training and test sets, each consisting of 50% of the total 2733 records. The authors utilized SVMs and ANNs, performing an extensive grid search with 900 parameter combinations for SVM and 700 for ANN. The study achieved average directional accuracies of 71.52% for the SVM and 75.74% for the ANN. Although this study demonstrates better MDAs than ours, possibly due to more comprehensive parameter optimization, a full comparison is not feasible because the test data evaluated by the model are different.

Promising systems that predict not only the stock trend but also the actual closing price have been highlighted in the study by Shaban et al. (2024). They introduced the Stock Market Prediction based on Deep Learning (SMP-DL) system, which consists of data preprocessing and a stock price prediction component. The prediction component integrates LSTM and Bidirectional Gated Recurrent Unit models. The study evaluates their model using Apple and Google stock data from March 2020 to April 2022, splitting the dataset into 80% for training and 20% for testing to benchmark it against seven state-of-the-art methods.

The SMP-DL system achieved an RMSE of 0.2883 and an R^2 of 0.9948 in predicting stock prices for the next 10 and 30 min, thus outperforming all other state-of-the-art methods. These other methods had RMSE values ranging from 0.3169 to 0.3638 and R^2 values ranging from 0.9458 to 0.9804.

The second-best method, with an RMSE of 0.3169 and an R^2 of 0.9804, was achieved with a Convolutional Extreme Learning Machine Model with Kernel Support (CKELM) developed by Agarwal et al. (2023). In their work, they demonstrated that this model achieves remarkable performance, with an accuracy of around 98.3%.

These studies indicate that our achieved accuracies are relatively modest, highlighting the need to expand our approach by considering additional factors. Therefore, our results suggest that while we demonstrated the potential for improvement through automation, achieving high accuracies remains challenging. However, these studies also reveal the issue noted in Jiang (2021), that their models do not integrate the forecasting problem with portfolio structuring, making them unsuitable for automated trading.

6 | DISCUSSION

Both the best static DQN models, regardless of whether 7 more years of training data were available, and the supervised learning algorithms failed to achieve test accuracies significantly greater than 50% (see Tables 7, 8 and 10). This may be due to the non-stationary nature of the stock price data. As researchers such as Krawczyk et al. (2020) and Souza et al. (2020) describe, historical price data can be irrelevant or even harmful to forecast models due to shifting data distributions. This is most evident from the fact that 7 more years of training data actually worsened the results (compare Table 7 with Table 8), although the inclusion of more training data is very helpful in most machine learning tasks.

Only the automation of the agents was able to significantly improve the prediction accuracy. We assume that the improvements over the static methods are due to updating the model parameters at specific points when the data distribution changed, whereas static models assume a constant data distribution. For the Adidas stock, the MDA could be improved from 51.14%–59.32% using strategy 3 (see Figure 17) and $n = 7$ (parameter adjustment every 7 days). The second-best MDA of 59.27% was achieved for the Adidas stock using strategy 5 and $n = 11$. Thus, it performed almost as well as strategy 3, despite requiring significantly less computing time. For the Allianz stock, strategy 5 using $n = 4$ was able to improve the MDA from 48.59% to 60.31% (see Figure 18). For the BASF stock, strategy 1 with $n = 20$ improved the MDA from 48.99% to 58.41% (see Figure 19). It can be seen that for each stock the optimal time interval n at which the agents were updated is different (see Table 9). This implies that for each stock the covariate shifts occurred at different times and therefore no optimal time interval n could be specified for all stocks. This may be due to the fact that stock prices can also be affected by individual factors, as has been shown in some studies such as those by Velankar et al. (2017) or Imansyah and Mustafa (2021) (Tables 10 and 11).

However, it is notable that for certain time steps n the test accuracies decreased (see Figures 17–19). This means that if an agent is trained with stock price data from the recent past every n days, for example, it may also become unstable. This is probably because the agents have also been updated unnecessarily, that is, the parameters have been adjusted, although the data distribution has not changed. These considerations imply that answering the question at which intervals DQNs should be updated is related to the question of when covariate shifts occur in time series data.

This indicates a need for further research to determine when data distribution changes in time series occur.

Raza et al. (2013, 2015) have already conducted investigations into this problem and developed noteworthy methods to address it. Raza et al. (2013) demonstrated with their first version of the method, which is based on an exponentially weighted moving average (EWMA) chart, that for a time series with 1000 observations, all time points at which the data distribution changes can be determined for certain smoothing constants λ.

However, their first version of the method had simultaneously resolved a number of false alarms, that is, it had also named time points at which the data distribution remained the same. Accordingly, they extended their method Raza et al. (2015) by additionally validating the time points at which the covariate shifts were suspected using the Kolmogorov–Smirnov test for univariate time series or Hotelling's T-squared test for multivariate time series. In this context, they demonstrated that this adjustment reduced the number of false alarms. Recent works, such as those by Musayev et al. (2024), propose sequential algorithms for detecting changes or anomalies in multivariate time series data. These techniques examine inter-channel connections, utilize dimensionality reduction through singular value decomposition, and show exceptional ability to detect standard anomalies.

Looking at the research of Raza et al. (2013, 2015), it becomes clear why strategy 5 (see Table 4) did not prove to be the best strategy for all stocks. This is because the methodology of the open-source framework Arangopipe is quite simple compared to the methodology of Raza et al. (2015), which itself even raises false alarms. One issue with the methodology from Arangopipe is that it does not check whether the estimated covariate shifts are statistically significant.

In terms of the update strategies (see Table 4), it can also be said that strategies 2 and 4 were the worst on average. Thus, it can be concluded that avoiding unnecessary updates is better for the dynamic process. This illustrates the importance of adjusting the parameters of the models only when the data distribution has actually changed. Because future updates could be negatively affected by useless adjustments in the past.

Regarding how many data points should be considered for parameter adjustment, the results did not provide a clear answer.

It is suspected that taking all previous data points into account is inefficient because computation time increases significantly over time, and not all data are likely relevant due to covariate shifts. It is possible that this question builds on answering the question of at what time points the covariate shifts occur. If it is known at what point in time the characteristics of the data distribution changes, it is possible to determine specific data points that are representative for the market change.

Regarding the influence of the window size on the MDA, similar results could be observed as for the results about the variation of the time step *n*. Some settings for window size proved to be better for some stocks than others (see Figures 20–22). Based on the results, it can further be concluded that higher window sizes need not lead to better results due to the consideration of more data. This is likely due to some data becoming irrelevant because of market dynamics, which consequently have a disruptive effect on the agent. In conclusion, it is recommended to test several window sizes.

It can also be observed that for certain time steps *n* and window sizes, the automated agents were able to improve test accuracy only for stock data measured in weekly and daily intervals (compare Figures 23 and 24 with Figures 20 and 21). For stock price data measured in hours (compare Figure 25 with Figure 22), the agent automation could not improve the results. Since it was generally observed that the results for both the automatic and static DQNs worsened as the price intervals decreased (see Figures 20–22), it is assumed that this deterioration is due to fluctuations in investor sentiment having a greater impact in smaller price intervals, thereby increasing the unexplained variance. This assumption is simultaneously consistent with Shah et al. (2020) claims that the stock market is more predictable and has less noise in the long run. To empirically verify this hypothesis, a follow-up study could examine whether the distribution of stock price data changes more frequently over time for smaller price intervals than for larger price intervals.

It is also evident that the test accuracies decrease for nearly all window sizes compared to the validation accuracies (see Figures 20–22). This is due to the fact that the parameters of the model were adjusted on the basis of the validation data and the test data may have an even less similar distribution to the training data than the validation data due to the existing market dynamics. Guo et al. (2018) also observed this when they realized that the accuracy of their static models had decreased over time.

Nevertheless, it must also be mentioned that our method also has a significant drawback. Our method has a much higher computation time than conventional methods where you train the model only once based on training data. In our method, the parameters of the model are adjusted *n* times, which increases the effort. And this higher effort does not always lead to better results, as we can also make unnecessary updates due to poor methods for determining covariate shifts or biased data. The comparison with previous systems (see section 5.7) has also shown that our results are relatively modest, highlighting the need to expand our approach by considering additional factors.

Furthermore, a serious limitation of our study is the lack of previous research studies on our exemplary defined method and selected data. This makes our work less comparable. Further research is therefore needed in this area. Another limitation is that not many more stocks could be analysed, which might have made the results more accurate and representative. The problem with many more stocks would have been that the calculation times for our intensive investigations would then no longer have been acceptable, as the calculation times for our few stocks already amounted to several weeks despite the use of the chargeable TPU from Google Colab Pro+.[6] However, future studies would not need to take so long, as their studies can build on ours, as they do not need to perform all of our preliminary research and test variants in terms of update strategies, window sizes or stock data intervals. For example, they would not have to update the model every *n* days and try different strategies, but only when covariate shifts occur.

During the course of the study, some notable additional insights were also gained, such as that Duelling DQNs can lead to good results when combined with a self-normalizing network architecture, or that the benefits of DRQNs do not necessarily outperform those of DQNs. During training, the DRQNs actually exhibited negative reward sums (see Figure A1), unlike the DQNs.

It is possible that the ability of the LSTM model to remember information over time could cause the DRQN models to remember irrelevant information due to covariate shifts and noise. Approaches to the correctness of the assumption can be validated by studies such as that of Elliot and Hsu (2017). They showed that the martingale model, whose prediction of the next value is based solely on the current value, outperforms the LSTM models in predicting the S&P 500 index.

7 | CONCLUSIONS AND FUTURE WORK

Automating machine learning models demonstrates significant potential for improving stock price forecasting accuracy, as demonstrated by this study and other related research. This highlights the need for models to dynamically adjust to market fluctuations, emphasizing the critical importance of timely updates.

After thorough analysis and discussion of our findings, we concluded that updating models at fixed intervals may not effectively capture market dynamics. This inefficiency arises because data distributions can shift unpredictably, and unnecessary adjustments to parameters can destabilize the model, leading to poor performance. Therefore, we recommend an adaptive update interval. Adjusting parameters only when significant shifts in covariates are detected is crucial for improving the accuracy of stock price predictions. Accordingly, there is a need to combine rapidly adaptable models, such as DQNs, to capture changing market dynamics with methods to determine when covariate shifts occur. Methods such as those of Raza et al. (2015) can be used, in which statistical procedures are additionally used to test whether the assumption of a shift in the data distribution is significant, or as in Musayev et al. (2024), where sequential algorithms are employed to incorporate inter-channel connections in multivariate time series. If the exact timing of market changes can be determined, timely decisions can be made regarding the initiation of adjustments.

Building on this, our approach should be further explored by incorporating additional stocks and features. This will enhance the understanding of automated deep reinforcement models as this field of research continues to expand. As more results emerge, we will gain a clearer insight into their potential.

As mentioned in the discussion, our methodology can be improved by employing more efficient methods to detect covariate shifts. This could significantly enhance the results by further reducing unnecessary updates and increasing the frequency of necessary updates when covariate shifts occur. In our opinion, this is the most crucial aspect for improving this method.

We do not dismiss the possibility that more efficient methods for automating the models exist; however, updating the parameters when a significant covariate shift occurs appears to be the most plausible approach. Therefore, we encourage other researchers to explore alternative methods if they have innovative ideas.

For future research, we recommend incorporating additional characteristics to reduce high unexplained variance, specifically by including behavioural datasets. Many researchers (Breaban & Noussair, 2018; Duxbury et al., 2020; Jin et al., 2019) emphasize the importance of investor sentiment in stock price prediction. We believe that the detection of covariate shifts can become more accurate with the inclusion of more behavioural datasets. Additionally, our approach can be enhanced with further methodological components and integrated into a larger system, potentially utilizing multi-agent systems as described by Li and Hai (2024).

After developing an efficient dynamic model, we recommend avoiding the mistake of evaluating the profit and risk of the trading strategy based solely on prediction results, as other studies have done according to Jiang (2021). Instead, we should leverage the advantage of reinforcement learning described by Singh et al. (2022), which integrates the prediction problem with the portfolio structuring task. This approach allows us to account for factors such as transaction costs, which, in combination with a robust adaptive model, leads to successful automated trading.

ACKNOWLEDGMENT

Open Access funding enabled and organized by Projekt DEAL.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Said Yasin https://orcid.org/0000-0002-0230-7095
Jamal Al Qundus https://orcid.org/0000-0002-8848-1632

ENDNOTES

1 https://pypi.org/project/yfinance/ .

2 https://pypi.org/project/pytrends/ .

3 https://www.gymlibrary.dev/content/environment_creation/ .

4 https://tensorforce.readthedocs.io/en/latest/ .

5 https://www.arangodb.com/2020/11/arangoml-part-4-detecting-covariate-shift-in-datasets/ .

6 https://colab.research.google.com/signup .

REFERENCES

Agarwal, V., Ravi Kumar, P., Shankar, S., Praveena, S., Dubey, V., & Chauhan, A. (2023). A deep convolutional kernel neural network based approach for stock market prediction using social media data. In *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 78–82). IEEE (Institute of Electrical and Electronics Engineers).

Alpaydin, E. (2022). *Maschinelles Lernen* (3rd ed.). De Gruyter Oldenbourg.

Amiri, R., Mehrpouyan, H., Fridman, L., Mallik, R. K., Nallanathan, A., & Matolak, D. (2018). A machine learning approach for power allocation in hetnets considering qos. In *2018 IEEE International Conference on Communications (ICC)*. IEEE (Institute of Electrical and Electronics Engineers).

Atsalakis, G. S., Protopapadakis, E. E., & Valavanis, K. P. (2016). Stock trend forecasting in turbulent market periods using neuro-fuzzy systems. *Operational Research*, *16*, 245–269.

Bajpai, S. (2021). Application of deep reinforcement learning for indian stock trading automation. *ArXiv*.

Breaban, A., & Noussair, C. N. (2018). Emotional state and market behavior. *Review of Finance*, *22*(1), 279–309.

Burkov, A. (2019). *Machine Learning kompakt: Alles, was Sie wissen müssen* (1st ed.). Mitp Verlag.

Camacho, J. D., Villaseñor, C., Alanis, A. Y., Lopez-Franco, C., & Arana-Daniel, N. (2019). Kadam: Using the kalman filter to improve adam algorithm. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Springer International Publishing.

Chollet, F. (2018). *Deep Learning mit Python und Keras Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek* (1st ed.). Mitp Verlag.

Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *ArXiv*. arXiv:1511.07289.

Dauphin, J. F., Dybczak, K., Maneely, M., Sanjani, M. T., Suphaphiphat, N., Wang, Y., & Zhang, H. (2022). Scalable approach using dfm, machine learning and novel data, applied to european economies. In *Nowcasting GDP* (Vol. 2022, p. 1). International Monetary Fund.

de Lima e Silva, P. C., Severiano, C. A., Alves, M. A., Silva, R., Weiss Cohen, M., & Guimarães, F. G. (2020). Forecasting in non-stationary environments with fuzzy time series. *Applied Soft Computing*, *97*, 106825.

Ditzler, G., Roveri, M., Alippi, C., & Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, *10*(4), 12–25.

Du, Y., Wang, J., Feng, W., Pan, S., Qin, T., Xu, R., & Wang, C. (2021). Adarnn: Adaptive learning and forecasting of time series. *ArXiv*. arXiv:2108.04443.

Duxbury, D., Gärling, T., Gamble, A., & Kla, V. (2020). How emotions influence behavior in financial markets: A conceptual analysis and emotion-based account of buy-sell preferences. *The European Journal of Finance*, *26*(14), 1417–1438.

Elliot, A., & Hsu, C. H. (2017). Time series prediction: Predicting stock price. *ArXiv*. arXiv:1710.05751.

Franceschetti, M., Lacoux, C., Ohouens, R., Raffin, A., & Sigaud, O. (2022). Making reinforcement learning work on swimmer. *ArXiv*. arXiv:2208.07587.

Gandhmal, D. P., & Kumar, K. (2019). Systematic analysis and review of stock market prediction techniques. *Computer Science Review*, *34*, 100190.

Gao, L. (2024). Comparison of dqn and double dqn reinforcement learning algorithms for stock market prediction.

Gao, Z., Gao, Y., Hu, Y., Jiang, Z., & Su, J. (2020). Application of deep q-network in portfolio management. In *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)* (pp. 268–275). IEEE (Institute of Electrical and Electronics Engineers).

Géron, A. (2019). *Hands-on machine learning with Scikit-learn, Keras, and TensorFlow concepts, tools, and techniques to build intelligent systems* (2nd ed.). O'Reilly Media.

Gu, Y., Shibukawa, T., Kondo, Y., Nagao, S., & Kamijo, S. (2020). Prediction of stock performance using deep neural networks. *Applied Sciences*, *10*(22), 8142.

Guo, Y., Han, S., Shen, C., Li, Y., Yin, X., & Bai, Y. (2018). An adaptive svr for high-frequency stock price forecasting. *IEEE Access*, *6*, 11397–11404.

Hausknecht, M., & Stone, P. (2017). Deep recurrent q-learning for partially observable MDPS. *ArXiv*. arXiv:1507.06527.

Hodson, T. O. (2022). Root-mean-square error (RMSE) or mean absolute error (MAE): When to use them or not.

Huyen, C. (2022). *Designing machine learning systems* (1st ed.). O'Reilly.

Imansyah, S., & Mustafa, M. H. (2021). The analysis of financial ratios effect on the stock price of consumer goods sector companies listed in kompas100 index. *Dinasti International Journal of Digital Business Management*, *2*(2), 371–383.

Jiang, W. (2021). Applications of deep learning in stock market prediction: Recent progress. *Expert Systems with Applications*, *184*, 115537.

Jin, Z., Yang, Y., & Liu, Y. (2019). Stock closing price prediction based on sentiment analysis and lstm. *Neural Computing and Applications*, *32*, 9713–9729.

Kara, Y., Acar, M., & Baykan, O. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul stock exchange. *Expert Systems with Applications*, *38*(5), 5311–5319.

Karunakaran, D., Worrall, S., & Nebot, E. (2020). Efficient statistical validation with edge cases to evaluate highly automated vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)* (pp. 1–8).

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *ArXiv*. arXiv:1412.6980.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. *ArXiv*. arXiv:1706.02515.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Woźniak, M. (2020). Ensemble learning for data stream analysis: A survey. *Information Fusion*, *37*, 132–156.

Kurani, A., Doshi, P., Vakharia, A., & Sha, M. (2021). A comprehensive comparative study of artifcial neural network (ann) and support vector machines (svm) on stock forecasting. *Annals of Data Science*, *10*, 183–208.

Lapan, M. (2020). *Deep reinforcement learning: Das umfassende Praxis-Handbuch* (1st ed.). Mitp Verlag.

Li, B. (2023). An explanation for the distribution characteristics of stock returns. *ArXiv*. arXiv:2312.02472.

Li, H., & Hai, M. (2024). Deep reinforcement learning model for stock portfolio management based on data fusion. *Neural Processing Letters*, *56*, 108.

Li, Y., Ni, P., & Chang, V. (2020). Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing*, *102*, 1305–1322.

Lin, G., & Shen, W. (2018). Research on convolutional neural network based on improved ReLU piecewise activation function. *Procedia Computer Science*, *131*, 977–984.

Liu, T., Ma, X., Li, S., Li, X., & Zhang, C. (2022). A stock price prediction method based on meta-learning and variational mode decomposition. *Knowledge-Based Systems*, *252*, 109324.

Mnih, V., Koray Kavukcuogl, D. S., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *ArXiv*. arXiv:1312.5602.

Musayev, A., Grigoriev, D., & Makshanov, A. (2024). Exploring sequential algorithms for anomaly detection in multivariate time series. In *2024 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)* (pp. 1050–1055). IEEE (Institute of Electrical and Electronics Engineers).

Nair, B. B., Dharini, N. M., & Mohandas, V. (2010). A stock market trend prediction system using a hybrid decision tree-neuro-fuzzy system. *2010 International Conference on Advances in Recent Technologies in Communication and Computing*, 381–385. IEEE (Institute of Electrical and Electronics Engineers).

Nguyen, D. H. D., Tran, L. P., & Nguyen, V. (2019). Predicting stock prices using dynamic lstm models. *Applied Informatics*, *1051*, 199–212.

Pesaran, H., & Timmermann, A. (1992). A simple nonparametric test of predictive performance. *Journal of Business and Economic Statistics*, *10*, 461–465.

Raza, H., Prasad, G., & Li, Y. (2013). Dataset shift detection in non-stationary environments using ewma charts. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE (Institute of Electrical and Electronics Engineers).

Raza, H., Prasad, G., & Li, Y. (2015). Ewma model based shift-detection methods for detecting covariate shifts in non-stationary environments. *Pattern Recognition*, *48*(3), 659–669.

Selvamuthu, D., Kumar, V., & Mishra, A. (2019). Indian stock market prediction using artificial neural networks on tick data. *Financial Innovation*, *5*(16), 16.

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, *90*, 106181.

Shaban, W. M., Ashraf, E., & Slama, A. E. (2024). Smp-dl: A novel stock market prediction approach based on deep learning for effective trend forecasting. *Neural Computing and Applications*, *36*, 1849–1873.

Shah, D., Campbell, W., & Zulkernine, F. H. (2018). A comparative study of lstm and dnn for stock market forecasting. *2018 IEEE International Conference on Big Data (Big Data)*, 4148–4155.

Shah, D., Isa, H., & Zulkernine, F. (2020). Stock market analysis: A review and taxonomy of prediction techniques. *International Journal of Financial Studies*, *7*(2), 1–22.

Shi, Y., Li, W., Zhu, L., Guo, K., & Cambria, E. (2021). Stock trading rule discovery with double deep q-network. *Angewandte Soft Computing*, *107*, 107320.

Singh, V., Chen, S. S., Singhania, M., Nanavati, B., Kumar Kar, A., & Gupta, A. (2022). How are reinforcement learning and deep learning algorithms used for big data based decision making in financial industries—a review and research agenda. *International Journal of Information Management Data Insights*, *2*(2), 100094.

Souza, V. M. A., dos Reis, D. M., Maletzke, A. G., & Batista, G. E. A. P. A. (2020). Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, *34*, 1805–1858.

Stewart, W. J. (2009). *Probability, Markov chains, queues, and simulation: The mathematical basis of performance modeling* (1st ed.). Princeton University Press.

Thakkar, A., & Chaudhari, K. (2021). A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions. *Expert Systems with Applications*, *177*, 114800.

van Hasselt, H., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double q-learning. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (pp. 2094–2100).

Velankar, N., Chandani, A., & Ahuja, A. K. (2017). Impact of EPS and DPS on stock price: A study of selected public sector banks of India. *Prestige International Journal of Management and IT & Sanchayan*, *6*(1), 111–121.

Wang, R., Dong, Y., Arik, S. Ö., & Yu, R. (2023). Koopman neural forecaster for time series with temporal distribution shifts. *arXiv*. arXiv:2210.03675. https://arxiv.org/abs/2210.03675

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. *arXiv*. arXiv:1511.06581.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards* (PhD thesis). King's College.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.

World Health Organization. (2020). WHO Director-General's opening remarks at the media briefing on COVID-19 - 11 March 2020. https://www.who.int/director-general/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19-11-march-2020

Zhang, J., & Lei, Y. (2022). Deep reinforcement learning for stock prediction. *Scientific Programming*, *2022*, 1–9.

Zhang, Q., Liu, Z., Wen, Z., Huang, D., & Xu, W. (2023). Cwa-lstm: A stock price prediction model based on causal weight adjustment. In *Advanced Intelligent Computing Technology and Applications: 19th International Conference, ICIC 2023* (pp. 421–432). Springer Nature Singapore.

Zhang, Y., Chen, C., Shi, N., Sun, R., & Luo, Z. Q. (2022). Adam can converge without any modification on update rules. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in neural information processing systems* (Vol. 35). Curran Associates Inc.

Zhong, X., & Enke, D. (2017). Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications*, *67*, 126–139.

AUTHOR BIOGRAPHIES

Said Yasin received his Master's degree in Data Science from the Freie Universität in 2022. Since then, he has worked in the financial sector, focusing on financial risk analysis. Alongside his professional work, he conducts research in natural language processing and computer vision. His research interests also include time series analysis and deep reinforcement learning.

Adrian Paschke Since 2008 Prof. Dr. rer. nat. Adrian Paschke is head of the Corporate Semantic Web group (AG-CSW) with a chair on semantic data intelligence at the institute of computer science, department of mathematics and computer science at Freie Universität Berlin (FUB). Since 2015 he additionally is director of the Data Analytics Center (DANA) at Fraunhofer FOKUS. He also is director of RuleML Inc. in Canada, and professorial member at the Einstein Center Digital Future (ECDF), the Dahlem Center for Machine Learning and Robotics (DCMLR), the Institut für Angewandte Informatik (InfAI) at University of Leipzig.

Jamal Al Qundus (Ph.D.) is an assistant Professor at Management Science/Business Intelligence and Data Analytics (BIDA). Dr. Jamal Al Qundus has pursued his PhD in Computer Science from the Free University of Berlin (FUB). He worked at Fraunhofer FOKUS Berlin in the field of Content and Data Curation. His research activities focus on knowledge and information interpretation in Artificial Intelligence (AI); Corporate Smart Insight Systems and Knowledge-Generation.

How to cite this article: Yasin, S., Paschke, A., & Al Qundus, J. (2025). Prediction of stock prices with automated reinforced learning algorithms. *Expert Systems*, *42*(2), e13725. https://doi.org/10.1111/exsy.13725

APPENDIX A

A.1 | FIRST PRELIMINARY INVESTIGATION

TABLE A1 Validation accuracies of all possible model combinations.

	Adidas			Allianz			BASF		
Model	DQN (%)	Double DQN (%)	Duelling DQN (%)	DQN (%)	Double DQN (%)	Duelling DQN (%)	DQN (%)	Double DQN (%)	Duelling DQN (%)
Model 1	55.81	53.10	53.87	54.65	56.58	56.91	54.26	56.20	55.03
Model 2	56.97	56.17	56.20	57.75	57.36	58.13	50.77	57.21	57.25
Model 3	57.36	58.52	56.97	59.68	56.20	59.38	60.07	58.52	59.93
Model 4	59.30	58.13	61.24	57.36	58.91	59.68	56.97	56.20	56.97

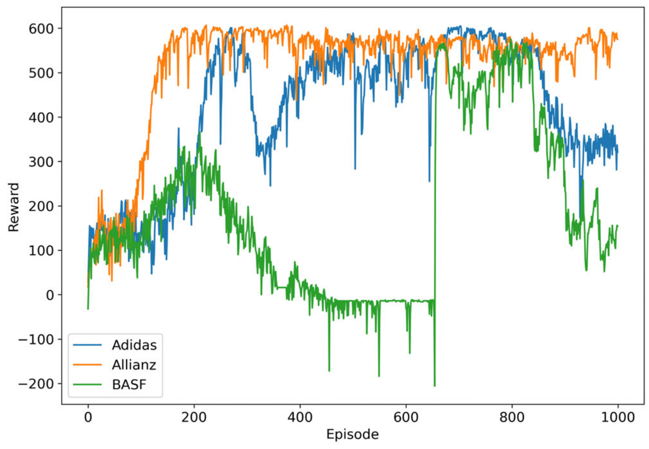

FIGURE A1 Reward during training for the DRQNs.

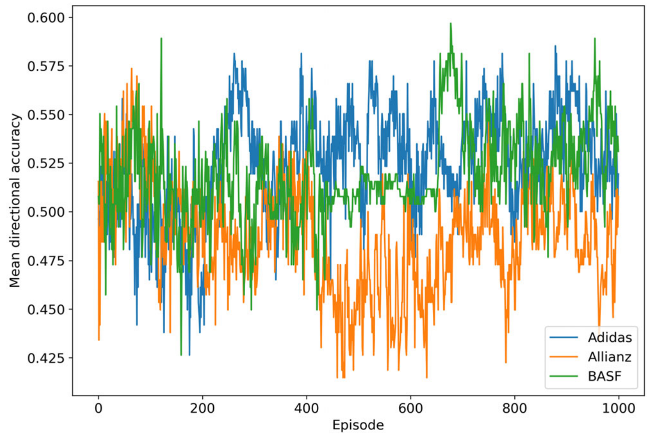

FIGURE A2 Mean directional accuracy during validation for the DRQNs.

APPENDIX B

B.1 | SECOND PRELIMINARY INVESTIGATION

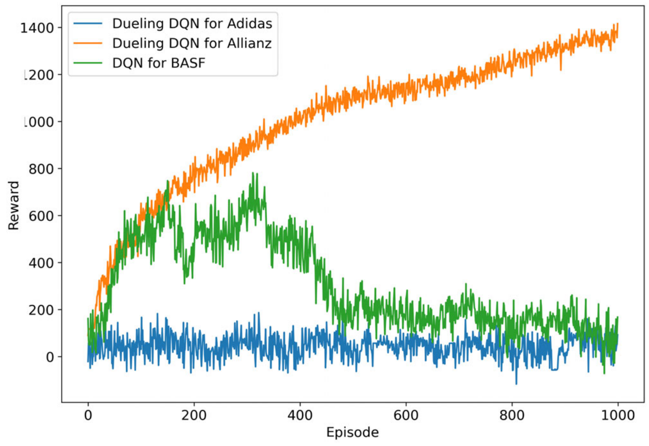

FIGURE B1 Reward during training with the extended training data and the best models.

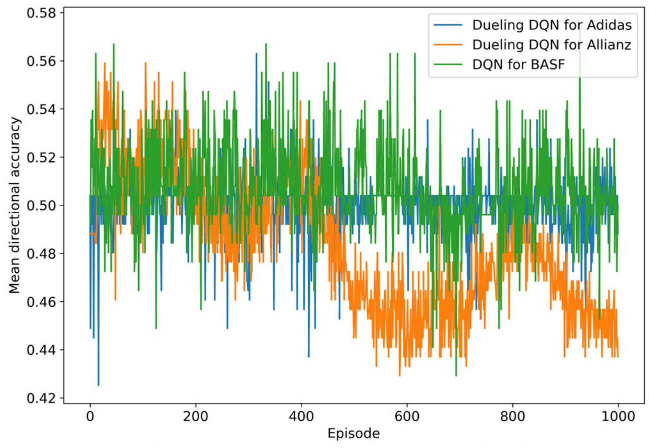

FIGURE B2 Mean directional accuracy during validation with the extended training data and the best models.