

Storage Location Assignment with Neural Embedding for Product Affinities and Sales Patterns

YUSUKE ASAI¹, KAIRI TAKAKI¹, SHIN KATAYAMA¹, KENTA URANO¹, (Member, IEEE),
TAKURO YONEZAWA¹, (Member, IEEE), and NOBUO KAWAGUCHI^{1,2}, (Member, IEEE)

¹Graduate School of Engineering, Nagoya University, Nagoya, Aichi Japan (e-mail: asayu@ucl.nuee.nagoya-u.ac.jp)

²Institute of Innovation for Future Society, Nagoya University, Nagoya, Aichi Japan

Corresponding author: Yusuke Asai (e-mail: asayu@ucl.nuee.nagoya-u.ac.jp).

This work was supported by Council for Science, Technology and Innovation, “Cross-ministerial Strategic Innovation Promotion Program (SIP), Development of foundational technologies and rules for expansion of the virtual economy”(JPJ012495).(funding agency: NEDO)
The authors would also like to thank TRUSCO Nakayama Corporation for their cooperation in the experiments.

ABSTRACT Warehouse storage location assignment affects operational efficiency, as order picking accounts for up to 55–65% of total warehouse costs. Most existing methods focus on a single factor such as product associations, seasonal patterns, or order frequencies – and either fix the layout statically or rely on continuous inventory relocation. We propose a dual-embedding framework that models product co-occurrence and weekly sales patterns in two parallel embedding pathways, trained in a single loop under a weighted combined loss and fused at inference. The resulting product representations enable a static storage layout to match the strongest dynamic-relocation baseline on real warehouse data, while requiring no inventory movement. We evaluate the method using one year of real warehouse data, measuring walking distance as the shortest path between two locations that respects the warehouse aisle structure. With dynamic relocation costs included, our approach reduces average walking distance by 37.3% compared to historical layouts. More importantly, our static layout (without any inventory movement) achieves a 36.9% reduction, statistically on par with ABC classification under dynamic relocation (36.5%). Under realistic $2 \times 5 \times$ relocation-cost multipliers, the static embedding layout significantly outperforms every relocation-based variant. This demonstrates that embedding-based assignment matches the best dynamic-relocation baseline at the published cost level and dominates it as relocation overhead grows. These results suggest that the proposed embedding-based approach captures stable and meaningful product relationships that remain effective even when demand fluctuates. This allows warehouse operators to maintain high picking efficiency without the operational complexity and labor costs associated with frequent inventory reorganization. In future work, we plan to incorporate demand forecasting models to further improve adaptability in complex warehouse environments.

INDEX TERMS Warehouse Storage Location Assignment, Product Embedding, Late Fusion, Order Picking, Warehouse Management, Supply Chain Management.

I. INTRODUCTION

AS supply chains undergo digital transformation [1]–[4], warehouses face increasing pressure to improve storage and picking efficiency. This pressure stems from booming e-commerce demand and labor constraints [5]. Order picking is the most labor-intensive warehouse operation, accounting for up to 55–65% of total operating expenses [6], [7]. This high cost share means that even modest gains in picking efficiency can significantly reduce overall warehouse operating costs. A key lever for such gains is storage location assignment.

How products are organized within the warehouse affects travel distances and fulfillment times. Optimizing storage location assignment is therefore critical for accelerating order throughput and cutting labor costs in distribution centers [8].

Two primary product characteristics influence optimal storage arrangements: product affinities and sales patterns. Product affinity refers to the tendency of certain items to be ordered together. For example, in an e-commerce setting, customers often buy complementary products such as batteries with electronics in the same order. Clustering such

products in nearby locations can shorten picker routes and improve order picking efficiency [9].

In contrast, demand-frequency slotting strategies prioritize products by pick frequency or sales volume. These strategies allocate prime locations to fast movers, ensuring that high-demand products are easy to access during peak periods [10].

Each strategy on its own can improve certain aspects of task performance. However, focusing on a single characteristic overlooks the multi-dimensional nature of warehouse operations. Real-world picking behavior reflects both what products are ordered together and when their demand fluctuates. This suggests that storage location assignment should consider these dimensions jointly rather than separately.

A layout based on product affinities might neglect temporal variations in demand. For instance, a product pair that sells together mainly in the winter should not occupy prime space year-round at the expense of summer-season top sellers. On the other hand, a layout optimized only for demand frequency might ignore products that are frequently picked in pairs or groups. This can cause pickers to travel extra distances when those items are stored far apart.

To address these challenges, we propose an embedding-based framework that balances product affinity learning with temporal sales modeling. Fig. 1 illustrates this framework. The key idea is to jointly model how products sell over time and how they relate to each other in customer orders. The framework groups similar products into clusters and assigns them to storage locations. It also incorporates radial sector assignment strategies and accessibility-prioritized location selection to minimize picking travel distances.

We evaluate on one year of real picking transactions from a Japanese B2B distributor (495,517 events, 36,729 SKUs), measuring walking distance as the shortest path between two locations that respects the warehouse aisle structure. The Weighted Sum approach reduces mean walking distance by 37.3% when dynamic relocation costs are charged, and 36.9% in its static configuration – statistically on par with ABC Merge+Cost (36.5%), and strictly superior under $2\times/5\times$ relocation-cost multipliers. The static ABC baseline, by contrast, degrades from 52.4% in period 1 to 13.2% in the final period.

Position relative to prior work. Network embeddings have been applied to a storage location assignment before: Qiu et al. [11] learn product embeddings from an order-co-occurrence graph and cluster them via K-means to place strongly related goods on the same rack. Dynamic-slotting methods (e.g., [12], [13]) adapt the layout to demand shifts. Each prior approach uses a single signal – co-occurrence or sales – whereas our framework learns both signals as parallel embeddings within a single model and shows that the resulting static layout matches the strongest dynamic-relocation baseline.

The main contributions of this paper are:

- 1) **Dual co-occurrence and sales-pattern embeddings with late fusion.** A skip-gram pathway and a sales-vector reconstruction pathway learn independent 64-

dimensional embeddings under a single weighted combined loss; the two representations are fused at inference by a tunable weight ω .

- 2) **Static-layout parity over dynamic ABC.** The static Weighted-Sum layout is statistically on par with ABC Merge+Cost (39.2% vs. 38.7%) and strictly superior under $2\times/5\times$ relocation costs – matching ABC's reduction without its relocation overhead.
- 3) **Comprehensive Evaluation.** We provide extensive experimental validation using both synthetic datasets and real warehouse operational data comprising 495,517 transactions of picking tasks across 36,729 SKUs.

The remainder of this paper is organized as follows. Section II reviews related work on storage location assignment problems and product embedding techniques in warehouse operations. Section III presents our dual-embedding methodology and storage location assignment framework. Section IV details the experimental setup, including synthetic data generation and real warehouse data characteristics, and analyzes performance results. Finally, Section V concludes with implications for warehouse management practice and directions for future research.

II. RELATED WORK

A. STORAGE LOCATION ASSIGNMENT

The Storage Location Assignment Problem (SLAP) has been studied using various approaches over the past decades [8], [9], [14]. Early research focused on mathematical optimization through integer programming models. For example, Frazee et al. [15] formulated SLAP as an integer programming model to minimize total travel time in order picking operations. This approach, while providing optimal solutions for small-scale problems, faces computational challenges when applied to real-world warehouse scenarios due to its NP-hard nature [16].

More recently, the field has witnessed a shift toward data-driven approaches [13], [14] that leverage historical order data to make storage decisions. These methods can be categorized into two streams: association-based approaches [16], [17] and trend analysis [18]. In the association category, Chuang et al. [19] pioneered the use of association rule mining to compute correlation between items, introducing the Association Index (AIX) metric to quantify item relationships. This approach was further refined by Pang et al. [20] who developed a data mining-based algorithm that simultaneously optimized order picking and put-away distances. However, these association-based methods can be sensitive to noise in historical data and may perform poorly when dealing with newly introduced products or rapidly changing consumer preferences. Additionally, they often require significant computational resources to process large-scale transaction data.

ABC trend analysis [21] represents another data-driven approach. It groups items into three categories based on turnover frequency: fast-moving (A), medium (B), and slow-moving (C). Zhou et al. [22] proposed combining ABC clas-

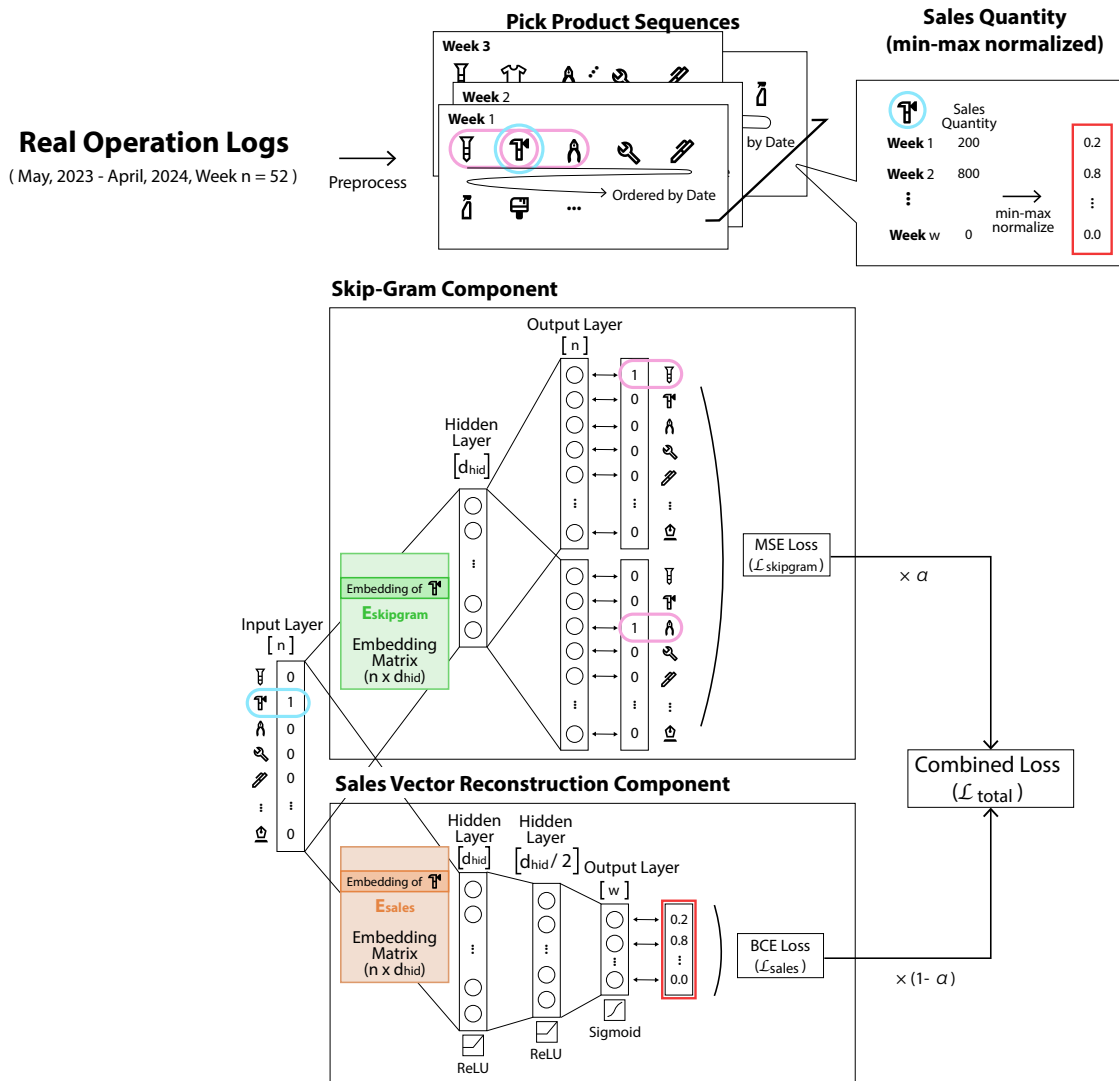


FIGURE 1. Dual-embedding framework for product representation. The skip-gram and sales vector reconstruction components use independent embedding matrices (E_{skipgram} and E_{sales}), trained in a single loop under the combined loss $\mathcal{L}_{\text{total}}$ and fused at inference.

sification with association rules to improve storage location decisions. This hybrid approach outperformed conventional ABC methods. However, it still struggles with products that have irregular demand patterns. It can also lead to poor space utilization when product volumes fluctuate significantly. Additionally, the fixed category boundaries in ABC analysis may not capture the continuous nature of demand variations.

ABC classification methods can become suboptimal under volatile demand [12]. Recent work [23], [24] has explored dynamic storage location assignment methods that adapt slotting decisions based on updated sales data or SKU interactions. Pierre et al. [10] and Kofler et al. [25] also proposed dynamic ABC variants that periodically reclassify items and relocate them based on stability, urgency, and demand shifts. These methods balance picking efficiency against relocation cost and show performance gains over static layouts. How-

ever, dynamic relocation requires additional labor to move items. It must also be carefully scheduled to avoid disrupting ongoing operations.

B. PRODUCT EMBEDDING TECHNIQUES

Word and item embeddings have evolved from early discrete vector representations (e.g., one-hot or bag-of-words) to neural network-based approaches. Traditional statistical techniques suffered from high dimensionality and sparsity, limiting their ability to generalize across contexts [26]. A breakthrough came with Word2Vec [27] which learns vector representations by predicting contextual co-occurrences. This approach showed that words sharing similar contexts can be mapped to similar positions in latent space.

These advances have been extended beyond natural language processing. Product embeddings in recommender

systems adopt Word2Vec-inspired architectures, such as Item2Vec [28] and Prod2Vec [29], to capture item relationships from transactional data.

In warehouse operations, Qiu et al. [11] applied neural network embeddings to product co-order graphs for the SLAP. Their approach clusters products based on co-occurrence patterns to optimize the placement of mobile racks. This improved order-picking efficiency. However, their model captures only relational proximity. It does not incorporate seasonal demand variation or product turnover rates.

A related line of work studies meta-embeddings, which combine embeddings trained independently from different sources into a single representation. Yin and Schütze [30] learn projections of multiple source embeddings into a shared space, and Coates and Bollegala [31] show that simple (weighted) averaging of source embeddings performs comparably to concatenation even though the source spaces share no common coordinate system, because independently trained high-dimensional embedding spaces are approximately orthogonal to each other, so averaging approximately preserves the distance structure of concatenation.

Our work extends these methods by introducing a dual-embedding framework that models transactional co-occurrences and seasonality in parallel pathways combined by late fusion. The Weighted Sum fusion we employ (Section III-C2) is an instance of meta-embedding averaging [31]. This enables more effective storage location assignment by capturing both product similarity and temporal demand behavior.

III. METHODOLOGY

We propose a dual-embedding framework for storage location assignment based on a late fusion strategy. We train the skip-gram embedding (E_{skipgram}) and the sales reconstruction embedding (E_{sales}) in a single training loop, then combine them at the final stage to construct the product representation for storage decisions.

A. WAREHOUSE DESCRIPTION AND ASSUMPTIONS

Our experimental framework models a multi-block warehouse layout based on the real environment shown in Fig. 2. The target floor consists of 1,462 storage shelves. Each location is identified by discrete (x, y) coordinates. The floor has a centralized depot as the primary shipping location and is divided into four operational zones (Zone 1, 2, 3, and 4) for zone-based picking and spatial clustering of related products.

Each storage shelf accommodates up to 30 stock keeping units (SKUs). For capacity calculations, we consider only the number of distinct SKUs per shelf, ignoring the quantity of each SKU. This simplifies the problem by treating each product type as occupying one slot regardless of inventory volume. The system can then focus on optimizing product co-location and accessibility rather than space utilization.

All storage and picking tasks are recorded in the Warehouse Management System (WMS) through barcode scanning. Workers scan product barcodes and location barcodes

during storage assignment and picking. Each picking task has a unique batching number that identifies which orders are grouped together. This provides operational records of what products are handled, when operations occur, who performs each task, and which orders are batched together. We use these records for embedding model training and evaluation.

The primary research objective is to minimize the total walking distance. Let (x_l, y_l) denote the coordinates of storage location l and let (x_d, y_d) denote the coordinates of the depot. To respect the constraint that picker travel follows the warehouse's aisle network, we evaluate this objective under a graph-based shortest-path metric. We construct an aisle-network graph $G = (V, E)$ with 1,890 nodes of four types: (i) the access points of all storage locations, (ii) cross-aisle junctions at the top and bottom of each aisle, (iii) shared-walkway nodes at the cross-aisles between vertically adjacent blocks, and (iv) the depot. Edges connect (i) consecutive access nodes within each aisle, (ii) adjacent aisles' junctions along each block's cross-aisles, (iii) each block's boundary junctions to the shared walkway, and (iv) the depot to its nearest junction. Each edge weight is the Euclidean length of the corresponding segment. The walking distance between any two locations a and b is the shortest-path length on G :

$$d(a, b) = \min_{\pi \in \Pi(a, b)} \sum_{(u, v) \in \pi} \|u - v\|_2,$$

where $\Pi(a, b)$ is the set of all paths from a to b in G . The optimization problem is then formulated as:

$$\begin{aligned} \text{minimize} \quad & \sum_{b \in B} \left[d((x_d, y_d), (x_{l_1}, y_{l_1})) \right. \\ & + \sum_{i=1}^{|E_b|-1} d((x_{l_i}, y_{l_i}), (x_{l_{i+1}}, y_{l_{i+1}})) \\ & \left. + d((x_{l_{|E_b|}}, y_{l_{|E_b|}}), (x_d, y_d)) \right] \end{aligned}$$



FIGURE 2. Our target warehouse layout and picking task illustration.

where B is the set of all picking batches, E_b is the set of picking events within batch b , and l_i is the storage location for the i -th picking event. Each batch starts from the depot (x_d, y_d) , visits all required picking locations in sequence, and returns to the depot.

Section IV-F compares this metric against three alternatives (Euclidean, rectilinear/Manhattan, and an S-shape model [32]) and shows that all four metrics yield qualitatively consistent rankings, differing from the graph metric by only a few percentage points. Optimizing picking routes within batches is a separate research problem [16], [33] beyond the scope of this study; we focus on storage location assignment rather than route optimization, which allows evaluation of storage strategy effectiveness independent of routing algorithms.

The accessibility of location l is defined as:

$$\text{accessibility}(l) = \frac{1}{\sqrt{(x_l - x_d)^2 + (y_l - y_d)^2}}$$

where (x_l, y_l) are the coordinates of location l and (x_d, y_d) are the depot coordinates. Higher accessibility scores indicate locations closer to the depot that require less travel time for picking. The depot serves as the reference point for all distance and accessibility calculations. Worker travel time is directly proportional to the aisle-network shortest-path distance defined above between the depot and storage locations; the accessibility heuristic above retains the Euclidean form because it is used only as a quick ranking score during storage assignment, not as a measure of actual walking. Our model assumes the following:

- Perfect inventory visibility with real-time tracking of product locations and quantities.
- Unrestricted worker access to any storage location.
- Deterministic travel times without congestion or operational delays.
- Consistent product physical characteristics that do not affect storage or retrieval.

These assumptions simplify routing decisions while maintaining realistic constraints, allowing the system to focus on optimizing spatial relationships and picking efficiency.

B. DUAL-EMBEDDING FRAMEWORK

Our model processes two data streams: temporal order sequences and sales data. Order sequences are created by sorting pick products by time within defined time windows. This produces a set of product co-occurrences. Sales data is organized as a matrix of n products by w weeks. The neural network has two pathways: a skip-gram component and a sales component ($d_{hid} = 64$). The skip-gram component captures which products appear together. The sales component learns to reconstruct weekly sales patterns from product embeddings. The skip-gram weight α , window size W , embedding dimension d_{hid} , and cluster count k are each empirically justified in Section IV-D via Friedman tests across seeds and per-treatment trade-off analyses.

1) Skip-Gram Component

The skip-gram component is based on Word2vec [27], adapted to learn product relationships from transactions. It uses separate target and context embedding layers that map product IDs to the same embedding dimension. This allows the model to capture how products relate to each other in transaction data. The model uses a sliding window over transaction sequences. For each position, the center product becomes the target, and nearby products within the window become context products. This creates target-context pairs that capture local co-occurrence patterns.

The model computes similarity using a dot product between target and context embeddings, followed by sigmoid activation. This produces probability values that represent how likely two products are to appear together in transactions.

2) Sales Vector Reconstruction Component

The sales vector reconstruction component learns to model sales patterns through a neural network. It first converts product IDs into dense vector representations using a sales embedding layer. This maps each product ID into a continuous vector space, allowing the model to learn product-specific sales patterns.

Before training, weekly sales data is normalized using min-max scaling. For each product, sales values are scaled to the range $[0, 1]$ by subtracting the minimum and dividing by the range. This preserves relative patterns while standardizing the scale across products with different sales volumes.

The product embeddings pass through three fully connected layers. The first layer transforms the embedding to 64 ($= d_{hid}$) units. The second layer reduces it to 32 ($= d_{hid}/2$) units. The final layer outputs sales values for each week. Each hidden layer uses ReLU activation and dropout (probability 0.2) to improve learning and prevent overfitting.

The output layer uses sigmoid activation to produce values in $[0, 1]$. The output shape is $[\text{batch size} \times \text{number of weeks}]$, representing sales patterns for each product over the target time period.

3) Architecture Summary

The full model has three trainable embedding tables – $\mathbf{E}_{\text{sales}}$, $\mathbf{E}_{\text{target}}$, and $\mathbf{E}_{\text{context}}$, each of shape $|V| \times d_{\text{emb}}$ where $|V|$ is the vocabulary size (30,000 products in our deployment) and $d_{\text{emb}} = 64$ – and the sales-predictor Multi-Layer Perceptron (MLP). The sales-predictor MLP has three fully connected layers of sizes $d_{\text{emb}} \rightarrow d_{\text{hid}} \rightarrow d_{\text{hid}}/2 \rightarrow w$, that is $64 \rightarrow 64 \rightarrow 32 \rightarrow 52$, where $w = 52$ is the number of training weeks; hidden layers use ReLU activation and a dropout probability of 0.2, and the output layer uses sigmoid activation. The skip-gram component has no additional parameters beyond $\mathbf{E}_{\text{target}}$ and $\mathbf{E}_{\text{context}}$: the co-occurrence probability for a (target, context) pair is the sigmoid of the dot product of their respective embedding rows. Total trainable parameters: $3 \cdot |V| \cdot d_{\text{emb}} + (d_{\text{emb}} \cdot d_{\text{hid}}) + (d_{\text{hid}} \cdot d_{\text{hid}}/2) + (d_{\text{hid}}/2 \cdot w) + \text{biases} \approx 5.77M$ for our $|V| = 30,000$ deployment.

C. EMBEDDING TRAINING PROCESS

The training process combines both components through a unified loss function. The sales component uses Mean Squared Error (MSE) loss to measure the difference between predicted and actual sales values. The skip-gram component uses Binary Cross-Entropy (BCE) loss to predict product co-occurrence, with positive pairs labeled as 1.0. The two losses are combined using a weighted average:

$$\mathcal{L}_{total} = (1 - \alpha) \times \mathcal{L}_{sales} + \alpha \times \mathcal{L}_{skipgram}$$

where α is the skip-gram weight. This allows the model to learn both sales patterns and product relationships together. The weight α controls the balance between the two tasks.

The model uses the Adam optimizer for training. Each batch contains product indices, sales vectors, and context product indices. During training, the model updates all embedding layers and network parameters together.

1) Loss-scale Balance.

The combined loss mixes Mean Squared Error (MSE; regression, unbounded above) with Binary Cross-Entropy (BCE; classification, bounded in $[0, \log 2]$ for sigmoid outputs and a constant target of 1.0). Because the two terms operate on different scales, α controls not only which task receives more emphasis but implicitly also the gradient magnitude balance between the two heads. Because the two pathways share no trainable parameters, $\mathcal{L}_{total}$ decomposes across pathways; α therefore acts as a relative scaling of each pathway's effective learning rate rather than a gradient-mixing weight on shared parameters. We treat α as a single empirically tuned hyperparameter and select $\alpha = 0.85$ via the sensitivity analysis described in Section IV-D. Adaptive loss-balancing schemes such as GradNorm [34] and the homoscedastic-uncertainty weighting of Kendall et al. [35] were not explored in this work; they may further improve the balance and are a natural direction for follow-up.

2) Embedding Fusion

After training, the skip-gram embedding ($\mathbf{E}_{skipgram}$; we use the target table $\mathbf{E}_{target}$ at inference) and the sales reconstruction embedding ($\mathbf{E}_{sales}$) are combined into the final product representation $\mathbf{E}_{final}$, which drives the clustering and storage-assignment decisions in the remainder of this section. We propose two fusion strategies.

1. Concatenation: The two vectors are concatenated:

$$\mathbf{E}_{final} = \mathbf{E}_{skipgram} \oplus \mathbf{E}_{sales}$$

This preserves full dimensionality and lets the clustering algorithm learn the weighting between product affinity and sales patterns.

2. Weighted Sum: The two vectors are combined using a weight ω :

$$\mathbf{E}_{final} = \omega \cdot \mathbf{E}_{skipgram} + (1 - \omega) \cdot \mathbf{E}_{sales}$$

where $\omega \in [0, 1]$ controls the balance between skip-gram ($\omega \rightarrow 1$) and sales ($\omega \rightarrow 0$) components.

The Weighted Sum is a weighted form of embedding averaging: when the source spaces are approximately orthogonal, as is typical of independently trained high-dimensional embeddings [31], the average approximates the distance structure of the corresponding weighted concatenation while halving the dimensionality.

3) Relationship Between Training-time α and Inference-time ω .

The skip-gram loss weight α and the Weighted-Sum fusion weight ω , equation for $\mathbf{E}_{final}$ control different aspects of the same skip-gram-vs-sales trade-off. α shapes what the embeddings learn – a higher α shifts training capacity toward co-occurrence prediction, so $\mathbf{E}_{target}$ encodes affinity more sharply at the expense of seasonality fidelity in $\mathbf{E}_{sales}$. ω shapes how the learned embeddings are combined at inference – $\omega \rightarrow 1$ recovers pure skip-gram, $\omega \rightarrow 0$ recovers pure sales, and intermediate values blend them. We tune the two hyperparameters separately and sequentially: α is fixed at 0.85 during training (chosen by the intrinsic-eval Friedman analysis in Section IV-D), and ω is then swept at inference time over the trained embeddings (Section IV, Fig. 11). Joint (α, ω) optimisation is left as future work; we expect the optimum to remain in the high- α , high- ω quadrant because the pure-skip-gram baseline ($\omega = 1$) already captures most of the Weighted-Sum reduction (see Section IV-E).

D. STORAGE LOCATION ASSIGNMENT OF PRODUCTS

Our method assigns storage locations based on product embeddings and warehouse layout. It consists of four phases: (1) period-based cluster assignment, (2) radial sector assignment, (3) accessibility-prioritized location selection, and (4) similarity-based overflow handling.

1) Period-Based Cluster Assignment

We use a dynamic clustering system where product assignments change over time. Each period spans T_{period} days, and clusters are reassigned based on demand patterns:

$$\text{period}(t) = \left\lfloor \frac{(t - t_{start})}{T_{period}} \right\rfloor + 1$$

where t is the timestamp and t_{start} is the simulation start date. For each period p and cluster c , we define valid storage locations as:

$$L_{c,p} = \{l \mid \text{cluster_period}_p(l) = c\}$$

where $\text{cluster_period}_p(l)$ indicates the cluster assignment of location l in period p .

2) Coordinate System and Sector Definition

We divide the warehouse into sectors using polar coordinates centered on the depot. Each storage location l is described by its radius and angle from the depot:

$$\begin{aligned} \text{radius}(l) &= \sqrt{(x_l - x_d)^2 + (y_l - y_d)^2} \\ \text{angle}(l) &= \arctan 2(y_l - y_d, x_l - x_d) \end{aligned}$$

where (x_l, y_l) are the coordinates of location l and (x_d, y_d) are the depot coordinates. Each sector is defined by an angular range $[\theta_i, \theta_{i+1})$ and a radial range $[r_j, r_{j+1})$. This gives each sector an identifier (i, j) for the i -th angular and j -th radial sector. Sector boundaries are set by dividing the coordinate ranges of available storage locations uniformly.

3) Cluster Representation and Assignment Strategy

Each cluster is represented by a centroid vector $\mathbf{c}_c$. The centroid is the normalized sum of embeddings from the top products in that cluster:

$$\mathbf{c}_c = \frac{\sum_{p \in P_c} \mathbf{e}_p}{\left\| \sum_{p \in P_c} \mathbf{e}_p \right\|_2}$$

where $\mathbf{e}_p$ is the embedding vector of product p , and P_c is the set of top N products in cluster c ranked by picking frequency.

Clusters are assigned to angular sectors based on similarity. Related clusters are placed in adjacent sectors. The cluster most similar to the period's top-picked products is assigned to the central angular sector. Remaining clusters are arranged based on their similarity to this reference cluster.

High-priority products are assigned to inner radial sectors to minimize travel distance from the depot. Priority levels range from 0 (highest) to $P_{\max}$. The radial assignment follows $r_{c,p} = \min(p_c, n_{\text{rad}} - 1)$, where p_c is the priority level of cluster c and n_{rad} is the total number of radial sectors.

4) Assignment Algorithm and Optimization

We score candidate sectors s for each cluster c in period p using three additive terms, each capturing a different intuition:

- **(1) Reference similarity:** clusters that match the period's top-picked products should sit near the warehouse's central angle. Term value is largest when (i) cluster c is similar to the period's reference cluster and (ii) sector s is close to the central angle.
- **(2) Cluster adjacency:** clusters similar to one another should sit in adjacent sectors. Term value is largest when s is angularly close to the sectors already chosen for similar clusters $c' \in \text{adj}(c)$.
- **(3) Temporal consistency:** a cluster should not move far from its previous-period angle, to avoid disruptive layout churn. Term value is largest when s is the same angular index as the cluster's position in period $p - 1$.

The corresponding closed form is

$$\begin{aligned} S(c, s, p) = & w_1 \cdot \text{sim}(c, \text{ref}_p) \cdot \left(1 - \frac{d_{\text{ang}}(s, s_{\text{center}})}{n_{\text{ang}}/2}\right) \\ & + w_2 \sum_{c' \in \text{adj}(c)} \text{sim}(c, c') \cdot \left(1 - \frac{d_{\text{ang}}(s, s_{c'})}{n_{\text{ang}}}\right) \\ & + w_3 \cdot \left(1 - \frac{|s_p - s_{p-1}|}{n_{\text{ang}}}\right), \end{aligned}$$

where $\text{sim}(c, c')$ is the cosine similarity between the centroids of clusters c and c' . The function $d_{\text{ang}}(s, s')$ gives the angular distance between sectors s and s' , and n_{ang} is the

total number of angular sectors. The weights w_1 , w_2 , and w_3 control reference similarity, cluster adjacency, and temporal consistency, respectively, with $w_1 + w_2 + w_3 = 1$. Each cluster c in period p is assigned to a sector $\text{sector}(c, p) = (a_{c,p}, r_{c,p})$, where $a_{c,p}$ and $r_{c,p}$ are the angular and radial sector indices.

5) Overflow Handling of Preferred Sectors

When a preferred sector is full, the algorithm finds an alternative location within a maximum Manhattan distance of d_{max} sectors. Products are assigned to the nearest available sector that satisfies $|a_{\text{new}} - a_{\text{pref}}| + |r_{\text{new}} - r_{\text{pref}}| \leq d_{\text{threshold}}$, where $d_{\text{threshold}}$ is the maximum allowed deviation from the preferred assignment. High-priority products ($p_i \leq P_{\text{high}}$) are assigned first to ensure they get optimal locations before sectors fill up.

Fig. 3 shows how product clusters are assigned across time periods in a warehouse with a central depot (red square). Higher-priority clusters are placed in inner radial sectors to minimize travel distance. Colored dots show products in preferred locations. X marks indicate overflow assignments when preferred sectors are full.

6) Tail Product Handling Strategy

Products with very low picking frequency are classified as "Tail Products". Concretely, we apply a per-period rank cutoff of $N_{\text{top}} = 10,000$: within each period, products ranked outside the top N_{top} by picking frequency in that period are marked "Unclustered". Since each of our 13 periods observes between 11,706 and 14,038 distinct picked SKUs, the cutoff selects approximately the most-picked 70–85% of in-period SKUs for clustered assignment and routes the remainder to unclustered storage. As shown in Fig. 4, products are ranked by picking frequency. High-frequency items keep their cluster assignments. Low-frequency items in the long tail are marked as "Unclustered". This keeps high-value storage zones for high-frequency products. If a tail product's picking frequency increases in a later period and crosses the N_{top} threshold, it can be reassigned to a cluster at the next period boundary.

7) Cold-Start Handling for New SKUs.

A SKU that has never been picked cannot have a meaningful embedding, because both pathways need historical picking data. We handle this with the Tail-Product mechanism: any product not in the trained vocabulary, or ranked below N_{top} in the current period, is treated as Unclustered and placed in the unclustered-zone overflow region by accessibility alone.

Two practical limitations follow. First, a new SKU's first appearance is placed by accessibility only, so co-pick opportunities can be missed. Second, the embedding only updates at re-training boundaries; warehouses with fast-changing catalogues may need monthly or shorter re-training.

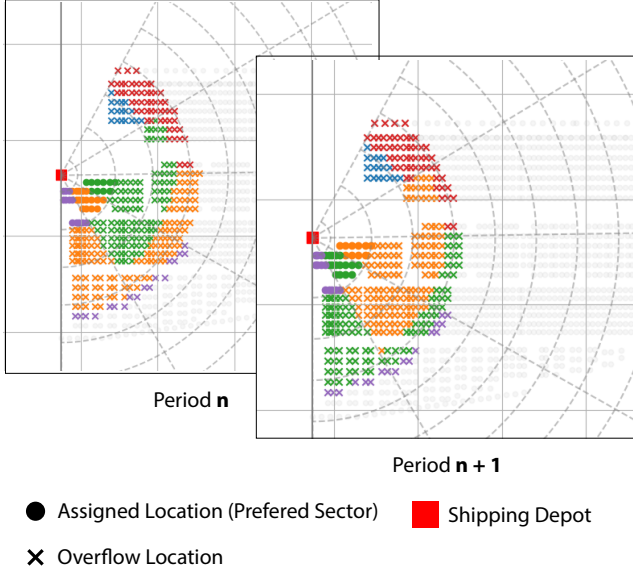


FIGURE 3. Cluster reassignment across warehouse periods showing radial-angular sector organization with period-based storage.

8) Accessibility-Prioritized Location Selection for Incoming Products

Within each cluster's storage area, we assign products based on accessibility scores. The assignment procedure follows two steps:

First, we try to store the product in a location that already contains the same product. We choose the most accessible one:

$$l_{\text{existing}}^* = \underset{l \in L_{c,p} \cap L_{\text{product}}}{\operatorname{argmax}} \operatorname{accessibility}(l)$$

where L_{product} is the set of locations currently storing the same product.

If no such location exists, we use similarity-based assignment. The similarity between products is:

$$\operatorname{similarity}(p_i, p_j) = \frac{\mathbf{e}_i \cdot \mathbf{e}_j}{\|\mathbf{e}_i\| \cdot \|\mathbf{e}_j\|}$$

where $\mathbf{e}_i$ and $\mathbf{e}_j$ are the embedding vectors of products i and j , respectively. The final assignment balances similarity and accessibility:

$$l^* = \underset{l \in L_{c,p}}{\operatorname{argmax}} \left[\max_{p_j \in \operatorname{inventory}(l)} \operatorname{similarity}(p_{\text{target}}, p_j) \cdot \operatorname{accessibility}(l) \right]$$

Replenishment Strategies: Merge vs. No Merge

Fig. 5 shows two replenishment strategies for the transition from period n to period $n + 1$. Products are categorized as “Clustered” (high picking frequency) and “Unclustered” (low picking frequency).

The Merge strategy relocates inventory only when a product switches between groups:

- **Unclustered → Clustered:** Existing inventory is moved to the designated clustered zone.
- **Clustered → Unclustered:** Inventory is moved to unclustered storage to free up high-value space.

If a product remains in the same group across periods, no relocation occurs. This minimizes handling costs while keeping stock consolidated.

The No Merge strategy keeps existing items in their original locations regardless of status changes. Incoming products are assigned based on accessibility without consolidating existing stock. This avoids relocation labor but may result in fragmentation, where a single product is split across multiple locations.

9) Overflow and Substitute Location Strategy for Incoming Products

When designated cluster locations are full, the system uses a multi-tier overflow strategy. First, we search for the same product in other locations for the current period:

$$L_{\text{same_product}} = \{l \mid p_{\text{target}} \in \operatorname{inventory}(l), \text{cluster_period}_p(l) \neq \text{none}\}$$

If found, we assign to the location with highest accessibility. Otherwise, we find the most similar product across all clustered locations:

$$(p^*, l^*) = \underset{(p, l) \in P_{\text{clustered}}}{\operatorname{argmax}} \operatorname{similarity}(p_{\text{target}}, p)$$

where $P_{\text{clustered}}$ is the set of all product-location pairs in clustered areas for the current period.

We then find the nearest unclustered location to this similar product:

$$l_{\text{substitute}}^* = \underset{l \in L_{\text{unclustered}}}{\operatorname{argmin}} \sqrt{(x_l - x_{l^*})^2 + (y_l - y_{l^*})^2}$$

where $L_{\text{unclustered}}$ is the set of unclustered locations, and (x_{l^*}, y_{l^*}) are the coordinates of the location containing the most similar product. This approach keeps overflow products close to their designated sectors and groups similar products together.

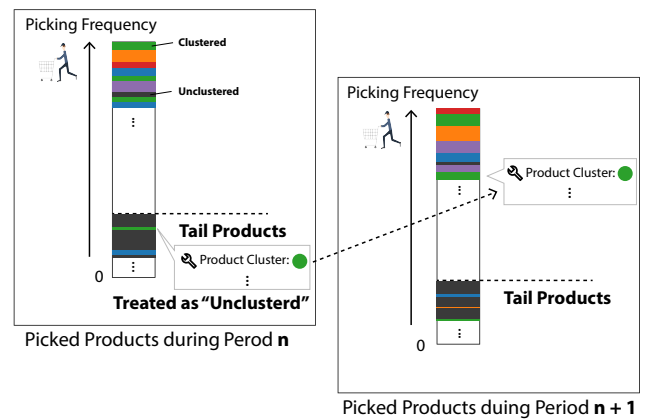


FIGURE 4. Handling of Tail Products. Low-frequency items are treated as “Unclustered” to simplify storage logic.

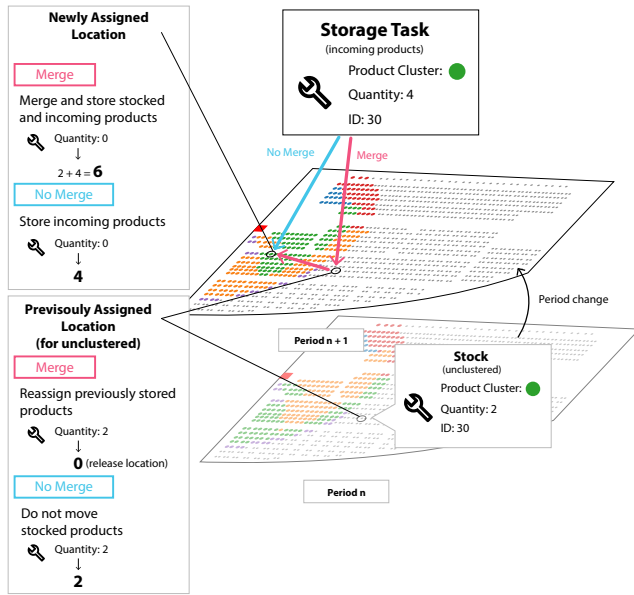


FIGURE 5. Two replenishment strategies during the transition from period n to period $n + 1$.

E. PICKING STRATEGY

Our picking strategy uses zone-based batching to minimize travel time. It combines temporal grouping, spatial optimization, and inventory management to improve picking efficiency.

1) Zone-Based Batch Formation

We group picking orders by time and location. Orders are first divided into time windows:

$$W_t = \left\lfloor \frac{t - t_{\text{day_start}}}{\Delta t} \right\rfloor$$

where t is the order timestamp, $t_{\text{day_start}}$ is the start of an operational day, and Δt is the time window duration.

Within each time window, orders are grouped by warehouse region. For each picking event e with product p , we select the most accessible location:

$$l_e^* = \underset{l \in L_p}{\operatorname{argmax}} \operatorname{accessibility}(l)$$

where L_p is the set of locations storing product p . The region is determined from the location:

$$\operatorname{region}(e) = \operatorname{parse_region}(l_e^*)$$

Batches are formed for each region-time combination:

$$B_{r,t} = \{e \mid \operatorname{region}(e) = r, \\ W_{\operatorname{timestamp}(e)} = t, \\ |B_{r,t}| \leq \operatorname{target_size}\}$$

where r is the warehouse region. The parameters Δt and $\operatorname{target_size}$ are set based on operational data.

2) Intra-Batch Organization

Within each batch, picking events are sorted by aisle, then by shelf:

$$\operatorname{sort_priority}(e) = (\operatorname{aisle}(l_e^*), \operatorname{shelf}(l_e^*))$$

where events are sorted in ascending order by aisle number, then by shelf number within each aisle. This provides a logical sequence of picking locations for generating efficient routes.

3) Distributed Picking Algorithm

When one location cannot fulfill a product request, the system picks from multiple locations in order of accessibility:

Algorithm 1 Distributed Picking Algorithm

```

1: Input: Product  $p$ , Required quantity  $Q$ , Event  $e$ 
2: Output: Success status, Pick records
3:  $Q_{\text{remaining}} \leftarrow Q$ 
4:  $\text{picks} \leftarrow \emptyset$ 
5:  $L_{\text{sorted}} \leftarrow \operatorname{sort\_by\_accessibility}(L_p)$ 
6: for  $l \in L_{\text{sorted}}$  do
7:    $Q_{\text{available}} \leftarrow \operatorname{inventory}(l, p)$ 
8:   if  $Q_{\text{available}} > 0$  then
9:      $Q_{\text{pick}} \leftarrow \min(Q_{\text{remaining}}, Q_{\text{available}})$ 
10:     $\operatorname{inventory}(l, p) \leftarrow \operatorname{inventory}(l, p) - Q_{\text{pick}}$ 
11:     $\text{picks} \leftarrow \text{picks} \cup \{(l, Q_{\text{pick}})\}$ 
12:     $Q_{\text{remaining}} \leftarrow Q_{\text{remaining}} - Q_{\text{pick}}$ 
13:    if  $Q_{\text{remaining}} = 0$  then
14:      return True,  $\text{picks}$ 
15:    end if
16:  end if
17: end for
18: return False,  $\emptyset$ 

```

4) Releasing Inventory Location

The system updates inventory after each pick. When a location becomes empty, the system decides whether to release it:

$$\operatorname{release}(l, p) = \begin{cases} \text{True} & \text{if } \operatorname{inventory}(l, p) = 0 \\ \text{False} & \text{otherwise} \end{cases}$$

Released locations are removed from the product's location set:

$$L_p \leftarrow L_p \setminus \{l \mid \operatorname{release}(l, p) = \text{True}\}$$

IV. EXPERIMENTS

This section evaluates the proposed embedding model using synthetic and real-world datasets. We first validate the model's ability to capture behavioral patterns and co-occurrence structures using synthetic data. We then compare the approach against ABC classification and operational baselines using one year of data. We test two embedding fusion strategies: Concatenation and Weighted Sum. Using a warehouse simulation that accounts for walking distance and relocation costs, we show that the Weighted Sum strategy

achieves better picking efficiency and stability than traditional methods.

A. SYNTHETIC DATA EXPERIMENTS

To evaluate the model, we constructed a synthetic dataset that mirrors real-world retail patterns. The dataset parameters were calibrated using actual warehouse data to ensure realistic behavior. Table 1 compares real and synthetic data characteristics. Fig. 6 shows the picking frequency distributions.

As shown in Table 2, the model uses a 64-dimensional embedding space and a 32-dimensional hidden layer. Training ran for 2 epochs using the Adam optimizer with a learning rate of 0.01. The skip-gram component uses a window size of 2 and a skip-gram weight (α) of 0.85.

TABLE 1. Comparison of Real and Synthetic Warehouse Data Characteristics

Characteristic	Real Data	Synthetic Data
Dataset Overview		
Total Products	36,729	30,000
Clustering Products	30,000	30,000
Total Transactions	495,517	553,546
Transactions for Training	445,392	553,546
Behavioral Clusters	5	5
Number of Weeks	52	48
Picking Frequency Statistics		
Mean Frequency	14.8	18.5
Median Frequency	8.0	10.0
Standard Deviation	29.0	35.3
Minimum Frequency	2	2
Maximum Frequency	1,888	834
Range	1,886	832
Percentile Analysis		
50th Percentile (picks)	8.0	10.0
80th Percentile	19.0	20.0
90th Percentile	32.0	33.0
95th Percentile	49.0	61.0
99th Percentile	112.0	191.0
Distribution Segmentation		
Low Freq. (2-15)	22378 (74.6%)	20,987 (70.0%)
Medium Freq. (16-50)	6201 (20.7%)	7,170 (23.9%)
High Freq. (51+)	1421 (4.7%)	1,843 (6.1%)
Long-tail Characteristics		
Power Law Alpha (α_p)	1.502	1.50
Min Power Law Frequency	3	2
Concentration (Top 20%)	High	High
Pareto Distribution	Yes	Yes
Co-occurrence Analysis		
Avg. Co-occurrence Prob.	16.2%	15.0%

a: Training cost and convergence.

At our 30,000-SKU vocabulary the model has approximately 5.77M trainable parameters and trains in ≈ 8 minutes per (configuration, seed) cell on a single NVIDIA RTX 3090 with 24 GB VRAM (10 epochs, batch size 32, learning rate

TABLE 2. Network Parameters for Synthetic and Real Data Training

Parameter	Value	Description
Embedding Dimension	64	Embedding vector size (d_{hid})
Hidden Dimension	32	Hidden layer size ($d_{hid}/2$)
Epochs	2	Training iterations
Learning Rate	0.01	Adam optimizer rate
Window Size	2	Skip-gram window
Skip-gram Weight	0.85	Embedding loss weight

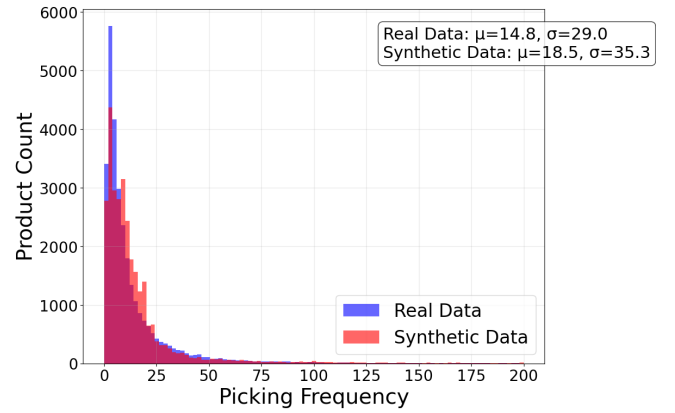


FIGURE 6. Comparison of Picking Frequencies of Real and Synthetic Data

10^{-3}). Fig. 7 plots the per-epoch skip-gram BCE and sales-head MSE for the published configuration ($\alpha = 0.85$, seed = 0): the skip-gram BCE converges within two epochs and the sales-head training MSE saturates by epoch 5, indicating that 10 epochs is comfortably past the convergence point. The validation sales MSE rises slightly across epochs because train and validation weeks are disjoint (see “training-time split” in Section IV-D); we treat the converged training loss together with the intrinsic-evaluation results, not the absolute validation MSE, as the convergence criterion.

1) Dataset Characteristics

The synthetic dataset comprises 30,000 products distributed equally across five distinct behavioral patterns, each containing 6,000 products with 48 weeks of sales history. This configuration simulates a full year of retail operations while ensuring balanced representation of different product categories. The dataset generates 550,431 transaction records, providing sufficient co-occurrence data for skip-gram learning while maintaining realistic transaction density patterns.

Fig. 8 shows the five sales patterns: upward trending (growth from 0.2 to 1.0), downward trending (decline from 1.0 to 0.2), steady (consistent around 0.5), seasonal (cosine wave with mid-year peaks), and volatile (high variance around a stable mean). Each pattern includes Gaussian noise to simulate market fluctuations. The shaded bands show ± 1 standard deviation.

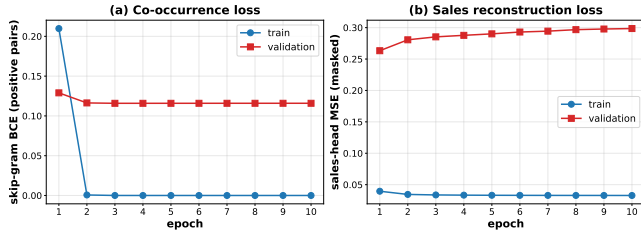
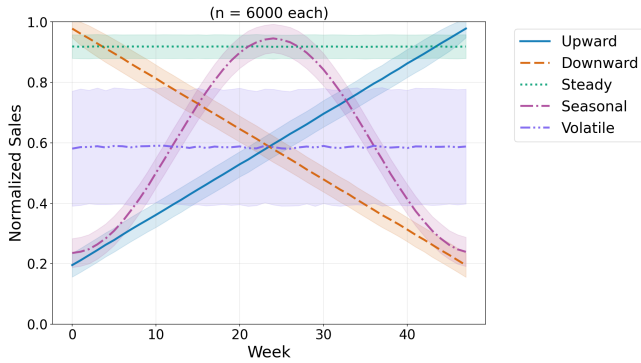
2) Co-occurrence Structure

Definition and measurement:

Co-occurrence analysis measures how often products from

TABLE 3. Cluster Analysis Results of Synthetic Data

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Upward	61 (1.02%)	129 (2.15%)	78 (1.30%)	5608 (93.50%)	124 (2.07%)
Downward	43 (0.72%)	109 (1.82%)	5528 (92.15%)	58 (0.97%)	262 (4.37%)
Steady	190 (3.17%)	5399 (90.03%)	161 (2.68%)	207 (3.45%)	43 (0.72%)
Seasonal	5678 (94.60%)	125 (2.08%)	59 (0.98%)	86 (1.43%)	52 (0.87%)
volatile	142 (2.37%)	213 (3.55%)	188 (3.13%)	197 (3.28%)	5260 (87.67%)

**FIGURE 7.** Per-epoch loss curves for the published configuration ($\alpha = 0.85$, seed = 0). (a) Skip-gram BCE on positive pairs: train collapses by epoch 2; validation BCE plateaus immediately. (b) Sales-head masked-MSE: train converges by epoch 5; the small rise in validation MSE is expected under the disjoint train/validation time-window split.**FIGURE 8.** Synthetic Data Patterns (5 classes).

different clusters appear together in picking sequences. We use short sequence analysis (co-occurrence within 2 consecutive picks).

Empirical calibration:

Transaction sequences are generated based on real warehouse data. Products within the same category have a 16.2% chance of appearing in the same transaction window. This value is the median of observed co-occurrence probabilities across cluster pairs, which ranged from 13.49% to 41.68%. Transaction frequency is proportional to normalized sales values. Products with higher weekly sales generate more transactions.

3) Picking Frequency Distribution

Target distribution characteristics:

The synthetic data targets real warehouse picking frequency distributions with long-tail behavior. The target distribution is derived from 30,000 products: mean 14.8 picks, median 8.0 picks, standard deviation 29.0, and range 2 to 1,888 picks. The distribution shows significant concentration, with 50%

of products receiving 8 or fewer picks.

Synthetic data calibration and validation:

The generated synthetic dataset achieves close approximation to real warehouse characteristics with mean frequency of 18.5 picks (target: 14.8), median of 10.0 picks (target: 8.0), and standard deviation of 35.3 (target: 29.0). The synthetic distribution ranges from 2 to 834 picks per product, capturing the essential long-tail structure while maintaining computational tractability.

Distribution segmentation validation:

The synthetic data replicates realistic segmentation: 70.0% low frequency (2-15 picks), 23.9% medium frequency (16-50 picks), and 6.1% high frequency (51+ picks). This matches warehouse operations where most products are slow-moving while a small fraction drives most activity. Percentiles confirm the long-tail shape: 50th at 10, 90th at 33, and 99th at 191 picks.

Power law modeling:

The distribution uses a segmented approach with shape parameter $\alpha_p = 1.50$ from maximum likelihood estimation. The generation process uses exponential distributions: 60% of products receive low frequencies (mean 6.0), 35% receive medium frequencies (mean 12.0), and 5% receive high frequencies (mean 80.0). This creates the heavy-tail behavior where extreme outliers coexist with many low-activity products.

4) Clustering Results for Synthetic Data

Our embedding model was trained for 2 epochs, achieving a training loss of 0.0022 and validation loss of 0.0019. The model identified 5 clusters, matching the ground truth. The Adjusted Rand Index was 0.802. Purity scores were: upward (0.935), downward (0.921), steady (0.900), seasonal (0.946), and volatile (0.877). Each cluster is dominated by one pattern: Cluster 0 by seasonal (94.60%), Cluster 1 by steady (90.03%), Cluster 2 by downward (92.15%), Cluster 3 by upward (93.50%), and Cluster 4 by volatile (87.67%). Sales prediction MSE varied by pattern: steady (0.0143), volatile (0.0387), upward (0.1092), seasonal (0.1238), and downward (0.1884).

The model learned distinct embeddings for each pattern, with purity scores above 0.87. Seasonal patterns were most predictable, while volatile trends were hardest to forecast.

B. REAL DATA EXPERIMENTS

The real picking data contains 495,517 entries from May 1, 2023 to April 28, 2024, covering 36,729 SKUs. This spans 13

periods of 4 weeks each. We select the top 30,000 products by picking frequency for training. The remaining products have only 1–2 picks over the entire period.

We compare our approach against four baselines:

- **Operational Baseline:** Actual picking tasks recorded from May 2023 to April 2024.
- **ABC Classification Baseline:** Traditional ABC classification that assigns products to tiers (A, B, C) based on picking frequency.
- **Pure Sales Baseline** ($\omega = 0$): The Weighted-Sum framework restricted to the sales-pattern component only (i.e. $E_{\text{final}} = E_{\text{sales}}$).
- **Pure Skip-gram Baseline** ($\omega = 1$): The Weighted-Sum framework restricted to the skip-gram component only (i.e. $E_{\text{final}} = E_{\text{skipgram}}$). The two pure baselines isolate the contribution of each embedding pathway; per-period reductions for both are reported alongside the other methods in Table 5.

Concatenation, also evaluated in Table 5, is the second proposed embedding-fusion variant (Section III) rather than a baseline.

The model uses the same parameters as the synthetic experiments (Table 2). This allows direct comparison between synthetic and real-world results.

1) Seasonality Distribution Analysis

To validate seasonal patterns, we analyzed picking logs for 30,000 products. We calculated the Autocorrelation Function (ACF) using weekly sales volume. A significant seasonal signal is defined as $ACF > 0.5$.

Table 4 shows the distribution of picking frequencies by ACF range. Over 88% of picking volume comes from products with low correlation ($ACF < 0.3$). Products above the 0.5 threshold show consistent demand recurrence ($ACF \approx 0.51$ – 0.64).

Seasonal products represent only 1.9% of inventory but have higher picking intensity: 25.99 picks per item compared to 15.81 for non-seasonal products. Given this difference, we next investigate how to combine these signals.

2) Fusion Strategy Evaluation

We evaluate the two fusion strategies defined in Section III-C2: Concatenation, which keeps both embeddings at full dimensionality, and Weighted Sum, whose weight ω we sweep over $[0, 1]$ to analyze its impact on clustering and storage assignment. Fig. 9 shows the clustering results under the Concatenation strategy.

3) Cluster Identification and Behavioral Patterns

Clustering analysis using concatenated embeddings reveals five product behaviors from the real-world data. Fig. 9 shows picked quantities that are min-max normalized per product, then averaged within each cluster. This ensures each product contributes equally regardless of sales volume.

Cluster 0 (5,019 SKUs): “Growth”:

Sales start near 0.025 and rise to 0.25 by the end of the period. Growth accelerates from January 2024.

Cluster 1 (7,387 SKUs): “Steady High-Volume”:

Sales remain stable between 0.10 and 0.16 throughout the period with minor variations.

Cluster 2 (6,903 SKUs): “Declining”:

Sales decline from 0.15 in May 2023 to 0.02 by early 2024, with a slight uptick at the end.

Cluster 3 (4,623 SKUs): “Seasonal Peak”:

Sales peak in July 2023 (0.19) and October 2023 (0.165), with low activity during winter and spring.

Cluster 4 (6,068 SKUs): “Late-Season Recovery”:

Sales are low to moderate for most of the period, with increases in December 2023–January 2024 (0.15) and a spike at the end (0.18).

4) Warehouse Simulation Implementation

The simulation runs from May 1, 2023 to April 28, 2024, divided into 13 periods.

Period-Based Product Selection:

Each period focuses on the top 10,000 most picked products. Since 11,706 to 14,038 SKUs are picked per period, this keeps cluster assignments relevant while managing complexity.

Top Cap Cluster Assign:

Products outside the top 10,000 lose their cluster assignments and become unclustered. Assignments update at each new period.

Initial Storage State:

Starting inventory is determined from real data by tracking storage additions and picking subtractions. Products with net negative quantities start with their minimum recorded amount; others start at zero. All products are initially placed in optimal locations using radial sector assignment. Locations are not reserved for products with zero initial quantity.

Daily Operations:

Each day processes storage first, then picking. Storage operations update inventory and locations. Picking operations process all daily tasks while tracking capacity.

Clustered Product Storage:

Clustered products go to locations assigned to their cluster. The system tries existing locations first, then high-accessibility locations. When cluster locations are full, it finds nearby unclustered locations using product similarity (threshold 0.5).

Unclustered Product Storage:

Unclustered products go to locations marked as “unclustered”. The system tries existing same-product locations first, then uses similarity calculations, then accessibility scores.

Capacity Management:

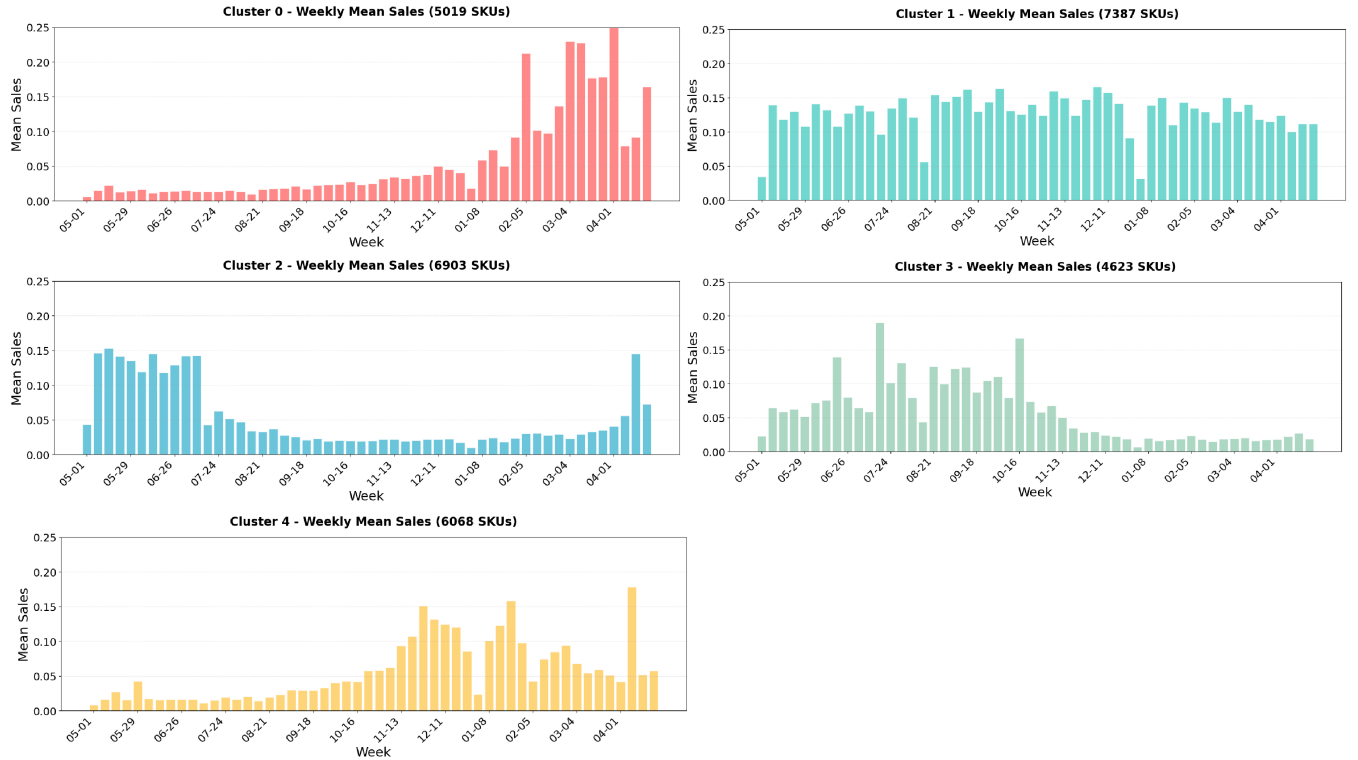
The warehouse has 1,462 shelves. Initial setup uses 26 SKUs per shelf; operations allow 30 SKUs per shelf. This 4-SKU buffer handles storage surges.

Backup Plans:

When storage fails, the system tries: same-product locations in other clusters, nearby substitute locations, then

TABLE 4. Comparison of Picking Frequency by Autocorrelation (ACF) Period

ACF Range	4-Week Pattern			13-Week Pattern			26-Week Pattern		
	Prods	Picks	Share (%)	Prods	Picks	Share (%)	Prods	Picks	Share (%)
< 0	19,454	226,969	47.30%	23,231	314,082	65.45%	25,783	367,030	76.48%
0–0.3	8,655	199,155	41.50%	6,024	152,254	31.73%	3,364	95,629	19.93%
0.3–0.5	1,612	43,775	9.12%	661	12,237	2.55%	648	13,848	2.89%
0.5–0.7	270	9,806	2.04%	80	1,284	0.27%	164	2,856	0.60%
0.7+	9	189	0.04%	4	37	0.01%	41	531	0.11%

**FIGURE 9.** Min-max normalized picked quantities averaged across all products within the clusters from May 2023 to April 2024, derived from the Concatenation embedding strategy.

accessibility-based placement.

5) Order Batching for Picking Tasks

We used 30-minute time windows to match real batching patterns. Validation across 13 periods showed a 1.39% difference in total walking distance compared to real batches, with period-wise variations from -3.37% to +1.96%. This confirms that the batching algorithm replicates real operations without systematic bias.

C. RESULTS

This section reports the walking-distance reductions achieved by each storage policy on the real warehouse data described in §IV-A, evaluated across the 13 four-week periods that together cover a full year of operations (May 2023 – April 2024, 495,517 picking transactions over 36,729 SKUs). All reductions are reported relative to the Operational Baseline (the actual historical picking distance over the same 13

periods), so a 37% reduction means the proposed layout would have completed the same year of orders with 37% less walking. Section IV-C3 below presents the headline performance comparison across the five methods of Table 5; the remainder of §IV then probes the robustness of that comparison with the following follow-up analyses:

- **Statistical Significance of the Hybrid vs. ABC Margin** (§IV-E) applies paired-*t*, Wilcoxon signed-rank, and sign tests to the 13 per-period reductions in order to determine whether the small mean margin between the static Weighted Sum layout and dynamic ABC reflects a real difference or is consistent with no real difference.
- **Sensitivity to the Relocation-Cost Multiplier** (§IV-G) sweeps a multiplier $\lambda \in \{0, 1, 2, 5\}$ on the per-event relocation cost in order to test whether the relative ranking of the methods is preserved when the true labour cost of physical relocation exceeds the per-relocation walking distance our simulation charges.

TABLE 5. Comparison of Walking Distance Reduction Rates (%) Relative to Operational Baseline (Oper. Base.), measured under the aisle-network shortest-path distance defined in Section III. NoM = No-Merge (static layout); M+C = Merge+Cost (dynamic relocation with cost charged). Five methods are compared: the three storage policies (ABC, Concatenation, Weighted Sum $\omega = 0.9$) and the two ablation endpoints of the Weighted-Sum framework (Pure Sales $\omega = 0$ and Pure Skip-gram $\omega = 1$).

Period	ABC		Concat.		Pure Sales $\omega=0$		Pure SG $\omega=1$		WS $\omega=0.9$		Oper. (km)
	NoM	M+C	NoM	M+C	NoM	M+C	NoM	M+C	NoM	M+C	
1	52.4	52.4	41.3	41.3	40.4	40.4	43.9	43.9	44.5	44.5	725.0
2	43.8	41.8	34.2	32.0	34.6	32.6	38.5	36.2	39.2	36.8	857.9
3	40.2	40.5	32.5	32.7	32.6	31.4	38.5	38.2	39.0	38.4	853.3
4	33.5	37.3	27.5	27.4	27.3	27.7	34.9	35.7	36.5	36.5	664.9
5	32.4	37.6	27.8	29.7	28.3	30.1	35.7	36.2	37.4	37.7	902.6
6	30.8	36.5	28.8	29.7	29.2	31.0	36.0	36.3	37.9	37.9	842.0
7	27.3	34.8	27.3	29.2	27.5	29.8	35.1	35.7	35.9	36.9	818.1
8	23.1	32.2	27.7	30.1	26.0	28.6	33.8	34.8	34.2	35.5	841.8
9	22.6	32.1	29.4	31.3	26.1	28.7	34.7	35.2	34.5	35.7	627.9
10	25.9	35.8	32.8	35.1	29.6	31.7	36.5	37.7	37.2	38.5	1037.0
11	19.3	32.4	31.5	33.8	28.2	30.3	35.4	36.4	35.4	36.9	859.2
12	15.8	30.5	30.9	32.9	29.1	31.3	33.8	34.9	34.0	35.1	869.0
13	13.2	30.6	28.7	29.8	28.9	31.3	33.3	34.4	33.6	34.2	891.0
Mean	29.3	36.5	30.8	31.9	29.8	31.2	36.2	36.6	36.9	37.3	830.0

- **Hyperparameter Sensitivity Analysis** (§IV-D) verifies that the four load-bearing hyperparameters of the framework – the skip-gram loss weight α , the cluster count k , the skip-gram window size W , and the embedding dimension d_{hid} – are each empirically justified by per-seed Friedman tests rather than chosen on a hunch.
- **Comparison Against Closed-Form Distance Approximations** (§IV-F) recomputes the headline comparison under Euclidean, rectilinear/Manhattan, and an aisle-aware S-shape metric to confirm that the comparative ranking does not depend on the choice of distance metric.

Fig. 10 compares the three storage policies against the Operational baseline period-by-period under the two Merge configurations.

1) ABC Classification Baseline

This baseline uses ABC classification to assign products to one of three frequency tiers – Class A (most frequent), Class B (mid-frequency), Class C (low frequency) – or to an Unclustered overflow bucket for the long tail (Section III-D6).

When a tier is full, products overflow to lower tiers: A overflows to B, then C, then Unclustered; B overflows to C, then Unclustered; C overflows to Unclustered only. The Unclustered bucket has no corresponding physical-zone identifier; it is the residual capacity left after the three tiers have been placed. For fair comparison, this baseline also uses the **Merge strategy** for dynamic relocation, with constraints for the tiered structure:

- **Selective Merge:** Inventory is consolidated *only* when a product transitions between an ABC class and unclustered (e.g., Unclustered $\rightarrow$ Class A, or Class C $\rightarrow$ Unclustered).

- **No Merge Between Classes:** Relocation is suppressed for transitions between classes. For example, if a product moves from A to B, existing inventory stays in the A-zone while new stock goes to the B-zone.

2) Merge Cost Calculation

To assess the cost of dynamic relocation, we calculate merge cost for each relocation event. The merge cost is the aisle-network shortest-path distance (Section III) between the old and new storage locations. Merge costs are included in the overall evaluation to determine the net effectiveness of dynamic strategies. In practice, the labor of physical relocation depends on additional factors beyond walking distance, including inventory volume, transport effort, and shelving time. To assess the robustness of our conclusions to this gap, Section IV-G reports a sensitivity analysis that scales the merge cost by a multiplier $\lambda \in \{0, 1, 2, 5\}$.

3) Performance Comparison: No Merge and Merge Approaches

Table 5 compares five methods — ABC Classification, the two proposed embedding-fusion variants (Concatenation and Weighted Sum $\omega = 0.9$), and the two pure-pathway ablation baselines (Pure Sales $\omega = 0$, Pure Skip-gram $\omega = 1$) — under two configurations: No Merge (static assignment) and Merge+Cost (dynamic relocation with labor penalties).

a: No Merge Performance Analysis:

In the No Merge scenario, the Weighted Sum ($\omega = 0.9$) method achieves the highest stability with a mean reduction of **36.9%**, followed by Concatenation (**30.8%**) and ABC (**29.3%**). The ABC method degrades over time in this configuration: it starts at 52.4% in Period 1 but drops to 13.2% by Period 13, because the static period-1 ABC layout becomes

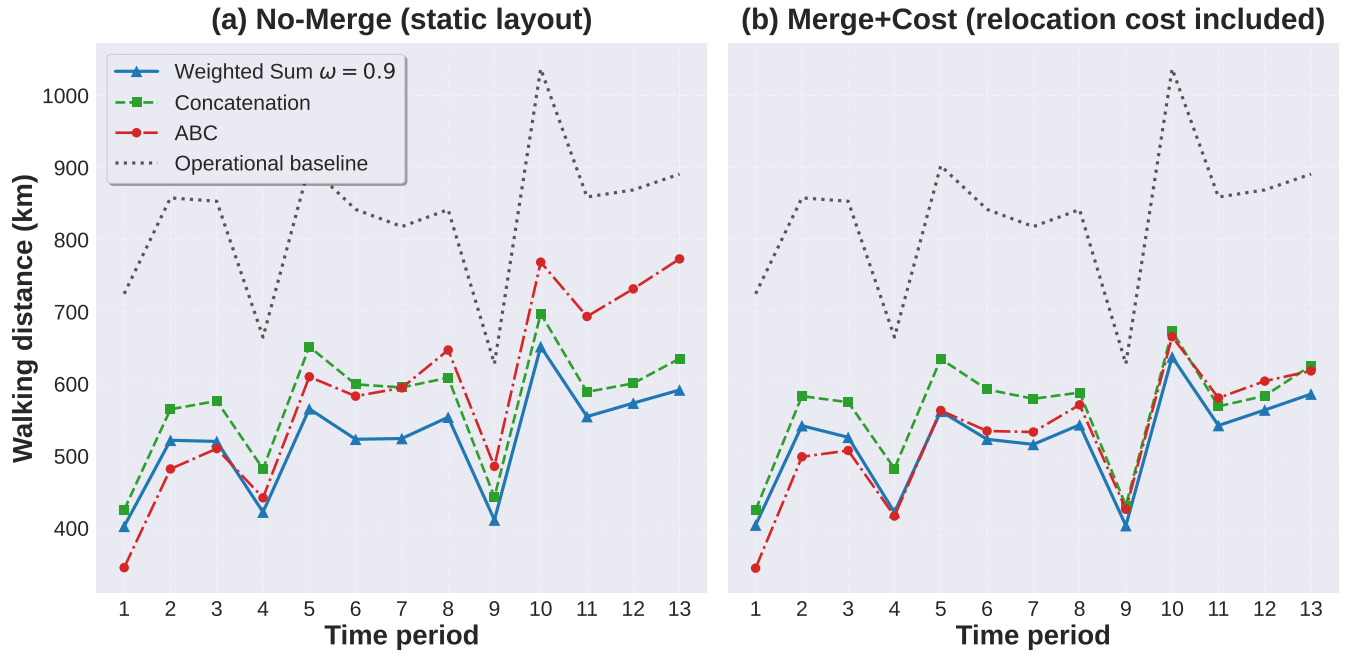


FIGURE 10. Period-wise walking distances (km) under the aisle-network shortest-path metric (Section III), shown for the two storage-policy configurations: (a) No-Merge (static layout) and (b) Merge+Cost (dynamic relocation with cost charged). Each panel plots Weighted Sum $\omega = 0.9$ (blue, triangle), Concatenation (green, square), ABC (red, circle), and the Operational baseline (grey). ABC degrades sharply in (a) and recovers under relocation; WS stays stable across both panels.

increasingly mismatched with later periods' demand. The Weighted Sum layout, in contrast, maintains stable performance across all 13 periods, and Concatenation also remains broadly stable but at the lower mean.

b: Net Efficiency Analysis (Merge+Cost):

With relocation costs included (Merge+Cost), the Weighted Sum ($\omega = 0.9$) method achieves the highest net efficiency of **37.3%**, followed by ABC (**36.5%**) and Concatenation (**31.9%**). Fig. 11 plots the mean reduction as ω is swept from 0 (pure sales) to 1 (pure skip-gram), together with the ABC and Concatenation reference lines; the peak at $\omega = 0.9$ under Merge+Cost is the source of the 37.3% figure above, and the No-Merge curve peaks at the same ω with a mean reduction of 36.9%. This shows that Weighted Sum identifies better locations than Concatenation, requiring less relocation to achieve peak efficiency.

c: Operational Implications and Stability:

A key finding is that the **Weighted Sum with No Merge (36.9%)** achieves walking-distance reduction on par with **ABC with Merge+Cost (36.5%)**, while requiring zero inventory relocation. The 0.4 percentage-point (pp) mean gap is not statistically significant on a paired test across the 13 periods (paired t : $\bar{d} = +0.34$ pp, $p = 0.70$, 95% CI $[-1.55, +2.23]$; Wilcoxon $p = 0.34$; full analysis in Section IV-E). The practical advantage of the Weighted Sum framework is therefore not a small mean-reduction edge, but the elimination of dynamic relocation overhead while delivering an equivalent reduction. The ABC system requires

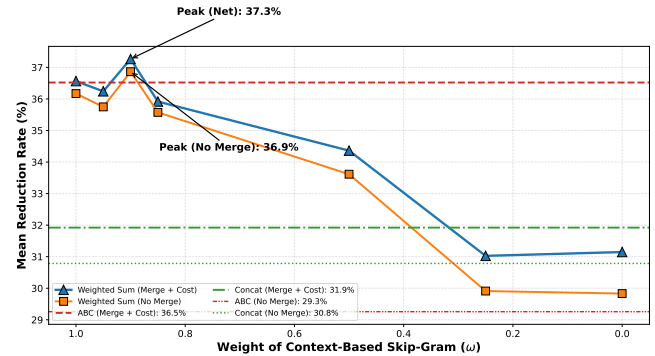


FIGURE 11. Comparison of Net Reduction Rates (Merge+Cost). The Weighted Sum strategy ($\omega = 0.9$) achieves the highest net efficiency of 37.3% under the aisle-network shortest-path metric.

constant merging to prevent degradation in the No-Merge configuration, and (as Section IV-G shows) becomes strictly inferior to the static Weighted-Sum layout once relocation cost rises above the paper's nominal $1 \times$ multiplier.

D. HYPERPARAMETER SENSITIVITY ANALYSIS

We analyze the sensitivity of the four load-bearing hyperparameters of the framework – the skip-gram loss weight α , the cluster count k , the skip-gram window size W , and the embedding dimension d_{hid} – with one-at-a-time sweeps holding the others at their published values. Each cell is trained with three random seeds, evaluated with the same intrinsic-eval battery on the trained embedding (Spear-

man seasonality¹, ridge probe R^2 , PMI-based association³, anisotropy⁴, norm–frequency Pearson⁵), and aggregated with the Friedman test across seeds.

a: $\alpha = 0.85$ (skip-gram loss weight).

$\alpha = 0.85$ is the largest skip-gram weight that improves co-occurrence quality without compromising the sales pathway. PMI-based association ρ on the target embedding rises monotonically in α and peaks at $\alpha = 0.85$ (0.241 $\rightarrow$ 0.269; Friedman $p = 0.010$, 100% per-seed agreement), while sales-seasonality alignment is invariant across $\alpha \in [0.10, 0.90]$ (Friedman $p = 0.94$). The choice therefore maximizes skip-gram quality at zero measurable cost on the sales head.

b: $k = 5$ (number of product clusters).

$k = 5$ is the largest value of k that simultaneously preserves operationally usable cluster sizes (smallest cluster: 4,630 products; at $k = 6$ this collapses to 1,693 – a $2.7\times$ reduction), achieves excellent stability (seed-ARI = 0.997, bootstrap-ARI = 0.92, both monotonically declining for $k \geq 7$)⁶, and delivers most of the Davies–Bouldin⁷ geometric improvement ($\Delta/k \approx 0.18$ for $k = 2 \rightarrow 5$, only $\Delta/k \approx 0.09$ for $k = 5 \rightarrow 10$).

c: $W = 2$ (skip-gram window size).

Across $W \in \{1, 2, 3, 4, 5\}$, the semantic-content intrinsic metrics (seasonality ρ , probe R^2 , association ρ) are flat: Friedman tests yield $p > 0.10$ for all three. Only the geometric-health metrics (anisotropy, norm–frequency $|r|$) show a significant monotonic improvement with W ($p = 0.017$, 100% per-seed agreement). W is therefore a low-leverage hyperparameter; $W = 2$ is indistinguishable from $W \in \{1, 3, 4, 5\}$ on content.

d: $d_{hid} = 64$ (embedding dimension).

Sweeping $d_{hid} \in \{32, 64, 128, 256\}$ reveals a statistically significant content-vs-geometry trade-off: probe R^2 col-

lapses from +0.130 at $d_{hid} = 32$ to -0.418 at $d_{hid} = 256$ (Friedman $p = 0.029$, 100% per-seed agreement), while anisotropy and norm–freq monotonically improve. $d_{hid} = 64$ is the smallest dimension that simultaneously preserves probe $R^2 > 0$ and reduces norm–frequency Pearson below +0.6; larger d_{hid} would yield a frequency-collapsed embedding that has lost the sales signal the contiguous-zone storage policy depends on.

E. STATISTICAL SIGNIFICANCE OF THE HYBRID VS. ABC MARGIN

To assess whether the headline 0.4 percentage-point margin between Weighted Sum No-Merge (36.9%) and ABC Merge+Cost (36.5%) is statistically supported, we apply paired tests to the 13 per-period reduction pairs of Table 5, with walking distance measured under the aisle-network shortest-path metric (Section III).

a: Headline comparison (Weighted Sum NoM vs. ABC Merge+Cost).

The 13 within-period differences are $\bar{d} = +0.34$ pp (Weighted Sum better), SD = 3.13 pp. Five periods favor ABC by up to 5.2 pp; eight periods favor Weighted Sum by up to 3.5 pp, so the across-period mean is small precisely because the signal is non-stationary. All three standard tests fail to reject the null of equal means: paired $t p = 0.70$, Wilcoxon signed-rank $p = 0.34$, sign test $p = 0.58$. The 95% paired- t confidence interval on the mean difference is $[-1.55, +2.23]$ percentage points. Cohen’s paired $d = 0.11$ (very small effect). The mean margin is therefore consistent with no real difference.

b: Validity check (Concatenation NoM vs. ABC Merge+Cost).

To confirm that the paired test has adequate power, we run the same procedure on Concatenation No-Merge (mean 30.8%) vs. ABC Merge+Cost (36.5%), where the manuscript reports Concatenation as the weaker of the two proposed embedding-fusion variants. Paired $t p = 0.0001$, Wilcoxon $p = 0.0005$, sign test $p = 0.003$, $\bar{d} = -5.73$ pp, CI $[-8.02, -3.45]$, Cohen’s $d = -1.52$. The test correctly rejects on this comparison, so the non-rejection on the headline is genuine evidence of no significant difference – not a false negative.

c: Apples-to-apples comparison (Weighted Sum Merge+Cost vs. ABC Merge+Cost).

If we align both policies on the Merge+Cost variant, the means are 37.3% vs. 36.5% (+0.7 pp), with paired $t p = 0.50$ and CI $[-1.61, +3.10]$ – still not significant. The 0.4 pp and 0.7 pp margins reported in Table 5 are best interpreted as “Weighted Sum and ABC Merge+Cost are comparable in mean walking-distance reduction”, not as a definitive ranking.

¹Spearman rank correlation between pairwise cosine similarity in the embedding and pairwise similarity of products’ 52-week sales vectors; measures whether embedding-similar products also follow similar seasonal sales patterns.

²Held-out R^2 of a ridge-regularised linear regression of each product’s 52-week sales vector on its learned embedding; measures how much of the sales signal is linearly recoverable from the embedding.

³Pointwise Mutual Information, $\text{PMI}(a, b) = \log \frac{P(a, b)}{P(a)P(b)}$, computed empirically from the pick log; the metric is the Spearman correlation between pairwise cosine similarity in the embedding and pairwise PMI.

⁴Mean cosine similarity of each embedding row to the global embedding mean; values near 1 indicate a collapsed cone that degrades cosine-based retrieval.

⁵Pearson correlation between each embedding’s L2 norm and the product’s log pick frequency; large $|r|$ means cosine similarity is partly tracking popularity rather than the intended affinity or seasonality structure.

⁶Both are the mean pairwise Adjusted Rand Index over 20 KMeans runs: seed-ARI varies only the random seed (algorithm-stability), while bootstrap-ARI varies the input via 80% product subsamples (data-perturbation stability).

⁷The Davies–Bouldin index averages each cluster’s intra-cluster spread relative to its distance from the nearest neighbouring cluster; lower is better (more compact, better-separated clusters).

d: Pure baselines.

The Weighted Sum framework outperforms the pure sales baseline ($\omega = 0$, mean 29.8% on NoM): $\bar{d} = +7.03$ pp, $p < 0.0001$, CI [+5.90, +8.16], 13/0 sign agreement, Cohen's $d = +3.76$. Against the pure skip-gram baseline ($\omega = 1$), the Weighted Sum is also better – under the rigorous graph metric the small refinement contributed by the sales pathway is statistically detectable: on No-Merge, $\bar{d} = +0.69$ pp (paired $t p = 0.002$, Wilcoxon $p = 0.001$, Cohen's $d = +1.09$, 12/13 sign agreement); on Merge+Cost, $\bar{d} = +0.70$ pp ($p = 0.0004$, Cohen's $d = +1.36$, 12/13 sign agreement). The skip-gram pathway still carries the bulk of the framework's gain, but the sales-pattern weighting contributes a small consistent refinement.

F. COMPARISON AGAINST CLOSED-FORM DISTANCE APPROXIMATIONS

Although the headline metric of the manuscript is the rigorous aisle-network shortest-path distance defined in Section III, prior SLAP work routinely uses simpler closed-form distance functions (Euclidean, rectilinear / Manhattan, or an S-shape model that treats within-aisle motion as $|\Delta y|$ and between-aisle motion via the nearer end-cross-aisle). To confirm that our comparative conclusions do not depend on the choice of metric, we recompute total walking distance for the two headline policies under all four metrics, holding the per-batch picking sequences fixed. For each policy, Operational uses the position-corrected historical pick log with the simulator's zone-based batching; ABC Merge+Cost uses the simulator's picking trace plus the matching storage log's relocation walks (the “+Cost” contribution, evaluated under each metric); Weighted Sum no-merge has no relocation events.

The per-policy reduction vs. Operational under each metric is summarised below:

Policy	Graph	Eucl.	Manh.	S-shape
ABC Merge+Cost	36.5%	38.7%	37.6%	34.3%
WSum $\omega=0.9$ NoM	36.9%	39.2%	38.0%	35.1%

All four metrics agree qualitatively: WSum $\omega = 0.9$ NoM matches or slightly exceeds ABC Merge+Cost while requiring no inventory relocation. The Euclidean approximation is a mild upper bound on the achievable reduction (~ 2 pp above the rigorous graph value), the S-shape model is a lower bound, and Manhattan sits between them; the WSum-over-ABC margin ranges from +0.4 pp (graph) to +0.8 pp (S-shape) – consistently small and never negative.

G. SENSITIVITY TO THE RELOCATION-COST MULTIPLIER

The merge cost we charge in Section IV-C2 is the aisle-network shortest-path travel between the old and new storage locations of each relocated item. This walking-distance figure underestimates true relocation labour (which also depends on inventory volume, transport effort, and shelving time). We therefore sweep a relocation-cost multiplier $\lambda \in \{0, 1, 2, 5\}$.

The relationship is exact:

$$\text{Merge} + \text{Cost}(\lambda) = (1 - \lambda) \cdot \text{Merge} + \lambda \cdot \text{Merge} + \text{Cost},$$

Policy (M+C variant)	$\lambda=0$	$\lambda=1$	$\lambda=2$	$\lambda=5$
ABC	39.7	36.5	33.3	23.5
Concat.	35.2	31.9	28.7	19.1
WSum $\omega=0.9$	40.0	37.3	34.5	26.4
WSum NoM (static, any λ)	36.9			

Paired tests vs. the static Weighted Sum No-Merge reference (36.9%, λ -independent because no relocation is performed) yield:

- $\lambda = 1$ (paper's nominal): $\bar{d} = +0.3$ pp vs. ABC, paired $t p = 0.70$ (not significant, as Section IV-E discusses).
- $\lambda = 2$: $\bar{d} = +3.6$ pp vs. ABC, paired $t p = 0.012$, 12 of 13 periods favour the static Weighted Sum (significant).
- $\lambda = 5$: $\bar{d} = +13.3$ pp vs. ABC, paired $t p = 0.0001$, CI [+8.3, +18.4], 12 of 13 periods favour the static Weighted Sum (highly significant).

At $\lambda = 5$, all three relocation-based variants (ABC M+C 23.5%, Concatenation M+C 19.1%, Weighted Sum M+C 26.4%) fall below their own No-Merge counterparts; the operationally sound choice at high relocation cost is to forgo relocation entirely, which makes the static Weighted Sum the strict winner since its No-Merge mean (36.9%) is the highest of any No-Merge configuration.

V. CONCLUSION

This research addressed the challenge of optimizing warehouse storage location assignment to reduce picking costs, which can constitute up to 65% of total warehouse operating expenses. We introduced two embedding-based strategies, **Concatenation** and **Weighted Sum**, that combine co-occurrence information (skip-gram) with temporal sales patterns. The Weighted Sum approach uses a tunable parameter ω to balance relational structure and sales velocity.

Our evaluation shows that the **Weighted Sum method** ($\omega = 0.9$) yields strong and stable performance. In the No Merge (static) configuration, it achieves a mean walking distance reduction of **36.9%**, maintaining stability across all 13 periods, whereas the ABC baseline without relocation degrades from 52.4% in period 1 to 13.2% by period 13. With relocation costs included (Merge+Cost), the Weighted Sum method achieves **37.3%**, ahead of Concatenation (31.9%) and statistically on par with ABC Merge+Cost (36.5%) – the 0.4–0.7 percentage-point mean margin is within the 95% paired- t CI of zero (Section IV-E). The substantive advantage of the proposed framework is therefore not a marginal mean-reduction edge, but the elimination of dynamic relocation overhead while delivering an equivalent reduction.

A key empirical insight of this study is that the effectiveness of each embedding source depends on the structure of the demand data, and the fusion weight ω makes this trade-off explicit and tunable. Our autocorrelation analysis showed that most products have negligible or negative seasonality across 4, 13, and 26-week horizons. Traditional sales-dependent methods like ABC analysis were heavily

influenced by non-seasonal noise. The noise leads to unstable placements and the sharp performance drop shown in the experiment. In contrast, the proposed framework can be tuned to match this demand structure. The success of $\omega = 0.9$ suggests that without strong seasonal signals, co-occurrence relationships from skip-gram provide a more stable foundation for storage assignment than sales history alone.

From a practical perspective, a key finding is that the **Weighted Sum without relocation (36.9%)** delivers walking-distance reduction comparable to **ABC with full Merge+Cost relocation (36.5%)** while requiring no inventory movement. Warehouse operators can therefore obtain a near-equivalent reduction without the labour costs of dynamic relocation, using existing warehouse management systems without infrastructure changes. We note that this operational advantage is demonstrated under the modelling assumptions enumerated below (perfect inventory visibility, deterministic uncongested travel, single-slot SKUs); transferring the reported gains to other warehouse environments requires these conditions to hold at least approximately, and should be validated per site.

While our simulation defines relocation costs by walking distance, this underestimates the true operational burden: physical relocation also requires moving an item's entire accumulated stock, whereas picking only handles the small quantities needed per order. We quantify the impact of this under-counting in Section IV-G by sweeping a relocation-cost multiplier $\lambda \in \{1, 2, 5\}$ on the Merge+Cost metric. At $\lambda = 2$ the static Weighted Sum significantly outperforms ABC Merge+Cost by +3.6 percentage points (paired t $p = 0.012$); at $\lambda = 5$ the gap reaches +13.3 percentage points ($p = 0.0001$).

This study has several limitations that bound the generality of the reported reductions. (i) **Single-slot SKU assumption**: the simulation treats each product as occupying one slot per shelf regardless of physical inventory volume; high-volume SKUs that occupy multiple slots in practice are therefore over-compacted in our model. (ii) **Perfect inventory visibility**: we assume real-time, error-free knowledge of every SKU's location and quantity; in practice, scanner errors and stock discrepancies would degrade the fidelity of any policy. (iii) **Deterministic travel times with no congestion**: walking distance is treated as proportional to per-pair aisle-network shortest-path distance, with no aisle-blocking, no inter-picker interference, and no fatigue effects. (iv) **Graph-distance approximation of the aisle skeleton**: walking distance is measured as the shortest path on a hand-crafted aisle-network graph (Section III) that captures rectangular blocks, the rounded boundaries of the 3C and 3D blocks at our deployment site, and the shared cross-aisle walkways between vertically stacked blocks. Local irregularities not encoded in this graph – end-cap shelving, in-aisle obstacles, temporary blockages – are not modelled. Section IV-F additionally reports the same reductions under closed-form Euclidean, Manhattan, and relaxed S-shape metrics; all four give qualitatively consistent rankings and differ by only a few

percentage points. (v) **Cold-start coverage**: products with no prior picking history are placed by accessibility alone via the Tail-Product mechanism (Section III-D6), so the affinity benefit of the framework only accrues after a SKU has picked enough times to be included in the next re-training boundary. (vi) **Computational footprint**: training the dual embedding model requires substantially more computation than ABC classification or other heuristic methods, which may be a deployment barrier for smaller operators. (vii) **Baseline coverage**: our experiments do not include direct comparisons with the network-embedding SLAP method of Qiu et al. [11] or the DRL-based dynamic storage assignment of Waubert de Puiseau et al. [13]; the reported results should therefore not be read as establishing superiority over the broader state of the art, and such benchmarking remains future work.

a: Real-world implementation considerations.

Deploying the proposed framework on a live warehouse-management system (WMS) raises three practical concerns. **System integration**: the framework consumes the same per-pick event records that any WMS already logs for compliance (timestamp, product ID, location, batch), so integration requires only an offline embedding-training pipeline plus a per-period assignment writer; no real-time inference path is needed. **Scalability**: at our 30,000-SKU vocabulary the trained model is ~ 5.77 M parameters, fits comfortably on a single commodity GPU, and re-trains in tens of minutes per period; the dominant cost is the offline simulation used to validate a candidate layout, which is parallelisable across periods. **Continuous SKU churn**: as Section III discusses, new SKUs are routed to the Unclustered zone until they accumulate enough picking history to be promoted at the next re-training boundary.

b: Position relative to prior baselines.

Three observations summarize the experimental section's findings in compact form. First, the static Weighted-Sum layout matches the strongest dynamic-relocation baseline (ABC Merge+Cost) in mean walking-distance reduction without performing any relocation, and this match is robust to a sensible range of relocation-cost multipliers (Section IV-G) and to two more realistic distance metrics (Section IV-F). Second, the skip-gram pathway alone accounts for nearly all of the framework's reduction relative to pure sales-pattern baselines (Section IV-E), with the sales-pattern component contributing a small but consistent refinement; this isolates co-occurrence as the load-bearing signal for SLAP on this dataset. Third, the four load-bearing hyperparameters (α , k , W , d_{hid}) are each empirically justified by per-seed Friedman tests rather than chosen on a hunch (Section IV-D).

Future work will focus on integrating real-time inventory volume data to address multi-slot storage constraints and incorporating predictive analytics for proactive storage assignment. We will also explore adaptive tuning of ω in response to demand volatility, joint (α, ω) optimisation, and adaptive

loss-balancing schemes (GradNorm, Kendall uncertainty) for the combined training objective.

REFERENCES

- [1] G. Büyüközkan and F. Göçer, "Digital supply chain: Literature review and a proposed framework for future research," *Computers in Industry*, vol. 97, pp. 157–177, 2018.
- [2] M. Ghobakhloo, M. Iranmanesh, B. Foroughi, M.-L. Tseng, D. Nikbin, and A. A. Khanfar, "Industry 4.0 digital transformation and opportunities for supply chain resilience: a comprehensive review and a strategic roadmap," *Production planning & control*, vol. 36, no. 1, pp. 61–91, 2025.
- [3] A. A. Tubis and J. Rohman, "Intelligent warehouse in industry 4.0—systematic literature review," *Sensors*, vol. 23, no. 8, p. 4105, 2023.
- [4] Y. Asai, Y. Mori, K. Higashiura, K. Yokoyama, S. Katayama, K. Urano, T. Yonezawa, and N. Kawaguchi, "Towards a real-time and energy-efficient edge ai camera architecture in mega warehouse environment," in *2024 IEEE 3rd Real-Time and Intelligent Edge Computing Workshop (RAGE)*. IEEE, 2024, pp. 1–6.
- [5] R. Maharjan and H. Kato, "Logistics and supply chain resilience of japanese companies: Perspectives from impacts of the covid-19 pandemic," *Logistics*, vol. 7, no. 2, p. 27, 2023.
- [6] R. De Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *European journal of operational research*, vol. 182, no. 2, pp. 481–501, 2007.
- [7] Y.-C. Ho, T.-S. Su, and Z.-B. Shi, "Order-batching methods for an order-picking warehouse with two cross aisles," *Computers & Industrial Engineering*, vol. 55, no. 2, pp. 321–347, 2008.
- [8] J. Reyes, E. Solano-Charris, and J. Montoya-Torres, "The storage location assignment problem: A literature review," *International Journal of Industrial Engineering Computations*, vol. 10, no. 2, pp. 199–224, 2019.
- [9] S. Islam and K. Uddin, "Correlated storage assignment approach in warehouses: A systematic literature review," *Journal of Industrial Engineering and Management*, vol. 16, no. 2, pp. 294–318, 2023.
- [10] B. Pierre, B. Vannieuwenhuysse, D. Dominanta, and H. V. Dessel, "Dynamic abc storage policy in erratic demand environments," *Jurnal Teknik Industri: Jurnal Keilmuan dan Aplikasi Teknik Industri*, vol. 5, no. 1, p. 1–12, Jul. 2004.
- [11] J. Qiu, Y. Si, Y. Chai, Y. Liu, D. Zhang, H. Han, and L. Wang, "Network-embedding based storage location assignment in mobile rack warehouse," in *Artificial Intelligence and Security: 5th International Conference, ICAIS 2019, New York, NY, USA, July 26–28, 2019, Proceedings, Part I*. Springer, 2019, pp. 630–639.
- [12] J. Li, M. Moghaddam, and S. Y. Nof, "Dynamic storage assignment with product affinity and abc classification—a case study," *The International Journal of Advanced Manufacturing Technology*, vol. 84, pp. 2179–2194, 2016.
- [13] C. Waubert de Puiseau, D. T. Nanfack, H. Tercan, J. Löbber-Plattfaut, and T. Meisen, "Dynamic storage location assignment in warehouses using deep reinforcement learning," *Technologies*, vol. 10, no. 6, p. 129, 2022.
- [14] J. C. Gutierrez, S. I. Polo Triana, and J. S. León Becerra, "Benefits, challenges, and limitations of inventory control using machine learning algorithms: literature review," *OPSEARCH*, pp. 1–33, 2024.
- [15] E. Frazee and G. P. Sharp, "Correlated assignment strategy can improve any order-picking operation," *Industrial Engineering*, vol. 21, no. 4, pp. 33–37, 1989.
- [16] J. Kim, F. Méndez, and J. Jimenez, "Storage location assignment heuristics based on slot selection and frequent itemset grouping for large distribution centers," *IEEE Access*, vol. 8, pp. 189 025–189 035, 2020.
- [17] Y. Li, F. A. Méndez-Mediavilla, C. Temponi, J. Kim, and J. A. Jimenez, "A heuristic storage location assignment based on frequent itemset classes to improve order picking operations," *Applied Sciences*, vol. 11, no. 4, p. 1839, 2021.
- [18] D. Winkelmann, F. Tolkmitt, M. Ulrich, and M. Römer, "Integrated storage assignment for an e-grocery fulfilment centre: Accounting for day-of-week demand patterns," *Flexible Services and Manufacturing Journal*, pp. 1–41, 2024.
- [19] Y.-F. Chuang, H.-T. Lee, and Y.-C. Lai, "Item-associated cluster assignment model on storage allocation problems," *Computers & industrial engineering*, vol. 63, no. 4, pp. 1171–1177, 2012.
- [20] K.-W. Pang and H.-L. Chan, "Data mining-based algorithm for storage location assignment in a randomised warehouse," *International Journal of Production Research*, vol. 55, no. 14, pp. 4035–4052, 2017.
- [21] H. Ravinder and R. B. Misra, "Abc analysis for inventory management: Bridging the gap between research and classroom," *American journal of business education*, 2014.
- [22] L. Zhou, L. Sun, Z. Li, W. Li, N. Cao, and R. Higgs, "Study on a storage location strategy based on clustering and association algorithms," *Soft Computing*, vol. 24, pp. 5499–5516, 2020.
- [23] X. Xu and C. Ren, "Research on dynamic storage location assignment of picker-to-parts picking systems under traversing routing method," *Complexity*, vol. 2020, no. 1, p. 1621828, 2020.
- [24] S. E. McInerney and V. Yadavalli, "Increasing warehouse throughput through the development of a dynamic class-based storage assignment algorithm," *South African Journal of Industrial Engineering*, vol. 33, no. 2, pp. 157–167, 2022.
- [25] M. Kofler, A. Beham, S. Wagner, and M. Affenzeller, "Robust storage assignment in warehouses with correlated demand," *Computational intelligence and efficiency in engineering systems*, pp. 415–428, 2015.
- [26] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.
- [27] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
- [28] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *2016 IEEE 26th international workshop on machine learning for signal processing (MLSP)*. IEEE, 2016, pp. 1–6.
- [29] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1809–1818.
- [30] W. Yin and H. Schütze, "Learning word meta-embeddings," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1351–1360.
- [31] J. Coates and D. Bollegala, "Frustratingly easy meta-embedding—computing meta-embeddings by averaging source word embeddings," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018, pp. 194–198.
- [32] R. d. Koster and E. V. D. Poort, "Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions," *IIE transactions*, vol. 30, no. 5, pp. 469–480, 1998.
- [33] H. Diefenbach, E. H. Grosse, and C. H. Glock, "Human-and-cost-centric storage assignment optimization in picker-to-parts warehouses," *European Journal of Operational Research*, vol. 315, no. 3, pp. 1049–1068, 2024.
- [34] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *International conference on machine learning*. PMLR, 2018, pp. 794–803.
- [35] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.



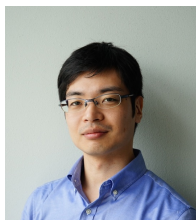
YUSUKE ASAI received his B.E. and M.E. degrees in Engineering from Nagoya University, Japan, in 2019 and 2021, respectively. He is currently pursuing the Ph.D. degree with the Department of Graduate School of Engineering, Nagoya University. His research interests include autonomous mobile robots (AMRs), multi-modal perception using LiDAR and vision systems, and warehouse logistics optimization, particularly focusing on storage location assignment problems.



KAIRI TAKAKI received his B.E. degree in Engineering from Nagoya University, Japan, in 2024. He is currently pursuing the M.E. degree in Engineering with the Department of Graduate School of Engineering, Nagoya University. His research interests include logistics facility planning and warehouse layout optimization, with a particular focus on meta-heuristic algorithms such as simulated annealing.



SHIN KATAYAMA is a project assistant professor in Graduate School of Engineering, Nagoya University, Japan. He received Ph.D. degree in engineering from Nagoya University, Japan in 2023. His current research interests include human-computer interaction, ubiquitous computing systems, and affective computing.



KENTA URANO (Member, IEEE) received his B.E., M.E. and Ph.D. degrees in engineering from Nagoya University, Japan in 2016, 2018, and 2021, respectively. From 2021, he is an assistant professor in Graduate School of Engineering, Nagoya University. His research interests include location based system, human activity recognition, real-world data modeling, and biosignal entertainment computing.



TAKURO YONEZAWA (Member, IEEE) is an associate professor in Graduate School of Engineering, Nagoya University, Japan. He received Ph.D. degree in the Media and Governance from Keio University in 2010. His research interests are the intersection of the distributed systems, human-computer interaction and sensors/actuators technologies. He is a member of IPSJ, IEEE, IEICE and ACM.



NOBUO KAWAGUCHI (Member, IEEE) received his B.E., M.E. and Ph.D. degrees in computer science from Nagoya University, Japan, in 1990, 1992, and 1995, respectively. From 1995 he was an associate professor in the department of Electrical and Electronic Engineering and Information Engineering, school of engineering, Nagoya University. Since 2009, he has been a professor in the graduate school of Engineering, Nagoya University. His research interests are in

the areas of human activity recognition, smart environmental system and ubiquitous communication system.