Prediction of stock prices with automated reinforced learning algorithms

Said Yasin[1], Adrian Paschke[1], and Jamal Al Qundus[1]

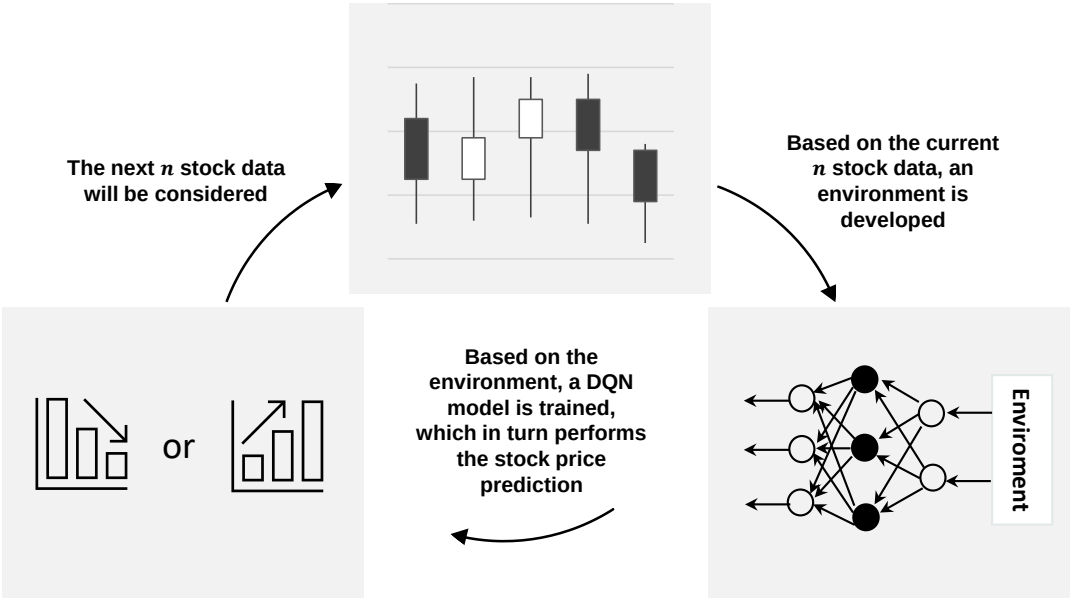
[1]Freie Universitat Berlin Institut fur Informatik

April 05, 2024

Graphical Abstract

Prediction of stock prices with automated reinforced learning algorithms

Said Yasin, Adrian Paschke, Jamal Al Qundus



Highlights

Prediction of stock prices with automated reinforced learning algorithms

Said Yasin, Adrian Paschke, Jamal Al Qundus

- Compared to ordinary supervised learning algorithms, the prediction accuracy could be increased by 10 %
- Unnecessary updates can have a negative impact on performance
- The Dueling DQN performed on average better than the Double DQN and the DQN

Prediction of stock prices with automated reinforced learning algorithms

Said Yasin[b], Adrian Paschke[a,b], Jamal Al Qundus[a,b,c]

[a]*Data Analytics Center, Fraunhofer FOKUS, Berlin, 10589, Germany*
[b]*Institute of Computer Science, Freie Universitaet Berlin, Berlin, 14195, Germany*
[c]*Business Intelligence and Data Analytics, German Jordanian University, Amman, 11180, Jordan*

Abstract

The prediction of stock price developments represents one of the greatest challenges in the analysis of time series data and has been investigated for decades using a wide variety of methods. Until now, only a few approaches have been able to establish themselves. The reasons for this vary. One of the challenges in stock price forecasting is that the desired model changes over time due to market dynamics. The objective of this paper is to predict stock prices using automated reinforcement learning algorithms. The models are automated by repeatedly retraining them using data from the recent past. Our approach is designed to capture the market dynamics. DQN models and their modified variants are used as the reinforcing learning algorithms. They have shown remarkable results in recent studies in stock price forecasting. In this paper we were able to show that for different DAX stocks, measured in daily intervals, the test accuracy can improve from 50.00 % to about 60 %. It was shown that for certain step sizes n and window sizes an improvement of the results can be achieved by a dynamic implementation. However, this only holds for stock price data that is measured at daily and weekly intervals.

Keywords: stock price prediction, reinforcement learning, automation, deep Q-learning

Email addresses: said_yasin1234@hotmail.com (Said Yasin), adrian.paschke@fokus.fraunhofer.de (Adrian Paschke), jamal.alqundus@gju.edu.jo (Jamal Al Qundus)

1. Introduction

The stock market is a market segment that has continuously increased in value over the past decades. The value of the world's stock portfolio grew from 2.9 trillion to 105.9 trillion in the period from 1980 to 2020 (Bundeszentrale für politische Bildung, 2021). In the course of this, investors interest in trading on the stock market and consequently how to predict the future stock price increased. After all, successfully predicting stock prices can lead to attractive profits if the right decisions are made (Gandhmal and Kumar, 2019). Thus, predicting stock price trends emerged as one of the major challenges in analyzing time series data (Gu et al., 2020). According to Gu et al. (2020) and Jiang (2021), a number of approaches, such as the use of machine learning techniques, have been investigated in recent decades to overcome this challenge. Despite all this research only a few models could successfully establish themselves in automatic stock trading (Gu et al., 2020).

In this context, predicting stock prices proved to be extremely difficult due to the non-stationary and chaotic nature of stock data (Gandhmal and Kumar, 2019). For example, the non-stationarity of the data means that the desired model for predicting stock prices changes over time due to the non-uniform distribution of the data (Chollet, 2018; Raza et al., 2015). Researchers such as Guo et al. (2018) were able to prove this empirically when they observed that their selected stocks had different distributions at different times. This phenomenon can be illustrated graphically in figure 1.

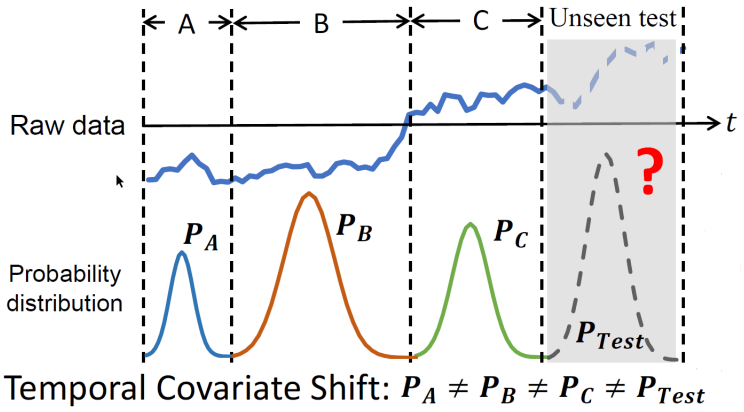


Figure 1: Covariate shift Source: Du et al. (2021)

This problem, according to Krawczyk et al. (2020) and Souza et al. (2020), raises the issue that historical data may not be relevant or even detrimental to the modeling of the forecast model. Despite this described problem, according to Du et al. (2021) there has been little research modeling time series from a distributional perspective. To address this problem of distribution of data changes over time, researchers such as the developer of the deep learning library Keras Chollet (2018) recommend that the required model be continually re-trained using data from the recent past. This is because static models trained using non-stationary data become unusable over time (Ditzler et al., 2015). In this respect, the existing market dynamics suggest that static models are not the appropriate models to capture changes in the market in the long run.

Against this background, this paper will deal with automated ML-models to address this problem of dynamic data changes over time. In our solution Deep Q-Networks (DQNs) developed by Mnih et al. (2013) are used because the adaptability and enormous performance demonstrated in works such as Thakkar and Chaudhari (2021), Bajpai (2021) or Zhang and Lei (2022) of the deep reinforcement models in combination with a dynamic implementation, could lead to remarkable results. In the study by Thakkar and Chaudhari (2021), the DQN was able to achieve an accuracy of almost 100%, while many other models, including LSTMs, RNNs and CNNs, had an approximate accuracy of 50%. These results are also confirmed by reviews such as that of Singh et al. (2022), which conclude that deep reinforcement approaches perform better than classical ML approaches, e.g. due to their high-level feature extraction property. Researchers such as Li et al. (2020) also confirmed that the theory of deep reinforcement learning is now widely accepted on the financial markets, but they pointed out that there are still some challenges to overcome, such as the exploration and exploitation of balance problem, slow convergence rate, space catastrophe and so on.

Thus, automated reinforcement learning has the potential to better adapt to market fluctuations and learn to make optimal decisions in uncertain environments. So far, supervised learning algorithms have been preferred for stock price forecasts (Shi et al., 2021). As a consequence, reinforcement learning algorithms have received less attention and, in particular, not in combination with an automated implementation.

No other study could be found in which DQN models are automated to adapt to the changing characteristics of non-stationary price data. This illustrates the uniqueness of the present work. The practical importance of this study is thus to show the first experimental results of how automated DQN models behave. The theoretical significance is to describe the advantages of reinforcement learning algorithms and dynamic implementation in one context.

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 subject of the paper:

> *To which extend can automated DQN models improve forecast accuracy?*

The remainder of the paper is organized as follows: In Section 2, we describe related work where ML models have also been used to automate and improve prediction accuracy, as well as other works on static models. In Section 3, the theoretical background of the DQN models is described. In Section 4, we describe the research methodology, which includes an explanation of the used data, the updating process, etc. In Section 5, we present our results, which are then discussed in detail in Section 6. The final sections 7 and 8 describe future work directions and the conclusions from our investigation.

2. Related Work

Since according to Jiang (2021), the prediction of stock price developments has been a research topic for a long time, there are some comprehensive literature studies, such as those of Kurani et al. (2021), Thakkar and Chaudhari (2021), Jiang (2021), Gandhmal and Kumar (2019) or Sezer et al. (2020), which tabulate the researches and developments already made. Literature studies, such as those by Kurani et al. (2021), for example, document investigations dating back to 1970. Based on the literature studies, it can be seen that a wide variety of models, such as 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, or a wide variety of hybrid models combined from the models just named, have been used. According to the literature reviews by Jiang (2021) or Sezer et al. (2020), the most common models used were recurrent models.

In addition, based on the above-mentioned literature studies, it was found that depending on which stocks, indices and time periods were studied, researchers were able to achieve accuracies of up to 99.9 % on test data. 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.

There are also literature reviews, such as by Singh et al. (2022), which document the developments of deep learning, reinforcement and deep reinforcement learning methods according to their various application areas in the financial industry (e.g. financial forecasts, financial text mining, portfolio management or credit risk) and compares traditional statistical methods with these algorithms. Both value-based methods such as Q-learning and DQN as well as policy-based algorithms such as Deterministic Policy Gradient (DPG) and Deep DPG were considered for the reinforcement learning algorithms. They concluded that reinforcement and deep reinforcement learning methods can effectively predict and recognize difficult market trends compared to traditional ML algorithms, with the significant advantage of high-level feature extraction properties and proficiency of the problem solver methods. They also mention the advantage that reinforcement learning allows us to create a more efficient framework that takes into account key market constraints by integrating the prediction problem with the portfolio structure task. Other researchers, such as Li et al. (2020), have also generally demonstrated the feasibility of deep reinforcement learning in financial markets, as well as the credibility and utility of strategic decision making, through experiments with U.S. stocks.

According to Gu et al. (2020), despite the versatility of the models and results obtained, only a few models were nevertheless able to prevail in automated stock trading. Review work, such as by Kurani et al. (2021) describe that studies,in recent decades made the mistake of not including external factors, such as current investor sentiment. The same report Jin et al. (2019).

Other reviews, such as that by Jiang (2021), describe that some studies have made the mistake of evaluating the profit and risk of the trading strategy only on the basis of the prediction result and not acting in favor of the portfolio (e.g. not taking transaction costs into account). Futhermore, there is the problem, according to Du et al. (2021), it is unexplored how to model time series from the perspective of distribution, although it is known that according to Gandhmal and Kumar (2019) stock price data are non-stationary and researchers such as Guo et al. (2018) have shown in tests that stock price data can have different data distributions at different times.

Related work that tries to improve the prediction accuracy of stock prices through a dynamic implementation is by Nguyen et al. (2019). They compare the prediction accuracies of dynamic and static LSTMs using four stocks listed in the NASDAQ stock market, which prices have been measured in daily intervals. In dynamic LSTM, in contrast to static LSTM, the model was updated after each prediction using an expanded training set. The training set was expanded by adding the current prediction of the LSTM to the training set. After approximately seven days of computation, the authors determined that automating the LSTMs had significantly improved the prediction accuracy. For the stock period from 2014 to 2019, the dynamic LSTMs were able to reduce the mean square error by 45.9 %. They also compared the performance capabilities of the static and dynamic LSTM models with respect to different window sizes. The authors concluded that the dynamic LSTMs tended to be less dependent on window size compared to the static LSTMs.

Guo et al. (2018) used five random stocks from the Shanghai Stock Exchange to explore whether automating the Support Vector Machine Regression (SVR) learning algorithm can improve prediction accuracy. The stock price data were measured in 2015, 2016 and 2017 and different stock price intervals were considered. They automated the SVR learning algorithm by using Particle Swarm Optimization (PSO). The idea behind PSO is to find the optimal solution through cooperation and information sharing among individuals. To counteract computation time, the parameters of the SVR learning algorithms were updated only when the model error exceeded a threshold θ. They compared their adaptive SVR learning algorithm with ordinary SVR learning algorithms and backpropagation neural networks and concluded that the adaptive SVR learning algorithm had better adaptability

and prediction accuracy. Thus, the adaptive SVR learning algorithm could reduce the prediction error for the different stocks by 8 to 41 %. Furthermore, they found in their study that the predictive accuracy of the static models became worse and worse as time progressed.

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, as described in section 1, no other study could be found in which DQNs are automatically adjusted to the changing characteristics of non-stationary stock price data.

Nevertheless, there are numerous papers, such as those by Shi et al. (2021), Zhang and Lei (2022), Dang (2019) or Bajpai (2021) that use static DQNs to predict stock prices with good performances compared to conventional methods. For example, Shi et al. (2021) showed how the double DQN outperformed LSTMs and SVMs in terms of various factors using various American and Chinese stock indices. Bajpai (2021) was also able to see the performance of the DQN using Indian stocks in her study. She makes the assumption, that due to the fact that stock markets are particularly stochastic and can change rapidly, DQN models perform better than conventional methods as a result of their rapid adaptability. Zhang and Lei (2022) considered all possible U.S. stocks in their study, and concluded that DQN models can perform better in turbulent market periods compared to traditional methods due to their adaptability.

Finally, it is important to point out the weaknesses of our literature review. Since the approach of this paper is highly exemplary, 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 autoamtized DQNs. Another important weakness of the above-mentioned work on automated ML models is that the very good results were achieved with a small sample, such as that of Guo et al. (2018) with five stocks or Nguyen et al. (2019) with four stocks, which is 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 preserving feature vector capturing the state (Burkov, 2019).

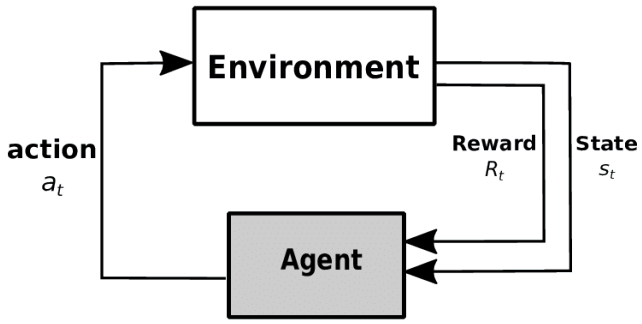


Figure 2: Agent and environment Source: Amiri et al. (2018)

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 $\{X_n,\ n = 0, 1, 2, ...\}$ is a stochastic process that satisfies the following Markov property for a finite set of states $\{x_n,\ n = 0, 1, 2, ...\}$:

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

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

For problems where the transition probabilities and the reward function of the underlying Markov decision process are unknown, Watkins (1989) introduced the Q-learning algorithm, whose convergence theorem is presented and proves in detail in Watkins and Dayan (1992). Q-learning works by an agent trying out an action in a certain state and recognizing the consequences in the form of the immediate reward or punishment. By repeatedly trying all actions in all states, the agent learns which actions are best overall, measured in terms of the long-term discounted reward. The Q-learning algorithm is defined as follows:

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

Where 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 $\max \mathrm{Q(s', a')}$ 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 updated its DQN algorithm, which it called 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, according to DeepMind, is supposed to be the max operation (see formula 2), because the approximated Q-values are often noisy and the use of 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:

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

A first neural network Q_θ, called the online network, evaluates the greedy strategy, i.e. 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. Dueling DQN

The Dueling 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 dueling 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 dueling 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. DRQN

The Deep Recurrent Q-Network (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

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

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

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

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

The study is composed of two preliminary and one main investigations. The first preliminary investigation examines which network architectures are powerful for the different DQN and their variants, because there are four different variants of DQNs and a wide range for the definition of their network architectures. Tensorforce[1] was chosen to access the DQN variants because it is very user-friendly. In the second preliminary investigation, the best models from the first preliminary investigation are trained using an extended training data set. This ensures that the lack of representativeness of the training data is not the reason for the poor results obtained later. After that, the main investigation begins, where the best models from the first preliminary investigation are automated. Five different strategies are tried for the automation process, which are explained in more detail in the Section 4.6. The study is then extended by examining how different stock price intervals and window sizes (time steps) affect automation results. Finally, supervised learning algorithms are also trained with the same data to provide further benchmarks, and a statistical test is performed to verify the improvements of the automated models. Figure 3 shows this worklow again.

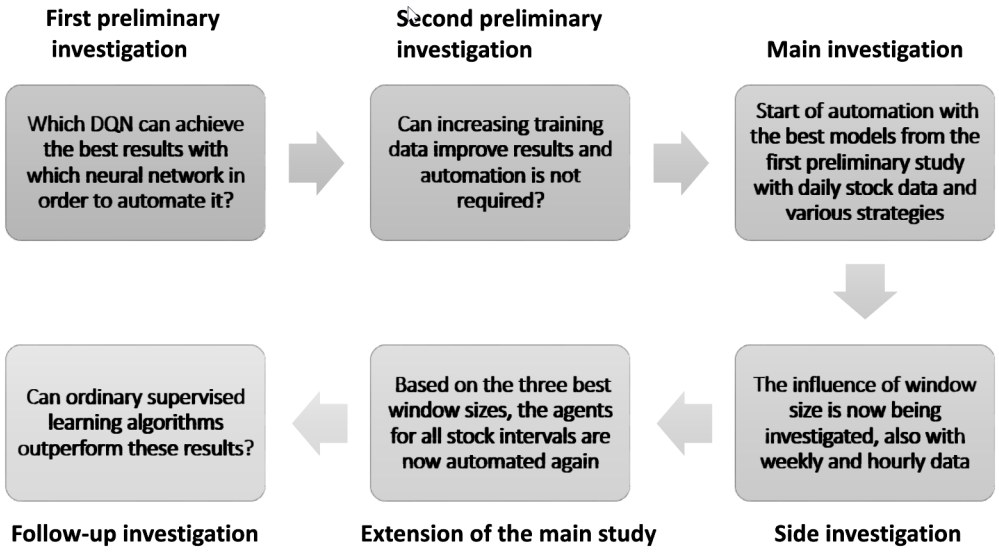


Figure 3: Research workflow

[1] https://tensorforce.readthedocs.io/en/latest/

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. Opening prices, highs, lows and closes, trading volume, DAX closing prices and Google Trends data via the search queries "crash" and the stock name (adidas, basf or allianz) are taken into account. 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 an sentiment indicator included in the prediction. The stock data is downloaded using the Python library yfinance[2] and the Google Trends data is downloaded using the Python library pytrends[3]. The training data is formed for the individual stocks using the data measured in the period 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 7 selected characteristics (5 historical stock data characteristics and 2 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.

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

[2] `https://pypi.org/project/yfinance/`
[3] `https://pypi.org/project/pytrends/`

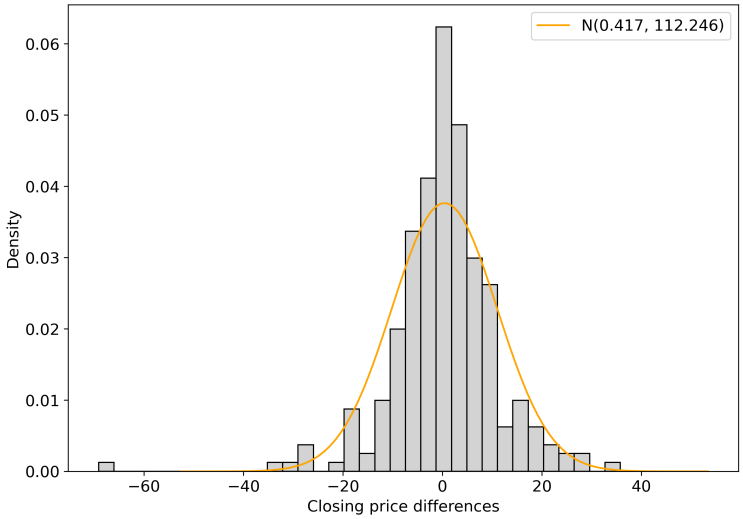


Figure 4: Histogram of weekly closing price differences for Adidas with $N(0.417, 112.246)$

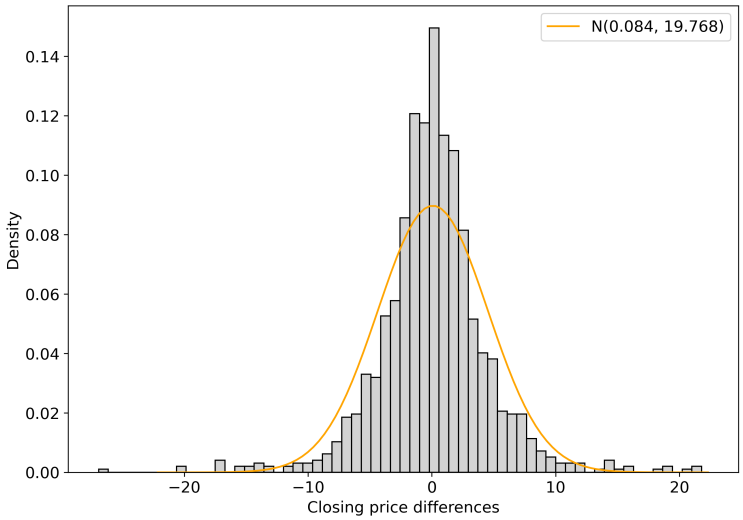


Figure 5: Histogram of daily closing price differences for Adidas with $N(0.084, 19.786)$

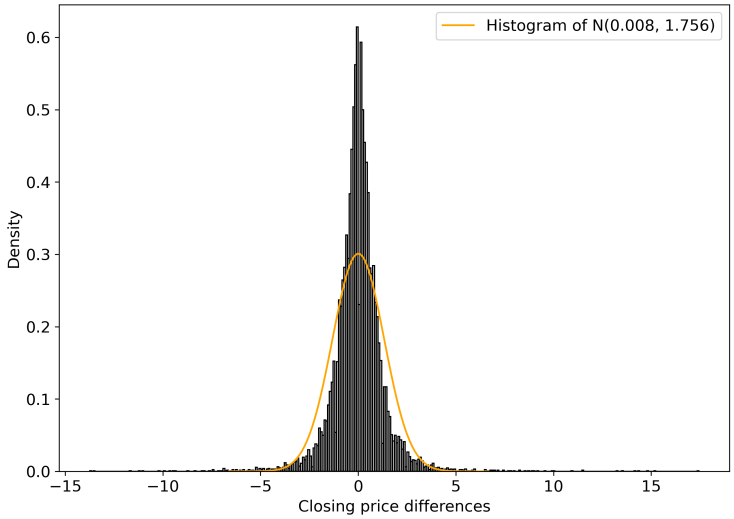


Figure 6: Histogram of hourly closing price differences for Adidas with $N(0.008, 1.756)$

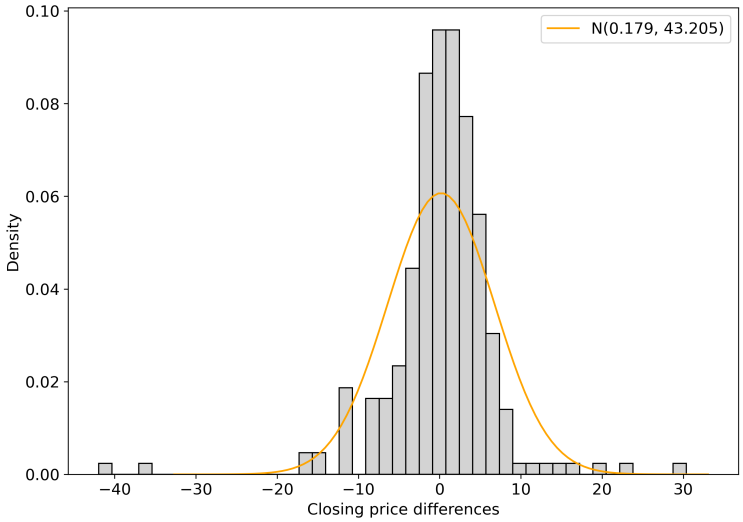


Figure 7: Histogram of weekly closing price differences for Allianz with $N(0.179, 43.205)$

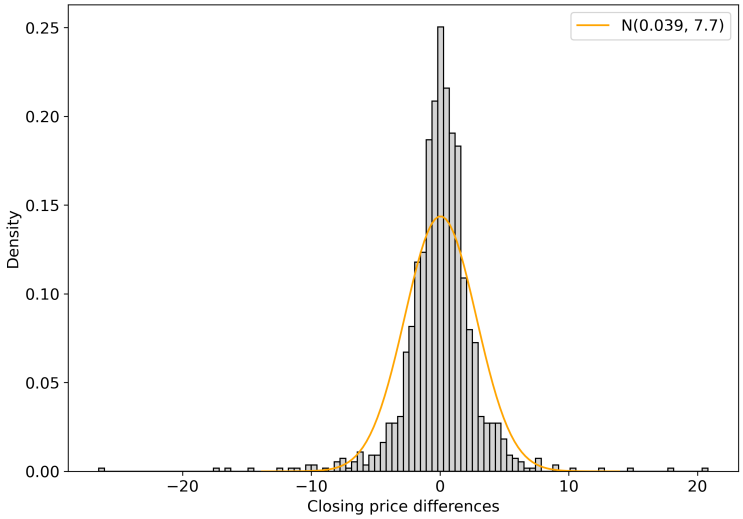


Figure 8: Histogram of daily closing price differences for Allianz with $N(0.039, 7.7)$

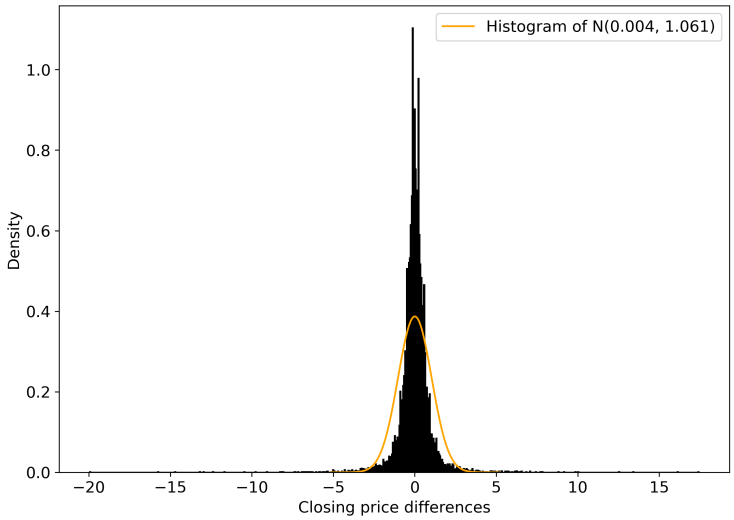


Figure 9: Histogram of hourly closing price differences for Allianz with $N(0.004, 1.061)$

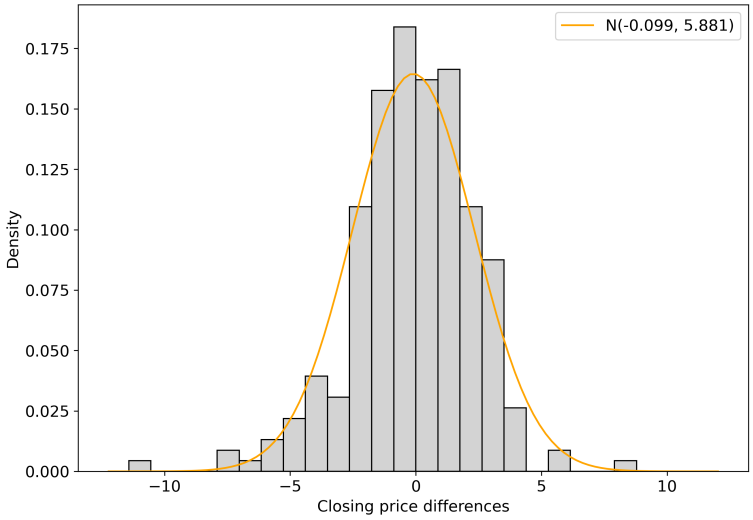


Figure 10: Histogram of weekly closing price differences for BASF with $N(-0.099, 5.881)$

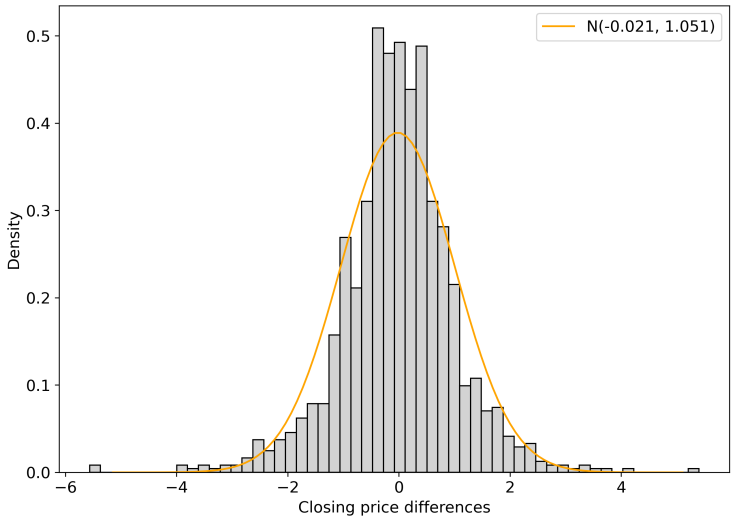


Figure 11: Histogram of daily closing price differences for BASF with $N(-0.021, 1.051)$

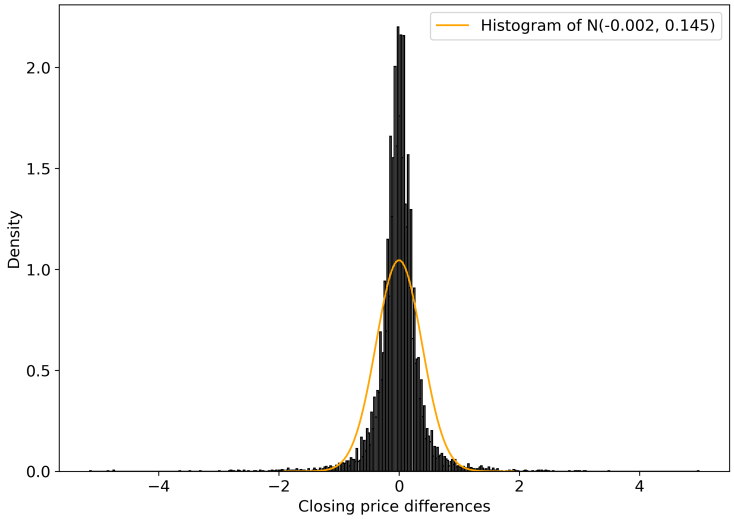


Figure 12: Histogram of hourly closing price differences for BASF with $N(-0.002, 0.145)$

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).

We can also observe this 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 figure 9 or 12), but they also all have a high peak at this point. For this reason, researchers also suggest using distributions for stock returns that are similar to a normal distribution but also have additional characteristics such as high peak and thick tail, such as the exponential power distribution (Li, 2023).

We hypothesize that if enough parameters and normally similar distributions are tested, a matching distribution to our samples can be found. However, the more important and difficult task is how to recognize and model changing distributions over time.

To get an idea of how difficult this task is, the sample means and variances for each month of the daily closing price differences of adjacent days t_n and t_{n+1} are plotted in figure 13 and 14.

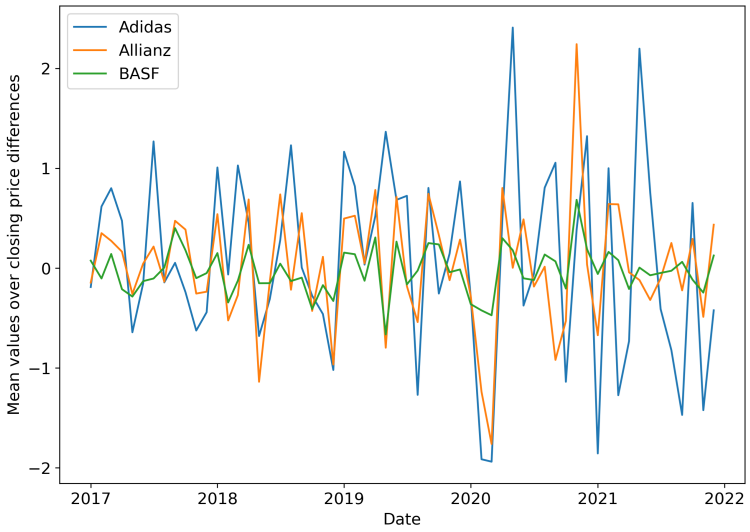


Figure 13: Monthly mean values of daily closing price differences

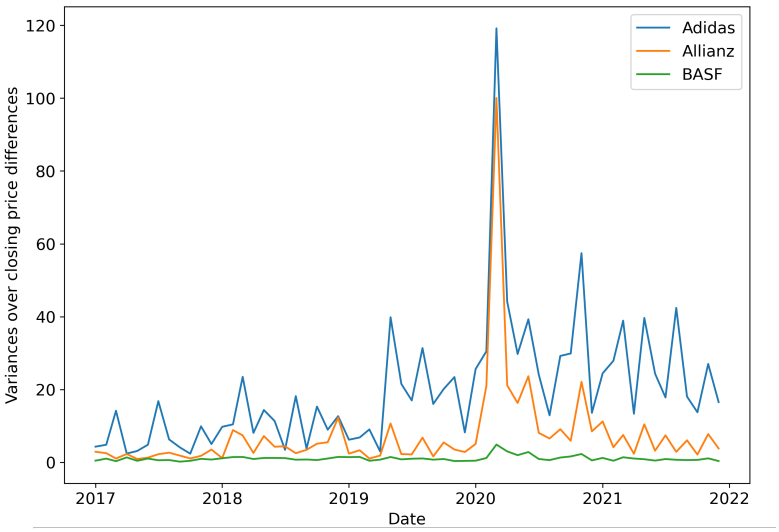


Figure 14: Monthly variances of daily closing price differences

It can be seen how the monthly means fluctuates between the values -2 and 2 and that the stocks Adidas and Allianz have greater volatility than BASF (see figure 13). It can also be seen that the mean values over the entire five years (see figure 5, 8 and 11) are close to 0, but can vary greatly 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 14). 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.

From the histograms it can also be seen that the sample variance decreases with finer price intervals (compare figure 4, 7 and 10 with figure 6 , 9 and 12). This means that the probability of a higher gain or loss increases with the time that a stock is in the portfolio. Due to the symmetry of the mean value of the normal distribution-like samples (see e.g figure 6 or 8), 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 confirm this, one can look at the proportion of positive and negative price direction changes from the entire data set.

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 1: Correlation between Google Trends data and stock closing prices

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

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 2 shows the Pearson correlation coefficients and their p-values between our Google Trends data and the closing prices for all price intervals.

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

Table 2: Correlation between Google Trends data and stock closing prices

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 α of 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 seen in the hourly data. This is probably due to the fact that the volume of data is highest in this period. Overall, these statistics suggest that the Google Trends data, particularly for the keyword "crash", will probably help us to increase the explainable variance.

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[4] by defining a custom agent environment, which is illustrated in the following figure:

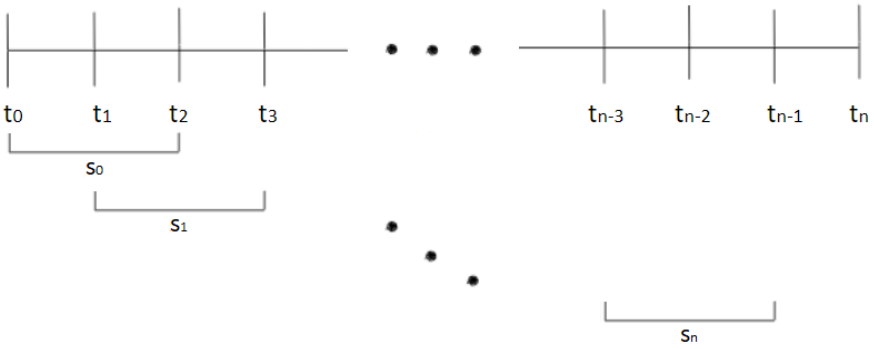


Figure 15: Example for self-defined environment with window size = 3

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 15). 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.

[4] `https://www.gymlibrary.dev/content/environment_creation/`

4.3. Neural Network architectures

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

Name of Model	First 1D convolutional layer			Second 1D convolutional layer		
	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 3: Overview of the Convolutional Neural Networks (CNNs) of the models

Name of Model	Hidden layers			LSTM layer			Dropout layers	
	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	Tanh	3	0.3

Table 4: Overview of the Neural networks of the models after the CNNs

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 4). 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, etc.

We used the ReLU function for the CNNs and for some hidden layers as activation function because it has a low computation time due to its definition $f(x) = max(0, x)$, is the most commonly used function and has a fast convergence speed (Lin and Shen, 2018). We have also used other activation functions for the hidden layers (see model 3 and 4), as there have been publications in recent years describing that they have found better activation functions, such as that of Clevert et al. (2015) via the ELU function, which allows now to consider negative inputs with an exponential function, or their extension of Klambauer et al. (2017) by the SELU function, which additionally standardizes their outputs.

We set the number of filters in models 2, 3, and 4 in ascending order (see table 3), which is also recommended in litteratures such as Géron (2019), so that the CNN gradually learns to recognize increasingly complex patterns, and we selected their kernel sizes experimentally. For the 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 16. 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).

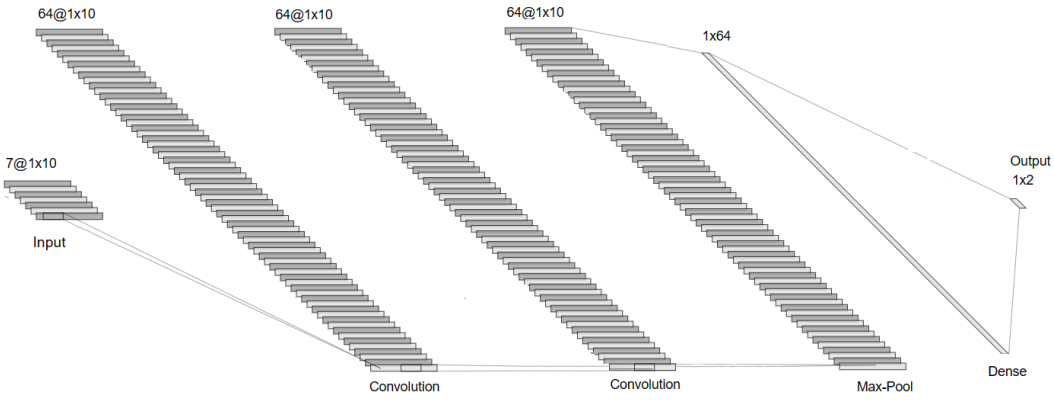


Figure 16: Model 1 with window size = 10

4.4. Hyperparameters

The following hyperparameters are used for the models:

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

Table 5: Hyperparameters of 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 5) also comes from various experiments and was also 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

For a better understanding of how the DQN of Mnih et al. (2013) works, its general algorithm is described:

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 $t = 0, ..., T$ for M episodes as follows:
    1. 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 $arg\ max_a\ Q_\theta(s, a)$.
    2. Perform the selected action a_t, change to the next state s' and receive the reward r.
    3. Save the transition information (s, a, r, s') in the replay buffer D.
    4. Extract a random mini-batch with K transition information from the replay buffer.
    5. 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 $y_i = r_i$.
    6. Calculate the network error $L(\theta) = \frac{1}{K} \sum_{i=1}^{K} (y_i - Q_\theta(s_i, a_i))^2$.
    7. 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 $t_i = t_0 + i * n$.

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:

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.

Table 6: Update strategies

The idea behind resetting the agent after each update in strategies 1 and 3 (see table 6) is to check whether the pre-trained goodness is useful. 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 checked using the open source framework Arangopipe from ArangoDB[5], as it can be used to determine a data shift value. The data shift value in this API is determined by developing a classifier that specifies the accuracy of how much it can distinguish between two datasets.

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

5. Results

5.1. First preliminary investigation

For all possible combinations of network architectures (see table 3 and 4) and DQN variants, the following models achieved the best results:

Stock	Best combination		MDA			RMSE		
	Variant	**Model**	**Training**	**Validation**	**Test**	**Training**	**Validation**	**Test**
Adidas	Dueling DQN	4	86,34 %	61,24 %	51,14 %	0.37	0.62	0.70
Allianz	Dueling 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

Table 7: The best models of the performance comparison

The performances of all other model combinations can be found in table A.12 in the Appendix A. This appendix also contains illustrations showing the training process of the DRQN (see figure A.26 and A.27).

5.2. Second preliminary investigation

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:

Stock	MDA			RMSE		
	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

Table 8: Performance based on the extended training data set

The training process from this preliminary investigation can be seen in Appendix B (see figure B.28 and B.29).

5.3. Main investigation

In the main study, agents are first automated using daily stock price data and different strategies. Figures 17, 18 and 19 thus show the accuracies of the automated agents per strategy and stock at different step sizes 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 figure 20, 21 and 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 figure 23, 24, and 25). On the following pages you can see these nine figures, for which a total of several weeks of computing time was required:

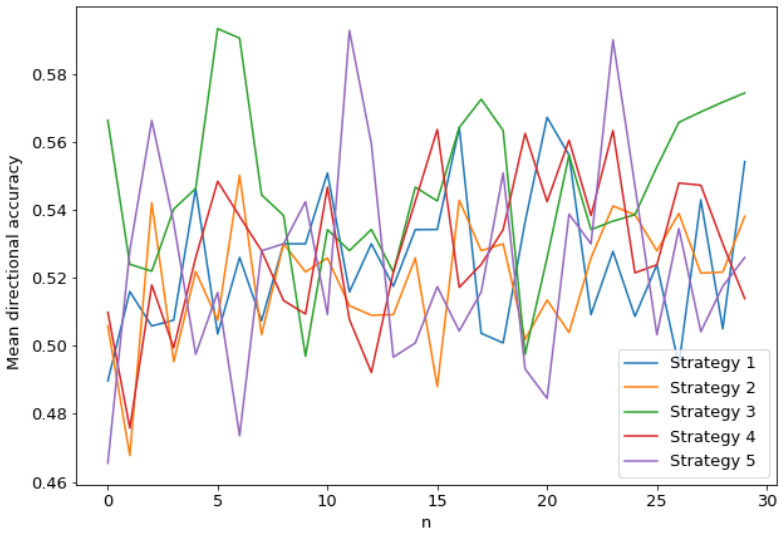


Figure 17: MDAs regarding the stock Adidas and different update strategies

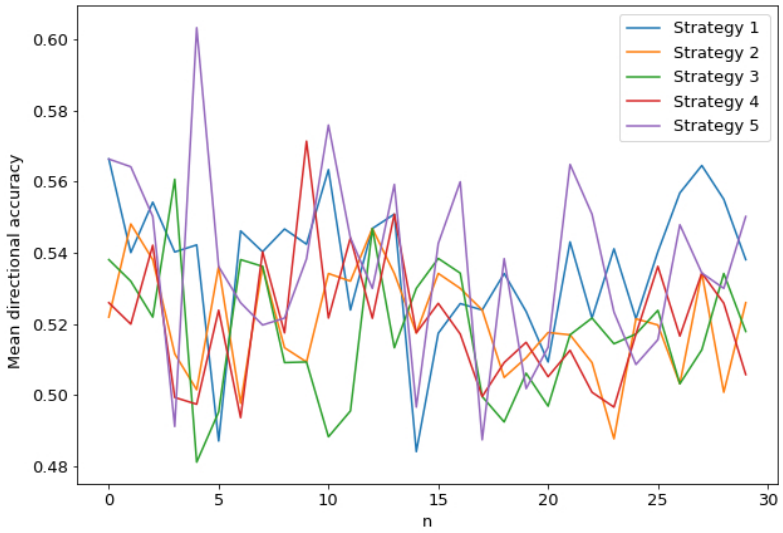


Figure 18: MDAs regarding the stock Allianz and different update strategies

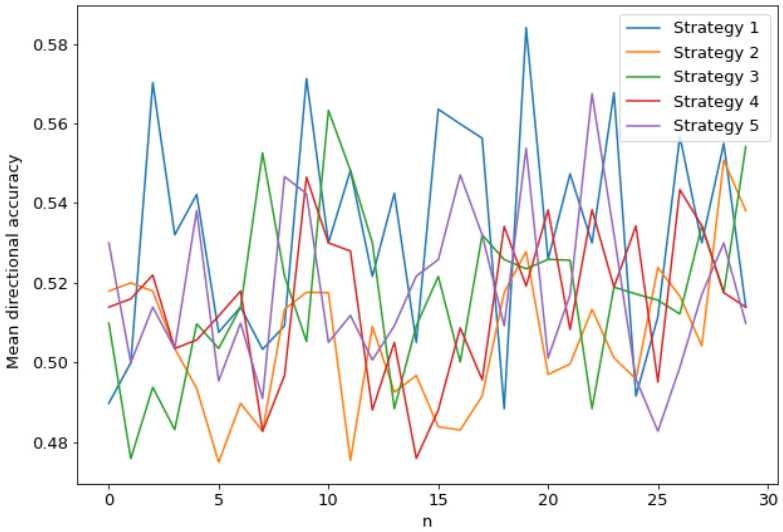


Figure 19: MDAs regarding the stock BASF and different update strategies

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

Table 9: Best strategys and time intervals for figures 17,18 and 19

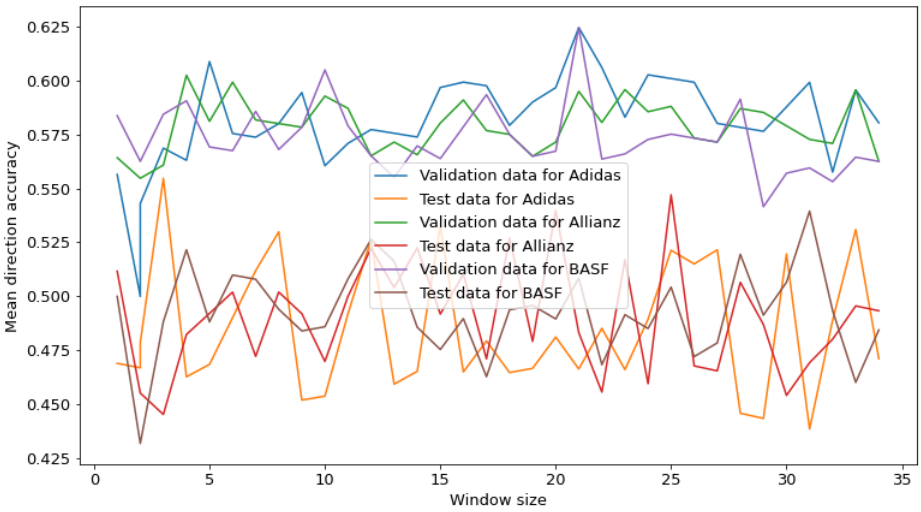


Figure 20: MDAs regarding different window sizes and daily measured stock data

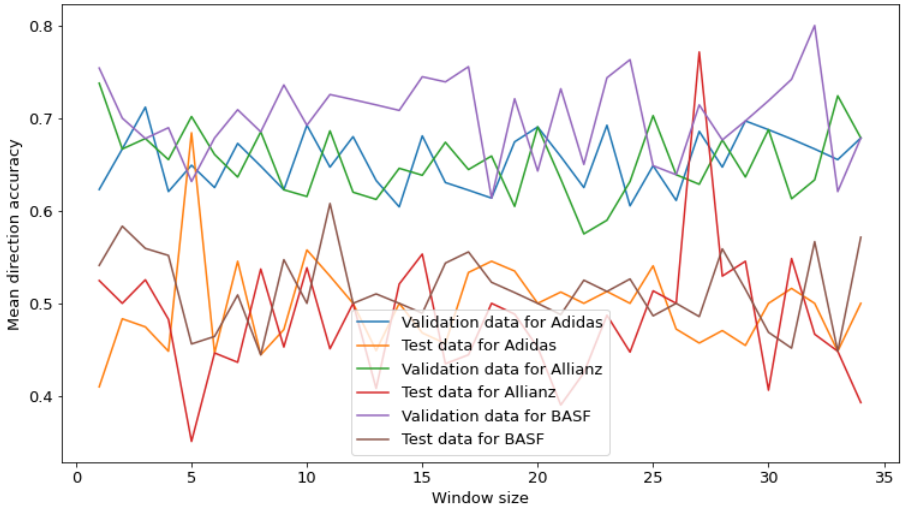


Figure 21: MDAs regarding different window sizes and weekly measured stock data

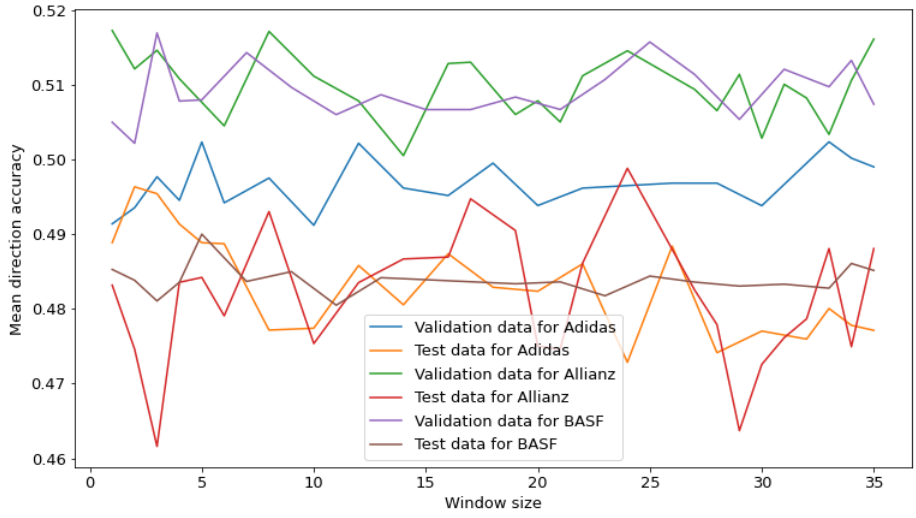


Figure 22: MDAs regarding different window sizes and hourly measured stock data

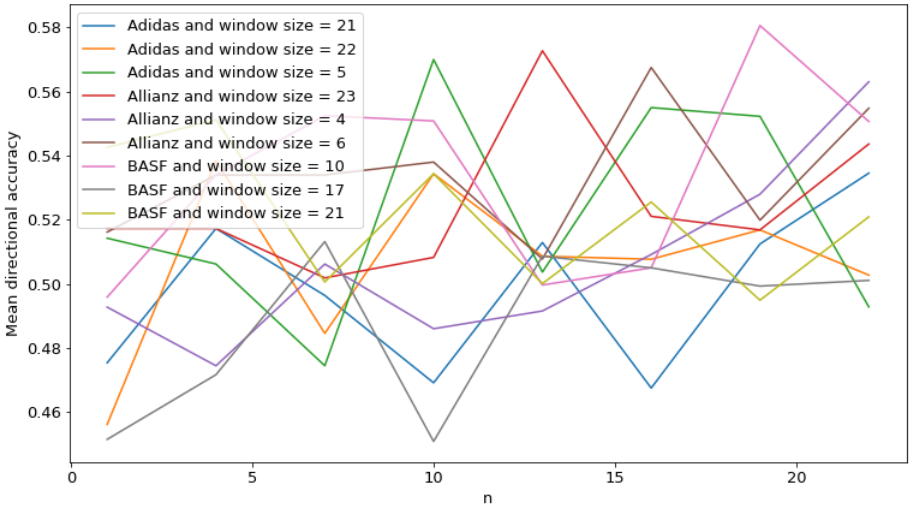


Figure 23: MDAs of automated agents regarding daily measured stock data and different window sizes

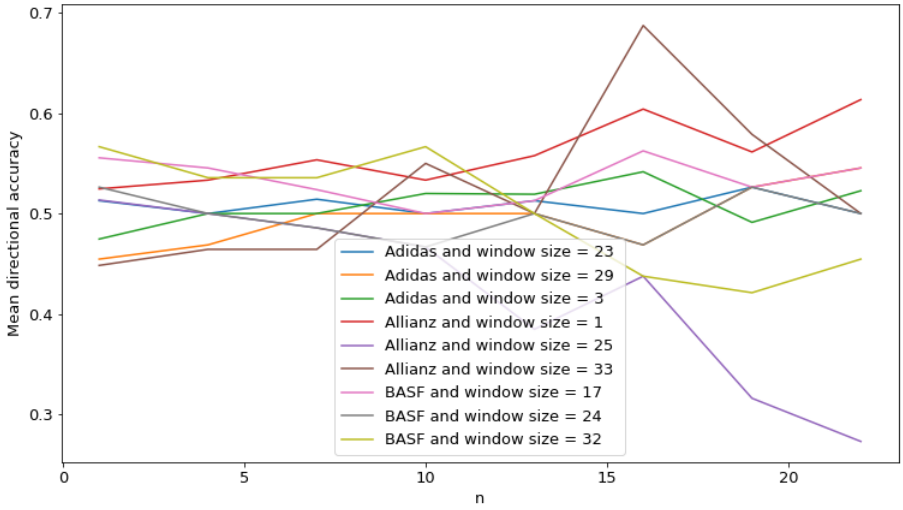


Figure 24: MDAs of automated agents regarding weekly measured stock data and different window sizes

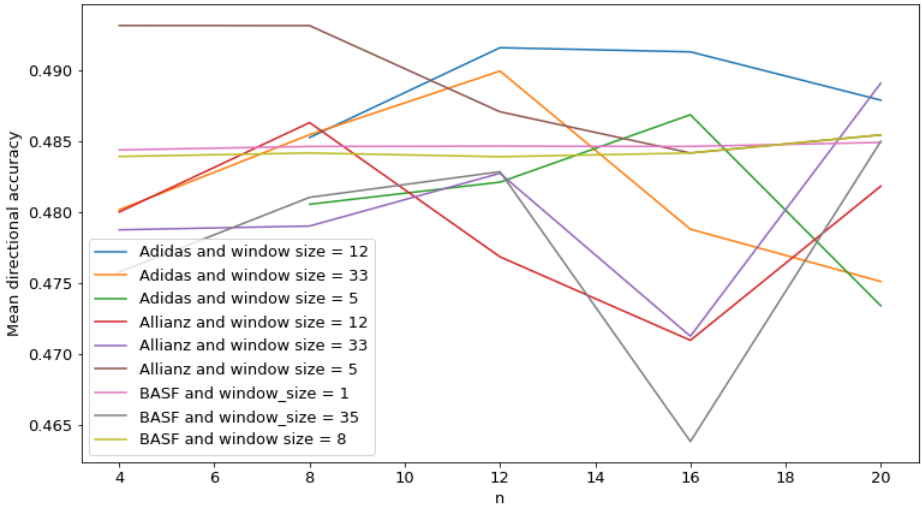


Figure 25: MDAs of automated agents regarding hourly measured stock data and different window sizes

5.4. Comparison with supervised learning algorithms

To compare how much better the automatic agents are compared to conventional methods, we also train and test three LSTMs, SVMs and logit models based on the same data split (see section 4.1) and hyperparameters (see table 5). The LSTM models are initialized with ten LSTM layers with 64 LSTM units, each of which has a dropout layer with a dropout rate of 0.4. The supervised learning algorithms were able to achieve the following results:

Stock	Test MDA			Test RMSE		
	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

Table 10: Performance of the supervised learning algorithms

5.5. 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-handed hypothesis test, the null hypothesis is rejected with a significance level α of 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).

Stock	Static model			Automated model		
	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

Table 11: Static verification of the improved accuracy

The P-values clearly show that at a significance level α of 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.

6. Discussion

Both the best static DQN models, whether or not 7 more years of training data were available, and the supervised learning algorithms failed to achieve test accuracies significantly greater than 50 % (see table 7, 8 and 10). The reason for this may be the non-stationarity of the stock price data. Because, as researchers such as Krawczyk et al. (2020) or Souza et al. (2020) describe, historical price data, due to the changing data distribution, can be irrelevant or even harmful for the modeling of the forecast model. 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.

The only thing that could significantly improve the results was the automation of the agents. We assume that the improvements compared to the static methods are due to the fact that we updated the parameters of the models at some points in time when the data distribution actually changed for a while compared to the data distribution assumed by the static models. For the Adidas stock, the MDA could be improved from 51.14 to 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 was almost as good as strategy 3 (despite a much shorter 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).

However, at the same time it is important to point out that for certain step sizes n the test accuracies have worsened (see figure 17, 18 and 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 irritated. This is probably because the agents have also been updated unnecessarily, i.e. 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 implies that research is needed to address the question of when the data distribution changes in time series data. Raza et al. (2013, 2015) have already carried out some investigations concerning this problem and have developed noteworthy methods. They could already show with their first version of the method Raza et al. (2013), which is based on an EWMA chart (Exponentially Weighted Moving Average Chart), for a time series with 1000 observations that for certain smoothing constants λ all time points in which the data distribution changes can be determined.

However, their first version of the method had simultaneously resolved a number of false alarms, i.e., 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 were able to show that this change reduced the number of false alarms.

Looking at the research of Raza et al. (2013, 2015) it becomes clear why strategy 5 (see table 6) 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 guesses of the covariate shifts are statistically significant.

In terms of the update strategies (see table 6), it can also be said that strategies 2 and 4 were the worst on average. Thus, it can be concluded that resetting 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.

For the question, how many data points should be considered for the adjustment of the parameters, the results could not give a clear picture. It is suspected that considering all previous data points is not efficient because the computation time increases enormously over time and probably not all data are 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 step size n. Some settings for window size proved to be better for some stocks than others (see figure 20, 21 and 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 probably due to the fact that, again, some data are no longer relevant because of market dynamics and thus have an irritating effect on the agent. In conclusion, it is recommended to test several window sizes.

It can also be observed that for certain step sizes n and window sizes, the automated agents were able to improve test accuracy only for stock data measured in weekly and daily intervals (compare figure 23 and 24 with figure 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 became smaller (see figure 20, 21 and 22), it is assumed that the reason for this is fluctuations in investor sentiment play a greater role in smaller price intervals, as this increases the unexplained variance. This assumption is simultaneously consistent with Shah et al. (2020) claim that the stock market is more predictable and has less noise in the long run. To empirically verify this conjecture, 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 can also be seen that the test accuracies in our studies decrease for almost all window sizes compared to the validation accuracies (see figure 20, 21 and 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) was also able to observe this when he realized that the accuracy of his static models had decreased over time.

Compared to other works that automated ML models, we achieved smaller improvements. Guo et al. (2018) was able to reduce the RMSE by 8 to 41 % with their adaptive SVR learning algorithm and Nguyen et al. (2019) the MSE by 45.9 % with their dynamic LSTM. We were able to reduce the RMSE for our stocks by about 8 to 12.5 % (compare table 7 with table 9). This is less than the other two researchers, but it must also be mentioned that our results are not comparable as we all selected different stocks and time periods to analyze. In order to investigate which approach is the best, we would all have to analyze the same database.

Our results can also be compared with other studies that use completely different approaches, i.e. without continuous parameter adjustment. For example, Thakkar and Chaudhari (2020) combined information from daily open price data from two exchanges (NSE and BSE) for three stocks, to combine LSTM models that achieve a maximum directional accuracy of 54.14 %. They used a time period from 2009 to 2019 with a data split of 80 % training data and 20 % test data. Their results are consistent with ours in that the directional accuracy is relatively low despite the longer time period used and the application of complex static models.

Zhong and Enke (2017) tried to predict the daily direction the SPDR S&P 500 ETF using 60 financial and economic factors as input by combining three dimensionality reduction techniques, which are principal component analysis (PCA), fuzzy robust principal component analysis (FRPCA) and kernel-based principal component analysis (KPCA) with artificial neural networks (ANN) and varying number of principal components. The period used was from 01.06.2003 to 31.05.2013 (2518 data records) with a data split of 70 % training data (1762 data records) and 15 % each of test and validation data (378 data records respectively). The ANN with PCA and 60 principal components achieved an accuracy of 58.1 %, the KPCA with 31 principal components an accuracy of 59.2 % and the KPCA with 60 principal components a accuracy of 58.4 %. This study shows that they were able to achieve a directional accuracy of more than 58 %, similar to ours. The main advantage of their study probably lies in the fact that they used many more input variables than we did (60 input variables versus 7 input variables). Presumably, our approach with more input variables could increase the explainable variance of our automated models.

However, there are also studies in the literature that show that they have achieved an MDA of over 70 % even with a low number of inputs. Using ANNs and SVMs and an intensive grid search of 900 and 700 parameter combinations, Kara et al. (2011) was able to achieve an average directional accuracy of 71.52 % for the SVM and 75.74 % for the ANN with 10 technical indicators as input in predicting the direction of movement of the daily Istanbul Stock Exchange National 100 Index. The period used for the performance comparison was from 02.01.1997 to 31.12.2007 (2733 data records) with a split of 50 % training data and 50 % test data. The study shows that, despite the assumption of stationarity, very good results can also be achieved with ordinary supervised models. However, as mentioned in the comparison with automated models, the selected database used to develop the models must be the same so that the aforementioned works are more comparable.

Nevertheless, it must also be mentioned that our method also has a significant disadvantage. 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.

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 analyzed, 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 Dueling 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 A.26), 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 irritating information due to covariate shifts and noise. Approaches to the correctness of the assumption can be given 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.

[6] `https://colab.research.google.com/signup`

7. Future work

In general, our approach should be further investigated, for example by using other stocks and features, so that automated deep reinforcement models are better researched, this field of research continues to grow and more results are available to give a better picture of their potential.

As mentioned in the discussion, our methodology can be extended by using more efficient methods to determine the covariate shift. This could significantly improve the results, as unnecessary updates could be avoided even more and the necessary updates in the case of covariate shifts could be increased. In our opinion, this is the most important point to improve this method.

We also do not want to rule out that there are better ways to automate the models, although updating the parameters when a significant covariate shift occurs seems most plausible. Therefore, we would like to encourage other researchers to try out other methods if ideas are available.

It is also recommended for future work to include additional characteristics in order to reduce the high unexplained variance. It is recommended to consider behavioral datasets, as according to many researchers such as Jin et al. (2019), Breaban and Noussair (2018) or Duxbury et al. (2020), investor sentiment is enormously important for stock price prediction. We assume that the detection of covariate shifts can become more accurate with more behavioral datasets.

After developing an efficient dynamic model, we recommend not to make the mistake of evaluating the profit and risk of the trading strategy based only on the prediction results, as other studies have done according to Jiang (2021). Rather, the advantage of reinforcement learning described by Singh et al. (2022) of integrating the prediction problem with the portfolio structure task should be used. This allows us to take factors such as transaction costs into account, which, in combination with a good adaptive model, leads to successful automated trading.

8. Conclusion

The automation of ML models seems to be a promising way for stock price forecasting, since this study, as well as related works, such as that by Du et al. (2021), de Lima e Silva et al. (2020) or Guo et al. (2018), indicate, that the desired model for stock price forecasting changes over time due to market dynamics. Therefore, it is very important to clarify at which times the model needs to be updated.

After an extensive analysis and discussion of our results, we found that updating at regular intervals is not necessarily the most effective way to capture market dynamics. This is because data distributions can change at unpredictable times and the unnecessary adjustments of parameters can irritate the agents, resulting in worse results. Accordingly, we recommend the time interval n to be adaptive. Adjusting parameters when significant covariate shifts are detected is therefore key to achieving greater accuracy in predicting stock price movements.

Accordingly, there is a need to combine rapidly adaptable models, such as DQNs, to capture changing market dynamics and 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. If it can be determined exactly when the market changes, a timely decision can be made whether to initiate an adjustment.

Appendix A. First preliminary investigation

Model	Adidas			Allianz			BASF		
	DQN	Double DQN	Dueling DQN	DQN	Double DQN	Dueling DQN	DQN	Double DQN	Dueling 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 %

Table A.12: Validation accuracies of all possible model combinations

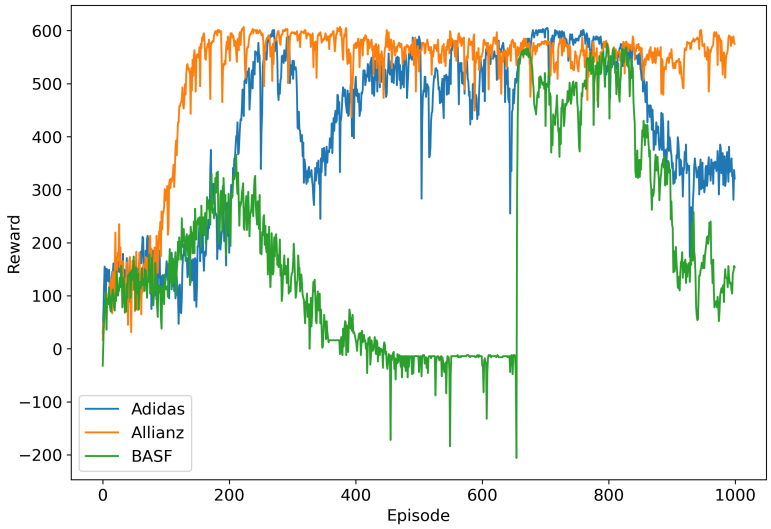


Figure A.26: Reward during training for the DRQNs

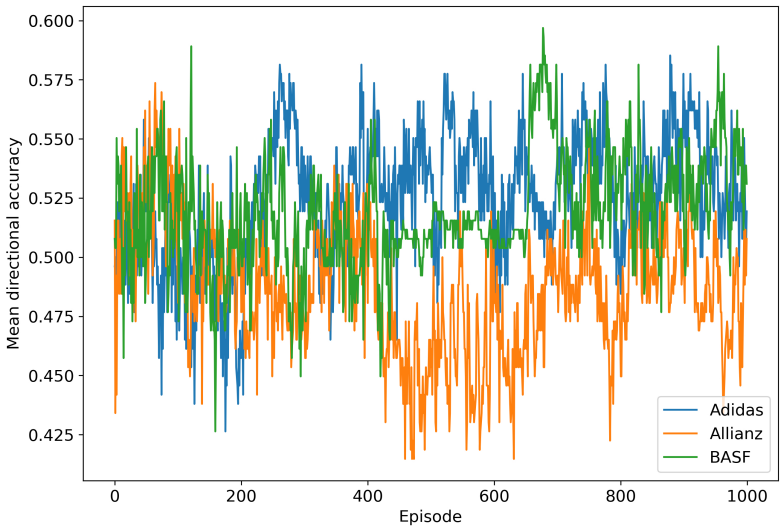


Figure A.27: MDA during validation for the DRQNs

Appendix B. Second preliminary investigation

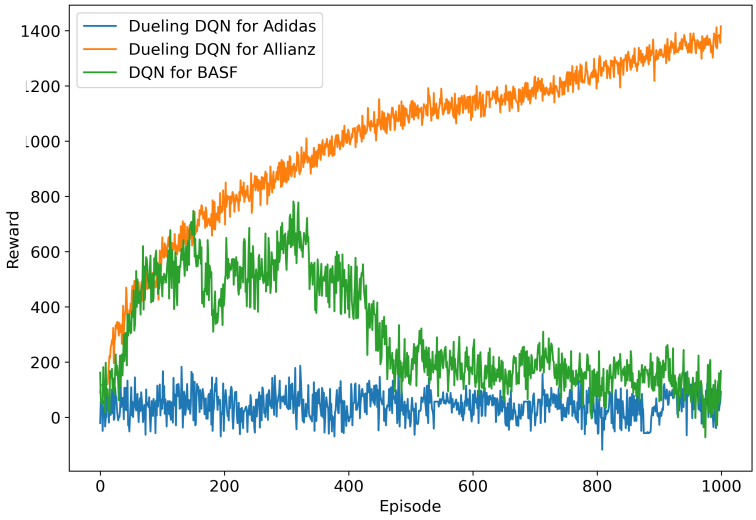


Figure B.28: Reward during training with the extended training data and the best models

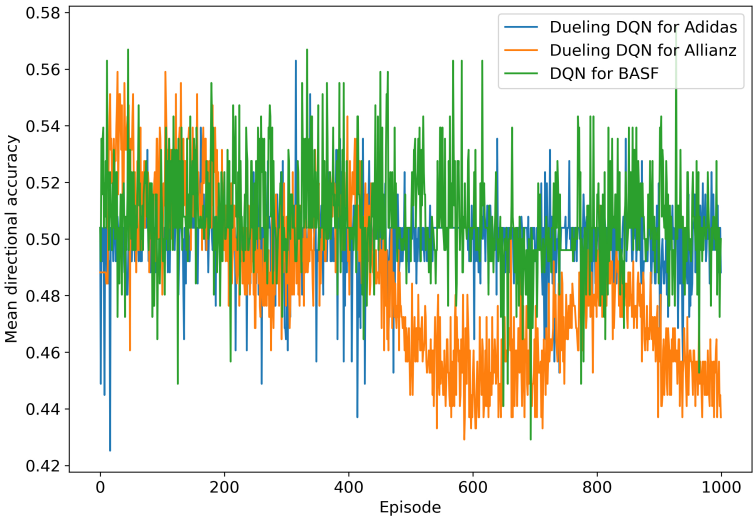


Figure B.29: MDA during validation with the extended training data and the best models

References

Alpaydin, E., 2022. Maschinelles Lernen. 3 ed., Berlin, Boston: 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).

Bajpai, S., 2021. Application of deep reinforcement learning for indian stock trading automation, in: ArXiv e-prints.

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

Bundeszentrale für politische Bildung, 2021. Aktienbestand und Aktienhandel. `https://www.bpb.de/nachschlagen/zahlen-und-fakten/globalisierung/52590/aktien`. Accessed on 12.12.2021.

Burkov, A., 2019. Machine Learning kompakt: Alles, was Sie wissen müssen. 1 ed., Frechen: 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.

Chollet, F., 2018. Deep Learning mit Python und Keras Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek. 1 ed., Frechen: Mitp Verlag.

Clevert, D.A., Unterthiner, T., Hochreiter, S., 2015. Fast and accurate deep network learning by exponential linear units (elus), in: ArXiv e-prints.

Dang, Q.V., 2019. Reinforcement learning in stock trading, in: International Conference on Computer Science, Applied Mathematics and Applications.

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. Washington, D.C.: 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, in: Applied Soft Computing 97.

Ditzler, G., Roveri, M., Alippi, C., Polikar, R., 2015. Learning in nonstationary environments: A survey, in: IEEE Computational Intelligence Magazine 10 (4), pp. 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, in: ArXiv e-prints.

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, in: The European Journal of Finance 26 (14), p. 1417–1438.

Elliot, A., Hsu, C.H., 2017. Time series prediction: Predicting stock price, in: ArXiv e-prints.

Franceschetti, M., Lacoux, C., Ohouens, R., Raffin, A., Sigaud, O., 2022. Making reinforcement learning work on swimmer, in: ArXiv e-prints.

Gandhmal, D.P., Kumar, K., 2019. Systematic analysis and review of stock market prediction techniques, in: Computer Science Review 34.

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

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

Géron, A., 2019. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems. 2 ed., Sebastopol: O'Reilly Media.

van Hasselt, H., Guez, A., Silver, D., 2015. Deep reinforcement learning with double q-learning, in: ArXiv e-prints.

Hausknecht, M., Stone, P., 2017. Deep recurrent q-learning for partially observable mdps, in: ArXiv e-prints.

Hodson, T.O., 2022. Root-mean-square error (rmse) or mean absolute error (mae): when to use them or not, volume 15.

Huyen, C., 2022. Designing Machine Learning Systems. 1 ed., Sebastopol: 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, in: Dinasti International Journal of Digital Business Management 2 (2).

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

Jin, Z., Yang, Y., Liu, Y., 2019. Stock closing price prediction based on sentiment analysis and lstm, in: Neural Computing and Applications 32, p. 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, in: Expert Systems with Applications 38 (5), pp. 5311–5319.

Karunakaran, D., Worrall, S., Nebot, E., 2020. Efficient statistical validation with edge cases to evaluate highly automated vehicles, in: ArXiv e-prints.

Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization, in: ArXiv e-prints.

Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S., 2017. Self-normalizing neural networks, in: ArXiv e-prints.

Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J., Woźniak, M., 2020. Ensemble learning for data stream analysis: A survey, in: Information Fusion 37, p. 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, in: Annals of Data Science.

Lapan, M., 2020. Deep Reinforcement Learning: Das umfassende Praxis-Handbuch. 1 ed., Frechen: Mitp Verlag.

Li, B., 2023. An explanation for the distribution characteristics of stock returns, in: ArXiv e-prints.

Li, Y., Ni, P., Chang, V., 2020. Application of deep reinforcement learning in stock trading strategies and stock forecasting, in: Computing, p. 1305–1322.

Lin, G., Shen, W., 2018. Research on convolutional neural network based on improved relu piecewise activation function, in: Procedia Computer Science 131, pp. 977–984.

Mnih, V., Koray Kavukcuogl and, D.S., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning, in: ArXiv e-prints.

Nguyen, D.H.D., Tran, L.P., Nguyen, V., 2019. Predicting stock prices using dynamic lstm models, in: Applied Informatics, pp. 199–212.

Pesaran, H., Timmermann, A., 1992. A simple nonparametric test of predictive performance, in: Journal of Business and Economic Statistics.

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.

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

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

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

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

Shah, D., Isa, H., Zulkernine, F., 2020. Stock market analysis: A review and taxonomy of prediction techniques, in: Stock Market Analysis: A Review and Taxonomy of Prediction Techniques 7 (2).

Shi, Y., Li, W., Zhu, L., Guo, K., Cambria, E., 2021. Stock trading rule discovery with double deep q-network, in: Angewandte Soft Computing 107.

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, in: International Journal of Information Management Data Insights 2 (2).

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, in: Data Mining and Knowledge Discovery 34, p. 1805–1858.

Stewart, W.J., 2009. Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling. 1 ed., New Jersey: Princeton University Press.

Thakkar, A., Chaudhari, K., 2020. Crest: Cross-reference to exchange-based stock trend prediction using long short-term memory, in: Procedia Computer Science 167, pp. 616–625.

Thakkar, A., Chaudhari, K., 2021. A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions, in: Expert Systems With Applications 117.

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, in: Prestige international journal of management and IT & Sanchayan 6 (1), pp. 111–121.

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., de Freitas, N., 2015. Dueling network architectures for deep reinforcement learning, in: ArXiv e-prints.

Watkins, C.J.C.H., 1989. Learning from delayed rewards. PhD Thesis. Cambridge: King's College.

Watkins, C.J.C.H., Dayan, P., 1992. Q-learning, in: Machine Learning 8, p. 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`, last accessed on 01/03/2024.

Zhang, J., Lei, Y., 2022. Deep reinforcement learning for stock prediction, in: Scientific Programming 2022, pp. 1–9.

Zhang, Y., Chen, C., Shi, N., Sun, R., Luo, Z.Q., 2022. Adam can converge without any modification on update rules, in: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (Eds.), Advances in Neural Information Processing Systems 35.

Zhong, X., Enke, D., 2017. Forecasting daily stock market return using dimensionality reduction, in: Expert Systems with Applications 67, pp. 126–139.