electronics

Article

# Hierarchical Learning for Closed-Loop Robotic Manipulation in Cluttered Scenes via Depth Vision, Reinforcement Learning, and Behaviour Cloning

Hoi Fai Yu *,† and Abdulrahman Altahhan *,†

School of Computer Science, University of Leeds, Leeds LS2 9JT, UK
* Correspondence: ryan.hoifai@gmail.com (H.F.Y.); a.altahhan@leeds.ac.uk (A.A.)
† These authors contributed equally to this work.

**Abstract**

Despite rapid advances in robot learning, the coordination of closed-loop manipulation in cluttered environments remains a challenging and relatively underexplored problem. We present a novel two-level hierarchical architecture for a depth vision-equipped robotic arm that integrates pushing, grasping, and high-level decision making. Central to our approach is a prioritised action–selection mechanism that facilitates efficient early-stage learning via behaviour cloning (BC), while enabling scalable exploration through reinforcement learning (RL). A high-level decision neural network (DNN) selects between grasping and pushing actions, and two low-level action neural networks (ANNs) execute the selected primitive. The DNN is trained with RL, while the ANNs follow a hybrid learning scheme combining BC and RL. Notably, we introduce an automated demonstration generator based on oriented bounding boxes, eliminating the need for manual data collection and enabling precise, reproducible BC training signals. We evaluate our method on a challenging manipulation task involving five closely packed cubic objects. Our system achieves a completion rate (CR) of 100%, an average grasping success (AGS) of 93.1% per completion, and only 7.8 average decisions taken for completion (DTC). Comparative analysis against three baselines—a grasping-only policy, a fixed grasp-then-push sequence, and a cloned demonstration policy—highlights the necessity of dynamic decision making and the efficiency of our hierarchical design. In particular, the baselines yield lower AGS (86.6%) and higher DTC (10.6 and 11.4) scores, underscoring the advantages of content-aware, closed-loop control. These results demonstrate that our architecture supports robust, adaptive manipulation and scalable learning, offering a promising direction for autonomous skill coordination in complex environments.

**Keywords:** robot manipulation; decision making; pushing; grasping; reinforcement learning; behaviour cloning

check for updates

Academic Editor: Cecilio Angulo

Received: 3 July 2025
Revised: 24 July 2025
Accepted: 27 July 2025
Published: 31 July 2025

**Citation:** Yu, H.F.; Altahhan, A. Hierarchical Learning for Closed-Loop Robotic Manipulation in Cluttered Scenes via Depth Vision, Reinforcement Learning, and Behaviour Cloning. *Electronics* **2025**, *14*, 3074. https://doi.org/10.3390/electronics14153074

## 1. Introduction

In both warehouse and household settings, object arrangements are often cluttered, imposing challenges to visually identifying and grasping objects. Humans adopt prehensile (grasping) and non-prehensile (pushing) actions to improve visibility and retrieve objects. Although extensive research has focused on grasping and pushing actions, these two fundamental actions have been less studied together. Grasping and pushing actions are complementary actions that humans use. Pushing breaks and rearranges the structure of cluttered objects to facilitate future grasping and helps the vision system to identify which ones are graspable [1]. Grasping actions can remove objects and create more space for future actions. These actions create a positive feedback loop for manipulation tasks.

Recent advances have enabled synergies between pushing and grasping. Ref. [2] proposed an end-to-end system to perform coordinated pushing and grasping actions. Following the research of [2], ref. [3] incorporated the push and grasp action into a goal-driven manipulation system, while ref. [4] utilised the push action to search for hidden objects. However, these studies relied on externally mounted vision sensors and no vision feedback was provided during the grasping or pushing executions, limiting the system's adaptability to changes. Humans continuously observe objects while manipulating objects. The closed-loop grasping and pushing action determines which objects are graspable or pushable and also requires planning of the trajectories to reach the targets. Although closed-loop operation is technically more challenging, it has the potential for better adaptability. Although several studies [5,6] have explored closed-loop manipulations such as grasping and repositioning objects, incorporating different closed-loop operations into one system remains largely unexplored.

Based on the motivations mentioned above, this paper proposed a systematic approach to learning synergies between closed-loop grasping and pushing using behaviour cloning and deep reinforcement learning. The system is formulated as a two-layered hierarchy: The high-level decision neural network (DNN) decides whether grasping or pushing actions should be taken, whereas low-level action neural networks (ANNs) comprise the grasping network and the pushing network used to execute decisions. The DNN learns the coordination between pushing and grasping through deep reinforcement learning. Using a $128 \times 128$ depth image captured from the robot's home position, the DNN outputs Q values for both actions. To enable closed-loop operations, the ANNs observe the current state and execute small adjustments ($d_x$, $d_y$, $d_z$ and $d_{yaw}$) each time. The grasping network moves the gripper tip to the desired grasping pose, while the pushing network directs it to the starting point of the push. The following are the contributions of this paper that helped us achieve the main objective.

Synergies Between Closed-Loop Operations : In contrast to previous work [2–4] that limits low-level actions to open-loop operations, in this study, the vision sensor is mounted on a robot arm to enable coordination between closed-loop grasping and pushing. This setup holds potential adaptability to dynamic environments. A hierarchy-based system is proposed to achieve coordination between closed-loop pushing and grasping. The DNN is responsible for decision making, while ANNs are responsible for executing the decisions by choosing target objects and generating corresponding trajectories.

Integration of Behaviour Cloning into ANNs: Both grasping and pushing networks are deep double Q-learning networks (DDQN), allowing ANNs to learn from self-explorations. Learning based on full self-exploration is time-consuming, since numerous trials and errors are required. To address this, the demonstration system is implemented to provide the correct trajectories. Furthermore, the demonstrated push trajectories demonstrate how to separate objects (Section 3.5), which is the main purpose of pushes in this study. Rank loss is introduced to prioritise demonstrated actions for effective behaviour cloning.

Training Pipeline Design: A curriculum-based training approach is introduced since the combination of incorrect decisions and actions seriously affects training efficiency. In phase 1, the demonstration system computes high-level decisions and low-level actions to rapidly collect demonstration examples and allow the ANNs to acquire basic manipulation skills through behaviour cloning. After 200 training episodes, ANNs interact with the environment to learn from self-explorations and correct demonstrations. Once ANNs are reliable, the DNN starts making high-level decisions. The demonstration program will intervene to demonstrate low-level action trajectories when ANNs fail to execute high-level decisions.

Novelty Contributions: This work presents a novel hierarchical framework for closed-loop robotic manipulation that integrates behaviour cloning and reinforcement learning to coordinate grasping and pushing actions. Unlike prior methods that rely on open-loop execution or external vision, our system uses an on-arm depth camera and a high-level decision network trained via reinforcement learning to select object-distribution-aware decisions. The low-level action networks are efficiently initialised using automatically generated demonstrations and subsequently refined via self-exploration, enabling scalable learning and robust control in cluttered scenes.

## 2. Related Work

This study requires autonomous decision making and robot manipulation, involving computer vision and artificial intelligence. Related works in these areas are briefly reviewed in this section.

CAD model-based grasping: Ref. [7] focused on localising objects by matching CAD data with a 3D point cloud. Ref. [8] extended the work by incorporating edge information to replace surface normals for planar industrial parts. However, this extension requires time-consuming feature extraction. To address this, ref. [9] proposed voxel structure-based extraction processes to speed up the overall process. These studies are limited to applications involving a well-defined set of objects.

Model-free grasping: Ref. [10] proposed a method that provides multiple vision views and motor signals to a Grasp-Q network to estimate the best grasp action. Another study [11] proposed an improved soft actor-critical (SAC) algorithm to grasp moving objects, demonstrating the superior performance of SAC compared with other baselines. Furthermore, ref. [12] extended the affordance map to 6D grasping and demonstrated that the proposed methodology can surpass 4-DoF planar grasps. Ref. [6] used human demonstration data to learn 6-DoF closed-loop grasping. Unlike the CAD model-based approach, the model-free approach does not involve CAD data but trains the agent based on various items, allowing the robotics system to have a better general grasping ability, even for unseen objects.

Pushing: Early research by [13,14] used analytical dynamics to generate pushing actions. Recent advances in deep learning have enabled various data-driven approaches. Ref. [15] integrate an optical tensile sensor and CNN pose prediction model into multiple feedback loop controllers to guide the pusher and object toward the target. Ref. [16] used reinforcement learning in simulations with a single object to generalise pushing policies to unseen objects. These studies focus on pushing a single object at a time or executing pushing to grasp objects, but they do not focus on separating objects by pushing in a cluttered situation.

Joint synergies between grasping and pushing: Ref. [17] proposed online reinforcement learning from scratch to select pushing and grasping actions. Their pipeline involved a hand-crafted feature extraction pipeline instead of automatic feature extractions from a deep neural network. Ref. [18] utilised a variational autoencoder (VAE) to automatically convert RGB images to 128-dimensional latent vectors and incorporated pushing to reposition the object in a graspable position for grasping. However, both demonstrations [17,18] only involved 1–2 object(s) in the experience instead of a cluttered arrangement. Ref. [14] introduced a push–grasp action in cluttered environments, which involves pushing the item until it fully rolls into the gripper. Ref. [2] proposed an end-to-end deep reinforcement learning framework to learn the goal-agnostic synergies between pushing and grasping in clutter environments. Similar research [19] has also been carried out to integrate push motion for future suction or grasping motion. Based on the study by [2], various improvements have been made. Ref. [20] proposed a multiple view observation approach to avoid camera blockage during grasping actions. Ref. [1] introduced a push-to-see system to interact with unstructured environments for better instance segmentation performance. Ref. [3] proposed goal-oriented grasping, while ref. [4] used push actions to search for partially occluded or hidden objects. Ref. [21] proposed a dual network and adopted a new reward strategy to achieve synergy of push and grasp action. However, most of this work assumed open-loop low-level operations (i.e., grasping and pushing) and hence limited adaptability to environmental changes.

Comparisons with other work: Refs. [2–4,21] are closely related to this article. The most significant difference is that, instead of using an external camera, the proposed approach mounts the vision sensor on the robot arm to perform closed-loop graspings and pushings. Low-level action networks must choose the target and generate the corresponding trajectories. In contrast, the networks designed by [2] do not have closed-loop trajectory planning ability. Another difference is the hierarchical system architecture: the DNN observes the object arrangement to make a binary decision (grasping or pushing), while the ANN outputs $dx$, $dy$, $dz$ and $dyaw$ for adjustment. Compared to previous studies [2–4,21], the dimension of the action space is smaller and independent of the dimensions of the work space, allowing better adaptability to larger working spaces. In addition, reinforcement learning and behaviour cloning are incorporated to accelerate the ANN training process. A demonstration program is designed to provide ANNs with correct examples, and a rank loss is introduced to prioritise demonstration actions.

## 3. Design and Methodology

### *3.1. Single Depth Image Design*

Our system makes grasp and push decisions based on a single depth image captured from a static home viewpoint. This design choice provides several practical benefits. For example, it significantly reduces computational and memory costs, simplifies the model architecture, and improves inference efficiency. In addition, a single viewpoint streamlines the generation of data sets, replay experience storage, and training dynamics, especially in a simulation-based environment. Importantly, both the low-level action execution networks and the high-level decision-making network rely on the same depth image input to guide their learning and execution, ensuring consistency and coordination across the hierarchy.

### *3.2. State Representations*

The grasp and push state of the ANNs is expressed as a combination of a depth image captured at time $t$ and the yaw angle of the gripper tip relative to the robot coordinate system (Figure 1). The DNN considers only the depth image captured in the home position as its state representation (Figure 2). The depth sensor is mounted on the robot arm and is orientated downward, orthogonal to the ground. The depth image helps both ANNs and DNNs to understand the objects' distribution, while the yaw angle guides the ANNs with adjusting the gripper's position and orientation for grasping and pushing.

Each depth image has a resolution of 128 $\times$ 128 pixels and covers an area of 0.4 m $\times$ 0.4 m at the home position. This implies that each pixel represents approximately a square with a length of 3.125 mm in real-world space, which provides sufficient granularity to distinguish object boundaries and to plan precise grasping and pushing motions.

Although lower resolutions could improve computational efficiency, they reduce the system's ability to resolve fine spatial details, especially for small gaps between objects or partially occluded surfaces. Reducing the resolution (e.g., to 64 $\times$ 64) can lead to poor grasping accuracy and increased decision errors due to the loss of visual detail. In contrast, increasing the resolution to 256 × 256 substantially increased memory and training time without significant improvement in task performance. Overall, 128 × 128 offers a practical balance between visual fidelity and computational cost, and it was empirically validated to support reliable closed-loop manipulation in our setup.

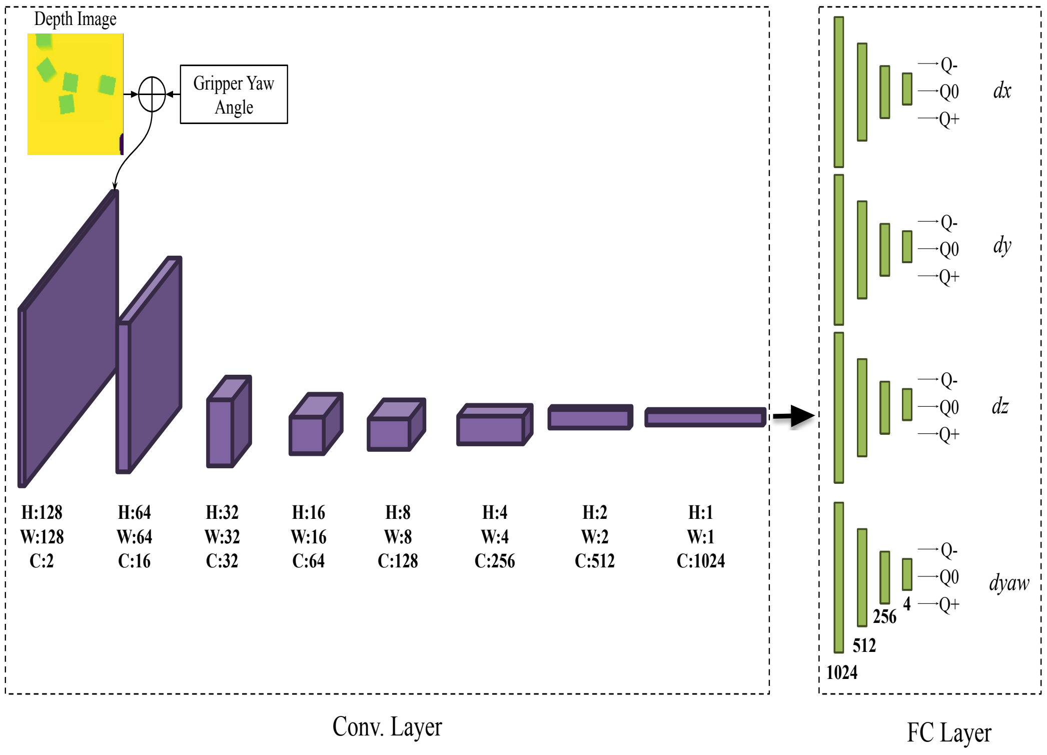

**Figure 1.** Architecture of the low-level action networks.

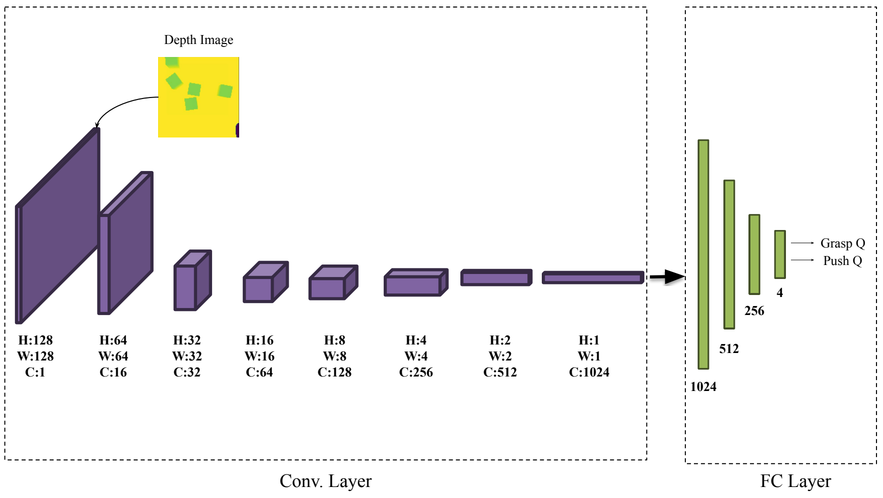

**Figure 2.** Architecture of the high-level decision network.

### 3.3. Outputs

High-Level Decision Neural Network: The DNN outputs are the corresponding Q values of the push and grasp actions (Figure 2). An exploration factor is set as 0.25 to explore alternative actions. This approach strikes a balance between exploration and exploitation.

Low-Level Action Neural Network: The grasping and pushing networks belong to ANNs and have four output branches corresponding to $d_x$, $d_y$, $d_z$ and $d_{yaw}$, relative to the robot coordinate system. Each branch contains three Q values (Figure 1): negative ($Q-$), zero ($Q0$) and positive ($Q+$) adjustments. For grasping and pushing networks, the adjustment ranges of $d_x$ and $d_y$ are $[-0.015, 0, 0.015]$ m, while the range of $d_z$ is $[-0.02, 0, 0.02]$ m. The adjustment range $d_{yaw}$ for the grasping and pushing networks are $[-7.5, 0, 7.5]°$ and $[-15, 0, 15]°$ respectively. The grasping network directs the gripper tip to the grasping pose. When the gripper tip reaches a height below a threshold of $h_g = 0.05$ m, the gripper closes for grasping. The pushing network guides the gripper to approach the starting pose of a push action. Once the gripper tip descends below a height threshold $h_p = 0.05$ m, the gripper moves forward by a fixed distance $d_p = 0.07$ m along the direction of gripper orientation, thus executing the push.

Justifications of ANNs' Adjustment Ranges: The adjustment ranges for $d_x$, $d_y$ ($\pm0.015$ m), $d_z$ ($\pm0.02$ m), and $d_{yaw}$ ($\pm7.5°$ for grasping, $\pm15°$ for pushing) were chosen to align with the principles of closed-loop motion control and strike a balance between motion precision and training efficiency. These small, quantised step sizes allow action neural networks (ANNs) to iteratively refine the gripper pose in response to visual feedback. This fine-grained control is essential for precise manipulation in cluttered scenes, where small positional or rotational errors can result in failed grasps or pushes.

Preliminary experiments revealed that larger step sizes introduced instability, frequent overshooting of target poses, and loss of visual tracking of target objects. In contrast, excessively small step sizes prolonged training and hindered policy convergence. The selected values were empirically validated to be large enough to facilitate efficient learning while maintaining the reliability of low-level motion execution. The simulation results confirm the effectiveness of these settings, resulting in a high training success rate in grasping (98.6%) and pushing (97.8%) (Figure 3).

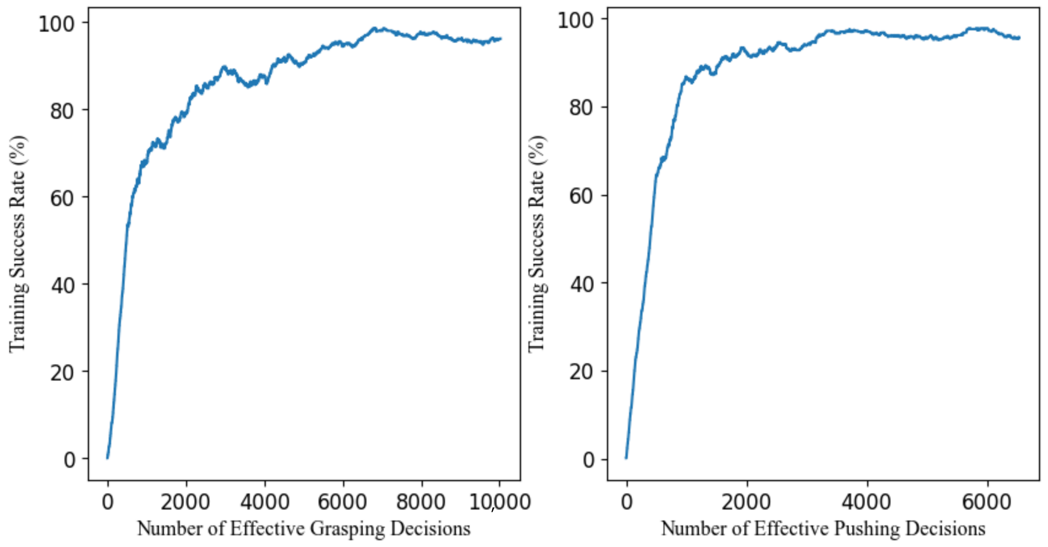

**Figure 3.** (**Left**) Grasping training success rate and (**Right**) pushing training success rate.

## *3.4. Overall System and Network Architecture*

Figure 4 illustrates how DNNs cooperate with ANNs to interact with the environment. The DNN is the first layer of the system that selects between grasping and pushing decisions based on the Q values (Figure 2). If the decision is grasping, the grasping network is triggered; otherwise, the pushing network is selected. Both networks share the same architecture (Figure 1), and the adjustment is made based on the Q values in each branch. The selected low-level actions are transmitted to the robot arm for interaction.

Modular Design: The modular separation between decision neural networks (DNNs) and action neural networks (ANNs) is both theoretically motivated and empirically beneficial. Robotic manipulation tasks in cluttered environments can be naturally decomposed into two subtasks: what to do (e.g., push vs. grasp) and how to do it (e.g., reach the target pose through continuous control). This separation aligns with the classical sense–plan–act paradigms and each module to specialise and optimise for its respective function. The DNN focuses on long-term planning and global scene understanding, while the ANNs refine precise motor control through short-term feedback.

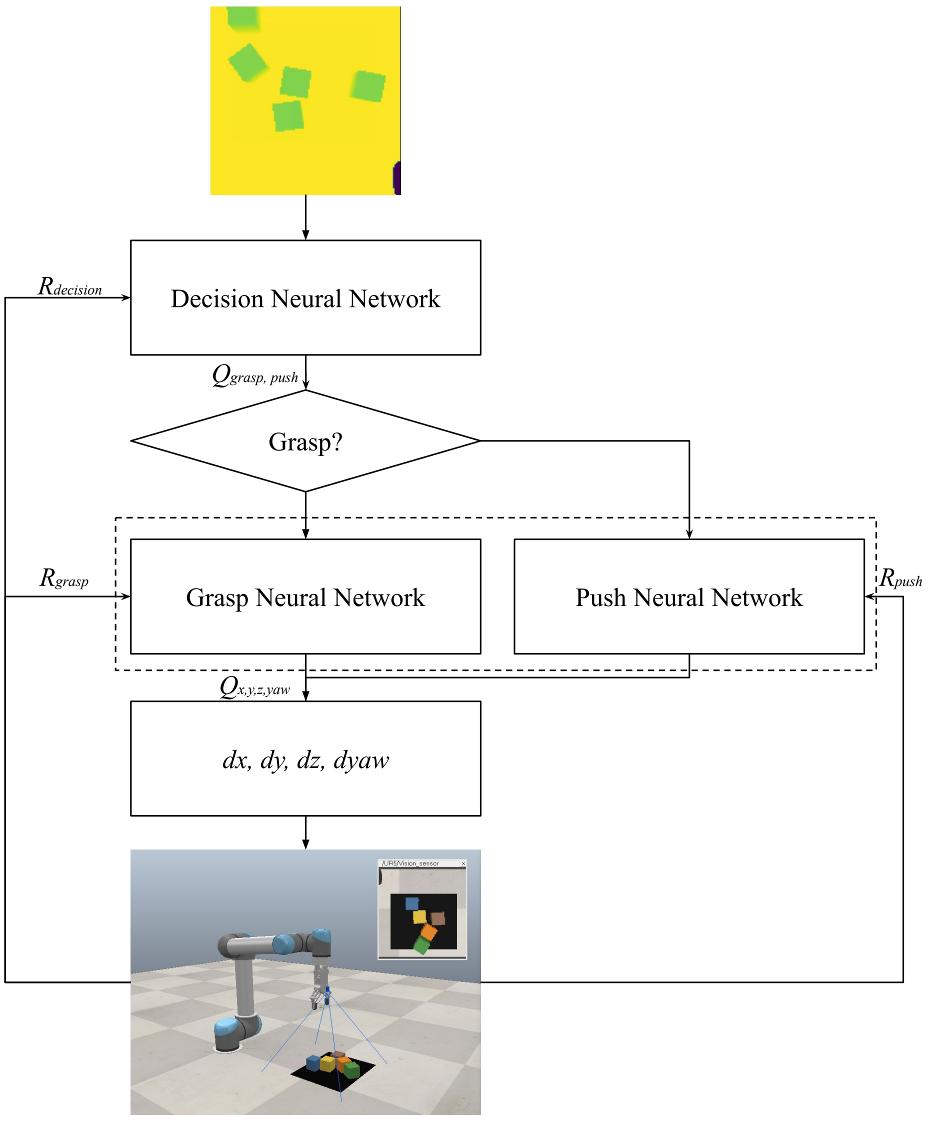

**Figure 4.** Architecture of the overall system.

Although end-to-end learning is theoretically appealing, it introduces significant strong coupling between decision logic and motion control, increasing sample complexity and making targeted debugging more difficult. In contrast, our phased training pipeline (Section 3.9) allows ANNs to first master reliable execution via behaviour cloning and reinforcement learning. Only after this foundation is established, the DNN begins learning high-level decision making through reinforcement learning. The high-level decision (i.e., push vs. grasp) usually needs to be made once per sub-goal, whereas low-level controls require iterative corrections. Separation of these processes avoids unnecessary computation and improves the overall efficiency of the system during inference. More importantly, this modular design promotes reusability and flexibility. Once trained, low-level ANNs can be reused in different high-level strategies or decision models, making the framework more flexible and modular for future extensions.

Over-Estimation Bias Issue: To mitigate the over-estimation bias problem, all ANNs and DNN are implemented using double deep Q-networks (DDQNs), where action selection and action evaluation are decoupled. This is a well-established method used to reduce over-estimation bias in Q-learning over noise value estimates during target computation. Each ANN target network is updated via soft updates to ensure a smoother evolution of the Q value. This gradual adjustment helps prevent abrupt value shifts, which could lead to over-estimated Q values, especially in the early training phases.

### *3.5. Demonstration Program*

Overview: The demonstration program is structured in two main stages: (1) generating trajectories for grasping and pushing, and (2) making high-level decisions based on the availability of trajectories. The program calculates the minimum distance ($d_n$) between each object and its neighbour. $d_n$ is computed by comparing the distance between the target's bounding box and its neighbouring bounding boxes. If $d_n$ is larger than a threshold (0.035 m), the target is sufficiently isolated from its neighbours, and a grasping trajectory is generated for that object. Otherwise, a push trajectory is generated for that object.

Extraction of Orientated Bounding Boxes: In this study, the Orientated Bounding Boxes (OBBs) used by the demonstration generator are not estimated from raw depth images but are directly retrieved from the simulation environment via its built-in API. This enables robust and precise access to ground-truth object poses and dimensions, regardless of occlusion or proximity to neighbouring objects. This is one of the key advantages of using a simulated environment in the early stages of AI system development: it provides reliable learning signals that would otherwise be difficult to obtain in cluttered scenarios in the real world.

Grasping Trajectories: Figure 5 (left) illustrates how to compute the grasping trajectories, where the red line (1) and the blue line (2) represent two possible grasping orientations and the black line is the gripper orientation. Grasp orientation 1 is chosen because of the minimum angular difference between grasp orientation 1 and the gripper orientation. The grasping trajectory is computed on the basis of the bounding box's centre and the chosen grasp orientation.

Pushing Trajectories: Figure 5 (right) illustrates how to compute the push trajectories, where the orange box and blue box are the target and its neighbour, respectively. Vectors 1, 2 and 3 are three possible pushing directions. The tip and tail are the end and starting points of pushing, respectively. Collision checks verify that only vectors 1 and 2 are safe for pushing by checking the starting points. The centre of the target (orange box) is offset by vectors 1 and 2 to simulate the resulting position caused by pushing. Vector 2 is chosen because it is the most effective way to separate the target from its neighbour. Then, a pushing trajectory is generated based on the gripper tip and the starting point of vector 2.

High-Level Decision Making: The high-level demonstration program selects grasping if grasping trajectories are available; otherwise, a push decision is chosen.

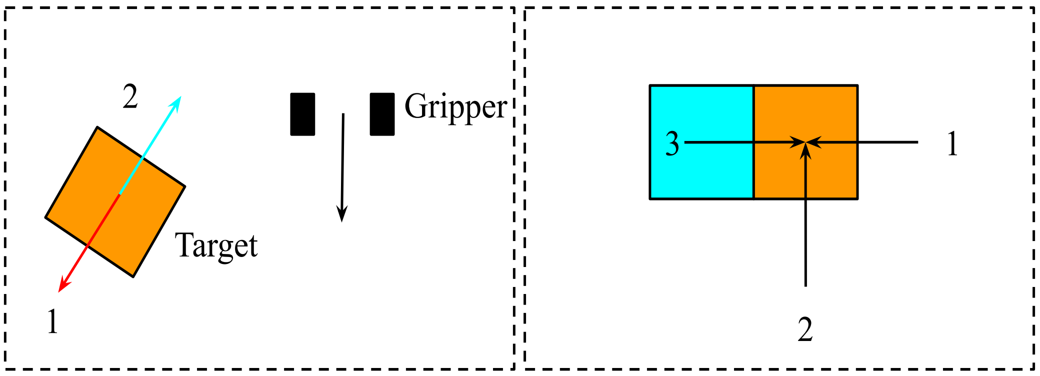

**Figure 5.** (**Left**) Grasping illustration and (**Right**) pushing illustration.

### *3.6. Rewards and Loss Function of DNN*

The reward structure of the DNN is designed to prioritise grasping and choose to push if necessary.

Grasp Rewards (Equation (1)): A grasp decision is correct if an object is grasped and lifted at least by 0.025 m successfully. A grasp decision is neutral if some objects are sufficiently separated ($d_n$ > 0.035 m) for grasping in the working space, but the grasping network fails to execute the decision. A neutral grasp reward is set to 0 to avoid penalising the DNN due to failed executions. The DNN is penalised when the decision is neither correct nor neutral.

$$r_{DNN,grasp} = \begin{cases} 1 & \text{if correct} \\ 0 & \text{else if neutral} \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

Push Rewards (Equation (2)): A push decision is correct when all objects are close ($d_n \leq 0.035$ m) to each other and an object is successfully pushed. In addition, a push decision is neutral when (1) graspable and non-graspable objects coexist in the working space or (2) all objects are close ($d_n \leq 0.035$ m) to each other; however, the pushing network fails to execute the decision in this scenario. Condition 1 prevents unnecessary pushing, while condition 2 avoids penalising the DNN due to poor executions. A push decision is considered incorrect when all objects are sufficiently separated ($d_n > 0.035$ m).

$$r_{DNN,push} = \begin{cases} 0.5 & \text{if correct} \\ 0 & \text{else if neutral} \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

Terminal Reward (Equation (3)): The terminal decision is the grasping decision that results in clearing all objects.

$$r_{terminal} = r_{DNN,grasp,correct} + 5 \tag{3}$$

Loss Function of the DNN: The online DNN $Q_{DNN}$ is updated based on the mean squared error (MSE) (Equation (4)), while the target network $Q_{DNN,target}$ is updated based on the soft update method (Equation (5)), and the update factor $\tau_{DNN}$ is set to 0.05 to ensure stable learning.

$$loss_{DNN,Q} = MSE(q, q_{target}) \tag{4}$$

$$Q_{DNN,target} = Q_{DNN} + (1 - \tau_{DNN})Q_{DNN,target} \tag{5}$$

### *3.7. Rewards and Loss Function of the ANNs*

Grasp Rewards (Equation (6)): A grasp action is successful if an object is grasped and lifted at least by 0.025 m successfully; otherwise, it is a failed action.

$$r_{ANN,grasp} = \begin{cases} 1 & \text{if successful} \\ -1 & \text{otherwise} \end{cases} \tag{6}$$

Push Rewards (Equation (7)): A push action is successful when objects ($d_n \leq 0.035$ m) are pushed. A push action is partially successful when objects with $d_n > 0.035$ m are chosen as pushing targets. This push action is rewarded with 0.1 to prioritise which objects should be pushed. A push action fails when no object is pushed.

$$r_{ANN,push} = \begin{cases} 1 & \text{if successful} \\ 0.1 & \text{else if partially} \quad \text{successful} \\ -1 & \text{otherwise} \end{cases} \tag{7}$$

Loss Function: This loss function ensures that the ANNs learn to prioritise the actions of demonstration. Meanwhile, both networks can learn from their mistakes and reinforce correct actions through self-exploration. The general loss function $loss_{ANN}$ is formulated as Equation (8). $loss_{ANN,Q,rl}$ and $loss_{ANN,Q,bc}$ are computed based on the MSE, as described in Equation (9).

The margin parameter $\rho$ = 0.1 is used within a rank-based loss (Equation (10)) to enforce a preference for expert (demonstrated) actions over non-expert alternatives during behaviour cloning. Specifically, for each output branch (e.g., $d_x$), the rank loss penalises any non-expert Q value $q_{b,i}$ + $\rho$ that exceeds the expert Q value $q_{b,e}$. This encourages the network to maintain a Q-value margin between the action of the expert and all others, effectively pushing the model to prioritise the paths of the experts.

The margin $\rho$ introduces tolerance: non-expert actions are permitted to approach, but not surpass the expert Q value, allowing flexibility in learning while preserving expert bias. Through empirical validation, $\rho$ = 0.1 was found to provide the best trade-off between imitation strength and adaptability. Smaller values slowed convergence due to weaker expert guidance, while larger margins overly constrained the policy and interfered with reinforcement learning updates.

In this formulation, $i$ = 0...2 corresponds to the discrete Q-value outputs ($Q-$, $Q0$, $Q+$) within each branch, while $b$ = 0...3 refers to the following four control branches: $d_x$, $d_y$, $d_z$ and $d_{yaw}$ (Figure 1). The soft update is used to update the target ANNs. $\tau_{ANN}$ = 0.05, which ensures a smooth and stable transition in Q-value targets during training.

$$loss_{ANN} = loss_{ANN,Q,rl} + loss_{ANN,Q,bc} + loss_{rank} \tag{8}$$

$$loss_{ANN,Q} = \frac{1}{4}\sum_{b=0}^{3} MSE(q_b, q_{b,target}) \tag{9}$$

$$loss_{rank} = \frac{1}{4}\sum_{b=0}^{3}\left(\frac{1}{3}\sum_{i=0}^{2}\begin{cases} min(q_{b,i} + \rho - q_{b,e}, 0) & \text{if } i \neq e \\ 0 & \text{otherwise} \end{cases}\right) \tag{10}$$

### 3.8. Experience Replay

A hybrid sample of experience replay is adopted. In total, 80% of chance adopts prioritised experience replay to accelerate the training process; otherwise, uniform sampling is used to ensure thorough learning from the buffers. For the ANNs, there are two experience buffers, the self-exploration experience and the demonstration experience. The self-exploration experiences are sampled based on $loss_{ANN,Q,rl}$, while the demonstration experiences are sampled based on $loss_{ANN,Q,bc}$ + $loss_{rank}$. For DNNs, the experiences are sampled based on $loss_{DNN,Q}$.

### 3.9. Training Pipeline and Hardware Setting

Training Setting: In each episode, five square cubes are randomly generated and placed closely together. An episode is complete when all objects are cleared from the working space or more than 50 low-level actions are executed without clearing all objects. The maximum step of each low-level action is set to 25 to ensure sufficient self-exploration. The low-level grasping action is completed when an object is grasped successfully or 25 steps are exhausted. In contrast, the low-level pushing action is completed when the constant-length pushing along the gripper's orientation, described in Section 3.3, is executed or when 25 steps are exhausted. When objects are out of the working space due to interactions, they are randomly reset to the working space. The maximum number of training episodes is set to 2000. The proposed system is trained in Pytorch v2.5.1 with an NVIDIA RTX 4060 (NVIDIA Corporation, Santa Clara, CA, USA) on a 12th Gen Intel Core i7-12650H CPU (Intel Corporation, Santa Clara, CA, USA). A curriculum-based training pipeline is proposed and structured as follows.

Object Choice: Cubes were intentionally selected as a controlled object category to isolate and evaluate the core contributions of this work, which consisted of analysing the hierarchical coordination between pushing and grasping, and the effectiveness of our decision-making and low-level closed-loop control networks. Cubes provide consistent contact surfaces and predictable dynamics during pushing, enabling reproducible evaluation across different experimental runs and baselines.

Phase 1 Full Demonstration: The demonstration program is responsible for all high-level decisions and low-level actions, allowing for the accumulation of low-level demonstration experience and allowing ANNs to acquire basic manipulation skills by behaviour cloning. After 200 training episodes, the training enters the next stage.

Phase 2 ANN Training based on Reinforcement Learning and Behaviour Cloning: ANNs attempt to grasp and push objects based on their learnt actions. The demonstration program provides demonstrations only when the ANNs fail to execute a grasp or push action. In this stage, reinforcement learning and behaviour cloning are integrated to update ANNs, as described in Section 3.7. Once the successful rate of both low-level networks reaches 70%, the system transitions to the next phase.

Phase 3 Full System Training: The DNNs begin taking over the responsibility of making high-level decisions, while the ANNs are tasked with executing the decisions made by the DNNs. The DNNs and ANNs learn together to complete an episode successfully. The demonstration program remains available but only demonstrates low-level actions when ANNs fail to execute the decisions.

Diversity of Training Samples: Multiple mechanisms are in place to ensure diversity in the replay buffer and prevent overfitting to robot-specific kinematics or scripted patterns. At the start of each training episode, five square cubes are placed within a bounded workspace. This setup generates a wide range of object configurations, ensuring that the demonstration trajectories adapt to diverse spatial arrangements rather than following a fixed set of paths.

As described in phase 1, demonstration data are primarily used in the early stages of training to bootstrap low-level action networks. After the first 200 episodes, the reliance on scripted trajectories is gradually reduced in phases 2–3. Demonstration data are only generated when the ANNs fail to successfully execute grasp or push actions. This gradual transition enables reinforcement learning to explore alternative strategies and expand the diversity of the trajectory.

Exploration is further encouraged through $\epsilon$-greedy policies with an exploration factor of 0.1 for ANNs and 0.25 for DNNs. This controlled stochasticity introduces variability in action selection, preventing overfitting to rigid trajectories and encouraging the discovery of novel state–action transitions. The replay buffers contain both demonstration and self-exploration data. In total, 80% of the samples are drawn using prioritised experience replay, while 20% are uniformly sampled (Section 3.8), helping to maintain buffer diversity and avoid the over-representation of repetitive trajectories.

### 3.10. Hyperparameter Setting

The soft update is used and $\tau$ is set to 0.05 to update all target networks based on empirical experiments. A smaller $\tau$ (e.g., <0.01) can result in overly conservative updates to DNN or ANNs, significantly slowing convergence and delaying meaningful policy improvement. In contrast, a larger $\tau$ (e.g., >0.1) can cause more abrupt updates to the target network, introducing instability or even oscillatory behaviour in the Q values.

The exploration factor ($\epsilon$) of the DNNs and ANNs are set as 0.25 and 0.1, respectively, to explore alternative solutions. They were empirically tuned on the basis of early stage experiments designed to balance exploration stability and convergence in their respective roles. A higher exploration rate was used to encourage broader policy exploration during high-level decision making, especially in the early stages of training. Since the DNN is responsible for switching between pushing and grasping actions, adequate exploration helps avoid premature convergence to suboptimal behaviour (e.g., always grasping or always pushing). A lower exploration factor was chosen to stabilise fine-grained control during low-level action execution. Because ANNs operate in a closed-loop manner with small motion adjustments, excessive exploration (e.g., 0.25) introduced erratic motion and reduced success rates. A value of 0.1 provided enough stochasticity to support continued learning while maintaining precise motion control. These values were selected to reflect the differing roles of DNNs (global strategy selection) and ANNs (local action refinement). The chosen settings led to stable training, high task success rates, and efficient decision sequences, as shown in our evaluation results.

The learning rates for the low-level action networks and the high-level decision networks are set to $1 \times 10^{-4}$. All networks use ADAM as an optimiser due to its efficiency and adaptive learning abilities [22]. The buffer sizes for the self-exploration experience, the demonstration experience, and the DNN experience are set to $2 \times 10^5$, $1 \times 10^5$, and $0.5 \times 10^5$ to ensure sufficient diversity for learning.

## 4. Results and Discussion

### 4.1. Training Results

The completion rate (CR) and the decisions taken for completion (DTC) are the metrics in the training used to choose the best DNN model, while the success rate (SR) is the metric used in the training to choose the best grasping and pushing network.

Completion Rate (CR): Equation (11) measures how many successful episodes are in the most recent 250 episodes (N). This metric measures whether the DNN's decision is effective enough to clear all objects in various scenarios.

$$CR = \frac{Number\,of\,successful\,episodes}{N} \tag{11}$$

Decisions Taken for Completion (DTC): Equation (12) measures the efficiency of DNNs in completing an episode. $N_s$ is the number of successful episodes within the most recent 250 episodes, while $N_{DNN}$ is the number of high-level decisions to remove objects in the workspace.

$$DTC = \frac{1}{N_s}\sum^{N} N_{DNN} \tag{12}$$

Success Rate (SR): Equation (13) measures how many successful grasping and pushing actions there are under the most recent $N_{effective}$ (500) effective grasping or pushing decisions. It is important to note that demonstration actions are not included in the calculation of the success rate for fair evaluations. An effective grasping decision is defined as (1) some objects being sufficiently separated ($d_n$ > 0.035 m) in the working space or (2) the actions of the grasping network resulting in a successful grasp even if no object is sufficiently separated. In addition, an effective push decision is made when some objects are closely positioned together ($d_n \leq$ 0.035 m) in the scene. Effective decisions for grasping and pushing are defined to ensure that the success rates of ANNs are not seriously underestimated or over-estimated due to ineffective high-level decisions.

$$SR = \frac{Number\ of\ successful\ actions}{N_{effective}} \tag{13}$$

Figure 6 (left) illustrates that the highest completion rate (CR) achieved is 100% in training. A general downward trend in the decisions taken for completion (DTC) is observed in Figure 6 (right), indicating that the overall efficiency of the system improves as training progresses. Training decisions decrease from a high of value of 12.1 to a low value of 8.5. Figure 3 shows an upward trend in the success rates of grasping and pushing networks, suggesting that their behaviours are improved by behaviour cloning and reinforcement learning. The highest success rates of grasping and pushing networks are 98.6% and 97.8%, respectively.

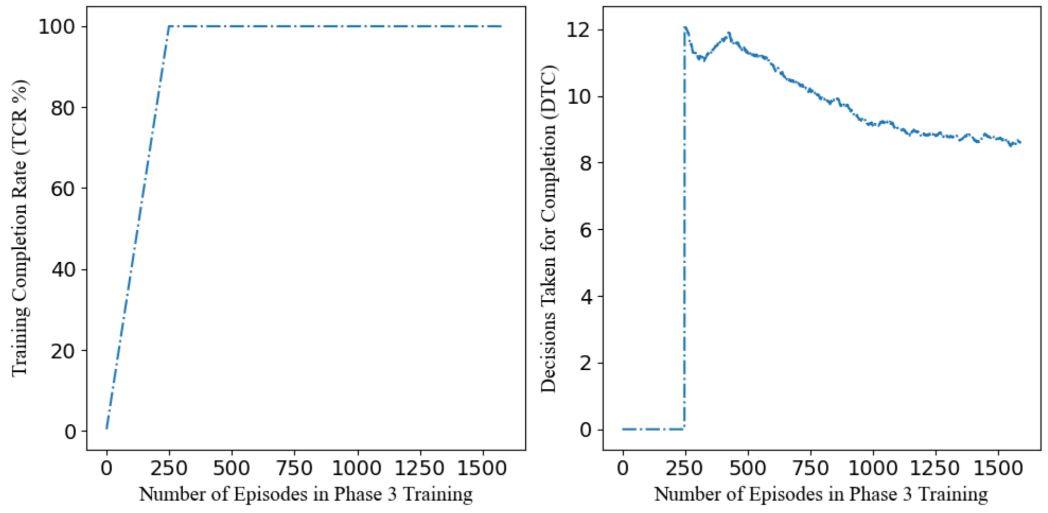

**Figure 6.** (**left**) DNN training completion rate (TCR) and (**right**) decisions taken for completions (DTC).

### 4.2. Evaluation Metrics

The completion rate (CR) and decisions taken for completions (DTC) described in Equations (11) and (12) are also used as metrics in the evaluation. The only difference is that $N$ in Equation (11) is set to 100 in the evaluation. The average grasping success rate per completion (AGS) (Equation (14)) is added to measure how effective the system is in making correct grasping decisions. $N_s$ in Equation (14) is the total number of successful test cases, while $G_{success}$ and $G_{attempt}$ in Equation (14) represent the amounts of successful grasps and grasp attempts per episode.

$$AGS = \frac{1}{N_s} \sum^{N_s} \frac{G_{success}}{G_{attempt}} \tag{14}$$

### 4.3. Proposed Baselines

Grasping-Only System: This baseline focuses on grasping actions without pushing. The trained grasping network is used for performance evaluation. This baseline aims to study how the grasping-only system completes the episode, especially when some objects are close to each other.

Sequential Grasping-Pushing System: This baseline follows a strictly alternating sequence of actions (grasp→push →grasp→push . . . ). This baseline does not adapt to the state of the environment but strictly alternates between two actions. The trained grasping and pushing networks are utilised in this baseline. This study aims to investigate the efficiency difference between the proposed system and the fixed sequence baseline.

Demonstration System: This baseline is the system designed as described in Section 3.5. The demonstration system handles all high-level decisions and low-level actions. Unlike the first two baselines, high-level decisions of the demonstration system are content-aware, as described in Section 3.5. Since position and orientation information are provided for the demonstration, the completion rate and average grasping success rate are perfect. Hence, this study focuses on whether the proposed system can outperform the demonstration program in terms of decision efficiency (DTC).

The above baselines are designed in a way to evaluate (1) the necessity of pushing action in a cluttered environment, (2) the importance of context-aware decision making, and (3) the effectiveness of the learning ability of the proposed system in surpassing the demonstration system.

### 4.4. Evaluation Results

Evaluation Setting: The evaluation is similar to the training setup (Section 3.9). The main difference is that the total number of evaluation episodes is set to 100. In addition, the demonstration program is no longer included to provide correct examples during the evaluation of the proposed system. All decisions are made and executed by the DNNs and the ANNs, respectively.

### 4.5. Discussions of Baselines

Grasping-Only Baseline: The grasping-only baseline achieved a CR of 100% but achieved the lowest AGS (86.6%). Based on observations ((https://github.com/ryanyu512/vision_based_synergies_between_closed_loop_grasping_and_pushing/blob/main/research2.0/grasp_only.gif) (accessed on 12 December 2024)), failed grasping attempts sometimes act as incidental pushes that help separate objects. In other words, pushing is still necessary. In addition, the DTC of the grasping-only baseline is the lowest (6.1), indicating the highest efficiency. In one way, efficiency is important, but low AGS may indicate more failed grasp attempts, increasing the risk of gripper collisions.

Sequential Grasping–Pushing Baseline: The sequential grasping–pushing system achieved a CR of 100%, and its DTC (10.6) was higher than that of the proposed system (7.8). The fixed alternating cycle of this baseline is independent of the current state, resulting in more unnecessary pushing actions. In contrast, the proposed system is aware of the object distribution and selects appropriate decisions, resulting in higher efficiency.

Demonstration System: The demonstration system baseline refers to a system that directly executes the hand-coded decisions and trajectories of the demonstration program during evaluation, without any learning. It fully replicates the scripted logic described in Section 3.5, including the hard-coded separability threshold ($d_n$) and rule-based trajectory planning.

In contrast, during training, the demonstration program is used only as the data source for behaviour cloning. It provides samples of low-level actions in the early phase and intervenes only when learning agents (ANNs) fail to act effectively (as described in Section 3.9). Once ANNs achieved a sufficient success rate, the DNNs began learning to make high-level decisions (push vs. grasp) via reinforcement learning to optimise long-term accumulated rewards, independent of any hand-coded decision logic.

As a result, the demonstration system is inherently conservative and deterministic. It often performs additional push actions to satisfy the hard-coded rule before grasping, which increases the number of high-level decisions taken (DTC = 11.4). In contrast, the learnt DNN policy is reward-driven and adaptive: it can identify opportunities to grasp objects earlier, even if they are slightly occluded or close to others, thus reducing the DTC (7.8) without compromising the grasping success rate. This demonstrates the benefit of reinforcement learning in improving decision efficiency beyond rule-based heuristics.

Figure 7 illustrates why the proposed system is more efficient in completing an episode. Based on Section 3.5, since object 1 and object 2 in Figure 7 (left) are too close ($d_n < 0.035$ m) to each other, at least one push action (labeled as 1:push) is required before grasping (labeled as 2:grasp) object 2. As can be seen in Figure 7 (right), the proposed system learnt to grasp (labeled as 1:grasp) object 2 directly, resulting in higher efficiency.

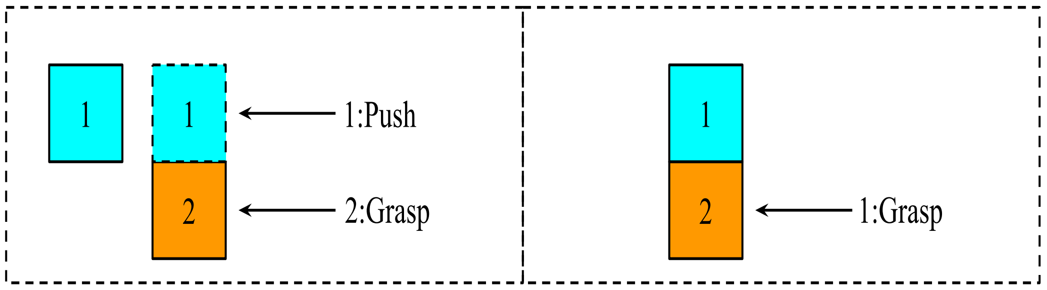

**Figure 7.** (**left**) Demonstration system procedure and (**right**) proposed system procedure.

The grasping-only baseline demonstrates the necessity of pushing actions in cluttered scenes, since some failed graspings act like pushes to separate objects. The comparison showed that context-aware decision making is important to reduce the number of decisions needed to be completed. The proposed system evolved to exceed the decision efficiency of the demonstration system, indicating the effectiveness of the proposed system design and training approach.

## *4.6. Discussion of Proposed System*

Single Depth Image Viewpoint: A single depth image from an initial static "home" viewpoint is sufficient for our system to make effective grasp and push decisions. Despite the minimalist observation model, the model demonstrates strong performance in all key metrics, such as achieving 7.8 DTC and 93.1% AGS, indicating that this sparse view approach provides adequate spatial and occlusion information to make effective grasp and push decisions, even in cluttered scenes. Depth imagery inherently captures 3D scene geometry and occlusions, enabling the system to infer object relationships without relying on temporal sequences or multiview input.

Independent ANN Output Branches: Each of the four low-level branches ($d_x$, $d_y$, $d_z$, $d_{yaw}$) selects the action with the highest Q value independently. Although this assumes independence across branches, both theoretical considerations and empirical results suggest that this design is sound and effective for the task. From a learning perspective, the reward functions for grasping and pushing (Equations (6) and (7)) directly penalise failed executions, which naturally guides the network toward selecting branch combinations that produce coherent, goal-directed movements. For example, if $d_x$ and $d_y$ are selected in conflicting directions or if dyaw is misaligned with the grasp axis, the resulting failure is penalised. Over time, the ANN learns to avoid such ineffective combinations through reinforcement.

From a theoretical point of view, the adjustment steps chosen are small and discrete (e.g., $\pm 0.015$ m in $d_x/d_y$, $\pm 0.02$ m in $d_z$, and $\pm 7.5°/15°$ in $d_{yaw}$). This granularity enables incremental corrections at each time step; closed-loop feedback can help the network recover from minor interbranch inconsistencies or non-commutative effects over multiple steps. Empirically, this independent-branch design has proven effective. As shown in Figure 3, the grasping and pushing networks achieve training success rates of 98.6% and

97.8%, respectively, indicating that interbranch coordination emerges through learning despite the assumption of independence. Moreover, this design significantly reduces the dimensionality of the action space, improving sample efficiency and training stability.

Avoid Conservative and Aggressive Decisions: During early training, high-level decisions are governed by a simple hand-designed threshold ($d_n$) as the DNN has not yet acquired sufficient competence. This threshold decides whether to grasp or push, allowing low-level action neural networks (ANNs) to first master the foundational skills. Once low-level ANNs reach a sufficient success rate, the DNN fully begins to take over high-level decision making and refine its decisions via a carefully designed reward system. The DNN receives only depth images and learns to maximise long-term rewards through reinforcement learning. A successful grasp yields a reward of +1.0, whereas a successful push yields +0.5. This reward structure inherently encourages the DNN to prefer grasping unless pushing leads to higher long-term returns (e.g., by enabling future grasps) and thereby avoids the DNN overfitting to the hand-designed threshold. Empirical results demonstrate that the DNN learns to generalise beyond the separability rule. As shown in Table 1 and Figure 7, the proposed system completes each episode using fewer high-level decisions (7.8 on average) compared to the rule-based demonstration system (11.4), which strictly follows the hard-coded threshold. This indicates that the DNN learns a more efficient and generalised policy than that encoded in the demonstration rule.

**Table 1.** Evaluation results' comparison.

| System | CR (%) | AGS (%) | DTC |
|---|---|---|---|
| Proposed System | 100 | 93.1 | 7.8 |
| Grasping-only | 100 | 86.6 | 6.1 |
| Sequential Grasping–Pushing System | 100 | 89.3 | 10.6 |
| Demonstration System | 100 | 100 | 11.4 |

The reward structure was carefully designed to strike a balance between encouraging efficient grasping while discouraging premature or risky grasp attempts that could result in suboptimal long-term outcomes. Although Equation (2) provides a reward of +0.5 for a successful push in crowded configurations, an ineffective grasp decision incurs a penalty of $-1$ (Equation (1)). This negative reward discourages the agent from repeatedly trying grasps that are unlikely to succeed. It effectively offsets the temptation to always grasp due to the higher positive reward (+1.0) and terminal bonus (+5.0), reinforcing the need to assess the object distribution before acting. The higher immediate reward for grasping (+1.0) and the terminal bonus (+5.0) encourage task completion, but the penalty for failed grasps ensures that the agent develops caution and learns to push when necessary.

Furthermore, the chosen $\tau = 0.05$ consistently produces a smooth and stable convergence of the decision neural network (DNN), as reflected in the steady reduction in decisions taken to completion (DTC) during training (Figure 6). In addition, the final system performance supports the stability of the learning process. Compared to the grasping-only system (86.6% of AGS), the proposed system can achieve 93.1% of AGS, while the DTC of the proposed system (7.8) is much lower than that of the conservative demonstration system (DTC = 11.4). Both results indicate that over-estimation bias has not seriously affected the proposed system to overly prioritise grasping or pushing.

Unnoticeable Over-Estimation Bias in ANNs: Throughout training, we did not observe instability or signs of value explosion. As shown in Figure 3, both the grasping and pushing ANNs converge to high success rates (98.6% and 97.8%), suggesting that the Q values are consistently scaled and the policy is well calibrated. The stability of the learning curves further confirms the absence of over-estimation issues.

## 5. Conclusions and Future Works

### 5.1. Conclusions

This study has proposed and implemented a hierarchical system to coordinate closed-loop grasping and pushing actions based on a depth sensor mounted on a robot arm. The DNN decides whether to perform a pushing or grasping action, while the ANNs execute high-level decisions. A curriculum-based training method was employed to train the hierarchical system, since the combination of incorrect decisions and actions seriously affects the efficiency of training. Both DNN and ANNs adopted double deep Q-learning to enable self-exploration. Behaviour cloning is integrated to accelerate the training process of ANNs, and rank loss is introduced to prioritise the demonstrated actions. A demonstration system was also designed and implemented to provide correct examples based on the object bounding box information, as described in Section 3.5. The highest success rates achieved by grasping and pushing networks are 98.6% and 97.8%, while the training completion rate is 100%, demonstrating the effectiveness of the design and training approach of the system architecture. This study also proposed three baselines for comprehensive evaluations, indicating the need to push actions in the cluttered environment to prevent damaging physical parts and the importance of content-aware decision making to reduce the number of decisions taken for completion.

### 5.2. Future Works

The limitations of this study point to several directions for future research. This study focused on 4D motions to push and grasp, which limit the adaptability of the proposed system to various scenarios. In addition to adjustments in the yaw orientation, the architecture of ANNs can be extended to include pitch and roll adjustments. Secondly, test objects are limited to square cubes in this study. Future work could expand the system to handle a variety of block shapes, such as rectangular and triangular shapes. The same training approach described in Section 3.9 can be applied to improve the effectiveness of the system. After learning to manipulate various block shapes, the proposed system can learn to manipulate irregular objects based on full self-explorations. Third, the current study only focuses on computing the gripper's 3D movements and orientation adjustments, without directly controlling the open and closed movements of the gripper. ANNs' architecture can be expanded to include open and closed options for the gripper. Lastly, in the current study, the current reward system rewards the system whenever a robot arm pushes the objects closely positioned to each other, regardless of whether the pushing action causes the objects to separate. To reinforce the agent in pushing objects for separation, an extra reward should be offered when pushing causes the separation of objects.

**Author Contributions:** Conceptualization, H.F.Y. and A.A.; methodology, H.F.Y. and A.A.; software, H.F.Y.; validation, H.F.Y.; formal analysis, H.F.Y.; investigation, H.F.Y.; resources, H.F.Y.; data curation, H.F.Y.; writing—original draft preparation, H.F.Y.; writing—review and editing, H.F.Y. and A.A.; visualization, H.F.Y.; supervision, A.A.; project administration, H.F.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The data used in this study were generated automatically through interactions between a robot arm and a simulated environment.

**Conflicts of Interest:** The authors declare no commercial or financial conflicts of interest related to this work.

## References

1. Serhan, B.; Pandya, H.; Kucukyilmaz, A.; Neumann, G. Push-to-See: Learning Non-Prehensile Manipulation to Enhance Instance Segmentation via Deep Q-Learning. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 1513–1519.
2. Zeng, A.; Song, S.; Welker, S.; Lee, J.; Rodriguez, A.; Funkhouser, T. Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 4238–4245.
3. Xu, K.; Yu, H.; Lai, Q.; Wang, Y.; Xiong, R. Efficient Learning of Goal-Oriented Push-Grasping Synergy in Clutter. *IEEE Robot. Autom. Lett.* **2021**, *6*, 6337–6344. [CrossRef]
4. Yu, H.; Lou, X.; Yang, Y.; Choi, C. IOSG: Image-Driven Object Searching and Grasping. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, MI, USA, 1–5 October 2023; pp. 3145–3152.
5. Ebert, F.; Dasari, S.; Lee, A.X.; Levine, S.; Finn, C. Robustness via Retrying: Closed-Loop Robotic Manipulation with Self-Supervised Learning. In Proceedings of the Conference on Robot Learning, Zürich, Switzerland, 29–31 October 2018; pp. 983–993.
6. Song, S.; Zeng, A.; Lee, J.; Funkhouser, T. Grasping in the Wild: Learning 6DoF Closed-Loop Grasping from Low-Cost Demonstrations. *IEEE Robot. Autom. Lett.* **2020**, *5*, 4978–4985. [CrossRef]
7. Buchholz, D.; Winkelbach, S.; Wahl, F.M. RANSAM for Industrial Bin-Picking. In Proceedings of the ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th Germain Conference on Robotics), Munich, Germany, 7–9 June 2010; pp. 1–6.
8. Choi, C.; Taguchi, Y.; Tuzel, O.; Liu, M.L.; Ramalingam, S. Voting-based Pose Estimation for Robotic Assembly Using a 3D Sensor. In Proceedings of the IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 1724–1731.
9. Yu, H.F.; Tang, X.; Chen, M. Method and Apparatus for Posture, Dimension and Shape Measurements of Objects in 3D Scenes. U.S. Patent 11321953, 3 May 2022.
10. Joshi, S.; Kumra, S.; Sahin, F. Robotic Grasping using Deep Reinforcement Learning. In Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE), Online, 20–21 August 2020.
11. Chen, P.; Lu, W. Deep reinforcement learning based moving object grasping. *Inf. Sci.* **2021**, *565*, 62–76. [CrossRef]
12. Hou, Y.; Li, J. Learning 6-DoF grasping with dual-agent deep reinforcement learning. *Robot. Auton. Syst.* **2023**, *166*, 104451. [CrossRef]
13. Mason, M.T. Mechanics and Planning of Manipulator Pushing Operations. *Int. J. Robot. Res.* **1986**, *5*, 53–71. [CrossRef]
14. Dogar, M.R.; Srinivasa, S.S. A Framework for Push-Grasping in Clutter. In Proceedings of the Robotics: Science and Systems, Los Angeles, CA, USA, 27–30 June 2011.
15. Lloyd, J.; Lepora, N.F. Goal-Driven Robotic Pushing Using Tactile and Proprioceptive Feedback. *IEEE Trans. Robot.* **2022**, *38*, 1201–1212. [CrossRef]
16. Wang, S.; Sun, L.; Zha, F.; Guo, W.; Wang, P. Learning adaptive reaching and pushing skills using contact information. *Front. Neurorobotics* **2023**, *17*, 1271607. [CrossRef] [PubMed]
17. Boularias, A.; Bagnell, J.A.; Stentz, A. Learning to Manipulate Unknown Objects in Clutter by Reinforcements. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; pp. 1336–1342. [CrossRef]
18. Zhang, H.; Liang, H.; Cong, L.; Lyu, J.; Zeng, L.; Feng, P.; Zhang, J. Reinforcement Learning Based Pushing and Grasping Objects from Ungraspable Poses. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2023), London, UK, 29 May–2 June 2023; pp. 3860–3866.
19. Danielczuk, M.; Kurenkov, A.; Balakrishna, A.; Matl, M.; Wang, D.; Martín-Martín, R.; Garg, A.; Savarese, S.; Goldberg, K. Mechanical Search: Multi-Step Retrieval of a Target Object Occluded by Clutter. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 1614–1621.
20. Mohammed, M.Q.; Kwek, L.C.; Chua, S.C.; Aljaloud, A.S.; Al-Dhaqm, A.; Al-Mekhlafi, Z.G.; Mohammed, B.A. Deep Reinforcement Learning-Based Robotic Grasping in Clutter and Occlusion. *Sustainability* **2021**, *13*, 13686. [CrossRef]
21. Shiferaw, B.A.; Shiferaw, B.A.; Srinivasagan, R. Synergistic Pushing and Grasping for Enhanced Robotic Manipulation Using Deep Reinforcement Learning. *Actuators* **2024**, *13*, 316. [CrossRef]
22. Kingma, D.P.; Ba, J.L. ADAM: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.