Heuristic-Based Agent to Solve The Online Three-Dimensional Container Loading Problem

Juan Sebastián Herrera-Cobo[1] and David Álvarez-Martínez[2]

[1] Independent Researcher, Berlin 10367, Germany sebashc_3712@hotmail.com
[2] Los Andes University, Bogotá D.C. 111711, Colombia d.alvarezm@uniandes.edu.co

Submitted to special session Learning to Optimize: RL as an optimizer

Abstract. The rapid growth of the global economy, driven significantly by e-commerce and complex logistics systems, creates an urgent need for increased process automation. Building upon previous heuristic-based approaches, this study proposes a Deep Reinforcement Learning (DRL) framework to address the Online Three-Dimensional Container Loading Problem (OCLP). We introduce a Dueling Deep Q-Network (DQN) agent that integrates an auxiliary Feasibility Masking mechanism to optimize space utilization. Unlike traditional methods relying solely on fixed logic, the proposed agent utilizes height maps and box dimensions to represent the state. The agent dynamically selects the optimal placement heuristic (e.g., stacking, best-fit, corner-based) for each time step, while an auxiliary mask head predicts and prunes invalid actions to accelerate training convergence and ensure physical stability. Computational experiments demonstrate that this hybrid approach, combining the adaptability of deep learning with established heuristics, improves volumetric efficiency and generalization in automated environments.

Keywords: Deep Reinforcement Learning · Dueling DQN · Feasibility Masking · Online 3D Container Loading · Heuristics.

1 Introduction

Three-dimensional loading decisions are at the core of modern fulfillment and warehousing operations, where items must be placed in a container under strict geometric and operational constraints. In many automated settings, items arrive sequentially on a conveyor or at a robotic cell, and the system must decide in real-time whether and where to place each item. Unlike offline bin packing, where all items are known *a priori*, the online variant requires immediate placement decisions without knowledge of future arrivals. This constraint significantly increases complexity, as purely greedy strategies often lead to suboptimal long-term space utilization.

We consider a single rectangular container of fixed dimensions. At each arrival, the system observes the current container surface—represented as height map and the dimensions of the incoming item. The decision is to place the item while enforcing non-overlap, boundary, and stability constraints. We study mainly the fixed orientation and without buffer variant.

Since automated packing cells are increasingly deployed under a Robots-as-a-Service (RaaS) model, the resulting policies must be industry-agnostic, remaining effective across highly heterogeneous item distributions without per-deployment calibration. Consequently, the policy must generalize to instances whose size distributions may differ substantially from those seen during training.

We propose a hybrid decision framework that maintains geometric placement logic via a portfolio of classical heuristics while learning *which heuristic to apply* at each step. Concretely, we train a Dueling Deep Q-Network (DQN) agent [7,10] that maps the container state and current item features to a discrete action, selecting one among several placement heuristics. To prevent wasted learning signals on infeasible decisions, we integrate a feasibility masking mechanism that prunes actions incapable of producing valid placements.

This paper makes the following contributions:

– We introduce a hybrid DRL policy that learns to select among a compact portfolio of placement heuristics, avoiding the complexity of a large continuous placement action space.

– We integrate feasibility masking to eliminate invalid heuristic actions at decision time, improving learning efficiency in constrained packing environments.
– We evaluate the resulting agent on synthetic and literature-derived instance families, comparing it against representative heuristic and learning-based baselines.

The remainder of the paper is organized as follows: Section 2 reviews related literature. Section 3 formalizes the problem and constraints. Sections 4–5 describe the reinforcement learning formulation and network architecture. Section 6 presents the experimental evaluation, and Section 7 concludes.

2 Literature Review

This section surveys literature relevant to our setting: online three-dimensional loading into a single container under practical constraints, focusing on learning-based decision policies and optional buffering.

Cutting-and-packing problems cover a wide range of objectives and operational assumptions. A central distinction is whether the goal is to minimize the number of containers (bin packing) or to maximize packed volume in a single container (knapsack/single-container loading) [11]. Realistic container loading also incorporates operational constraints such as orientation rules, stability requirements, and handling constraints [2]. These practical constraints motivate solution approaches that explicitly account for feasibility during decision-making.

Online packing differs from offline loading because decisions must be made sequentially. This is particularly impactful in three dimensions, where early placements may create unusable cavities or induce fragmentation. Zhao et al. [15] address this setting with a constrained Deep Reinforcement Learning (DRL) approach that predicts feasibility and selects placements from a discretized set of candidate actions. Huertas et al. [5] extend the problem by incorporating buffering and repacking variants, though their solution remains heuristic-driven rather than learned.

Learning-based approaches must reconcile adaptivity to diverse sequences with strict feasibility under geometric and stability constraints. While Zhao et al. [15] combine constrained RL with feasibility prediction, their policy operates over a discretized geometric space that can remain prohibitively large.

Our work follows a different design: instead of learning geometric placements directly, we learn a high-level heuristic selection policy. This reduces the action space to a small discrete set while leveraging established placement heuristics. We apply feasibility masking so the agent only compares actions that can generate valid placements.

Wong et al. [12] propose HHPPO, combining Proximal Policy Optimization with two heuristic modules—an Extreme Point Sorting Method (EPSM) and Deepest Bottom Left with Fill (DBLF)—integrated directly into the reward signal. A key difference from our approach is their use of penalty-based reward shaping to discourage infeasible placements, whereas we employ explicit feasibility masking to prune invalid actions before they reach the policy. Furthermore, their evaluation is conducted on custom simulation models with a limited set of object types, rather than on the standard `cut1`, `cut2`, and `rs` benchmarks [15], which precludes direct numerical comparison. Our work differs by defining a discrete heuristic-selection action space with feasibility masking, evaluated on established benchmarks to facilitate reproducibility.

Table 1 summarizes representative related works.

3 Problem Definition

This section describes the Online Three-Dimensional Container Loading Problem (OCLP), including the physical setup, decision process, and constraints.

3.1 Packing Scene

The packing environment consists of a single rectangular container with fixed internal dimensions $L \times W \times H$. Items arrive sequentially from an external source and must be placed inside this container. The container uses a right-handed coordinate system with the origin at one corner: the x-axis runs along L, the y-axis along W, and the z-axis along H. The floor lies at $z = 0$.

Table 1. Representative literature on online single-container 3D loading.

Ref.	Setting	Decisions	Method	Limitation vs. our work
[1]	Semi-online 3D heuristics	Heuristic choice	Heuristics	No DRL masking
[4]	Online 3D dynamic arrivals	Place by rule	Heuristic	Fixed logic; no learning
[5]	Online single-container	Place/defer	Heuristics	No learned policy
[15]	Online 3D packing	Discretized placements	Constrained DRL	Geometric action space
[14]	Online 3D with waiting	Place/defer	DRL	No heuristic-portfolio
[13]	Robot packing	Reliable placements	DRL	No heuristic selection
[9]	Online 3D pack/unpack	Pack and repack	DRL	Different action model
[12]	Online 3D optimization	RL + heuristics	Deep Q-learning	No feasibility masking

Each item i is a rectangular cuboid with dimensions (l_i, w_i, h_i). Items may be rotated about the vertical axis (swapping l_i and w_i), but vertical flipping is prohibited—a *this side up* constraint reflecting practical requirements such as fragility or weight distribution. The effective base dimensions become (l'_i, w'_i), while h_i remains fixed. A placed item i at position (x_i, y_i, z_i) occupies the region $[x_i, x_i + l'_i] \times [y_i, y_i + w'_i] \times [z_i, z_i + h_i]$.

3.2 Online Arrival and Decision Process

Items arrive one at a time in an unknown order. At each time step t, the system observes the current item dimensions (l_t, w_t, h_t) and the collection of available empty spaces, then makes an irrevocable decision: place the item or defer it to the buffer.

3.3 Geometric Constraints

Every placement must satisfy two geometric requirements:

1. **Boundary constraints:** Ensure the item fits entirely within the container: $0 \leq x_i$, $x_i + l'_i \leq L$, and analogously for the y and z axes with bounds W and H.
2. **Non-overlap constraints:** Require that any two placed items i and j occupy disjoint regions. This holds if and only if they are separated along at least one axis—that is, $x_i + l'_i \leq x_j$ or $x_j + l'_j \leq x_i$ (separation in x), and similarly for y and z.

3.4 Stability Constraint

Placements must satisfy a physical stability requirement. We enforce a *minimum support coverage* rule: the base of each item must be supported over a sufficient fraction of its area. Let $A_{\text{base}}(i) = l'_i \cdot w'_i$ and let $A_{\text{sup}}(i)$ be the portion resting on the floor or top faces of other items. A placement is stable if:

$$A_{\text{sup}}(i) \geq \alpha \cdot A_{\text{base}}(i), \tag{1}$$

where $\alpha = 0.6$ (60% minimum support).

This support check is applied independently to each new placement against the current container state, which includes the floor and the top faces of all previously placed items. Once an item passes the support check and is placed, it becomes a rigid part of the container state for all subsequent placements. In particular, we do not perform recursive load-path analysis: if item A supports item B, and a later item C is placed on

B, only C's direct support from B (and possibly A) is evaluated—A's own support is not re-verified. This is a standard simplification adopted in comparable works [15,5]; incorporating full rigid-body stability with cascading load analysis would require physics simulation, which we identify as a direction for future work (see Section 7).

3.5 Objective

The goal is to maximize volumetric utilization:

$$U = \frac{\sum_{i \in \mathcal{P}} l_i \cdot w_i \cdot h_i}{L \cdot W \cdot H}, \tag{2}$$

where $\mathcal{P}$ denotes successfully placed items.

4 Reinforcement Learning Structure

We formulate the problem as a Markov Decision Process (MDP) defined by the tuple (S, A, R, P, γ).

4.1 State Space (S)

The state s_t is a composite representation capturing the container's geometry, the current item, and heuristic feasibility information. It consists of two components:

1. **Height Map ($\mathbf{H}_t$)**: A grid $\mathbf{H}_t \in \mathbb{R}^{L \times W}$ representing the current stacking height at every coordinate (x, y). This provides a dense spatial representation of the container's surface profile.
2. **Vector Features ($\mathbf{v}_t$)**: A feature vector $\mathbf{v}_t \in \mathbb{R}^8$ containing:
   - Normalized item dimensions: $(l_t/L, w_t/W, h_t/H_{max})$.
   - **Heuristic Proxy Scores**: A set of five scalars $\{s_1, \ldots, s_5\}$ derived from the auxiliary mask prediction (see Section 5.4). These scores indicate the predicted feasibility confidence for the placement location chosen by each heuristic, allowing the agent to gauge the viability of a heuristic before selection.

It is important to note the complementary roles of these two representations. The height map $\mathbf{H}_t$ records only the topmost occupied height at each (x, y) coordinate—a 2.5D surface projection, not a full 3D voxel grid. Consequently, internal cavities beneath placed items are *not* visible in the height map. However, these cavities *are* tracked by the maximal empty spaces list $\mathcal{F}$ (Section 4.2), which maintains exact 3D free-space geometry including voids below overhanging items. This is a deliberate design trade-off: the CNN receives a compact spatial representation from which it learns surface-level patterns (e.g., flatness, wall proximity), while the placement heuristics operate on the full 3D geometry through the empty spaces list. The result is that the learning component focuses on high-level spatial reasoning, while exact geometric feasibility—including cavity utilization—is delegated to the heuristic layer.

4.2 Action Space (A)

Rather than learning exact placement coordinates, the agent selects from a discrete set of $|A| = 5$ high-level placement heuristics. Each heuristic encapsulates a distinct packing strategy: given the current item (with effective dimensions l', w', h) and the list of feasible empty spaces ES, the heuristic deterministically computes a placement position (x, y, z) according to its scoring criterion. This design reduces the action space from a high-dimensional continuous domain to a small discrete set while leveraging well-established packing logic. With this smaller, discretized action space, the agent is incentivized to focus on finding robust, high-level policies—learning *how* to pack—rather than searching for "magical spots" in the coordinate space. By abstracting the exact placement to heuristics, we prevent the agent from overfitting to specific coordinate patterns that do not generalize, allowing it to navigate the optimization landscape more effectively.

Let $\mathcal{F} \subseteq ES$ denote the subset of empty spaces where the current item can be feasibly placed (satisfying boundary, non-overlap, and stability constraints). Each heuristic $a \in A$ defines a scoring function $\sigma_a(e)$ over candidate spaces $e \in \mathcal{F}$, and selects the placement that optimizes this score. The five heuristics are:

1. Stacking Heuristic. This heuristic prioritizes vertical consolidation by placing items as high as possible within the container. It selects the empty space with the maximum base height:

$$e^* = \arg\max_{e \in \mathcal{F}} \ z_e. \tag{3}$$

By building upward from existing items, stacking creates flat, consolidated layers that preserve contiguous floor-level space for larger items arriving later. This strategy is particularly effective when items have similar base dimensions.

2. Best-Fit Heuristic. Best-fit minimizes the vertical residual space above the placed item, seeking locations where the item nearly reaches the container ceiling or the base of an item above. The scoring function is:

$$e^* = \arg\min_{e \in \mathcal{F}} \ (H_e - h), \tag{4}$$

where H_e is the available height of empty space e and h is the item height. This heuristic reduces vertical fragmentation by filling tall gaps with appropriately sized items, leaving minimal unusable overhead space.

3. Semi-Perfect-Fit Heuristic. This heuristic targets placements that minimize local volumetric waste by considering fit quality across all three dimensions. It scores each candidate space by the sum of dimensional residuals:

$$e^* = \arg\min_{e \in \mathcal{F}} \ \big[(L_e - l') + (W_e - w') + (H_e - h)\big]. \tag{5}$$

A placement is considered "semi-perfect" when the item dimensions closely match the empty space dimensions, leaving little residual volume. This approach reduces fragmentation by consuming spaces that would otherwise be difficult to fill with future items.

4. Corner-Based Heuristic. The corner-based heuristic maximizes contact between the placed item and existing boundaries—container walls or faces of previously placed items. Let $c(e)$ denote the number of item faces that would be flush against a wall or neighbor when placed in space e. The selection criterion is:

$$e^* = \arg\max_{e \in \mathcal{F}} \ c(e). \tag{6}$$

Placements in corners (contact on three faces) are preferred over edge placements (two faces), which in turn are preferred over interior placements. This strategy consolidates items against boundaries, preserving large contiguous regions in the container interior for future arrivals.

When the agent selects action a, the corresponding heuristic is executed on the current state. If the heuristic finds a valid placement, the item is placed and the environment transitions to the next state. Actions corresponding to heuristics that cannot produce any feasible placement are masked prior to selection (see Section 5.2), ensuring the agent only chooses among currently viable strategies for geometry and maximal spaces.

5. Complex Fit Heuristic. This heuristic prioritizes the "replicability" of a placement by analyzing the geometry of the remaining empty space. It seeks positions where the distance to the nearest obstacle (boundary or taller item) is an exact multiple of the item's dimensions, thereby minimizing non-usable "residue" gaps. The optimal placement e^* is selected by maximizing the modular fit in both lateral dimensions:

$$e^* = \arg\max_{e=(x,y) \in \mathcal{F}} \big(\mathbb{I}\big(L_x \pmod w = 0\big) + \mathbb{I}\big(L_y \pmod d = 0\big)\big) + \epsilon(e), \tag{7}$$

where L_x and L_y represent the total available length from the placement coordinate to the nearest boundary along the x and y axes, w and d are the item's dimensions, $\mathbb{I}$ is the indicator function, and $\epsilon(e)$ serves as a tie-breaker based on contact maximization and vertical minimization. By enforcing modularity, this strategy facilitates dense packing patterns for batches of identical items.

4.3 Space-Aware Heuristic Scoring

A distinguishing feature of our approach is that each placement heuristic incorporates a *space preservation* criterion. The heuristics evaluate candidate placements based on the quality of the maximal empty spaces that would remain after placement. We define the *average maximal space volume* after placing item i at position (x, y, z) as:

$$\bar{V}_{ES}(x, y, z) = \frac{1}{|ES'|} \sum_{e \in ES'} L_e \cdot W_e \cdot H_e, \tag{8}$$

where ES' denotes the updated list of maximal empty spaces after the placement. Placements that yield higher $\bar{V}_{ES}$ preserve larger, more usable regions for subsequent items.

Each heuristic a evaluates candidate empty spaces using a composite score that balances the primary criterion $\sigma_a(e)$ with the space preservation objective. For a candidate placement in empty space e, the effective score becomes:

$$\tilde{\sigma}_a(e) = \sigma_a(e) + \omega \cdot \bar{V}_{ES}(e), \tag{9}$$

where $\omega > 0$ weights the importance of preserving large empty spaces. This formulation encourages placements that:

- Avoid creating small, fragmented cavities that are difficult to fill
- Maintain contiguous regions suitable for larger future items
- Reduce geometric waste from irregular residual spaces

Myopic placement strategies that only consider immediate fit quality often fragment the container, leaving no room for larger items arriving later. By leveraging the maximal space definition within both the reward structure and the feasibility mask, we obviate the need for computationally expensive look-ahead simulations, such as those found in Monte Carlo Tree Search approaches [15]. Our aim is to maintain the model's logic as efficient and lightweight as possible, deriving implicit foresight from the geometric representation rather than explicit simulation. This allows the agent to balance immediate volume gain against long-term space preservation. The principle is especially important for heterogeneous sequences: preserving a few large empty spaces accommodates a wider range of future item sizes than many small fragmented spaces of equivalent total volume.

4.4 Reward Function (R)

The reward function balances volumetric efficiency with surface flatness to encourage stable stacks. The reward at time t is:

$$r_t = \frac{V_{item}}{V_{container}} + 0.1 \cdot \left(\frac{F_{new} - F_{old}}{L \cdot W} \right), \tag{10}$$

where F is the "Maximal Flat Area" metric calculated on the height map. If the agent attempts an action that yields no valid placement (e.g., the heuristic cannot find a spot), a penalty is applied:

$$r_t = -0.5 \quad \text{(if invalid)}. \tag{11}$$

5 Neural Network Architecture

This section provides a detailed description of the neural network architecture, input processing, and hyperparameters to enable full reproducibility. The agent employs a Dueling Deep Q-Network (DQN) with multi-modal input processing and an auxiliary mask prediction head.

5.1 Input Representation and Preprocessing

The network receives three distinct input modalities corresponding to the current container state and the item to be packed.

*1. Height Map (**H**)* The agent utilizes a grid-based height map representation of the pallet. For a pallet of size $L \times W$, the state is represented as a matrix $\mathbf{H} \in \mathbb{R}^{L \times W}$, where each entry h_{ij} represents the current stack height at position (i, j). The height map is normalized by the maximum container height H_{max}:

$$\tilde{\mathbf{H}} = \frac{\mathbf{H}}{H_{max}}. \tag{12}$$

This normalized map is treated as a single-channel image $(1, L, W)$ and processed via the convolutional encoder.

*2. Item Dimensions (**d**)* The dimensions of the current box to be packed are given as a vector $\mathbf{d} = (l, w, h)$. These are normalized relative to the container dimensions:

$$\tilde{\mathbf{d}} = \left(\frac{l}{L}, \frac{w}{W}, \frac{h}{H_{max}} \right). \tag{13}$$

5.2 Network Architecture

The agent utilizes a custom *ProNet* architecture designed to process the hybrid state representation. The network consists of two input branches that fuse into a shared representation for the value and advantage heads, while a separate mask head branches directly from the spatial features.

Spatial Branch (Height Map) The normalized height map $\tilde{\mathbf{H}}$ $(1 \times L \times W)$ is processed by a 3-layer Convolutional Neural Network (CNN). Unlike standard increasing-channel architectures, we utilize a uniform feature depth to preserve spatial resolution for the mask head:

1. **Conv2D**: $1 \to 64$ channels, 3×3 kernel, padding 1, ReLU.
2. **Conv2D**: $64 \to 64$ channels, 3×3 kernel, padding 1, ReLU.
3. **Conv2D**: $64 \to 64$ channels, 3×3 kernel, padding 1, ReLU.
4. **Flatten**: Output vector size $64 \cdot L \cdot W$.

Vector Branch The vector input $\mathbf{v}_t \in \mathbb{R}^8$ (Item Dims, Buffer, Proxy Scores) is processed by a Multi-Layer Perceptron (MLP):

– Linear $(8 \to 64) \to$ ReLU $\to$ Linear $(64 \to 64) \to$ ReLU.

Heads and Fusion The flattened spatial features and the processed vector features are concatenated to form the latent vector $\mathbf{z}$.

– **Value Head** $V(s)$: Linear(size($\mathbf{z}$) $\to 512) \to$ ReLU $\to$ Linear($512 \to 1$).
– **Advantage Head** $A(s, a)$: Linear(size($\mathbf{z}$) $\to 512) \to$ ReLU $\to$ Linear($512 \to |A|$).

Auxiliary Mask Head To accelerate training, an auxiliary mask head predicts a pixel-wise placement quality map. This head branches *only* from the spatial feature map $\Phi_s \in \mathbb{R}^{64 \times L \times W}$ (extracted by the three-layer CNN encoder), ensuring the prediction relies solely on the current height map geometry. The network structure is:

– Flatten$(64 \cdot L \cdot W) \to$ Linear$(64 \cdot L \cdot W \to 512) \to$ ReLU
– Linear$(512 \to 2 \cdot L \cdot W) \to$ Reshape$(2, L, W)$

The output provides logits for two rotation channels $(0°, 90°)$ across the $L \times W$ grid. During training, this head is supervised using a Binary Cross Entropy (BCE) loss with a weight of $\lambda = 0.15$. The ground-truth mask is generated dynamically by identifying placement positions that satisfy a 60% surface support constraint and maximize the average flat area of the resulting stack.

Design Iteration: The representation of the mask target underwent several design iterations, specifically comparing binary versus continuous (float) targets. While a floating-point representation theoretically allows the model to learn subtle physical gradients, empirical testing revealed that it significantly slowed convergence. The model spent excessive capacity approximating the physics engine rather than optimizing high-level policy decisions. Consequently, a binary mask was adopted to provide a sharper, more efficient learning signal.

5.3 Loss Function

The total loss is a weighted sum of the Deep Q-Learning loss and the auxiliary mask loss:

$$\mathcal{L}_{total} = \text{SmoothL1}(Q, Y_{target}) + \beta \cdot \text{BCEWithLogits}(\hat{M}, M_{GT}), \tag{14}$$

where $\beta = 0.15$. The mask loss is calculated only for steps where a valid box exists ($\sum \text{dims} > 0$).

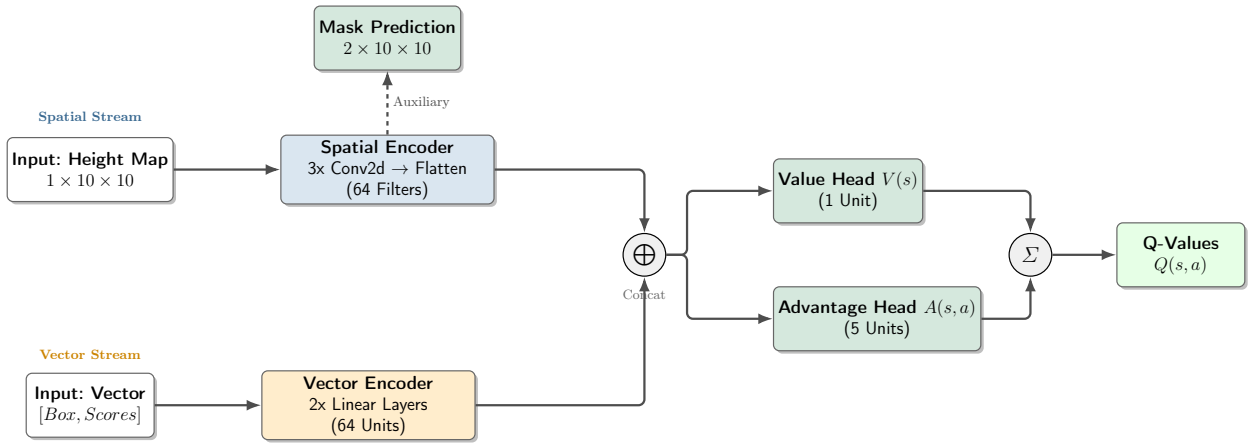

Fig. 1. Simplified architecture of the `ProNet` model. The 3-layer Convolutional stack and the MLP layers are grouped into encoder blocks for clarity. The spatial stream features are shared between the Q-learning fusion layer and the auxiliary Mask prediction head.

5.4 Heuristic Proxy Scores via Masking

The output of the Mask Head is used to generate the "Heuristic Proxy Scores" found in the state vector. For each heuristic h, we simulate its selection logic using the predicted feasibility mask $\sigma(\hat{M})$ as a probability surface. The probability value at the coordinate (x_h, y_h) that heuristic h *would* choose is fed back into the network input. This creates a loop where the agent learns to correlate high mask confidence with successful heuristic execution.

Distinction from Zhao et al. [15]. While our auxiliary head shares architectural elements with the feasibility predictor of Zhao et al. [15]—namely, a CNN trained with Binary Cross Entropy to produce a per-pixel output over the container footprint—the *role* of the mask within the RL system is fundamentally different. In Zhao et al., the policy outputs a distribution over discretized (x, y) placement coordinates, and the predicted feasibility mask is applied as a *hard filter on the action distribution*: infeasible pixels are zeroed out and the softmax is renormalized, so the mask operates directly on the policy's action space. In our formulation, the action space consists of five high-level heuristics, not pixel coordinates; the mask therefore *cannot* filter the action distribution, because the two live in different spaces. Instead, the mask serves two distinct purposes that are absent in [15]: (i) it acts as a *soft spatial prior* that restricts each heuristic's internal

candidate search to high-confidence regions, with a fallback to the full pallet (see Algorithm 1); and (ii) it augments the *state representation* via five proxy scores—one per heuristic—that let the agent anticipate each heuristic's viability before committing to an action. This state-side feedback loop is the principal novel contribution; the mask in our system is trained in order to produce these state features, and its supervision signal is correspondingly richer than pure feasibility: the ground-truth labels positions that both satisfy the 60% support constraint *and* maximize the resulting flat-area metric (Section 5.2), encoding *placement quality* rather than mere feasibility.

5.5 Action Selection with Spatial Fallback

We employ a soft-biased ϵ-greedy strategy where Q-values are augmented by spatial compatibility scores $\mathbf{b}$, derived from the predicted mask $\hat{M}$. To ensure robustness, execution follows a two-stage fallback mechanism: the agent first attempts to apply the selected heuristic strictly within high-confidence regions ($\hat{M} > 0.5$). If no valid placement exists, it relaxes the constraint and retries the heuristic over the full pallet before penalizing the agent (Algorithm 1).

Algorithm 1 Spatially Biased Execution with Fallback

Input: State s_t, Mask Model M_ϕ, Threshold $\tau = 0.5$
$\hat{M} \leftarrow (M_\phi(s_t) > \tau)$ ▷ Binary constraint mask
$\mathbf{b} \leftarrow \text{HeuristicScores}(M_\phi(s_t))$
Select Action:
if $u \sim U(0,1) < \epsilon$ **then**
    $a_t \sim \text{Uniform}(\mathcal{A})$
else
    $a_t \leftarrow \arg\max_a (Q(s_t, a) + \beta \cdot \mathbf{b}_a)$
end if
Execute:
if valid move found using a_t constrained by $\hat{M}$ **then**
    Execute move, observe r_t, s_{t+1}
else if valid move found using a_t (unconstrained) **then**
    Execute move, observe r_t, s_{t+1} ▷ Fallback
else
    $r_t \leftarrow -0.5$; Terminate episode ▷ Penalty
end if

6 Experimentation

6.1 Synthetic Data Generation

We employ a Genetic Algorithm (GA) to generate a training dataset of 50,000 episodes, ensuring generalization beyond standard benchmarks. The GA mixes instances from the `cut_1`, `cut_2`, and `rs` datasets [15].

- **Population**: 10,000 episodes, evolved over 100 generations.
- **Fitness**: $F = (1 - |\text{Fill} - 0.85|) + 0.12 \cdot \text{Diversity}$.
- **Mutations**: Reorder ($p = 0.15$), Switch Episodes ($p = 0.10$), Dimension Perturbation ($p = 0.08$).

To prevent overfitting to the generation order, training batches are shuffled globally. While prior references often lack transparency regarding the specific data distributions used to train their models, making reproducibility difficult, we explicitly employ this highly synthetic, artificially generated data. This strategy forces the agent to learn robust, generalized packing features rather than overfitting to the structured patterns of standard benchmarks.

The GA operates on 50,000 episodes over 100 generations:

1. **Selection**: Tournament Selection ($k = 3$); top 10% preserved as elites.
2. **Crossover**: Two parents combine by alternating items; child length selected from one parent.
3. **Mutation**: Three operators—Reorder ($p = 0.15$): swaps positions within episode; Switch ($p = 0.10$): swaps items between episodes; Dimension ($p = 0.08$): perturbs box dimensions.
4. **Fitness Function**:

$$F(ep) = (1 - |\text{Fill}(ep) - 0.85|) + \alpha \cdot \frac{N_{unique}}{N_{total}} \tag{15}$$

targeting 85% fill rate with $\alpha = 0.12$ weighting diversity.

6.2 Training Process

We conducted experiments in an Apple Silicon (M4, 16GB RAM) environment. We trained our models using distributed training across 8 parallel environments [3] utilizing MPS acceleration. The training duration was 30 epochs, where each epoch consisted of a sample of 10,000 episodes drawn from the synthetic pool of 50,000. We used Adam Optimizer [6]. The distributed training increase the speed from around 3 episodes per second to 9 episodes per second.

During the training process, we monitored utilization in both training and validation sets per epoch, the Q and mask losses, the learning rate decay, and the distribution of heuristics utilized (Fig. 2). Additionally, we periodically sampled container states to audit the placement logic and verify non-overlap constraints, as illustrated in Fig. 3.

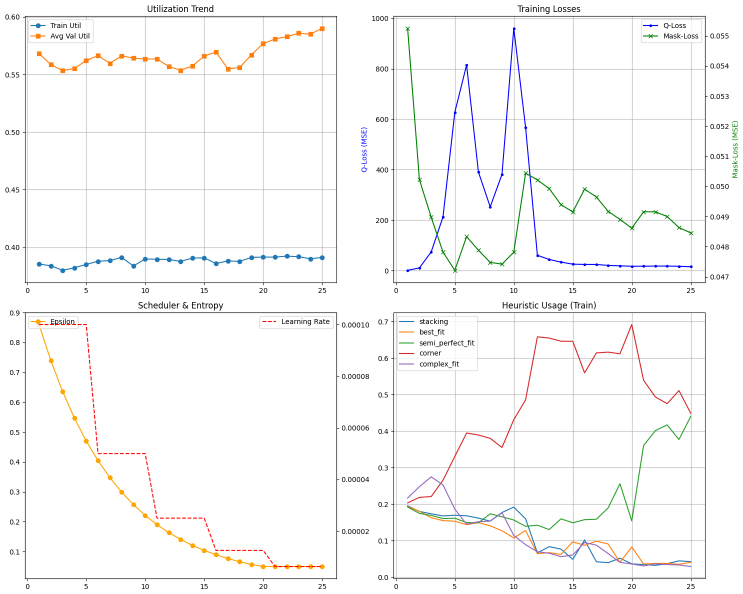

Fig. 2. Training Process Monitoring for Heuristic Based Agent

6.3 Results and Analysis

We compare our results with the main reference in the literature that addresses the same set of instances without buffering. We used a fixed orientation constraint for all items across these experiments. This adheres to the standard protocol for measuring performance in the classic bin packing literature, ensuring that improvements in utilization are attributable to the decision policy's landscape optimization rather than the flexibility of rotation.

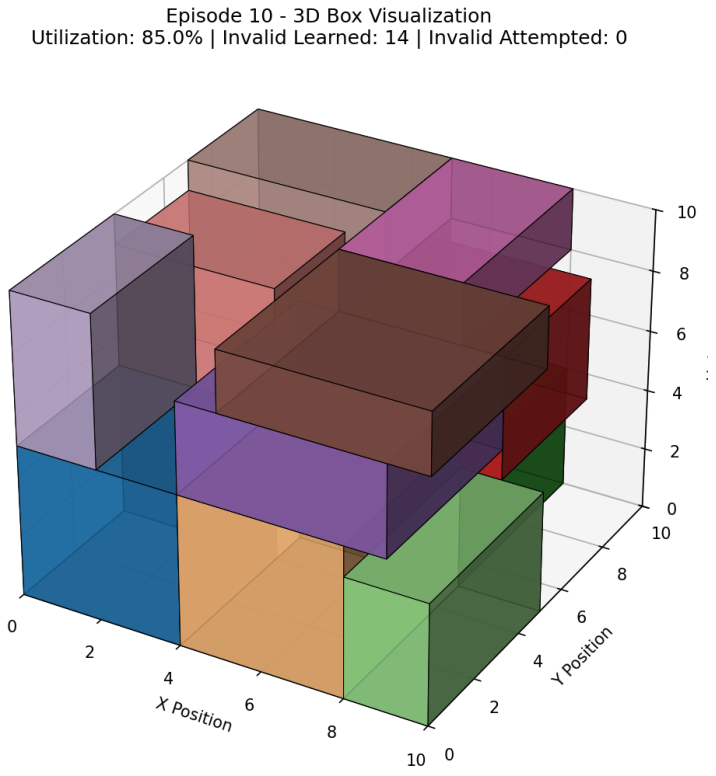

Fig. 3. Sample of Container in instance rs

For completeness, we briefly define the heuristic baselines appearing in Table 2. **Boundary Rule** places items along container walls following a left-bottom-back priority, always seeking contact with existing boundaries. **BPH (Best-fit Packing Heuristic)** is a greedy strategy that minimizes the vertical residual space above each placed item. **LBP (Layer-Based Packing)** partitions the container into horizontal layers and fills each layer using 2D strip-packing heuristics. All three baselines are evaluated under the online, fixed-orientation setting and are described in detail by Huertas et al. [5].

Table 2. Average Utilization Comparison, Fixed Orientation (%)

Reference	**cut1**	**cut2**	**rs**
Boundary Rule (Online)	41.2	40.8	34.9
BPH (Online)	51.9	49.2	35.4
[5]	57.2	57.7	49.4
[15]	73.4	66.9	50.5
Corner-Only (Ours)	56.7	57.9	50.1
Proposed Methodology	62.3	62.3	**52.4**
[8]	74.00	69.7	54.5
LBP (Online)	59.1	59.5	54.7

As shown in Table 2, our agent achieves competitive results, particularly on the challenging `rs` instances. Notably, we outperform the comparable Deep Reinforcement Learning baseline of Zhao et al. [15] on the heterogeneous `rs` set (52.4% vs 50.5%). The approach in [5] relies solely on heuristics without adaptive selection, while [15] determines exact coordinates, sacrificing heuristic efficiency. By integrating feasibility masking with

the maximal space representation, our model balances immediate feasibility with the preservation of usable space.

The Corner-Only ablation (Table 2) isolates the contribution of the learned selector. When the agent is restricted to the corner heuristic alone, utilization reaches 56.7%/57.9%/50.1% on `cut1`/`cut2`/`rs`. The full agent improves over this single-heuristic baseline by 5.6, 4.4, and 2.3 percentage points, respectively. This gap demonstrates that the agent learns meaningful heuristic-switching behavior rather than simply defaulting to corner placement. While corner placement dominates in early time steps—when the container is empty and wall contact is maximized—the agent increasingly selects alternative heuristics (stacking, best-fit, complex-fit) as the container fills and the packing geometry becomes more constrained.

It is worth noting that in homogeneous datasets like `cut1` and `cut2`, coordinate-based agents often achieve high performance by effectively "playing Tetris" with predictable shapes. However, when items are highly irregular (as in `rs`), these patterns disappear. The `rs` benchmark is the most practically relevant, as real-world item distributions are inherently heterogeneous. Our results suggest that while coordinate-based methods excel on simple instances, our heuristic-selection agent offers superior robustness on the complex `rs` set compared to other learning-based approaches. The learned selector's value emerges precisely in these hard cases, where no single heuristic suffices across the full packing sequence. Furthermore, the agent was trained exclusively on GA-generated synthetic data and tested on the standard benchmarks—a harder generalization setting than same-distribution evaluation—which underscores the policy's robustness.

7 Conclusion

This paper presented a hybrid Deep Reinforcement Learning framework for the Online 3D Container Loading Problem that bridges the gap between adaptive learned policies and robust heuristic methods. The core contribution is a Dueling DQN agent that learns to select among a portfolio of placement heuristics rather than predicting raw coordinates, reducing the action space while preserving geometric reasoning. Three key design choices drive performance: (i) a state representation based on height maps; (ii) feasibility masking that eliminates invalid actions based on geometry and maximal spaces during both training and inference; and (iii) space-aware heuristic scoring that preserves high-quality remaining volumes for future placements. An ablation restricting the agent to the corner heuristic alone confirms that the learned selection policy adds 2–6 percentage points of utilization beyond the strongest individual heuristic, validating the value of adaptive heuristic switching.

Experimental evaluation on benchmark instances demonstrates that the proposed method outperforms the primary DRL baseline on heterogeneous sequences where adaptive heuristic selection proves most valuable. The integration of synthetic data generation via Genetic Algorithms further ensures generalization across diverse item distributions, addressing a critical requirement for industry-agnostic deployment in automated packing systems.

Several directions remain for future research. First, implementing an Actor-Critic framework within the same network could offer stability improvements. Second, implement *active buffer management*, where the agent learns a retention policy for difficult items. Third, replacing geometric support constraints with *physics-aware packing*—incorporating rigid-body dynamics, center-of-mass stability, and item fragility—would bridge the gap to real-world robotic deployment.

References

1. Ali, S.M., Ramos, A.G., Carravilla, M.A., Oliveira, J.F.: Heuristics for online three-dimensional packing problems and algorithm selection framework for semi-online with full look-ahead. Applied Soft Computing **151**, 111168 (2024). https://doi.org/10.1016/j.asoc.2023.111168
2. Bortfeldt, A., Wäscher, G.: Constraints in container loading–a state-of-the-art review. European Journal of Operational Research **229**(1), 1–20 (2013). https://doi.org/10.1016/j.ejor.2012.12.006
3. Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., Zhokhov, P.: Openai baselines. https://github.com/openai/baselines (2017)
4. Ha, C.T., Nguyen, T.T., Bui, L.T., Wang, R.: An online packing heuristic for the three-dimensional container loading problem in dynamic environments and the physical internet. In: Applications of Evolutionary Computation (EvoApplications 2017). Lecture Notes in Computer Science, vol. 10200, pp. 140–155 (2017). https://doi.org/10.1007/978-3-319-55792-2_10

5. Huertas, J.M., Pantoja-Benavides, G., Valero, S., Álvarez-Martínez, D.: Approaches for the on-line three-dimensional knapsack problem with buffering and repacking. Mathematics **12**(20), 3223 (2024). https://doi.org/10.3390/math12203223
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017), https://arxiv.org/abs/1412.6980
7. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). https://doi.org/10.1038/nature14236
8. Qi, M., Zhang, L.: Online 3d packing problem based on bi-value guidance. Journal of Computer and Communications **11**(7), 156–173 (2023). https://doi.org/10.4236/jcc.2023.117010
9. Song, S., Yang, S., Song, R., Chu, S., Li, Y., Zhang, W.: Towards online 3d bin packing: Learning synergies between packing and unpacking via drl. In: Proceedings of the 6th Conference on Robot Learning (CoRL 2023). Proc. of Machine Learning Research, vol. 205, pp. 1136–1145 (2023)
10. Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., de Freitas, N.: Dueling network architectures for deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 1995–2003 (2016)
11. Wäscher, G., Haußner, H., Schumann, H.: An improved typology of cutting and packing problems. European Journal of Operational Research **183**(3), 1109–1130 (2007). https://doi.org/10.1016/j.ejor.2005.12.047
12. Wong, C.C., Tsai, T.T., Ou, C.K.: Integrating heuristic methods with deep reinforcement learning for online 3d bin-packing optimization. Sensors **24**(16), 5370 (2024). https://doi.org/10.3390/s24165370
13. Xiong, H., Ding, K., Ding, W., Peng, J., Xu, J.: Towards reliable robot packing system based on deep reinforcement learning. Advanced Engineering Informatics **57**, 102028 (2023). https://doi.org/10.1016/j.aei.2023.102028
14. Zhang, J., Shuai, T.: Online three-dimensional bin packing: A drl algorithm with the buffer zone. Foundations of Computing and Decision Sciences **49**(1), 63–74 (2024). https://doi.org/10.2478/fcds-2024-0005
15. Zhao, H., She, Q., Zhu, C., Yang, Y., Xu, K.: Online 3d bin packing with constrained deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence (2021). https://doi.org/10.48550/arXiv.2006.14978