A Comparative Study of Trading Using Deep Q-Learning and Double Deep Q-Learning in Niche Markets

John Cao William Ekberg Nils Kuhn

Abstract

This study assesses the potential of Deep Q-Learning (DQN) and Double Deep Q-Learning (Double DQN) for trading in niche financial markets, characterized by lower liquidity and reduced competition from algorithmic traders. We formulate the trading problem as a Markov Decision Process (MDP) with varying action spaces, including discrete allocation levels, binary market participation, and incremental adjustments. Our experiments, conducted on historical data from the First North Growth Market and Novotek, assess the profitability and stability of learned policies. The results provide insights into the suitability of reinforcement learning for niche markets and highlight key trade-offs between different MDP formulations. Our code can be found at our Github repository. [1]

1. Introduction

Algorithmic trading accounts for a large portion of the daily transactions within stock markets, with AI algorithms already outperforming human traders and reshaping the industry (Deng et al., 2016). As these algorithms become more advanced, competing in major markets such as technology stocks, the US market, and European exchanges has become increasingly difficult. Leading quantitative trading firms account for a significant portion of the daily market trades, deploying machine learning and data-driven methods to enable large-scale transactions on these highly liquid markets. These automated transactions have dramatically transformed the dynamics of the markets, resulting in effects such as increased liquidity due to high trading rates, and a rise in market correlations, where multiple assets move in tandem to each other (Addy et al., 2024). To this end, we hypothesize that less-explored areas of the market may offer new opportunities where reinforcement learning can provide a greater advantage compared to its deployment in well-established sections. Such assets are characterized by their low trading volumes and narrow audience of traders, promising a more pristine trading environment which is less dictated by the effects of mainstream automated trading.

[1] `https://github.com/WilliamEkberg/super-trading-bot`

Our goal is to investigate the efficacy of different RL algorithms and MDP formulations in these niche markets. We seek to gauge their performance in such environments, and determine which ones yield the highest profits. Using the results of this investigation, we aim to establish the potential of RL to learn profitable strategies in niche, underutilized markets. This paper is structured as follows:

- Section 2 gives a brief summary of related work within this domain.
- Section 3 presents our methodology, outlining the datasets, algorithms and MDP formulations.
- Section 4 details our experimental setup and simulation results.
- Section 5 concludes this paper.

2. Related Work

There exists a large corpus of literature within the field of learning-based trading. Modern methods mostly rely on neural architectures designed for time-series modeling, such as Recurrent Neural Networks (RNNs), LSTMs and Convolutional Neural Networks (CNNs) (Selvin et al., 2017). Such methods are often trained using features extracted from sliding-window observations over the asset data. The Transformer architecture was later introduced to address the common issues of RNNs, LSTMs and CNNs, such as exploding/vanishing gradients and information loss due to pooling (Wang et al., 2022). Reinforcement learning approaches to stock trading unify the prediction and allocation steps, by modeling the market as an MDP in which an agent can execute transactions. Deep Q-Learning and Double Q-Learning are two popular algorithms in the RL setting, serving as a basis for many frameworks (Théate & Ernst, 2021; Ning et al., 2021; Carta et al., 2021; Massahi & Mahootchi, 2024).

3. Approach

We choose Deep Q-Learning (Mnih et al., 2013) and Double Q-Learning (Mnih et al., 2013) as our main RL algorithms. The environment in which the agent acts is a time series of market data, which in its most basic form consists of the closing prices of a given stock after each day. We augment this information using another time series representing a related market index to enrich the state representation. Given our environment, we define the MDP for the trading task as follows:

- **Action Space $\mathcal{A}$:** We investigate three definitions of the action space $\mathcal{A}$, each governing how the agent allocates percentages of its portfolio to the market:

  1. **Discrete Allocation Levels**: The agent selects from a fixed set of investment percentages: $\{0\%, 10\%, 20\%, \ldots, 100\%\}$.
  2. **Binary Market Participation**: The agent chooses between staying out of the market (0% invested) or being fully invested (100% invested).
  3. **Incremental Adjustment**: The agent can increase, decrease, or maintain its current investment level in increments of 10%.

  Each action space represents a different trade-off between flexibility and complexity, impacting the agent's learning dynamics and portfolio performance, as discussed in section 3.1.

- **State Space $\mathcal{S}$:** Given a set window of size k, we define the state space as the set of windows of the same size trailing each closing price in the time series, i.e. for an entry x_t, the corresponding state $s_t = \{x_{t-k}, ..., x_t\}$. We pad any negative time steps by copying x_0. The raw price values at each time point are then augmented by transforming each entry in a window as

$$x'_t = \sigma(x_t - x_{t-1}), \tag{1}$$

  where x'_t is the transformed value of x_t and σ denotes the sigmoid function. The difference between subsequent values acts as a finite-difference approximation, providing information to the model about the direction of the market at each time step. To mitigate large disparities between inputs, the sigmoid function is applied to squeeze the value of the state variables between 0 and 1. Furthermore, we apply the same procedure to an index related to the asset we aim to predict. This additional information is then concatenated to the state vector to encode insights into larger market movements.

  A slight modification to the state space is made for the action space denoted "Incremental Adjustment". In that case, it might be important for the agent to know its current investment position, as it cannot fully exit or enter the market within a single day. Therefore, the current investment percentage is added to the state space.

- **Rewards $\mathcal{R}$:** The agent's reward is determined by the change in its portfolio value from the previous to the current day. This change is then normalized by a hyperparameter η and squeezed into a value between 0 and 1 using Tanh.

$$r_t = \tanh(\frac{\text{portfolio change}}{\eta}). \tag{2}$$

  For action space denoted "Discrete Allocation Levels", the reward is modified by incorporating a risk aversion function. This modification incentivizes the agent to adopt a more conservative allocation when uncertainty about the portfolio's movement is high. In that case, the agent receives a higher expected reward for choosing lower investment percentages. This should reduce the variance of it's performance and ensures that the agent makes use of all its actions. The exact formulas are detailed in section 3.2.

- **Dynamics $\mathcal{P}$:** For a given time t, the transition dynamics are defined as

$$P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_t) = 1. \tag{3}$$

  In other words, the states transition deterministically from one time step to the next, no matter the action of the agent at the previous time step (which is what a real-world scenario would look like, assuming that the volume being traded is low enough to be negligible in relation to the rest of the market). In the case of "Incremental Adjustment", the invested percentage in the next state is determined by the current action but dynamics are still deterministic.

The MDP is defined in these terms to compare different strategies of the agent. The first action space, "Discrete Allocation Levels," provides the agent with maximum flexibility in managing its portfolio, closely resembling the options of an actual investment manager who can invest in only a single stock. As the agent learns Q-values, it would, in theory, converge to investment allocations of either 0% or 100%. A brief proof supporting this behavior is presented in the following lemma.

Lemma 1 *Assume that $R(s_t, 100\%) > 0$, and the portfolio allocation percentage $0 \leq p < 100\%$. Then*

$$Q^\pi(s_t, 100\%) > Q^\pi(s_t, p), \; \forall s_t \in \mathcal{S}. \tag{4}$$

Proof:

$$\begin{aligned}
Q^\pi(s_t, 100\%) &= R(s_t, 100\%) + \gamma V^\pi(s_{t+1}) \\
Q^\pi(s_t, 100\%) &> pR(s_t, 100\%) + \gamma V^\pi(s_{t+1}) \\
Q^\pi(s_t, 100\%) &> Q^\pi(s_t, p) \qquad \blacksquare
\end{aligned} \tag{5}$$

Lemma 1 show that if the expected reward of investing into the market is greater than 0, then the agent maximizes it's expected reward by investing it's total portfolio. If $R(s_t, 100\%) < 0$, than it's trivial that staying out of the market is the only option that prevents the agent from the loss, as it is not able to invest a negative percentage. If $R(s_t, 100\%) = 0$, there is no difference between the actions of the agent. Therefore, to maximize it's expected reward the only acceptable options become 0% and 100%.

3.1. Action and State Space Design

The second action space, "Binary Market Participation," was motivated by Lemma 1. Reducing the number of actions the agent needs to learn might additionally enhance performance. When using this action space, we use $\eta = 150$ for the reward. In experiments with both Discrete Allocation Levels and Binary Market Participation, we observe that the agent tend to engage in what is known as "day trading," where the agent attempts to predict whether the market will rise or fall the following day. In a real portfolio management scenario, however, one might prefer the agent to adopt a more long-term strategy, which would be easier to interpret. The "Incremental Adjustment" strategy aims to encourage this behavior. Under this approach, the agent is restricted to buying or selling only 10 percent of its portfolio at a time. Additionally, the agent has the option to hold its current position. This smaller adjustment in portfolio allocation could potentially reduce the variance of the performance.

Since the defined action space requires the agent to take at least 10 days to fully enter or exit the market, the agent's current portfolio becomes crucial information for determining the next step. As a result, we incorporated the past investment percentage into the current state of the MDP. Additionally, due to the altered behavior associated with this strategy, we adjusted the reward function slightly, using $\eta = 40$.

Using this defined MDP, the goal is for the agent to learn a trading policy based on past market data, which can then be generalized to novel data from the same asset. The customized DQN algorithm tailored to our MDP is provided in Algorithm 1.

Remark 1 *The implementation of Double-DQN is identical to DQN, with the exception that* $y_i = r(\phi_j, a_j) + \gamma Q_\theta(\phi_{j+1}, \arg\max'_a)$. ◇

Remark 2 *The design of our state space was inspired by the method outlined in* `https://github.com/pskrunner14/trading-bot/tree/master`. ◇

Algorithm 1 Deep Q-Learning For Stock Trading

```
Initialize replay buffer D with capacity N,
Initialize neural network $Q_\theta$,
Initialize target neural network $Q_{\theta^-}$,
Set $\theta = \theta^-$
for episode = 1, M do
  Load the first time window from the asset data $w_1$
  Pre-process $w_1$ to obtain $\phi_1 = \phi(w_1)$.
  for $t = 1, T$ do
    With probability $\epsilon$ select a random action $a_t$
    otherwise select $a_t = \arg\max_a Q_\theta(\phi(w_t), a)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and state $w_{t+1}$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_i = \begin{cases} r(\phi_j, a_j), & \text{for terminal } \phi_{j+1} \\ r(\phi_j, a_j) + \gamma \max_{a'} Q_\theta(\phi_{j+1}, a'), & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_i - Q_{\theta^-}(\phi_j, a_j))$
    Every C steps set $\theta^- = \theta$
  end for
end for
```

3.2. Risk Aversion Penalty

In our initial tests, the agent learned to select only up to three distinct actions for interacting with the MDP. The underlying idea for partial investment in the market is to reduce risk. To address both the practical and theoretical challenges, we implemented a risk aversion function before squeezing the reward to a range between 0 and 1.

$$r_{\text{risk aversion}} = -\exp[-\frac{\text{portfolio change}}{150}] + 1 \tag{6}$$

$$r = \tanh(2 * r_{\text{risk aversion}}) \tag{7}$$

Risk aversion is implemented using a concave function, such as $f(x) = -e^{-x}$, which induces heavy penalties for low rewards while being robust to outliers in the high reward domain. This approach discourages excessive risk-taking by disproportionately punishing negative outcomes. Adding an offset of 1 ensures that the reward is zero if the portfolio does not change.

3.3. Datasets

The datasets used in this study can be viewed as the environment in which the agent operates. We assume that the agent's actions, such as buying or selling a stock, do not impact the market. This assumption is reasonable since, even in smaller markets, the total trading volume significantly exceeds the agent's trades. The initial portfolio volume of our agents is 10,000 SEK (Swedish Krona), roughly equivalent to $1,000 based on the exchange rate as of March 17, 2025.

The datasets used in this study are as follows:

- **FirstNorth Growth Market**: The FirstNorth Growth market encompasses roughly 550 traded nordic companies (fir, 2025). We use its closing value over the past 6 years for training and validation. The first 5 years (2019-2024) are used as training data, and the last year (2024-2025) as test data. The estimated trading volume of this asset is around 30 million dollars per day, based on the combined trading volume of its underlying assets. Both the training data and the test data can be seen in Figure 1.
- **Novotek**: A Swedish IT and automation company with a low daily trading volume of around 30k dollars (nov, 2025). The same split into 5-year training data (2019-2024) and 1-year test data (2024-2025) was made for Novotek. Both the train data and the test data can be seen in Figure 2.
- **OMX 30**: Sweden's largest stock index, serving as a useful indicator of the country's economic performance (omx, 2025). This dataset, obtained from Yahoo Finance, was used to provide additional context to the state representation.

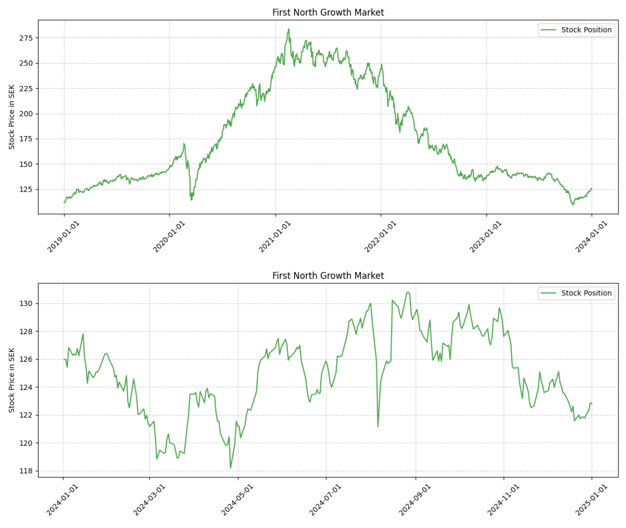

Figure 1. The train data (above) and the text data (below) of the First North Growth Market.

By holding the stock for the full duration with an initial investment of 10,000 SEK (Swedish kronor), the starting

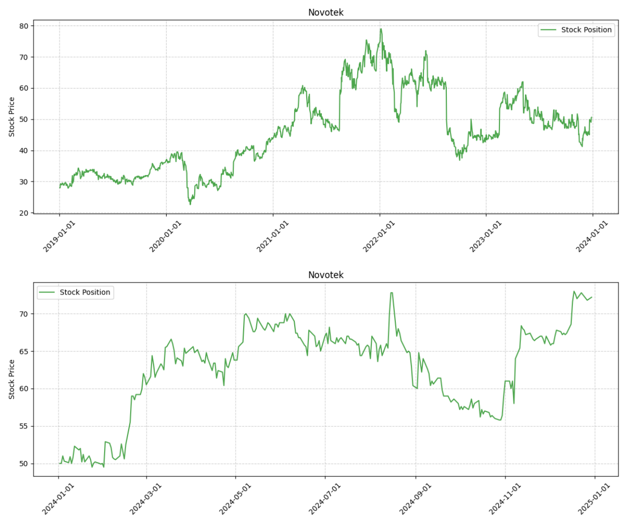

Figure 2. The train data (above) and the text data (below) of Novotek.

capital of our agents, one would have generated the following profits.

- FirstNorth growth market within 1 year: -249.64 SEK
- FirstNorth growth market within 5 years: 1246.89 SEK
- Novotek within 1 year: 4440.00 SEK
- Novotek within 5 years: 8068.20 SEK

3.4. Parameterizing the Q-Function using a Transformer

In standard Deep Q-Learning, the Q-function is parameterized using feed-forward multi-layer perceptrons. In addition to the standard approach, we implement a transformer variant of the Q-function. This involves modifying the state space by representing the observation as a sequence of feature embeddings derived from each time step within the sliding window. This experimental implementation is motivated by the strong time-series modeling capabilities of Transformers. We intend to investigate whether the attention mechanism can help the model generalize better compared to plain feed-forward architectures and potentially lead to higher returns.

4. Experimental Results

4.1. Experimental Setup

For our experiments, we implemented a five-layer feed-forward neural network for both DQN and Double-DQN and a transformer-based model. The neural network architecture consists of an input layer corresponding to

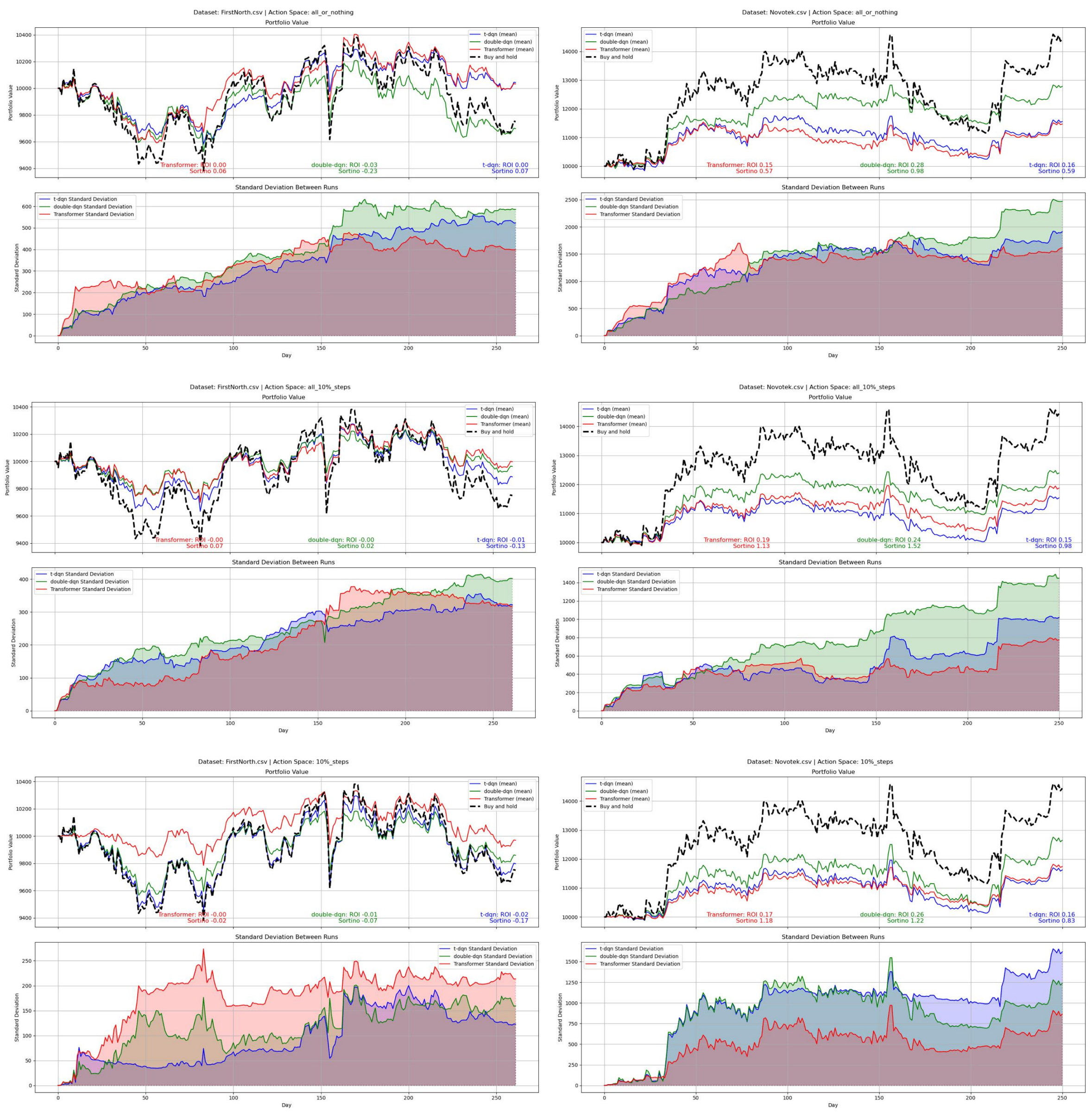

Figure 3. Mean portfolio values and their standard deviations under the different polices learned by the respective RL algorithms. The left column shows the results on the First North dataset, while results for Novotek is shown in the right column.

the observation space, followed by three hidden layers of size 40, and an output layer corresponding to the action space. The observation space is represented as a vector of size $2(\text{window size} - 1)$, with the first half corresponding to the current dataset and the second half to OMX30. In the case of "Incremental Adjustment," an additional value representing the current investment percentage is included. The action space consists of 11, 2, or 3 discrete actions, depending on the strategy. Each layer of the feed-forward network consists of a fully connected linear layer, ReLU activation functions, and layer normalization. The hyperparameters were fine-tuned for each method to ensure effective learning while preventing overfitting to the training data. The transformer model was configured with 3 layers and 2 attention heads. Due to having a higher number of parameters, it was trained for three epochs, whereas the feed-forward neural network was trained for seven epochs. This distinction is crucial, as transformers tend to overfit more quickly due to their parameter complexity. All agents use discount factor of $\gamma = 0.95$ and an epsilon greedy exploration strategy. For the "Binary Market Participation" strategy with the feed-forward network, the exploration parameters are set as follows: an initial exploration rate of 0.5, an exploration decay factor of

0.995, a minimum exploration rate of 0.05, and a learning rate of 10^{-4}. Given the higher parameter count in the transformer model, we reduce the learning rate to 10^{-6} and set the initial exploration rate to 1 to accommodate the lower number of training epochs. For the Binary Market Participation and Incremental Adjustment action space, we increased the initial and minimum exploration rate, reduced the decay rate and increased the learning rate slightly. These modifications account for the larger state space in "Incremental Adjustment" and the broader action space in "Discrete Allocation" Levels.

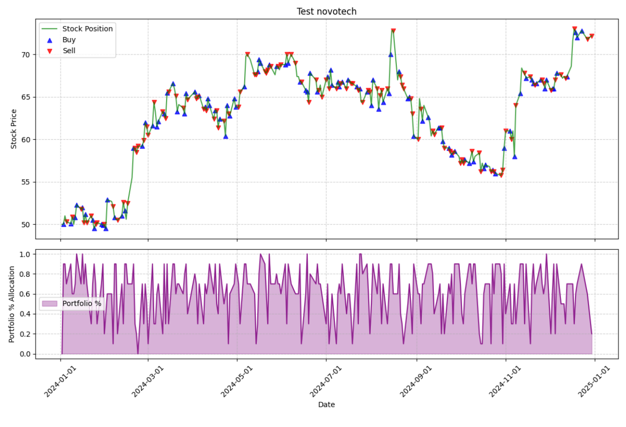

Figure 4. Behavior of an agent in "Discrete Allocation Levels"

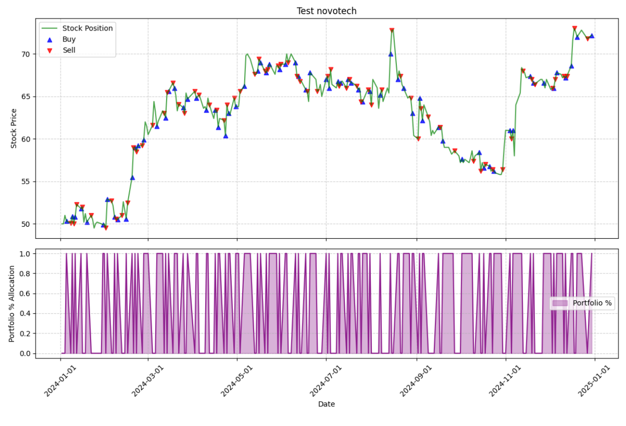

Figure 5. Behavior of an agent in "Binary Market Participation"

Figure 4 to 6 illustrate the behavior of agents across different action spaces, highlighting how the available actions influence trading decisions. The agent exhibits a clear tendency toward day trading, particularly in the "Binary Market Participation" setting, as shown in figure 5. In contrast, the "Incremental Adjustment" strategy, depicted in figure 6, constrains the agent to adopt a more long-term investment approach. "Discrete Allocation Levels" provide the greatest flexibility, allowing the agent to make a broader range of

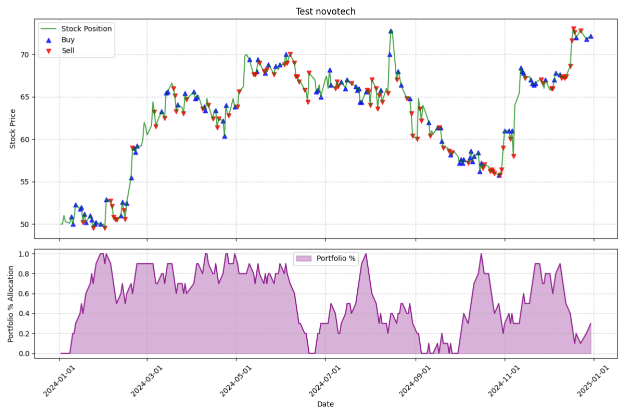

Figure 6. Behavior of an agent in "Incremental Adjustment"

decisions. Notably, the agent only occasionally exits the market entirely under this strategy.

The experiment was conducted by running 11 trials for each agent. The averaged results of all the runs are presented in Figure 3 (the upper part of each graph). The standard deviation of the results across all runs was calculated for each time step and displayed below the averaged results to provide an indication of the agents' consistency. The Return on Investment (ROI) was calculated for each run and averaged to compare the expected performance of the agent.

$$\text{ROI} = \frac{\text{Net Profit}}{\text{Cost of Investment}} \tag{8}$$

The Sortino Ratio is a risk-adjusted performance measure that takes into account the standard deviation of negative rewards. For simplicity, we assumed a risk-free rate of 0. The Downside Deviation measures the standard deviation of each run, but only considers negative rewards. This metric is particularly useful, as large positive rewards do not negatively affect the agent's performance and should not be penalized. The Sortino Ratios across all runs were averaged to compare the expected performance of the agent.

$$\text{Sortino Ratio} = \frac{\text{Annual Reward} - \text{Risk-Free Rate}}{\text{Downside Deviation}} \tag{9}$$

$$\text{Downside Deviation} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\max(0, -R_i))^2} \tag{10}$$

4.2. Results

Table 1. Comparison of ROI and Sortino Ratios for FirstNorth

Action Space	Method	ROI (%)	Sortino Ratio
None	Buy and hold	-0.025	-0.17
Binary	DQN	0.00	0.07
Binary	Double DQN	-0.03	-0.23
Binary	Transformer	0.00	0.06
Discrete	DQN	-0.01	-0.13
Discrete	Double DQN	0.00	0.02
Discrete	Transformer	0.00	0.07
Incremental	DQN	-0.02	-0.17
Incremental	Double DQN	-0.01	-0.07
Incremental	Transformer	0.00	0.02

Table 2. Comparison of ROI and Sortino Ratios for Novotek

Action Space	Method	ROI (%)	Sortino Ratio
None	Buy and hold	0.44	1.53
Binary	DQN	0.16	0.59
Binary	Double DQN	0.28	0.98
Binary	Transformer	0.15	0.57
Discrete	DQN	0.15	0.98
Discrete	Double DQN	0.24	1.52
Discrete	Transformer	0.19	1.13
Incremental	DQN	0.16	0.83
Incremental	Double DQN	0.26	1.22
Incremental	Transformer	0.17	1.18

First North Growth Market:

Buy-and-hold baseline: The baseline strategy of simply buying and holding the index throughout the year 2024-2025 would have resulted in a slight loss of 2.5% in portfolio value, as shown in Table 1.

Action-Space Comparisons:

- **Binary space:**

  As shown in figure 3 (top-left), the RL agents occasionally avoided deeper downward dips. Both the DQN and the transformer-based models outperformed the price of the underlying asset. The ROI for Double-DQN (-0.03) was close to the buy and hold ROI at -0.025 compared to the Transformer and DQN with an ROI of 0.00 seen in table 1. Additionally, the transformer model had a slightly lower standard deviation than both DQN and Double-DQN for this strategy.

- **Discrete Allocation Levels:** As seen in Table 1, both Double-DQN and the Transformer-based model achieved an ROI of 0.00%, matching or slightly improving on DQN's -0.01%. The Transformer-based policy had a modest improvement in the Sortino Ratio over DQN, suggesting it better handled downside volatility. Also the standard deveation of both the transformer based model and DQN was slightly lower then for double-DQN as shown in figure 3 (middle-left). However, all the models outperformed the underlying asset.

- **Incremental:** Using the incremental action space, table 1 shows that the Transformer-based model performed the best, with a 0.00% ROI and a small positive Sortino Ratio of 0.02, indicating it managed to avoid significant drawdowns. By contrast, DQN and Double DQN hovered slightly below zero with -0.02% and -0.01% respectively. In terms of standard deviation the Transformer-based model had a higher value compared to DQN and double-DQN.

Novotek stock:

Buy-and-hold baseline: The baseline strategy of buying and holding the index over the year 2024-2025 would have resulted in a profit of 44% of portfolio value seen in table 2.

- **Binary:** As shown in figure 3 (top-right), the models captured portions of Novotek's strong upward trend even though they traded into and out of the market. According to Table 2, Double-DQN performed the best with a final ROI of 0.28%, compared to 0.16% for DQN and 0.15% for the Transformer. Double-DQN exhibited a better Sortino Ratio of 0.98 than DQN or Transformer, although it had a slightly higher standard deviation between runs. None of the models outperformed the buy-and-hold baseline of 0.44%.

- **Discrete Allocation Levels:** In Table 2, Double DQN again achieved the highest ROI of 0.24% among RL approaches, followed by the Transformer-based method with 0.19% and DQN at 0.15%.The Double-DQN method also had a Sortino Ratio of 1.52, which is almost the same as the buy-and-hold Sortino ratio of 1.53. The Transformer's Sortino Ratio of 1.13 was also quite competitive, indicating it handled downside volatility well, but it was still below the buy-and-hold ratio of 1.53. As shown in Figure 3 (middle-right), all three RL methods had reduced variance compared to an all-in strategy, but they gave up some absolute gains in a strongly bullish market.

- **Incremental:** Using the incremental strategy, the agents captured large portions of the upside while at the same time keeping the standard deviation down. Table 2 shows Double-DQN at 0.26% ROI, DQN at 0.16%, and the transformer-based method at 0.17%. Although none matched the 0.44% buy-and-hold result, the risk-adjusted performance was relatively good. Double DQN had a Sortino Ratio of 1.22, suggesting it avoided some significant drawdowns. Figure 3 (bottom-right) shows that the Transformer's standard deviation among runs was somewhat lower than the other RL methods, suggesting a more stable policy learning but with slightly lower average returns.

4.3. Holistic Analysis

Noticeable patterns to be extracted from the above analysis of the results include the observation that Double DQN generally delivers more robust performance, validating prior findings that it mitigates overestimation bias and offers better risk-adjusted returns. Transformer-based Q-learning often matched or surpassed DQN in downward or more volatile conditions. This supports the superior time series modeling capabilities of the transformer architecture, which in this case might have enabled the model to learn a more optimal Q-function. Finally, action-space design (Discrete Allocation Levels, Binary Market Participation, or Incremental Adjustment) significantly impacts both returns and volatility control. For downward and volatile markets, e.g the FirstNorth index, partial or incremental exposures help preserve capital, shielding against overly negative ROIs. In a strongly rising market, however, buy-and-hold remains hard to beat in terms of absolute ROI.

5. Conclusion

This study demonstrates that simpler reinforcement learning methods hold potential for trading in niche markets characterized by lower daily trading volumes and reduced algorithmic competition. We showed that incorporating risk aversion is crucial for enabling full portfolio management flexibility, whereas in a reduced action space, pure reward signals may suffice. Through numerical experiments, we demonstrate that our methods effectively protect portfolios in declining markets while capitalizing on opportunities presented by upward trends.

6. Team Contribution

Our team collaborated constructively, supporting each other throughout the project. We all contributed to developing the initial trading agent. John set up the new datasets and expanded the observation space for the final version. William focused on clear data visualization, selecting datasets, choosing markets, and ensuring all experiments could be executed together. Nils implemented the agent's action spaces and explored different subsets of parameters for the models. Additionally, we all contributed to writing the report and creating the poster, working collaboratively to support one another throughout the process.

References

First north all-share sek, 2025. URL https://www.nasdaq.com/european-market-activity/indexes/firstnorthsek?id=SE0001718701.

Novotek, 2025. URL https://www.novotek.com/.

Omx 30, 2025. URL https://finance.yahoo.com/quote/%5EOMX/.

Addy, W. A., Ajayi-Nifise, A. O., Bello, B. G., Tula, S. T., Odeyem, O., and Falaiye, T. Algorithmic trading and ai: A review of strategies and market impact. *World Journal of Advanced Engineering Technology and Sciences*, 11(1):258–267, 2024.

Carta, S., Ferreira, A., Podda, A. S., Recupero, D. R., and Sanna, A. Multi-dqn: An ensemble of deep q-learning agents for stock market forecasting. *Expert systems with applications*, 164:113820, 2021.

Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2016.

Massahi, M. and Mahootchi, M. A deep q-learning based algorithmic trading system for commodity futures markets. *Expert Systems with Applications*, 237:121711, 2024.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M.

Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Ning, B., Lin, F. H. T., and Jaimungal, S. Double deep q-learning for optimal execution. *Applied Mathematical Finance*, 28(4):361–380, 2021.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V. K., and Soman, K. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pp. 1643–1647. IEEE, 2017.

Théate, T. and Ernst, D. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173:114632, 2021.

Wang, C., Chen, Y., Zhang, S., and Zhang, Q. Stock market index prediction using deep transformer model. *Expert Systems with Applications*, 208: 118128, 2022.