DeepStock: Reinforcement Learning with Policy Regularizations for Inventory Management

Yaqi Xie* **Xinru Hao†** **Jiaxi Liu‡** **Will Ma§** **Linwei Xin¶** **Lei Cao†** **Yidong Zhang†**

1 Introduction

Inventory control is a foundational problem in the field of Operations Research. Historically, the focus has been on stylized inventory and demand models that allow for the derivation of optimal policies. In modern times, these models can be significantly enhanced by leveraging high-dimensional contextual data. Even though optimal policies are no longer analytically defined, Deep Reinforcement Learning (DRL) offers a promising general methodology for learning performant policies that act on this high-dimensional data.

However, the success of applying DRL to inventory control has been mixed [see Gijsbrechts et al., 2025]. Indeed, a common approach is to frame inventory control as a generic sequential decision-making problem and then apply off-the-shelf DRL methods, which leads to actions that are difficult to interpret. More importantly, a dealbreaker is that the performance of DRL is highly sensitive to hyperparameters, whose tuning is time-consuming.

In this paper, we show how these dealbreakers can be mitigated through policy regularizations, where we encode simple intuitions from inventory theory into the policy learned by DRL. This not only reduces the black-box nature of DRL, but drastically speeds up training and improves final performance. In fact, our policy regularization techniques have enabled the full-scale deployment of DRL at Alibaba. As of October 2025, our algorithm manages inventory replenishment for 100% of the products (both domestic and international) sold by Alibaba on its B2C e-commerce platform, Tmall, covering over 1 million SKU-warehouse combinations.

1.1 Description of DRL and Policy Regularizations

A DRL method learns a policy π, represented by a (deep) neural network, that can decide an inventory ordering action from any state. In our problem, the state of an SKU at a time t includes exogenous features $x_t \in \mathbb{R}^m$, which consists of both static attributes (e.g., product category, demand scale, supplier, lead time, review period, profit margin) and dynamic attributes that evolve over time (e.g., upcoming promotions, seasonality, recent social media trends). In addition, the state includes endogenous information I_t about upcoming inventory shipments, which depend on prior actions (i.e., previously placed orders). This rich state space allows for meta-learning across SKU's with diverse characteristics, such as in demand scale (e.g., 10 vs. 10,000 weekly sales), lead time (domestic vs. international), and general demand pattern (e.g., fast-moving vs. long-tailed). Neural networks can process this high-dimensional information into useful representations while ignoring noisy signals.

*Booth School of Business, University of Chicago, Chicago, USA. yaqi.xie@chicagobooth.edu

†Taobao & Tmall Group, Hangzhou, China. xinru.hxr@taobao.com;huaju.cl@taobao.com;yidongzster@gmail.com

‡School of Economics, Sichuan University, Chengdu, China. liujiaxi@stu.scu.edu.cn

§Graduate School of Business, Columbia University, New York, USA. wm2428@gsb.columbia.edu

¶School of Operations Research and Information Engineering, Cornell University, Ithaca, USA. lx267@cornell.edu

[5]YX, XH, and JL are co-first authors with equal contribution. WM and LX provided academic guidance, while LC and YZ offered industry leadership.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.

When a DRL algorithm is applied off-the-shelf, the policy π is a typically single neural network that outputs an order quantity $\pi(I_t, x_t)$ based on the input state I_t, x_t.

Our first policy regularization imposes that the order quantity takes the functional form $\pi(I_t, x_t) = \mu_{\text{BASE}}(I_t, x_t) - \mathsf{tot}(I_t)$, where μ_{BASE} is the learned neural network, and tot denotes the total inventory (including upcoming shipments) contained in I_t. Here, BASE stands for "base stock", where $\mu_{\text{BASE}}(I_t, x_t)$ represents the target level for the total inventory, and the order quantity $\pi(I_t, x_t)$ equals this target level minus the total inventory $\mathsf{tot}(I_t)$ we already have. Intuitively, the learned target $\mu_{\text{BASE}}(I_t, x_t)$ should depend mostly on the exogenous features x_t which forecasts the upcoming demand, while the inventory information I_t is incorporated into the decision through the $\mathsf{tot}(I_t)$ term.

Our second policy regularization imposes the functional form $\pi(I_t, x_t) = \mu_{\text{COEFF}}(I_t, x_t)^\top \mathsf{feat}(x_t)$, where μ_{COEFF} is a learned neural network with an m'-dimensional output, providing Coefficients for m' features extracted from $x_t \in \mathbb{R}^m$ by the mapping $\mathsf{feat}(x_t) \in \mathbb{R}^{m'}$. At Alibaba, we have $m' = 5$, with $\mathsf{feat}(x_t)$ comprised of historical and forecasted demands in the near and distant horizons. Intuitively, the desired order quantity should exhibit a positive relationship with these features.

Finally, we can combine both of our policy regularizations, in which case we impose the order quantity to take the functional form $\pi(I_t, x_t) = \mu_{\text{BOTH}}(I_t, x_t)^\top \mathsf{feat}(x_t) - \mathsf{tot}(I_t)$, with μ_{BOTH} being the learned neural network with an m'-dimensional output.

1.2 Main Results

Our policy regularizations can be tested in conjunction with any DRL method for training the neural network that defines the policy. We test DRL methods, DDPG (Deep Deterministic Policy Gradient; see Lillicrap et al., 2015) which is an off-policy actor-critic algorithm, and SL (Supervised Learning; see Madeka et al., 2022, Alvo et al., 2023) which leverages demand uncensoring to treat inventory management as a supervised learning problem.

We demonstrate three main takeaways:

I. Policy regularizations improve the performance of DRL methods, *especially when hyperparameter tuning is limited*.
II. Policy regularizations *redefine the narrative* on whether SL is actually desirable compared to traditional DRL methods (DDPG) for inventory.
III. Policy regularizations enable a *unified, full-scale* deployment of DRL.

We demonstrate Takeaway I on Alibaba's offline data with and without regularization, after each hyperparameter trial. The improvement with regularization is more drastic under a limited number of hyperparameter trials, even though the best performance after a large number of trials is also improved for both DDPG and SL. To explain this finding, our regularizations encode into the policy human intuitions that prevent obvious blunders: our BASE regularization discourages large orders when we already have a lot of inventory in I_t, while our COEFF regularization ensures that larger historical and forecasted demands imply larger orders. This helps all hyperparameter configurations but especially the weaker ones, where learning is hindered by obvious blunders.

For Takeaway II, policy regularizations improve both traditional DRL and SL, but the improvement is more significant when hyperparameter tuning is limited, which is more likely to be the case for traditional DRL than for SL. We find that SL has a tendency to overfit, because it is not learning intermediate structure in the form of a Cost-to-go function or Q-function from a state. Given enough IID trajectories or long trajectories with consistent patterns, this overfitting subsides and SL performs no worse than traditional DRL (with regularization), and better if hyperparameter tuning has not plateaued. That being said, Alibaba still found DDPG to beat SL when training on 50,000 90-day SKU trajectories, suggesting that data in practice is highly non-IID across SKU's and have inconsistent patterns over time.

We demonstrate Takeaway III by reporting the impact of policy regularizations on Alibaba's deployment of DRL in the real world. Alibaba proceeded to deploy DDPG with our policy regularizations, which generally had the best performance in its routine offline tests, as exemplified by the results on Alibaba's offline data presented in this paper. Although Alibaba has previously tinkered with DRL as reported in Liu et al. [2023], that work was focused on uncertain yield for inventory orders, caused by shortages during the Covid-19 pandemic. In addition, that DRL version was not general-purpose:

it required grouping similar products and training separate models per group. By contrast, we are operating under more normal conditions, and our policy regularizations have allowed for a full-scale deployment of DRL that is *unified*, where we can meta-learn a single policy across all SKU's by leveraging their features, instead of separating them into groups such as fast-moving and long-tail SKU's.

2 Model Setup

We describe the inventory control model used to train policies from offline data, which is realistic and used by Alibaba.

- We consider a horizon of length $T = 90$, where each time period t corresponds to a single physical day. For each SKU, we assume that the review period and lead time L are fixed positive integers that remain constant over time and are exogenous to the inventory policies.
- Let $\xi = (x_t, d_t)_{t=1}^T$ denote a SKU trajectory, where $x_t \in \mathbb{R}^m$ and $d_t \in \mathbb{R}_{\geq 0}$ represent the feature vector and realized demand at time t, respectively. In practice, m is approximately 190, and the feature vector x_t includes updated demand signals and forecasts, which are assumed to subsume all information contained in the past observations $x_1, d_1, \ldots, x_{t-1}, d_{t-1}$. Demands are treated as uncensored using a fixed statistical uncensoring algorithm.
- Let $I_t = (I_t^0, I_t^1, \ldots, I_t^L) \in \mathbb{R}_{\geq 0}^{L+1}$ denote the inventory state at the start of time t, where I_t^0 is the on-hand inventory and $(I_t^1, \ldots, I_t^L)$ represents the inventory pipeline, i.e., the orders placed at times $t - L, \ldots, t - 1$, respectively. Inventory I_t^1 arrives in time to serve the present demand at time t.
- A policy π maps the current state $s_t = (I_t, x_t) \in \mathbb{R}_{\geq 0}^{L+1} \times \mathbb{R}^m$ to a non-negative order quantity. We note that the learned policy is stationary, i.e., independent of t. This is because time-specific information (e.g., promotions) is already encoded in the feature vector x_t, and most products do not require finite-horizon planning in practice. We also note that the policy is invoked only during review periods; the order quantity is set to zero otherwise.
- The inventory dynamics follow a lost-sales model, given by $I_{t+1}^0 = \max\{I_t^0 + I_t^1 - d_t, 0\}$. The inventory pipeline is updated according to $I_{t+1}^1 = I_t^2, \ldots, I_{t+1}^{L-1} = I_t^L$, and the newly placed order is set as I_{t+1}^L.
- We evaluate two long-term metrics over the entire trajectory ξ: the on-shelf rate $\rho_{\mathsf{os}}(\pi, \xi) = \frac{1}{T}\sum_{t=1}^T \mathbb{1}\left\{I_t^0 > 0\right\}$ and the turnover time $\rho_{\mathsf{tt}}(\pi, \xi) = \frac{\sum_{t=1}^T I_t^0}{\sum_{t=1}^T \min\{I_t^0 + I_t^1, d_t\}}$. The on-shelf rate is the fraction of days the SKU was in stock. The turnover time equals the average on-shelf inventory divided by the average sales, reflecting the average number of days a unit stays on the shelf. Higher is better for on-shelf rate, while lower is better for turnover time. These are key inventory metrics for Alibaba that are practically measurable, unlike the theoretical notions of lost-sales and holding costs that can be difficult to quantify. To guide learning, we combine the metrics $\rho_{\mathsf{os}}(\pi, \xi)$ and $\rho_{\mathsf{tt}}(\pi, \xi)$ into an ad hoc loss function $\ell(\pi, \xi)$, which is calibrated to on-shelf and turnover "targets" specified by managers.

3 Training and Testing on Offline Data

DRL methods. We train inventory policies offline using two DRL methods: DDPG and SL. DDPG learns from off-policy experiences (transitions) to update the "critic" (the Q function, a deep neural network estimating the final reward when the current policy takes an action from a given state), and then uses this critic to update the "actor" (i.e., the policy). Between updates, we simulate the current policy on the historical trajectories ξ and add the resulting transitions to the buffer. In contrast, SL is specifically designed to exploit the fact that the counterfactual performance of any inventory policy on any historical SKU trajectory ξ can be evaluated due to uncensored demand. This reduces the problem to supervised learning, where we can directly train a policy π to minimize the average loss $\ell(\pi, \xi)$ over historical SKU trajectories ξ. We note that in SL, each trajectory is always treated as a whole, with no value or Q functions defined intermediate states.

Our main focus is to compare the policy regularizations introduced in Section 1.1. Each combination of DRL method (DDPG, SL) and regularization type (NONE, BASE, COEFF, BOTH) defines a

	DDPG				SL			
	NONE	BASE	COEFF	BOTH	NONE	BASE	COEFF	BOTH
$\Delta^{\mathsf{test}}\rho_{\mathsf{os}}(\pi^*)$ (%)	-10.10	-6.03	-4.41	0	-2.10	-2.18	-1.74	-1.91
$\Delta^{\mathsf{test}}\rho_{\mathsf{tt}}(\pi^*)$ (days)	6.13	6.46	-0.41	0	-1.25	-2.81	3.80	0.23

Table 1: On-shelf rates and turnover times for the 8 final policies on test data.

learning algorithm, which outputs an inventory policy given training and validation data as input. We use $\pi^{\text{DRL,REG}}$ to denote the final policy learned by the method DRL $\in \{\text{DDPG}, \text{SL}\}$ with the regularization REG $\in \{\text{NONE}, \text{BASE}, \text{COEFF}, \text{BOTH}\}$.

Training and validation. We let $\mathcal{D}^{\mathsf{train}}$ and $\mathcal{D}^{\mathsf{validate}}$ denote the collections of historical SKU trajectories used for training and validation, respectively. In the results shown here, $\mathcal{D}^{\mathsf{train}}$ contains 50,000 randomly selected SKU trajectories from July to September 2024, while $\mathcal{D}^{\mathsf{validate}}$ contains 5,000 randomly selected trajectories of other SKUs from the same 90-day horizon. Given a policy π, we define $L^{\mathsf{train}}(\pi)$, $L^{\mathsf{validate}}(\pi)$ as its average loss over the trajectories in $\mathcal{D}^{\mathsf{train}}$, $\mathcal{D}^{\mathsf{validate}}$, respectively.

For each of the 8 learning algorithms, we train over multiple iterations, using a batch of trajectories from $\mathcal{D}^{\mathsf{train}}$ in each iteration. $L^{\mathsf{train}}(\pi)$ is approximated using the trajectories in the batch, and each iteration yields an update of the policy π. $L^{\mathsf{validate}}(\pi)$ is then evaluated for the updated policy π over $\mathcal{D}^{\mathsf{validate}}$ every 10,000 episodes for DDPG algorithms and 5,000 episodes for SL algorithms, where an episode corresponds to processing one training trajectory. Training stops if the validation loss does not improve after one epoch, defined as one full pass through the 50,000 training data. Upon stopping, we output the policy π from one epoch ago with lowest $L^{\mathsf{validate}}(\pi)$, and output π as the final learned policy. Further details are omitted here due to space limitations.

We take the final policies output by the 8 algorithms and evaluate them on test trajectories. In the results displayed here, $\mathcal{D}^{\mathsf{test}}$ contains 5,000 randomly selected SKU trajectories from February to April 2024, which is chronologically later than the training horizon. We forgo the ad hoc loss function used for training and validation and instead directly report the key metrics of interest: on-shelf rate and turnover time. For each of the 8 final policies π, we report $\Delta^{\mathsf{test}}\rho_{\mathsf{os}}(\pi) := \frac{1}{|\mathcal{D}^{\mathsf{test}}|}\sum_{\xi\in\mathcal{D}^{\mathsf{test}}}\big(\rho_{\mathsf{os}}(\pi,\xi) - \rho_{\mathsf{os}}(\pi^{\text{DDPG,BOTH}},\xi)\big)$, and $\Delta^{\mathsf{test}}\rho_{\mathsf{tt}}(\pi) := \frac{1}{|\mathcal{D}^{\mathsf{test}}|}\sum_{\xi\in\mathcal{D}^{\mathsf{test}}}\big(\rho_{\mathsf{tt}}(\pi,\xi) - \rho_{\mathsf{tt}}(\pi^{\text{DDPG,BOTH}},\xi)\big)$. The first metric is measured as the % of days the SKU did not stock out, with 1% roughly corresponding to one day over the 90-day horizon. The second metric is measured in days and represents the average time a unit of inventory remains on the shelf. For all 8 policies, we report these metrics in comparison to $\pi^{\text{DDPG,BOTH}}$ in Table 1 to mask Alibaba's absolute numbers, noting that $\Delta^{\mathsf{test}}\rho_{\mathsf{os}}(\pi^{\text{DDPG,BOTH}}) = \Delta^{\mathsf{test}}\rho_{\mathsf{tt}}(\pi^{\text{DDPG,BOTH}}) = 0$.

As shown in Table 1, having BOTH regularizations produces by far the best version of DDPG, while having the BASE regularization produces the best version of SL. This supports our Takeaway I from the Introduction. To see the result of Takeaway II, note that without regularizations, $\pi^{\text{DDPG,NONE}}$ is much worse than $\pi^{\text{SL,NONE}}$; with regularizations, $\pi^{\text{DDPG,BOTH}}$ is comfortably better than $\pi^{\text{SL,BASE}}$ (and $\pi^{\text{SL,BOTH}}$), recalling that Stockout Rate is twice as important as Turnover Time.

Finally, related to Takeaway I, one may wonder whether the results in Table 1 are from "limited hyperparameter search". To give some perspective, one hyperparameter trial for DDPG in this setting (with data size $\approx 50,000$ trajectories $\times$ 90 days $\times$ 190-dimensional features) takes about 24 wall-clock hours internally at Alibaba. Table 1 shows test results for every $\pi^{\text{DRL,REG}}$ after 10 sequential trials, with each trial choosing new hyperparameters based on the outcome of the previous trial, over 10 physical days. Although Alibaba could have potentially improved the hyperparameter search with some parallelization, which may decrease the improvement of $\pi^{\text{DDPG,BOTH}}$ over $\pi^{\text{DDPG,NONE}}$, this would only increase the advantage of DDPG over SL. Regardless, training is expensive and slow at this scale, and the results in Table 1 depict a realistic outcome for hyperparameter tuning in practice at Alibaba.

References

M. Alvo, D. Russo, and Y. Kanoria. Neural inventory control in networks via hindsight differentiable policy optimization. *arXiv preprint arXiv:2306.11246*, 2023.

J. Gijsbrechts, R. N. Boute, J. A. Van Mieghem, and D. Zhang. Ai in inventory management: The disruptive era of drl and beyond. *Available at SSRN 5199616*, 2025.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

J. Liu, S. Lin, L. Xin, and Y. Zhang. Ai vs. human buyers: A study of alibaba's inventory replenishment system. *INFORMS Journal on Applied Analytics*, 53(5):372–387, 2023.

D. Madeka, K. Torkkola, C. Eisenach, A. Luo, D. P. Foster, and S. M. Kakade. Deep inventory management. *arXiv preprint arXiv:2210.03137*, 2022.