Adaptive Inventory Control under Non-Stationary Retail Demand: a Deep Reinforcement Learning Study on the M5 Dataset

Yongjia Xie
Xi'an Jiaotong-Liverpool University, Suzhou, China

Abstract:Retail demand is not stable over time. Its average level and its variability change because of trend, seasonality, promotions, and shifts in consumer behavior. A fixed inventory rule that is tuned on past demand can become wrong when the demand level moves to a new range. This paper studies that problem with real data. We take one high volume product from the public M5 Walmart dataset and build a daily, single product, lost sales inventory environment that is driven by the real demand of that product. We train a deep reinforcement learning agent with Proximal Policy Optimization to set the daily order quantity. We compare the learned policy against two classic policies, a base stock policy and an (s, S) policy, both tuned on the training period. The held out test period for this product has a much lower mean demand than the training period, which makes it a clear test of adaptation. The deep reinforcement learning policy lowers the total cost on the test period by 13.5 percent, from 19705 to 17052 in the same cost units. Almost all of this gain comes from holding cost, which the agent cuts by about 48 percent because it stops over stocking once demand falls. The cost gain comes with a lower service level. The fill rate of the learned policy is 88.2 percent against 99.1 percent for the fixed rule, which reflects the chosen cost weights. The result shows that an adaptive learned policy can track a new demand level and reduce cost, while a fixed policy keeps acting on an old level. We report the full setup, the data statistics, and the results so the study can be reproduced.

Keywords: Inventory Optimization; Non-Stationary Demand; Deep Reinforcement Learning; Proximal Policy Optimization; M5 Dataset; Retail Supply Chain; Service Level

1. Introduction

Inventory control sits at the center of retail operations. The manager must decide how much to order and when, so that demand is met at a low cost. Two costs pull in opposite directions. Holding too much stock raises the holding cost. Holding too little stock causes stockouts and lost sales. A good policy balances the two. The classic answer is a parametric rule such as a base stock level or an (s, S) pair. These rules are simple and they have strong theory behind them when demand is stable and well behaved.

Real retail demand is often not stable. Figure 1 shows the daily demand of a single grocery product over five years. The average level rises, peaks, and then falls to a lower band. The variability changes with the level. There are weekday effects and month effects. This kind of demand is non-stationary, which means its statistical pattern itself changes over time. A fixed rule has a problem here. The rule is tuned on history. If the demand level later moves to a new range, the fixed rule keeps ordering for the old range. It can carry too much stock when demand falls, or run short when demand rises.

Deep reinforcement learning offers a way to handle this. A reinforcement learning agent does not use a single fixed parameter. It learns a policy that maps the current state of the system to an action. If the state includes recent demand and a short forecast, the agent can change its orders as the demand level moves. Recent studies report that deep reinforcement learning can match or beat classic inventory policies, and that it is well suited to demand that keeps changing.

This paper makes the question concrete with real data and a clean experiment. We pick one high volume product from the M5 Walmart dataset. We build an inventory environment that steps through the real daily demand of this product. We train a Proximal Policy Optimization agent to order each day. We compare it against a base stock policy and an (s, S) policy that are both tuned on the same training window. We then test all three on a held out window that the policies did not see during tuning or training.

The contributions of this paper are as follows. First, we run a fully reproducible empirical study on public data. We name the exact product, the dates, the cost parameters, and the train and test split, so others can repeat the work. Second, we show a clear and honest result. The learned policy lowers total cost by 13.5 percent on the test window. The gain comes mostly from holding cost, which the agent cuts by about 48 percent. The reason is adaptation. The test window has a lower demand level than the training window, and the agent scales its orders down while the fixed rule does not. Third, we report the cost against service trade off in full. The learned policy reaches a lower cost at a lower fill rate. We explain why this happens through the cost weights, and we show how to move the operating point.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 defines the problem. Section 4 describes the data and the demand analysis. Section 5 presents the method. Section 6 reports the experiments and results. Section 7 discusses the findings and limits. Section 8 concludes.

2. Related Work

2.1 Demand Forecasting and Inventory with Learning Methods

Demand forecasting is the base of inventory decisions. Classic models such as moving average, exponential smoothing, and ARIMA work well for stable series but lose accuracy when the pattern changes. Recent reviews of supply chain forecasting report a clear move from statistical models to machine learning and deep learning. A 2024 review in Applied System Innovation by Douaioui and colleagues studied 119 papers from 2015 to 2024 and found that deep learning methods now lead the field for demand forecasting in supply chains. Long Short-Term Memory networks, proposed by Hochreiter and Schmidhuber in 1997, are strong at learning long range patterns in time series. The Temporal Fusion Transformer, proposed by Lim and colleagues in 2021, mixes static features, known future inputs such as holidays, and observed history, and it produces both forecasts and useful interpretation.

2.2 Deep Reinforcement Learning for Inventory Decisions

Deep reinforcement learning frames inventory control as a sequential decision problem. The agent reads the system state, picks an action, and gets a reward. Oroojlooyjadid and colleagues built a deep Q network for the beer game and showed strong results against classic policies. Geevers and colleagues applied deep reinforcement learning to multi echelon inventory optimization in 2024. Proximal Policy Optimization, proposed by Schulman and colleagues in 2017, is a stable policy gradient method that works with continuous actions, and it is now a common choice for this kind of control.

The closest work to ours is by Dehaybe, Catanzaro, and Chevalier, published in 2024 in the European Journal of Operational Research. They study inventory optimization under non-stationary and uncertain demand, and they show that a deep reinforcement learning policy adapts to demand that keeps changing, where a fixed policy fails. Our study follows that theme, but we ground it in a named real product from the M5 dataset and report the exact trade off.

2.3 Joint Forecasting and Inventory, and Resilience

A growing line of work joins forecasting and inventory in one system rather than running them in separate steps. Yang and colleagues, in a 2025 paper in Sensors, built a multi agent deep reinforcement learning framework that jointly optimizes demand forecasting and inventory management in retail supply chains. They combine transformer based demand modeling with reinforcement learning agents that coordinate inventory across the network. On the resilience side, Lu and colleagues, in a 2025 study in Symmetry, proposed a multi echelon inventory framework based on Proximal Policy Optimization that builds the frequency, duration, and impact of disruptions directly into the optimization. These works show that deep reinforcement learning is a strong tool both for changing demand and for disruption.

2.4 Position of This Paper

Many works report results on private or simulated data, which makes them hard to repeat. We use a public dataset and a named product, and we report every parameter. We also keep the model simple so the effect of adaptation is easy to see. This makes the study a clean and honest reference point.

3. Problem Definition

We study a single product at a single stocking location. Time is discrete and one period is one day. At the start of each day the system has on hand inventory and a pipeline of orders that were placed earlier and have not yet arrived. The manager chooses an order quantity for the day. The order arrives after a fixed lead time of two days. Demand that is not met from stock on hand is lost. This is a lost sales model, which fits fast moving retail.

The cost in each day has three parts. The holding cost is charged on the inventory that remains at the end of the day. The shortage cost is charged on each unit of demand that is not met. The ordering cost is the unit cost paid for each ordered unit. In this study the holding cost is 0.1 per unit per day, the shortage cost is 1.0 per unit, and the unit ordering cost is 0.2 per unit. There is no fixed cost per order. These weights set the balance between stock and service. The ratio of shortage cost to the sum of shortage and holding cost is 1.0 divided by 1.1, which is about 0.909. In newsvendor terms this implies a cost optimal service level near 91 percent. We return to this number when we read the results.

We frame the problem as a Markov Decision Process. The state contains the recent demand history, the on hand inventory, the pipeline, a short demand forecast, a weekday signal, and a promotion flag. The action is the order quantity for the day. The reward is the negative of the total daily cost. The goal is a policy that maps the state to an order and that minimizes the expected total cost over a long horizon while keeping a good service level. Service level is measured by the fill rate, which is the share of demand met from stock on hand.

4. Data and Demand Analysis

4.1 Source and Product

We use the M5 Forecasting Accuracy dataset from the M5 competition organized by Makridakis and colleagues. It holds daily unit sales for Walmart stores in three United States states, with three data files for sales, prices, and the calendar. From this dataset we select a single product to manage. We pick the product with the highest total sales, which gives a series with enough volume for a meaningful inventory study. The selected product is FOODS_3_090 in store CA_3, in California. The series runs from 29 January 2011 to 24 April 2016, which is 1913 daily observations.

4.2 Demand Statistics

Table 1 reports the demand statistics. The mean daily demand is about 131 units and the standard deviation is about 109 units. The coefficient of variation is 0.83, which is high. The series has 359 days with zero demand, which is 18.8 percent of all days. Most of these zeros fall in the early part of 2011, before the product was actively stocked in the store. After that period the demand becomes regular and high.

Table 1. Demand Statistics for FOODS_3_090 in Store CA_3.

Statistic	Value
Days	1913
Date range	2011-01-29 to 2016-04-24
Mean daily demand	**130.95 Units**
Standard deviation	**108.61 Units**
Median	126 units
Maximum	763 units
Coefficient of variation	0.83
Zero demand days	359 (18.8 percent)

4.3 Evidence of Non-stationarity

The demand is non-stationary in the way that matters for inventory, which is a shifting mean and a shifting variance over time. Figure 1 shows the daily demand with a 28 day rolling mean and a 28 day rolling standard deviation. The rolling mean rises through 2012, reaches a clear peak in late 2013, and then settles into a lower band from 2014 onward. The rolling standard deviation follows the same shape, so the variability is tied to the level.

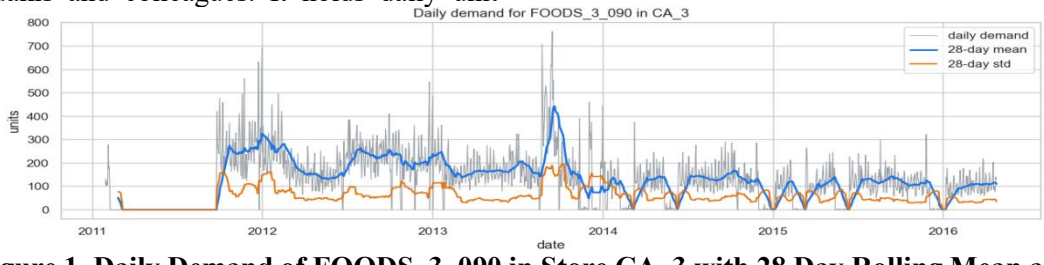

Figure 1. Daily Demand of FOODS_3_090 in Store CA_3 with 28 Day Rolling Mean and Standard Deviation.

We confirm the shift with a simple segment analysis. We split the series into three equal parts in time. The mean of the first third is 140.3 units, the second third is 155.0 units, and the last third is 97.5 units. The standard deviation falls from 130.0 in the first third to 64.3 in the last third. The demand level in the last part of the history is much lower and much less variable than in the middle part. This is the kind of regime change that breaks a fixed inventory rule. We also ran the Augmented Dickey-Fuller test. The test statistic is about minus 4.17 and the p value is about 0.0007. This rejects the presence of a unit root, so the series is not a random walk. We note this for completeness. The unit root test and the property that matters here are not the same thing. A series can have no unit root and still have a mean and a variance that move across long segments, as the rolling plot and the segment analysis show. For inventory, the moving level is what creates the risk for a fixed policy, and that is what the segment results capture.

4.4 Seasonality

Figure 2 shows the weekday and month patterns. Demand is highest on weekend days. The mean on the first weekday key, which is Saturday in the M5 calendar, is about 175 units, while the mid week low is about 105 units. There is also a month pattern. Demand peaks in September at about 191 units and is lowest in May at about 84 units. These patterns are stable enough to be useful features for the agent, so we include a weekday signal in the state.

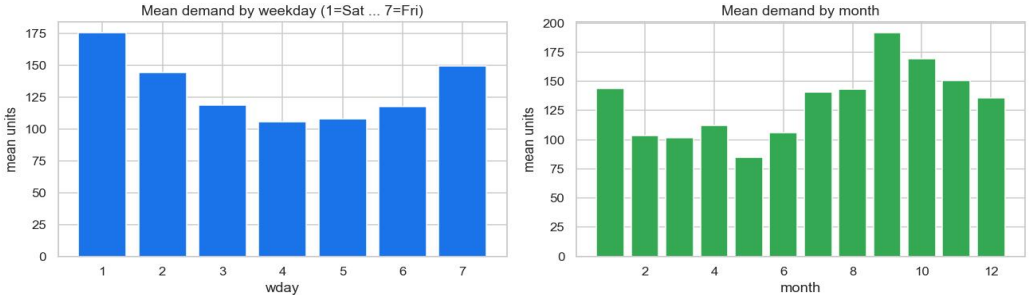

Figure 2. Mean Demand by Weekday (left) and by Month (right).

Figure 3 shows the autocorrelation of the demand. It has clear spikes at multiples of seven days, which match the weekly cycle, and a slow decay, which reflects the persistence of the demand level. These features support the use of recent demand lags and a weekday signal in the state of the agent.

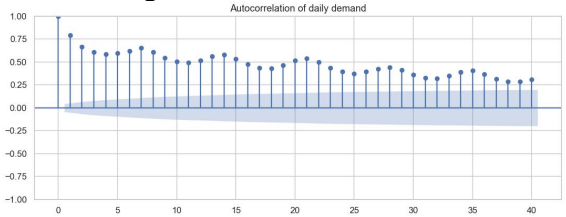

Figure 3. Autocorrelation of the Daily Demand up to Lag 40.

5. Method

5.1 Overview

We build a Gymnasium inventory environment that is driven by the real demand series of the product. The agent observes the state, chooses an order, and the environment returns the next state and the reward. We train the agent with Proximal Policy Optimization from the Stable-Baselines3 library. We then compare the trained policy against tuned base stock and (s, S) policies.

5.2 State, Action, and Reward

The state is a vector of 13 values. It has the demand of the last 7 days, the current on hand inventory, the total quantity in the pipeline, a short forecast equal to the mean demand of the last 28 days, a weekday sine and a weekday cosine, and the promotion flag for the day, which is the SNAP indicator for the store state. We normalize the inventory, pipeline, demand, and forecast values by the mean demand of the series. This keeps the inputs in a stable numeric range and helps training.

The action is a single value between 0 and 1. The environment scales it to an order quantity between 0 and an order cap. The order cap is three times the mean daily demand, which is wide enough to cover spikes. The reward is the negative of the total daily cost, which is the sum of holding, shortage, and ordering cost. We divide the reward by the mean demand so its size stays moderate during training. The cost values that we report in the results are kept in real units, not in the scaled reward units.

5.3 Dynamics

Each day the environment does the following steps. First it receives any order that was placed two days earlier and adds it to the on hand inventory. Second it places the new order chosen by the agent and adds it to the pipeline. Third it draws the real demand of the day from the series. Fourth it meets demand from stock on hand, and any unmet demand is lost. Fifth it computes the three costs and the reward. The environment then moves to the next day.

5.4 Training

We split the series in time. The first 80 percent, which is 1530 days, is the training window. The last 20 percent, which is 383 days from 8 April 2015 to 24 April 2016, is the held out test window. During training the agent starts each episode at a random day in the training window and runs for 180 days, so it sees many parts of the changing series. We train for 50000 steps with default Proximal Policy Optimization settings, a discount factor of 0.99, and a learning rate of 3 times 10 to the minus 4. We fix the random seed for repeatability.

6. Experiments and Results

6.1 Baselines and Metrics

We compare three policies. The base stock policy orders the inventory position up to a fixed level S every day. The (s, S) policy orders up to S when the inventory position falls below s. We tune S, and the pair s and S, by a grid search on the training window, choosing the values that give the lowest training cost. The deep reinforcement learning policy is the trained Proximal Policy Optimization agent. We measure total cost, broken into holding, shortage, and ordering cost, and we measure the fill rate, which is the service level. We also report a cost index, where the base stock cost is set to 100 for easy reading.

6.2 Training Behavior

Figure 4 shows the training reward over episodes. The scaled reward rises from about minus 400 at the start to about minus 120, and it then stays flat. The agent completed about 280 episodes in the 50000 steps. The flat tail shows that the policy reached a stable level and did not keep improving, so the training budget was enough for this single product.

Figure 4. PPO Training Reward over Episodes. the Blue Line Is a 20 Episode Moving Average.

6.3 Main Results

Table 2 reports the results on the held out window. The base stock policy and the (s, S) policy gave the same result. The grid search found that the best (s, S) pair behaves like the base stock policy for this product and this cost setting, so the two rows match. This happens when holding is cheap relative to shortage and the lead time is short, because then it is best to order back up to the target almost every day.

Table 2. Policy Results on the Held out Test Window (8 April 2015 to 24 April 2016).

Policy	Total cost	Holding	Shortage	Ordering	Fill rate	Cost index
Base-stock	19705.4	12026.6	334.7	7344.1	0.991	100.0
(s, S)	19705.4	12026.6	334.7	7344.1	0.991	100.0
DRL (PPO)	17052.2	6229.2	4326.5	6496.4	0.882	86.5

The deep reinforcement learning policy reached the lowest total cost. Its cost index is 86.5, which is a 13.5 percent reduction against the fixed rule. The gain is driven by holding cost. The learned policy carried far less stock, so its holding cost fell from 12026.6 to 6229.2, which is a drop of about 48 percent. Its ordering cost also fell a little, from 7344.1 to 6496.4. The trade off appears in the shortage cost and the fill rate. The learned policy let more demand go unmet, so its shortage cost rose from 334.7 to 4326.5, and its fill rate fell from 99.1 percent to 88.2 percent.

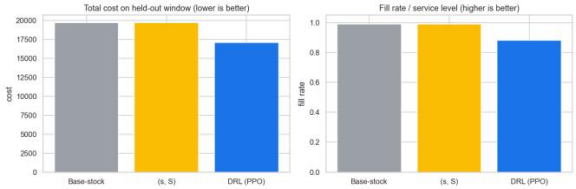

Figure 5. Total Cost (Left) and Fill Rate (Right) of the Three Policies on the Held out Window

6.4 Why the Learned Policy Wins

The reason for the gain is adaptation to the

demand shift. Recall from Section 4 that the demand level fell in the last part of the history. The test window has a mean demand of about 98 units per day, while the middle of the training period had a mean near 155 units per day. The base stock level S was tuned on the whole training window, so it is set for a higher demand level. When the test window arrives with lower demand, the fixed rule keeps ordering toward the high target and carries too much stock. This is why its holding cost is large and its service is near perfect at 99 percent.

The deep reinforcement learning agent does not use a single fixed target. Its state includes the recent demand and the 28 day forecast. When the demand level drops, these inputs drop, and the agent orders less. It tracks the new lower level and avoids the excess stock. This is the exact behavior that the reference work by Dehaybe and colleagues describes for non-stationary demand, now shown on a named real product.

6.5 Reading the Service Level

The lower fill rate of the learned policy is not a failure. It reflects the cost weights. With a holding cost of 0.1 and a shortage cost of 1.0, the cost optimal service level is near 91 percent, as we computed in Section 3. The base stock policy at 99 percent service is above this point, so it spends too much on holding to buy service that the cost weights do not justify. The learned policy at 88 percent service is close to the implied target, just slightly below it, which is why it reaches a lower total cost. In short, the fixed rule over serves and over spends, while the learned policy operates near the cost optimal point. A manager who wants higher service can raise the shortage weight in the reward, which will push the agent to hold more stock and lift the fill rate.

7. Discussion and Limitations

The study gives a clear message, but it also has limits that future work should treat. The first point is the size of the experiment. We studied one product at one store. This keeps the effect of adaptation easy to see and easy to reproduce, but it does not prove the result for a whole assortment. Different products have different demand shapes, lead times, and cost weights. The next step is to run the same method across many products and to report the distribution of the cost gain, not a single number.

The second point is the trade off between cost and service. The learned policy reached a lower cost by accepting more stockouts. This is correct under the given cost weights, but in practice many retailers set a hard service target. To respect such a target, the reward should be changed, for example by raising the shortage weight or by adding a penalty when the fill rate falls below a floor. We expect this to lift the service level at the cost of some holding, which moves the operating point along the same trade off.

The third point is that the base stock and (s, S) baselines collapsed to the same policy for this product and cost setting. This is a real and expected outcome for cheap holding and short lead time, but it means we did not test the (s, S) form in a regime where it differs from base stock. A product with a fixed order cost, or a longer lead time, would separate the two and give a richer baseline comparison.

The fourth point is that we used a fixed lead time and no supply side shocks. The demand side was real and non-stationary, but the supply side was simple. A natural extension is to add a stochastic lead time or a disruption window where the lead time grows, and then to measure how fast each policy recovers. This would turn the study into a resilience study, in the spirit of the recent disruption focused work.

The fifth point is training cost and safety. Deep reinforcement learning needs training and tuning, and a learned policy can behave in unexpected ways in states far from its training range. A safe deployment should keep a simple fallback rule that takes over in such states, and should test the policy carefully before it controls real orders.

8. Conclusion

This paper studied inventory control under non-stationary retail demand with real data. We took one high volume product from the public M5 Walmart dataset and built a daily, single product, lost sales inventory environment driven by its real demand. We trained a Proximal Policy Optimization agent to set the daily order and compared it against tuned base stock and (s, S) policies on a held out window. The learned policy reduced total cost by 13.5 percent. The gain came mostly from holding cost, which fell by about 48 percent, because the agent tracked the lower demand level in the test window while the fixed rule kept ordering for an old, higher level. The gain came with a lower fill rate, which

sits near the service level implied by the cost weights. The study shows that when demand is non-stationary, an adaptive learned policy can find the new operating point and cut cost, while a fixed policy cannot. Future work should scale the method to many products, add a service target to the reward, and extend the setting to supply side disruptions for a full resilience study.

Data and Code Availability

The demand data comes from the public M5 Forecasting Accuracy competition dataset on Kaggle. The product studied is FOODS_3_090 in store CA_3. The inventory environment, the training script, and the evaluation are implemented in Python with the Stable-Baselines3 and Gymnasium libraries. The figures in the text are the demand time series (Figure 1), the weekday and month seasonality (Figure 2), the demand autocorrelation (Figure 3), the training reward curve (Figure 4), and the policy comparison (Figure 5).

References

[1] Dehaybe, H.; Catanzaro, D.; Chevalier, P. Deep Reinforcement Learning for inventory optimization with non-stationary uncertain demand. European Journal of Operational Research, 2024, 314(2), 433-445.

[2] Yang, Y.; Wang, M.; Wang, J.; Li, P.; Zhou, M. Multi-Agent Deep Reinforcement Learning for Integrated Demand Forecasting and Inventory Optimization in Sensor-Enabled Retail Supply Chains. Sensors, 2025, 25(8), 2428. https://doi.org/10.3390/s25082428

[3] Lu, X.; Wang, H.; Peng, Z.; Liao, C.; Liu, C. Dynamic Optimization of Multi-Echelon Supply Chain Inventory Policies Under Disruptive Scenarios: A Deep Reinforcement Learning Approach. Symmetry, 2025, 17(12), 2078. https://doi.org/10.3390/sym17122078

[4] Oroojlooyjadid, A.; Nazari, M.; Snyder, L. V.; Takac, M. A Deep Q-Network for the Beer Game: Deep Reinforcement Learning for Inventory Optimization. Manufacturing and Service Operations Management, 2022, 24(1), 285-304.

[5] Geevers, K.; van Hezewijk, L.; Mes, M. R. K. Multi-echelon inventory optimization using deep reinforcement learning. Central European Journal of Operations Research, 2024, 32, 653-683.

[6] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347, 2017.

[7] Lim, B.; Arik, S. O.; Loeff, N.; Pfister, T. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting, 2021, 37(4), 1748-1764.

[8] Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. Neural Computation, 1997, 9(8), 1735-1780.

[9] Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. The M5 competition: Background, organization, and implementation. International Journal of Forecasting, 2022, 38(4), 1325-1336.

[10] Snyder, L. V.; Atan, Z.; Peng, P.; Rong, Y.; Schmitt, A. J.; Sinsoysal, B. OR/MS models for supply chain disruptions: A review. IIE Transactions, 2016, 48(2), 89-109.

[11] Douaioui, K.; Oucheikh, R.; Benmoussa, O.; Mabrouki, C. Machine Learning and Deep Learning Models for Demand Forecasting in Supply Chain Management: A Critical Review. Applied System Innovation, 2024, 7(5), 93.

[12] Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; Dormann, N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. Journal of Machine Learning Research, 2021, 22(268), 1-8.