Modern Supply Chain Research and Applications

Optimizing multi-tier supply chain ordering with LNN+XGBoost: mitigating the bullwhip effect

Chunan Tong
University of Maryland, College Park, Maryland, USA

Received 27 August 2025
Revised 12 July 2026
Accepted 20 August 2026

Abstract

Purpose – Supply chain ordering must balance profit, customer service and upstream order variability (the bullwhip effect). This study evaluates whether a two-stage pipeline–a Liquid Neural Network (LNN) encoder with an adaptive-leak recurrent cell followed by an XGBoost readout–offers a useful operational trade-off for multi-tier ordering, rather than assuming hybrid superiority.

Design/methodology/approach – An LNN encoder represents temporal demand dynamics and XGBoost regresses its frozen latent states; the encoder is explicitly distinguished from canonical Liquid Time-Constant and closed-form continuous-time networks. Under a leakage-free protocol, hyperparameters are selected on a separate development seed and frozen before testing on untouched days from ten unseen demand seeds. A matched linear-readout ablation, paired statistics with multiplicity correction, an observed-order DataCo replay, cost sensitivity analysis and an end-to-end latency benchmark are used.

Findings – Transformer ($1.738M), XGBoost ($1.737M) and LNN+XGBoost ($1.737M) attain nearly identical mean system profit, with no Holm-adjusted difference between the hybrid and either baseline. Relative to the exact-matched linear readout, LNN+XGBoost gains $14,477 (95% CI $4,811–$24,143; d_z = 1.07), although the family-wise Holm result ($p = 0.072$) does not indicate a statistically significant difference. Its manufacturer demand-referenced variance ratio is 2.61 versus 2.44 (XGBoost) and 2.48 (Transformer), with a fill rate of 0.996. The Q-network minimizes variance (1.43) but sacrifices profit and service, the DataCo replay favors standalone XGBoost and the hybrid is not latency-optimal.

Originality/value – The study provides the first leakage-free, ablation-controlled evaluation of an adaptive-leak-LNN–XGBoost ordering pipeline and characterizes a profit–service–bullwhip trade-off with explicit operating conditions rather than claiming universal superiority.

Keywords Bullwhip Effect, Inventory Management, Liquid Neural Network (LNN), XGBoost, Independent Test, Multi-Echelon Ordering

Paper type Research article

1. Introduction

Supply chain ordering must manage fluctuating demand, inventory imbalance and upstream variance amplification. Long short-term memory (LSTM) and reinforcement learning can model dynamics but impose training and tuning costs; XGBoost is efficient on tabular features but does not natively encode sequences. This motivates testing whether a compact temporal encoder followed by tree regression offers a useful operational trade-off.

Liquid time-constant (LTC) and closed-form continuous-time networks use input-dependent state dynamics inspired by biological nervous systems (Hasani *et al.*, 2020, 2022). Their hidden states respond adaptively to incoming sequences after training; this state adaptation does not imply online updating of model weights. The present encoder borrows the adaptive-state and time-constant ideas but is presented as an adaptive-leak LNN variant rather than a reproduction of the canonical LTC or CfC equations. We therefore call the overall

Funding: This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Modern Supply Chain Research and Applications
Emerald Publishing Limited
2631-3871
DOI 10.1108/MSCRA-08-2025-0054

encoder a *liquid neural network* (LNN) and disclose its adaptive-leak cell as the implementation used in this study.

This study evaluates a two-stage LNN + XGBoost model that sequentially composes liquid-inspired dynamic feature extraction with tree regression in a cascaded pipeline, not a jointly optimized architecture. The research question is: how does a frozen adaptive-state encoder followed by an XGBoost readout affect profit, service and upstream order amplification under an independently tested multi-tier ordering simulation? The latency benchmark in Section 4.4 tests, rather than assumes, whether a compact encoder translates into faster inference.

The study makes three bounded contributions. First, it tests a previously unreported adaptive-leak-LNN–XGBoost ordering pipeline and uses a matched linear-readout ablation to identify the incremental value of the tree stage rather than attributing performance to unspecified "hybrid synergy." Second, it evaluates profit, service and both chained and demand-referenced bullwhip measures on an untouched test period across ten unseen demand seeds, with effect sizes and multiplicity correction. Third, it combines a multi-tier simulation with an observed-order replay, economic sensitivity analysis and end-to-end latency measurement, thereby identifying both the pipeline's useful operating region and the conditions under which simpler models are preferable.

2. Related work and research gap

This section reviews machine learning and deep learning techniques for supply chain forecasting and ordering optimization – continuous-time and liquid-inspired recurrent models, LSTM, transformers, XGBoost and hybrid models – and identifies the gap addressed by the proposed pipeline.

2.1 Continuous-time and liquid-inspired recurrent models

LTC networks model neuron dynamics using liquid time constants so that hidden states respond continuously to sequential inputs (Hasani *et al.*, 2020). Hasani *et al.* (2022) subsequently derived closed-form continuous-time networks and reported speedups for those specific architectures and tasks. These findings motivate compact adaptive-state encoders, but do not justify labeling every leaky recurrent cell as a canonical LTC/CfC implementation or assuming lower end-to-end latency. We accordingly distinguish the simplified LNN used here from LTC/CfC and measure the complete GPU–CPU pipeline directly.

To our knowledge, LNN encoders with adaptive-leak cells followed by tree readouts have not been evaluated for multi-tier ordering with explicit profit, service and bullwhip outcomes. This bounded empirical gap motivates the present study; it is not a claim that the simplified encoder is a canonical LTC/CfC implementation.

2.2 LSTM and hybrid models in supply chain forecasting

LSTM captures long-term dependence (Graves, 2013; Benidis *et al.*, 2022), while recent supply-chain hybrids combine clustering, recurrent networks, convolution or attention (Ji *et al.*, 2024; Cao *et al.*, 2024; Jahin *et al.*, 2024). These studies primarily improve forecasts; they provide less evidence on closed-loop multi-tier ordering, bullwhip measures and end-to-end deployment cost.

2.3 Transformer models in supply chain forecasting

Transformer models use self-attention to model long-sequence dependence, although simpler time-series baselines can remain competitive (Zeng *et al.*, 2022). Wibawa *et al.* (2022) illustrated a complementary convolutional approach, and Aguiar-Pérez and Pérez-Juárez (2023) reviewed deep demand forecasting in another operational domain. This study directly

measures whether a compact LNN encoder plus XGBoost offers a useful outcome–efficiency trade-off.

2.4 XGBoost in supply chain forecasting

XGBoost is accurate and efficient on many tabular prediction tasks (Chen and Guestrin, 2016). Its feature-based regression is a natural second stage after temporal encoding, while standalone XGBoost supplies an essential simpler baseline. Whether the encoder adds value is therefore tested by both standalone XGBoost and a matched linear readout rather than assumed from model descriptions.

2.5 Deep reinforcement learning in supply chain optimization

Deep reinforcement learning has been applied to multi-echelon ordering, including discrete-action deep Q-network (DQN) policies for the beer game (Oroojlooyjadid *et al.*, 2017). The present reward-trained Q-network comparator does not reproduce that canonical discrete-action design and is labeled accordingly.

2.6 Related studies on ordering optimization strategies

Ordering optimization typically integrates demand forecasting, safety stock and cost management. Chaharsooghi *et al.* (2008) used reinforcement learning for beer-game ordering; Silver *et al.* (1998) established the classical safety-stock and demand-forecast foundation for multi-tier ordering; and Bertsimas and Thiele (2006) developed robust optimization under demand uncertainty. These approaches adjust orders from forecasted demand to optimize profit but do not incorporate an LNN or XGBoost.

2.7 Hybrid models and research gaps

Convolutional and hybrid forecasting studies demonstrate the value of learned temporal representations (Zhao and Wang, 2017; Borovykh *et al.*, 2017), and modern optimization frameworks enable systematic tuning (Akiba *et al.*, 2019). To our knowledge, no prior study has evaluated the specific LNN–XGBoost cascade used here for multi-tier ordering. The unresolved question is not merely predictive accuracy, but whether this cascade improves profit and service without worsening bullwhip or deployment cost on data not used for model selection.

3. Methodology

This section follows the audited data flow in Figure 1: demand generation, propagation, feature engineering, inventory-target modeling, order selection, development-only hyperparameter tuning and untouched-test evaluation.

3.1 Consumer demand generation

Consumer demand combines a baseline, seasonal and weekly cycles, and Gaussian noise:

$$D(t) = \text{constant} + \text{seasonal fluctuation} + \text{weekly fluctuation} + \text{noise}$$

Demand is constrained to be nonnegative and generated under fixed seeds; Section 4.2 gives the numerical specification.

3.2 Demand propagation across layers

The four layers are consumers, retailers, distributors and manufacturers. Each operational echelon receives the immediately downstream order:

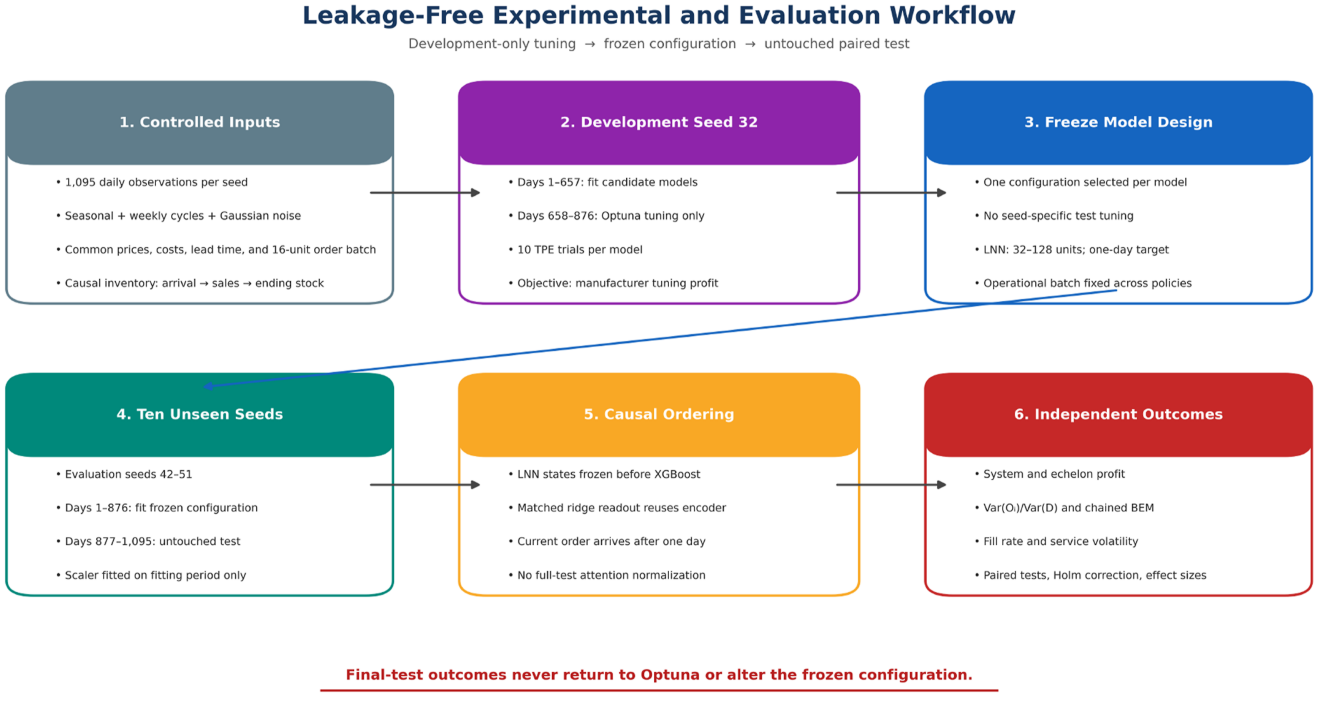

Figure 1. Overall pipeline: input generation, two-stage LNN → XGBoost modeling, development-only Optuna tuning, frozen-model order simulation and untouched-test evaluation

$$D_i(t) = O_{i-1}(t) \quad \text{for} \quad i = 1, 2, 3$$

This cascade permits direct measurement of incremental and system-wide order amplification.

3.3 Feature engineering

Features comprise current and lagged orders, inventory, sales, recent volatility, seasonality and normalized time. They are Min–Max normalized and processed in ten-day windows.

3.4 Forecasting and order decision process

3.4.1 Inventory-target models. The learned models estimate a one-day-ahead inventory target under the causal reference dynamics. This horizon matches the fixed one-day replenishment lead time. The compared models are:

(1) *Liquid neural network + XGBoost (LNN + XGBoost):* for input x_t, activation a_t, learned base leak λ_0 and time constant τ, the implemented update is

$$\begin{aligned} a_t &= \mathrm{ReLU}\,(W_{in}x_t + W_{rec}s_{t-1}), \\ \lambda_t &= \mathrm{clip}[\lambda_0\{1 + \mathrm{sd}\,(x_t)\}, 0.1, 0.9], \\ u_t &= (1 - \lambda_t)s_{t-1} + \lambda_t a_t, \\ s_t &= \mathrm{LayerNorm}\left[u_t + \frac{dt}{\tau}(-u_t + a_t)\right]. \end{aligned}$$

Here $dt = 0.1$, W_{rec} is initialized with 50% sparsity and τ is learned. This is a discrete, liquid-inspired adaptive-leak cell, an LNN variant rather than the canonical LTC or CfC equation. The encoder is trained first; its states are frozen and flattened, and XGBoost then predicts the inventory target. There is no feedback from XGBoost to the encoder.

(1) *XGBoost, LSTM and transformer:* tabular, recurrent and attention-based baselines.

(2) *Q-network comparator:* a reward-based neural baseline with replay and exploratory action proposals; it is not labeled as canonical DQN.

3.4.2 Safety stock calculation. Safety stock is dynamically computed to mitigate demand variability:

$$SS_i(t) = SS_{\text{base}} + SS_{\text{factor},i} \cdot \sigma_{D_i}(t)$$

Where SS_{base} is the buffer, $SS_{\text{factor},i}$ is layer-specific and $\sigma_{D_i}(t)$ is recent demand volatility.

3.4.3 Causal inventory flow and order selection. At the start of day t, the order placed at $t - 1$ arrives. Available inventory is $A_i(t) = I_i(t-1) + O_i(t-1)$, realized sales are $S_i(t) = \min\{A_i(t), Q_i(t)\}$ and ending inventory is $I_i(t) = A_i(t) - S_i(t)$, where $Q_1(t) = D(t)$ and $Q_i(t) = O_{i-1}(t)$ for $i > 1$. Thus, a newly placed order does not arrive on the same day.

All policies use the same operational order batch of 16 units; this is distinct from the neural training mini-batch. Nonclassical policies evaluate batch-feasible candidates between a safety-stock-adjusted lower bound and an upper bound derived from 1.5 times the recent mean incoming order. Each candidate is valued one lead-time step ahead using the predicted inventory target and the same revenue, purchase, holding and shortage accounting:

$$P = \text{Revenue} - \text{PurchaseCost} - \text{HoldingCost} - \text{ShortageCost}$$

The selected nonnegative order is rounded upward to the common operational batch:

$$O_i(t) = 16\lceil \frac{O_i(t)}{16} \rceil.$$

The classical comparator uses an echelon order-up-to rule under the same lead time and order batch.

3.5 Hyperparameter optimization

Hyperparameters are selected once on a separate development demand seed (32). Within that seed, days 1–657 fit candidate models and days 658–876 tune them through ten Optuna TPE trials per model. The objective is manufacturer cumulative profit on that development-only tuning period:

$$\text{Objective} = \sum_{t=1}^{T} \text{Profit}_3(t)$$

The resulting configuration is frozen before final evaluation. For each unseen evaluation seed (42–51), the model is fitted on days 1–876 and evaluated only on untouched days 877–1,095. No final-test outcome enters hyperparameter selection. Tuned parameters include architecture size, learning rate, training mini-batch, epochs and safety-stock baseline; the operational order batch remains fixed at 16 for every policy.

3.6 Feature correlation analysis

For descriptive transparency, pairwise Pearson correlations are computed among the six displayed input features selected for each model family and echelon and visualized in Appendix 3. Coefficients range from -1 to $+1$, and the unit diagonal is each variable's self-correlation. This is an association analysis rather than SHapley Additive exPlanations (SHAP) or feature attribution: it does not measure a feature's contribution to a prediction, establish predictive importance or support causal conclusions.

3.7 Performance evaluation

The system's effectiveness is evaluated using an efficiency metric comparing actual to theoretical maximum profit:

$$\text{Efficiency}_i(t) = \frac{\text{Profit}_i(t)}{\text{TheoreticalProfit}_i(t)}$$

Where:

$$\text{TheoreticalProfit}_i(t) = D_i(t) \cdot (P_i - C_i)$$

Assumes perfect demand fulfillment without holding or shortage costs. A 7-day moving average smooths daily fluctuations:

$$\text{Efficiency}_i^{MA}(t) = \frac{1}{7} \sum_{k=t-6}^{t} \text{Efficiency}_i(k), \quad t \geq 7$$

Additional metrics include:

(1) *Inventory turnover:* Sales divided by average inventory.

(2) *Service level:* Proportion of demand fulfilled.

(3) *Shortage cost:* Costs from unmet demand.

(4) *Holding cost:* Expenses from inventory storage.

(5) *Order volatility:* Standard deviation of orders.

(6) *Inventory-target MAE:* Mean absolute error of the one-day inventory-target prediction.

Because mitigating the bullwhip effect is a central objective, we use two complementary measures. For seed r, the system-wide *variance ratio* compares manufacturer orders with end-consumer demand: $\mathrm{VR}_r = \mathrm{Var}(O_{3,r})/\mathrm{Var}(D_r)$; values above 1 indicate amplification, and lower is better. We report the mean and standard deviation of VR_r over ten seeds. The *per-layer, chained bullwhip effect measure (BEM)* isolates the amplification introduced at each echelon:

$$\mathrm{BEM}_i = \begin{cases} \mathrm{Var}(O_1)/\mathrm{Var}(D), & i = 1, \\ \mathrm{Var}(O_i)/\mathrm{Var}(O_{i-1}), & i = 2, 3, \end{cases}$$

So that BEM_i measures how much layer i amplifies the order variance it receives from layer $i-1$ (with layer 1 measured against consumer demand). The chained ratios are distinct from the system-wide layer-3-to-consumer ratio. For each seed, peak amplification is $\max(O_3)/\max(D)$. At layer i, fill rate is $\sum_t S_i(t)/\sum_t Q_i(t)$, where S_i is realized sales and $Q_1 = D$, $Q_i = O_{i-1}$ for $i > 1$; the reported fill rate is the mean across layers. Service volatility is analogously the cross-layer mean of the standard deviation of $S_i(t)/Q_i(t)$. All summary values are computed directly from saved validation tensors.

Because every model uses the same ten unseen demand seeds and untouched test days, inference preserves pairing. Friedman tests assess omnibus differences; Kendall's $W = \chi^2/[N(k-1)]$ quantifies rank concordance. All pairwise profit comparisons use paired t-tests with Holm family-wise-error correction. For the targeted matched hybrid-versus-linear comparison, we additionally report $d_z = \bar{d}/s_d$, a 95% confidence interval, and an exact Wilcoxon signed-rank test as a distribution-robust check.

4. Experimental research and analysis

4.1 Experiment purpose

The purpose is to evaluate six learning configurations: LNN + XGBoost, its matched linear-readout ablation, standalone XGBoost, LSTM, transformer and a Q-network decision-learning baseline. A classical echelon order-up-to policy provides a nonlearning benchmark. Performance is evaluated across retailer, distributor and manufacturer layers on data excluded from tuning.

4.2 Experiment method

The supply chain is structured with four layers:

(1) Layer 0: Consumers

(2) Layer 1: Retailers

(3) Layer 2: Distributors

(4) Layer 3: Manufacturers

Each demand realization contains 1,095 days. Model selection and final evaluation are separated at both seed and time levels:

(1) Development seed 32: days 1–657 for fitting candidates and days 658–876 for 10-trial Optuna tuning.

(2) Unseen evaluation seeds 42–51: days 1–876 for fitting the frozen configuration and days 877–1,095 for final testing.

The ten evaluation seeds and their final 219-day periods are never used by Optuna. Hyperparameters are common across all evaluation seeds, eliminating seed-specific best-of-ten selection and preserving an untouched paired test.

Initial conditions include:

(1) Initial inventory for each layer: 100 units

(2) Order fulfillment lead time: Fixed at 1 day

The cost and price structure is as follows:

(1) Unit costs: [0, 30, 45, 60] for layers 0–3, respectively

(2) Unit prices: [0, 70, 100, 130] for layers 0–3, respectively

(3) Holding cost rate: 0.03 per unit per day

(4) Shortage cost rate: 0.03 per unit per day

Consumer demand at layer 0 combines deterministic cycles and random noise as a controlled synthetic process:

$$D(t) = 50 + 20\sin\left(\frac{2\pi t}{90}\right) + 5\sin\left(\frac{2\pi t}{7}\right) + \mathcal{N}(0,3)$$

Where t is the time step, and $\mathcal{N}(0,3)$ is Gaussian noise. Demand is constrained to be nonnegative, and fixed random seeds (42–51 for 10 runs) ensure reproducibility.

The experiment compares six learning configurations:

Feature engineering involves constructing a 10-dimensional feature vector per layer, including:

(1) Current demand, lagged orders ($t-1$, $t-2$), lagged inventory ($t-1$, $t-2$), lagged sales ($t-1$)

(2) Volatility indicators: Standard deviation of orders and demand over the past 5 days

(3) Seasonal indicator: $\sin\left(\frac{2\pi t}{90}\right)$

(4) Normalized time: $\frac{t}{1095}$

The scaler is fitted only on the fitting period and applied unchanged to tuning or test observations. Features use a 10-day sliding window. Test-time attention normalization over the full sequence is not used, preventing future observations from influencing current representations.

The training process is as follows:

(1) Models estimate the next-day inventory target, matching the one-day lead time.

(2) For LNN + XGBoost, the encoder is trained first using MSE loss, frozen and followed by XGBoost on flattened latent states. The matched linear ablation reuses the exact frozen encoder and ordering configuration.

(3) The Q-network comparator is trained with a profit-plus-service reward and replay.

Optuna performs ten trials per model only on development seed 32. The objective is tuning-period manufacturer profit. The single selected configuration is frozen across evaluation seeds. Training mini-batches of 4 or 8 affect optimization only; every policy faces the same 16-unit operational order batch.

During untouched testing (days 877–1,095), orders are selected causally each day:

(1) *Inventory-target prediction:* Models estimate the next-day reference inventory target; the order signal is exponentially smoothed with $\alpha = 0.3$.

(2) *Safety stock calculation:*

$$SS = \text{safety\_stock\_base} + 1.0 \times \sigma_D(t-10:t)$$

Where σ_D is the standard deviation of demand over the past 10 days.

(1) *Order range exploration:* Candidate orders use the common 16-unit batch and are bounded by the safety-stock-adjusted signal and 1.5 times the recent mean incoming order. Candidates are evaluated one lead-time step ahead; a new order never enters same-day inventory.

(2) *Profit calculation:*

$$\text{Profit} = \text{Revenue} - \text{Purchase Cost} - \text{Holding Cost} - \text{Shortage Cost}$$

The order that maximizes profit is selected and adjusted to the nearest multiple of 16 units to satisfy batch constraints (see Table 1).

4.3 Experiment results

Results below use only untouched days 877–1,095 of evaluation seeds 42–51. Hyperparameters were frozen before these demand realizations were evaluated. Table 5

Table 1. Comparison of six learning configurations

Model	Components	Configuration details
LNN + XGBoost	LNN encoder (adaptive-leak cell)	32–128 units (step 32), Xavier initialization, adaptive leak, learned τ, AdamW (learning rate 10^{-5} to 10^{-3}), 10–20 epochs
	XGBoost	Regressor on flattened frozen states, 50–150 trees, maximum depth 3–5, learning rate 0.01–0.3
LNN linear-readout ablation	Frozen LNN + ridge readout	Reuses the exact evaluation-seed encoder weights, state window, frozen hyperparameters and ordering parameters; only XGBoost is replaced by ridge regression ($\alpha = 1$)
Standalone XGBoost	XGBoost	Trained on engineered features, 50–150 trees, maximum depth 3–5, learning rate 0.01–0.3
LSTM	Long Short-Term Memory	64–256 hidden units (step 64), 1–3 layers, one-day inventory target, Adam optimizer, gradient clipping (maximum norm 0.5)
Transformer	Transformer	Model dimensions 64–256, 2–8 attention heads, 1–3 layers, multi-head attention (dropout rate 0.1), gradient clipping (maximum norm 0.1)
Q-network baseline	Reward-trained neural network	64–256 hidden units, replay buffer 20,000, exploratory action proposals, and profit-plus-service reward; treated as a Q-network-style comparator rather than a canonical discrete-action DQN

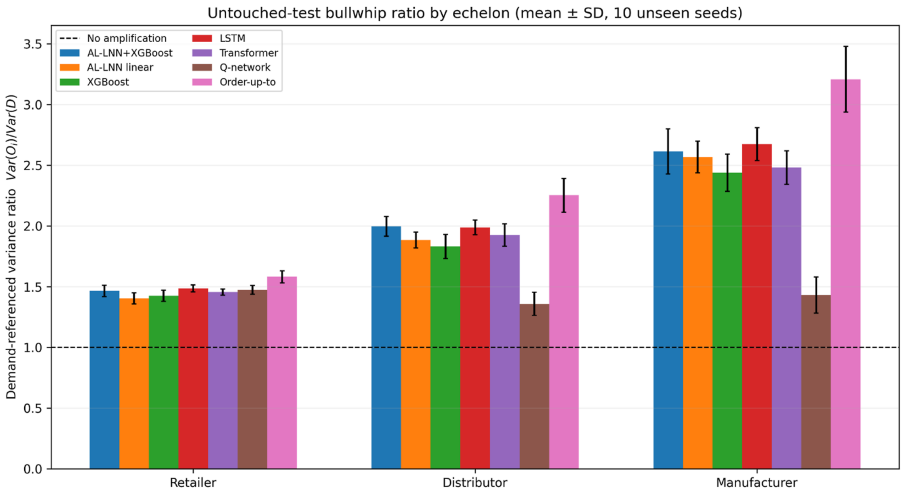

Figure 2. Untouched-test demand-referenced variance ratio $\mathrm{Var}(O_i)/\mathrm{Var}(D)$ by echelon (mean ± SD across 10 unseen seeds)

Table 2. Demand-referenced bullwhip ratio $\mathrm{Var}(O_i)/\mathrm{Var}(D)$ (mean ± SD over ten untouched test seeds). Lower is better; values above one indicate amplification relative to consumer demand

Model	Layer 1	Layer 2	Layer 3
LNN + XGBoost	1.47 ± 0.05	2.00 ± 0.08	2.61 ± 0.19
LNN linear	1.40 ± 0.05	1.88 ± 0.07	2.57 ± 0.13
XGBoost	1.43 ± 0.05	1.83 ± 0.10	2.44 ± 0.15
LSTM	1.48 ± 0.03	1.99 ± 0.06	2.67 ± 0.14
Transformer	1.46 ± 0.03	1.92 ± 0.09	2.48 ± 0.14
Q-network	1.47 ± 0.04	1.36 ± 0.09	1.43 ± 0.15
Order-up-to	1.58 ± 0.05	2.25 ± 0.14	3.21 ± 0.27

Table 3. Chained BEM $\mathrm{Var}(O_i)/\mathrm{Var}(O_{i-1})$ (mean ± SD). Layer 1 uses consumer demand as its denominator

Model	Layer 1	Layer 2	Layer 3
LNN + XGBoost	1.47 ± 0.05	1.36 ± 0.04	1.31 ± 0.06
LNN linear	1.40 ± 0.05	1.34 ± 0.03	1.36 ± 0.05
XGBoost	1.43 ± 0.05	1.28 ± 0.04	1.33 ± 0.02
LSTM	1.48 ± 0.03	1.34 ± 0.03	1.34 ± 0.04
Transformer	1.46 ± 0.03	1.32 ± 0.05	1.29 ± 0.03
Q-network	1.47 ± 0.04	0.92 ± 0.05	1.05 ± 0.07
Order-up-to	1.58 ± 0.05	1.42 ± 0.05	1.42 ± 0.09

reports system-wide profit – defined here as the sum of the three-echelon operating margins under fixed layer prices and costs, computed identically for every model (see the definition following Table 5) – while Figure 2 and Tables 2 and 3 report demand-referenced and chained order amplification (see Tables 4 and 5).

The Q-network has the lowest manufacturer-to-consumer variance ratio (1.43) but also the lowest profit and fill rate, illustrating that variance damping alone is not sufficient. Among the

Table 4. System-wide bullwhip and service metrics (mean ± standard deviation over 10 seeds). Variance ratio and peak amplification compare layer-3 manufacturer orders with consumer demand. Fill rate and service volatility are first computed per layer and then averaged across layers

Model	Variance ratio	Peak amplification	Fill rate	Service volatility
LNN + XGBoost	2.612 ± 0.186	1.263 ± 0.066	0.996 ± 0.001	0.037 ± 0.003
LNN linear	2.568 ± 0.131	1.426 ± 0.126	0.991 ± 0.005	0.055 ± 0.011
XGBoost	2.438 ± 0.153	1.223 ± 0.029	0.995 ± 0.001	0.044 ± 0.004
LSTM	2.673 ± 0.136	1.345 ± 0.105	0.997 ± 0.001	0.037 ± 0.005
Transformer	2.480 ± 0.137	1.223 ± 0.029	0.996 ± 0.002	0.037 ± 0.006
Q-network	1.430 ± 0.149	0.816 ± 0.019	0.953 ± 0.004	0.092 ± 0.002
Order-up-to	3.207 ± 0.270	1.552 ± 0.390	0.990 ± 0.002	0.056 ± 0.007

Table 5. Untouched-test system-wide cumulative profit (mean ± 95% CI, $N = 10$ unseen seeds) and one-day inventory-target MAE (mean ± SD). System-wide profit is a stylized aggregate echelon margin (see text), reported identically for every model

Model	Mean cumulative profit (USD)	Inventory-target MAE
Transformer	1, 737, 797 ± 4, 471	11.57 ± 0.85
XGBoost	1, 737, 362 ± 4, 352	16.95 ± 0.44
LNN + XGBoost	1, 737, 106 ± 3, 121	14.79 ± 0.40
LSTM	1, 735, 030 ± 3, 701	19.14 ± 0.96
LNN linear	1, 722, 629 ± 10, 832	21.53 ± 2.57
Order-up-to	1, 719, 828 ± 7, 906	24.45 ± 0.87
Q-network	1, 617, 572 ± 7, 236	7.11 ± 0.39

high-profit models, XGBoost has the lowest manufacturer ratio (2.44), followed by transformer (2.48), LNN + XGBoost (2.61) and LSTM (2.67). The classical order-up-to policy has the highest ratio (3.21). LNN + XGBoost improves fill rate and service stability relative to its matched linear readout, but its variance ratio is slightly higher (2.61 versus 2.57).

Transformer has the highest mean test profit ($1.738 M), followed closely by XGBoost and LNN + XGBoost. Their confidence intervals overlap, and Holm-adjusted paired comparisons show no difference between LNN + XGBoost and XGBoost or transformer (both $p = 1.000$). The matched readout contrast favors XGBoost by $14,477 (95% CI $4,811-$24,143; $d_z = 1.07$; paired $t_9 = 3.39$, raw $p = 0.008$; exact Wilcoxon $p = 0.0039$), but its Holm-adjusted $p = 0.072$ does not cross the family-wise 0.05 threshold. Thus the targeted effect is sizable and directionally consistent, but not multiplicity-robust in this ten-seed test.

Here *system-wide profit* is the sum of the retailer, distributor and manufacturer echelon operating margins (revenue less purchase, holding and shortage costs) under fixed layer prices and costs. Because each echelon's purchase cost is charged at its own layer cost rather than netted against the downstream layer's sale price, this is a stylized aggregate echelon profit for cross-model comparison, not a consolidated entity profit net of internal transfers; all models are scored identically, so the comparison is consistent. The final column reports the one-day inventory-target mean absolute error (MAE), the prediction accuracy underlying each policy.

Notably, the inventory-target MAE does not track profit rank: the Q-network has the lowest MAE (7.11) yet the lowest profit, while transformer, XGBoost and LNN + XGBoost span 11.6–17.0 MAE at nearly identical profit. This confirms that prediction accuracy alone does not determine ordering-policy value, motivating the profit, service and bullwhip outcomes as the primary criteria.

Table 6. Friedman repeated-measures tests on paired untouched-test outcomes across six learning configurations

Outcome	$\chi^2(5)$	p	Kendall's W
System profit	33.60	2.86×10^{-6}	0.672
Variance ratio	33.60	2.86×10^{-6}	0.672
Fill rate	37.21	5.45×10^{-7}	0.744
Service volatility	37.54	4.66×10^{-7}	0.751

Raw outcomes, not a model-set-dependent composite, are used for inference. Friedman tests across the six learning configurations reject equal ranks for all four primary outcomes (Table 6). The classical policy remains a descriptive benchmark outside these six-model omnibus tests.

These omnibus effects primarily separate the Q-network's low-variance/low-service profile from the high-profit models. They do not establish superiority among transformer, XGBoost, LNN + XGBoost and LSTM, whose mean profits differ by less than 0.2%.

The Pearson heatmaps in Appendix 3 show that the direction and magnitude of pairwise feature associations vary across model families and echelons. Several order, demand, sales and inventory variables are moderately or strongly correlated, while some displayed inputs are weakly related. These patterns may reflect shared temporal structure and engineered lags, but they are not interpreted as model explanations or causal mechanisms.

4.4 Runtime and computational cost

We benchmark the actual frozen configurations selected by the development procedure, using a batch of 32 ten-step sequences, ten warm-up passes, 100 timed repetitions and Compute Unified Device Architecture (CUDA) synchronization on an RTX 4060 Laptop GPU and Intel i7-13700H (Table 7). Both LNN pipelines include GPU-to-CPU state transfer and their CPU readout. The 128-unit LNN has 18,049 neural parameters, far fewer than the selected transformer (1.78 M), but LNN + XGBoost is the slowest measured pipeline at 20.26 ± 2.75 ms per batch. Parameter count therefore does not imply end-to-end speed.

4.5 Proof-of-concept replay using observed DataCo order quantities

As an observed-order replay, we use the public DataCo Smart Supply Chain dataset (Constante *et al.*, 2019), comprising 180,519 order-line records whose order dates span 2015–01 through 2018–01. Order Item Quantity is aggregated to a daily series by calendar day, giving approximately 1,127 consecutive daily buckets; missing days are retained as zero observed quantity, and values above the 99th percentile of the daily series are winsorized to that threshold to limit the influence of extreme spikes. These transaction-derived quantities are a demand proxy: they may reflect realized orders or sales and need not equal unconstrained

Table 7. Actual frozen-configuration inference benchmark (batch 32, sequence length 10; mean ± SD over 100 repetitions after ten warm-ups)

Model	Neural parameters/config	Batch latency (ms)	Samples/s
LNN + XGBoost	18,049 + 150 trees	20.260 ± 2.753	1,579
LNN linear	18,049 + ridge	13.282 ± 2.972	2,409
XGBoost	150 trees, depth 4	7.016 ± 0.685	4,561
LSTM	274,689	1.511 ± 0.499	21,184
Transformer	1,780,865	7.053 ± 3.548	4,537
Q-network	10,689	0.198 ± 0.083	161,980

latent demand. The first 60% of the chronological series (the earliest calendar days) is used for fitting and the final 40% for replay over ten model-initialization seeds (42–51). Price, cost, holding penalty, shortage penalty, lead time and batch size are experimental assumptions rather than observed operating records. Because this single-series replay has one demand stream rather than the multi-echelon cascade of the primary experiment, we evaluate the deployment-relevant policies for that setting – the classical base-stock benchmark, standalone XGBoost, LSTM, standalone LNN and the LNN + XGBoost hybrid – which retrain directly on the single-echelon trace without structural change. We omit the transformer, whose multi-head attention configuration was tuned for the multi-tier sequence structure and would require separate re-optimization on this trace, and the Q-network, whose reward and action design is defined over the multi-echelon simulation rather than a fixed observed-order series. The exercise tests a policy against an empirical temporal trace, not against a complete real multi-echelon inventory history.

Table 8 reports mean ± standard deviation. All policies are loss-making under the assumed cost configuration, so the comparison concerns loss mitigation. The deterministic base-stock and XGBoost policies repeat exactly and therefore have zero initialization variance. The hybrid reduces the base-stock loss by 35.9% and raises fill rate from 0.602 to 0.622, but standalone XGBoost has the smallest mean loss and highest fill rate, while LSTM has the lowest variance ratio. The replay demonstrates feasibility under an observed temporal trace, not external validation of a complete industrial supply chain and not superiority.

Because the DataCo ordering rules do not take monetary costs as inputs, we conduct a transparent ex-post accounting sensitivity analysis on the same saved trajectories rather than imply policy re-optimization. Table 9 varies shortage penalty from 0.5 to 2.0 times price and doubles holding cost. The hybrid improves on base-stock under every scenario, but standalone XGBoost remains the most profitable policy. Thus, the feasibility result is not an artifact of one cost setting, while the conclusion that the hybrid is not universally preferable is also stable (see Table 9).

5. Discussion and recommendations

On untouched test periods, transformer, XGBoost and LNN + XGBoost are practically tied in mean profit, with overlapping intervals and no Holm-adjusted pairwise differences.

Table 8. Closed-loop DataCo comparison (mean ± standard deviation over 10 initialization seeds). All policies are loss-making under the assumed cost configuration; less-negative profit is better

Policy	Total profit ($)	Fill rate	Bullwhip ratio
Base-stock	−1, 245, 695 ± 0	0.602 ± 0.000	0.480 ± 0.000
XGBoost-Opt	−708, 205 ± 0	0.626 ± 0.000	0.513 ± 0.000
LSTM-Opt	−784, 491 ± 64, 816	0.622 ± 0.003	0.353 ± 0.040
LNN-Opt	−786, 868 ± 300, 645	0.621 ± 0.012	0.445 ± 0.063
Hybrid-Opt (LNN + XGBoost)	−797, 921 ± 36, 242	0.622 ± 0.001	0.532 ± 0.056

Table 9. DataCo cost sensitivity: mean total profit in USD millions over ten initialization seeds. Policy trajectories are held fixed; only their accounting valuation changes

Scenario	Base	XGB	LSTM	LNN	Hybrid
Low shortage penalty	2.097	2.436	2.390	2.399	2.378
Baseline costs	−1.246	−0.708	−0.784	−0.787	−0.798
High shortage penalty	−7.932	−6.996	−7.134	−7.158	−7.150
High holding cost	−1.252	−0.773	−0.834	−0.813	−0.855

LNN + XGBoost has a high fill rate but more manufacturer variance amplification than XGBoost and transformer. The matched linear contrast yields a large standardized profit difference and an exact Wilcoxon result in the same direction, but the $p = 0.072$ Holm result prevents a family-wise significance claim. The Q-network provides the strongest variance damping only by accepting materially lower profit and fill rate. DataCo cost scenarios favor standalone XGBoost, and the selected LNN + XGBoost pipeline is slowest in the measured inference benchmark. The defensible conclusion is competitive performance with explicit trade-offs, not dominance or proven nonlinear-readout superiority.

5.1 Managerial implications

The results support only a conditional, locally re-estimated deployment rule. If low latency and tabular inputs dominate, standalone XGBoost is the simpler candidate. The LNN pipeline is a candidate only when a local matched-ablation test shows material value and the organization accepts additional pipeline overhead. If variance amplification dominates, lower-variance baselines should be examined. These are screening recommendations, not universal prescriptions; deployment requires local lead times, capacities, costs and prospective validation.

5.2 Limitations

Several limitations temper these findings. First, the primary experiment uses synthetic demand, a fixed one-day lead time and no supplier capacity constraints. DataCo contributes an observed order-quantity trace but not actual inventory states, lost demand, capacities, costs or lead times; its sensitivity is ex-post accounting on fixed trajectories. Second, the implemented LNN uses a simplified adaptive-leak recurrent cell rather than a canonical LTC/CfC implementation, and the cascade is not jointly optimized. Third, Optuna targets manufacturer profit on one separate development seed, which protects the final test but may not span all regimes. Fourth, ten evaluation seeds limit power for small effects, as illustrated by the ablation's raw and multiplicity-adjusted results. Fifth, runtime covers one hardware configuration. Future work should use multiple industrial datasets, stochastic lead times, capacity constraints, rolling-origin external validation, cost-aware retraining and canonical continuous-time comparators.

5.3 Data and code availability

The DataCo source is identified by DOI 10.17632/8gx2fvg2k6.5. The synthetic generator, development seed (32), evaluation seeds (42–51), chronological split, frozen hyperparameters, per-seed test arrays and analysis outputs are included in the reproducibility package accompanying the revision. The audited environment used Python 3.12.10, PyTorch 2.6.0+cu124, XGBoost 3.1.2, Optuna 4.5.0, scikit-learn 1.6.1, pandas 2.3.3 and NumPy 2.2.6. Inference hardware and timing protocol are reported in Section 4.4.

6. Conclusion

This study evaluates a two-stage LNN + XGBoost pipeline under a leakage-free development/test design. Across 10 unseen demand seeds, Transformer, XGBoost and LNN + XGBoost achieve nearly identical mean test profit. The hybrid improves mean profit by $14,477 over its exact-matched linear readout with a large standardized effect, but that contrast does not remain significant after correction across all profit pairs. XGBoost and transformer also produce lower manufacturer variance ratios, while the Q-network trades substantial profit and service for stronger variance damping. DataCo replay and every accounting scenario favor standalone XGBoost, and the actual frozen hybrid is not latency-optimal. LNN + XGBoost is therefore a competitive but more complex option whose value must be established locally; it is not a universal replacement for simpler models.

Supplementary material
The supplementary material for this article can be found online

References

Aguiar-Pérez, J.M. and Pérez-Juárez, M.A. (2023), "An insight of deep learning based demand forecasting in smart grids", *Sensors*, Vol. 23 No. 3, p. 1467, doi: 10.3390/s23031467.

Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M. (2019), "Optuna: a next-generation hyperparameter optimization framework", *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Anchorage, AK, pp. 2623-2631, doi: 10.1145/3292500.3330701.

Benidis, K., Rangapuram, S.S., Flunkert, V., Wang, Y., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Aubet, F.-X., Callot, L. and Januschowski, T. (2022), "Deep learning for time series forecasting: tutorial and literature survey", *ACM Computing Surveys*, Vol. 55 No. 6, pp. 1-36, doi: 10.1145/3533382.

Bertsimas, D. and Thiele, A. (2006), "A robust optimization approach to inventory theory", *Operations Research*, Vol. 54 No. 1, pp. 150-168, doi: 10.1287/opre.1050.0238.

Borovykh, A., Bohte, S. and Oosterlee, C.W. (2017), "Conditional time series forecasting with convolutional neural networks", arXiv preprint arXiv:1703.04691, doi: 10.48550/arXiv.1703.04691.

Cao, K., Zhang, T. and Huang, J. (2024), "Advanced hybrid LSTM-transformer architecture for real-time multi-task prediction in engineering systems", *Scientific Reports*, Vol. 14 No. 1, p. 4890, doi: 10.1038/s41598-024-55483-x.

Chaharsooghi, S.K., Heydari, J. and Zegordi, S.H. (2008), "A reinforcement learning model for supply chain ordering management: an application to the beer game", *Decision Support Systems*, Vol. 45 No. 4, pp. 949-959, doi: 10.1016/j.dss.2008.03.007.

Chen, T. and Guestrin, C. (2016), "XGBoost: a scalable tree boosting system", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, pp. 785-794, doi: 10.1145/2939672.2939785.

Constante, F., Silva, F. and Pereira, A. (2019), "DataCo smart supply chain for big data analysis (version 5) [data set]", *Mendeley Data*. doi: 10.17632/8gx2fvg2k6.5.

Graves, A. (2013), "Generating sequences with recurrent neural networks", arXiv preprint arXiv: 1308.0850.

Hasani, R., Lechner, M., Amini, A., Rus, D. and Grosu, R. (2020), "Liquid time-constant networks", arXiv preprint arXiv:2006.04439 [cs.NE], doi: 10.48550/arXiv.2006.04439

Hasani, R., Lechner, M., Amini, A., Liebenwein, L., Ray, A., Tschaikowski, M., Teschl, G. and Rus, D. (2022), "Closed-form continuous-time neural networks", *Nature Machine Intelligence*, Vol. 4 No. 11, pp. 992-1003, doi: 10.1038/s42256-022-00556-7.

Jahin, M.A., Shahriar, A. and Amin, M.A. (2024), "MCDFN: supply chain demand forecasting via an explainable multi-channel data fusion network model", arXiv preprint arXiv:2405.15598 [cs.LG], doi: 10.48550/arXiv.2405.15598

Ji, X., Li, J., Liu, X., Ding, Z., Zhang, L., Deng, J., Shi, J. and Ma, Q. (2024), "A novel K-means-LSTM hybrid model of supply chain inventory management", *Proceedings of the 5th International Conference on Computer Information and Big Data Applications*, pp. 419-425. doi: 10.1145/3671151.3671227.

Oroojlooyjadid, A., Nazari, M.R., Snyder, L.V. and Takáč, M. (2017), "A deep Q-network for the beer game: a deep reinforcement learning algorithm to solve inventory optimization problems", arXiv preprint arXiv:1708.05924 [cs.LG].

Silver, E.A., Pyke, D.F. and Peterson, R. (1998), *Inventory Management and Production Planning and Scheduling*, 3rd ed., Wiley.

Wibawa, A.P., Utama, A.B.P., Elmunsyah, H., Pujianto, U., Dwiyanto, F.A. and Hernandez, L. (2022), "Time-series analysis with smoothed convolutional neural network", *Journal of Big Data*, Vol. 9 No. 1, p. 44, doi: 10.1186/s40537-022-00599-y.

Zeng, A., Chen, M., Zhang, L. and Xu, Q. (2022), "Are Transformers effective for time series forecasting?", arXiv preprint arXiv:2205.13504 [cs.LG], doi: 10.48550/arXiv.2205.13504

Zhao, K. and Wang, C. (2017), "Sales forecast in e-commerce using convolutional neural network", arXiv preprint arXiv:1708.07946, doi: 10.48550/arXiv.1708.07946

Corresponding author
Chunan Tong can be contacted at: matttong@126.com