Article

PolyDexFrame: Deep Reinforcement Learning-Based Pick-and-Place of Objects in Clutter

Muhammad Babar Imtiaz *, Yuansong Qiao and Brian Lee

Software Research Institute, Technological University of the Shannon, Midlands Midwest, N37 HD68 Athlone, Ireland
* Correspondence: muhammad.babarimtiaz@tus.ie

Abstract: This research study represents a polydexterous deep reinforcement learning-based pick-and-place framework for industrial clutter scenarios. In the proposed framework, the agent tends to learn the pick-and-place of regularly and irregularly shaped objects in clutter by using the sequential combination of prehensile and non-prehensile robotic manipulations involving different robotic grippers in a completely self-supervised manner. The problem was tackled as a reinforcement learning problem; after the Markov decision process (MDP) was designed, the off-policy model-free Q-learning algorithm was deployed using deep Q-networks as a Q-function approximator. Four distinct robotic manipulations, i.e., grasp from the prehensile manipulation category and inward slide, outward slide, and suction grip from the non-prehensile manipulation category were considered as actions. The Q-function comprised four fully convolutional networks (FCN) corresponding to each action based on memory-efficient DenseNet-121 variants outputting pixel-wise maps of action-values jointly trained via the pixel-wise parametrization technique. Rewards were awarded according to the status of the action performed, and backpropagation was conducted accordingly for the FCN generating the maximum Q-value. The results showed that the agent learned the sequential combination of the polydexterous prehensile and non-prehensile manipulations, where the non-prehensile manipulations increased the possibility of prehensile manipulations. We achieved promising results in comparison to the baselines, differently designed variants, and density-based testing clutter.

Keywords: polydexterous; deep reinforcement learning; prehensile; non-prehensile; robotic manipulation; Markov decision process; deep Q-network; fully convolutional network; pixelwise-parameterization; DenseNet-121

check for updates

Citation: Imtiaz, M.B.; Qiao, Y.; Lee, B. PolyDexFrame: Deep Reinforcement Learning-Based Pick-and-Place of Objects in Clutter. *Machines* **2024**, *12*, 547. https://doi.org/10.3390/machines12080547

Academic Editor: Sergey Y. Yurish

Received: 12 July 2024
Revised: 5 August 2024
Accepted: 6 August 2024
Published: 11 August 2024

1. Introduction

The concept of Industry 4.0 has been prevalent for over a decade now [1]. The fourth industrial revolution, centered on automation, introduced key technologies such as Digital Twins, Cyber–Physical Systems, Smart Manufacturing, and the Internet of Things (IoT). These innovations have become integral to implementing automated processes within industries. With this revolution of industry, one can clearly see the direction and future of industries, as described by Warren Bennis [2]: "The factory of the future will have only two employees, a man and a dog. The man will be there to feed the dog. The dog will be there to keep the man from touching the equipment." The proof-of-concept of the Industry 4.0 revolution can be witnessed all over the world, for example, Europe's Industry 4.0, China's Made in China 2025, USA's Advanced Manufacturing, Japan's Super Smart Society 5.0, etc. One can witness different terminologies such as smart factories, smart manufacturing, and smart industry in the literature, which fall under the umbrella of Industry 4.0. In these domains, while achieving the goal of automation, the involvement of artificial intelligence (AI) and the utilization of robotics are quite evident.

Robotics has been a part of industry for almost five decades, evolving in various forms over time [3]. The operations usually performed by industrial robots can be divided into three categories: material handling, processing operation, and assembly and inspection [4]. Tasks included in the material handling category are material transfer and machine loading and unloading. In processing operations, robotic arms are used to perform a process on a medium such as welding and painting. In assembling and inspection operations, products are assembled, and their quality is ensured. In all these categories, the most frequent and repetitive robotic manipulation is the pick-and-place operation. In addition to pure industrial robots, even collaborative robots, also known as cobots, are specially designed to work alongside human workers, mostly to perform pick-and-place tasks. Over the years, along with pure industrial robots, collaborative robots have become the focus of attention of industry due to their ability to quickly learn on the job and trouble-free reprogramming features. It is being witnessed that with the rapid growth of industries and e-commerce, pick-and-place or bin-picking robotic arms are in demand to achieve automation in various factories and warehouses [5].

Usually, robotic manipulations are categorized into two categories: prehensile manipulation [6,7] and non-prehensile manipulation [8]. The key difference between the two is that the former implies grasping or capturing and the latter lacks the element of grasping and instead adopts pushing or propelling. Figure 1 clearly showcases the difference between both categories. Our previous work [9] showed that a well-coordinated approach between these two types of robotic manipulations can significantly enhance success rates. Therein, we found that non-prehensile manipulation such as sliding can navigate through tightly packed objects, facilitating subsequent grasping. Meanwhile, prehensile manipulation such as grasping can efficiently displace objects, ensuring collision-free and precise movements for non-prehensile manipulations, such as sliding or pushing. Previously, much research was conducted separately on prehensile and non-prehensile robotic manipulations. However, there remains a substantial research gap and ample opportunity to explore the integration of these two types of robotic manipulations that we targeted in our previous work [9].

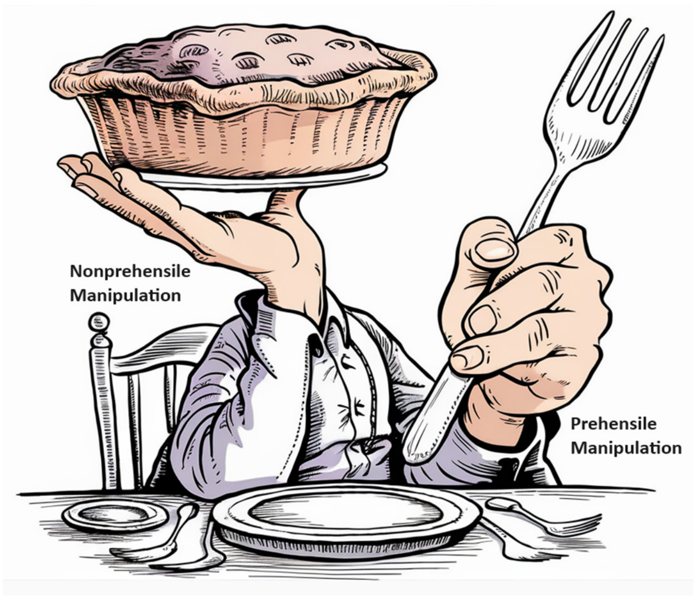

Figure 1. The difference between prehensile and non-prehensile manipulations.

When we consider the combination of prehensile and non-prehensile robotic manipulation, the role of non-prehensile manipulation can vary, for example, modifying the object poses, creating gaps between objects, breaking down a clutter of objects for further manipulation, etc. In most existing research combining prehensile and non-prehensile robotic manipulation, the varying role of non-prehensile manipulation has not been thoroughly examined, neither in model-based work [10] nor in data-driven approaches [11]. The primary difference between the model-based and data-driven is that the former relies entirely on static models, whereas the latter involves models trained through learning from relevant data. In prehensile manipulation, some approaches learn from previous experiences [12] while others rely on grasp stability measures [13]. However, most current approaches lack the ability to sequentially learn a combination of prehensile and non-prehensile manipulations, which could enhance the success rate of pick-and-place tasks.

In this research study, we extend our previous work [9] and explore the possibility of integrating previous work on the agent learning synchronization of grasping and sliding with suction-based grasping, which technically falls under the category of non-prehensile manipulation. To extend the previous work, we develop a polydexterous framework, which uses different robotic grippers on a robotic arm to perform multiple robotic manipulations. The major objectives or contributions of this research study are as follows:

- A deep reinforcement learning-based end-to-end framework, known as PolyDexFrame, enables the agent to learn the sequential combination of jaw-gripper and suction cup/pad prehensile and non-prehensile robotic manipulations, thus allowing the pick-and-place of regular- and irregular-shaped objects successfully.
- The jaw-gripper grasping, bidirectional sliding, and suction-cup-based vacuum gripping are learned together in such a sequential manner where each robotic manipulation further enhances the chances of the other manipulations.
- Optimal policy convergence is achieved through Q-learning by training joint end-to-end CNNs in a self-supervised mode with the help of a pixelwise-parameterization technique.

An important question here regards the need or reason behind extending our research by integrating the suction-based non-prehensile manipulation with the jaw-gripper-based prehensile and non-prehensile manipulations. There are several scenarios in industry and warehouses where there is clutter to be picked and placed, comprising various kind of objects; some may be deformable, soft, and flexible objects, whereas others may be rigid ones. When a regular jaw gripper faces problems in accurately grasping the deformable, soft objects, the suction cup/pad can be the right option due to its gentleness, adaptability to the soft deformable object's surface, and reduced pressure points [14]. Therefore, the agents' learning to combine the prehensile and non-prehensile manipulations of different grippers, i.e., to be polydexterous, in a sequential manner which may lead to an enhanced success rate of the pick-and-place task is highly beneficial. Therefore, the novelty of this extended research work is evident in the joint learning of the jaw-gripper grasping, bidirectional sliding, and suction-cup-based vacuum gripping for the pick-and-place of regular- and irregular-shaped clutter.

2. Related Work

In this research study, we target the amalgamation of robotic gripper-based grasping, sliding, and vacuum gripping with the help of a suction cup/pad. Therefore, we examine these three elements addressed individually thus far in the existing body of literature.

The robotic gripper-based grasping has been studied and practiced for a long time now. A lot of the literature consists of various model-based approaches. Some of these approaches identified the merit of individual grasps based on their ability to control the movement of the object [15], whereas a couple of them explored the resistive role of contact forces by modeling them in the process of grasping [16]. In some approaches, the main idea is to relate point clouds with the grasps computed beforehand dynamically on the run for the sake of the pose estimation of the objects under consideration [17]. The pre-computed grasps are mostly generated with the help of 3D object models which make the real-world adaptation much better [18]. The obvious limitation of these approaches is that a large amount of in-depth information is necessary beforehand for them to work, such as the shape of the objects, poses of the objects, pre-computed contact points on the objects, etc. But when the objects are novel and unknown, then it is impossible to meet this mandatory condition, which makes these approaches unsuitable. On the other hand, researchers have been working on data-driven approaches instead in recent years. Various data-driven approaches are evident, in which agents are trained in a model-free manner to identify the potential grasps after learning the visual features instead of requiring any in-depth information beforehand [12]. Approaches like [19] present the idea of using models pretrained to learn other robotic manipulations to be used for the learning of the desired robotic manipulation. The approach in [12] is completely deep-learning-based and tends to calculate pixelwise affordances in order to make the model learn the optimal grasping policy.

Like the history of prehensible manipulation, the relevant literature suggests that non-prehensile manipulation has also been around for a long time. The early approaches developed in the non-prehensile domain used the assistance of frictional forces in modeling [20]. Even though these approaches played a vital role in the future development of the non-prehensile research area, the modeling assumptions made during the process, such as non-uniform distribution and varying frictional forces, proved to be insufficient for real-world settings [21]. These limitations causing hurdles in adapting to real-world settings led to the exploration of data-driven approaches for the learning of non-prehensile manipulations [22]. Most of the techniques presented in these approaches revolved around dealing with a single object. However, exploring dense clutter remains a complex challenge in this domain.

The vacuum gripping with the help of a suction cup/pad is one of the widely deployed non-prehensile manipulations in warehouses and industries [23,24]. In the three popular events of the Amazon Picking Challenge [25], most approaches were based on vacuum gripping using a suction cup/pad. The Amazon Picking Challenge 2015 featured more than half of its approaches using suction grippers, including the winning team [25], which achieved suction gripping by pushing objects to the sides and walls. In [26], the authors utilized 6D pose estimation to perform vacuum gripping using a suction gripper near the center of the object surface. In [27], both 6D models and manual heuristics were employed for suction gripping at potential grasp coordinates. Fully convolutional networks were used by [12] to link visual inputs to the probabilities of suction grasps. In [28], the authors used two different masks to achieve grasp estimation from a single depth image. These masks represented the contact and collision regions, respectively. Another approach, [29], involved training convolutional neural networks to predict the grasp quality by classifying potential suction grasps in cloud points. A dataset comprising 2.8 million point clouds, suction grasps, and relevant labels was created for this purpose. Additionally, double-stream convolutional neural networks have been used to calculate suction point probabilities in clutter, with the aid of manual annotations [30]. Despite these advancements, most deep-learning-based approaches still rely on data manually annotated by humans.

The research area combining prehensile and non-prehensile robotic manipulations has not been fully explored yet, but there is no doubt about its efficacy. The study presented in [8] can be seen as a pioneering model-based approach that tried to combine prehensile and non-prehensile manipulations. It performs the combination of push–grasp, where the grasping is achieved while the pushing through clutter, with the hope of increasing the chances of grasping. However, this approach requires a large amount of relevant in-depth information beforehand, whereas the proposed approach in this research study requires absolutely no information beforehand since it learns in a self-supervised mode. Some other combination-based approaches [11] have also been presented where the grasping pose and locations are predefined, and the role of the non-prehensile pushing is to bring the objects to these pre-designed grasp locations. In these approaches, in addition to the limitation of a static grasping part, the pushing part is also not learned from scratch. A reinforcement-learning-based approach [31] has also been presented using handcrafted features to make the agent learn and select from prehensile and non-prehensile manipulations to pick-and-place the clutter. In this approach, object detection relies on the segmentation of RGB-D images, after which potential manipulations/actions are sampled for each detected object, and for each sampled action, handcrafted features are extracted before performing that manipulation. The downside of this approach is that it is only suitable for clutter composed mostly of convex objects. Another limitation of the approach is the requirement for prior information regarding the pushing actions, such as motion prediction through a simulator. Using the reinforcement learning, some singulation techniques [32] have also been presented, restricted to regular shapes and limited non-prehensile manipulation.

In the research area of combining prehensile and non-prehensile robotic manipulations, little has been carried out in combining suction-based gripping non-prehensile manipulation with jaw-gripper-based prehensile and non-prehensile manipulations. The proposed approach in this paper addresses this research gap. This work is essentially an extension of the research conducted in [9], which combined jaw-gripper-based prehensile and non-prehensile manipulations, and it extends the work in [7], where the pixelwise-parametrization technique was adopted for prehensile manipulation. The role of Q-learning in this work prior to adopting the pixelwise-parameterization technique was presented in [33,34].

3. Background

In this section, we briefly review the basic concepts of the cutting-edge technologies of deep learning, reinforcement learning, and their amalgamation, known as deep reinforcement learning, involved in this end-to-end data-driven approach.

3.1. Deep Learning

Deep learning is a very well-known member of the machine-learning family because of its exceptional performance in various domains. Deep learning is not limited to computer vision problems, its application can be found in many fields, including climate science, medical science, bioinformatics, drug design, material inspection, machine translation, natural language processing, speech recognition, social network filtering, etc. Deep learning has even outperformed human experts in board games. In other words, deep learning has introduced the world to many new ways of data processing, analysis, and manipulation. The word “deep” in deep learning refers to the presence of multiple layers in a single network. Whereas in machine learning, feature extraction from input data is performed manually by experts, deep learning extracts higher-level features automatically as the input data pass through the multiple layers of the deep neural network, as shown in Figure 2. Here, input data refers to raw images. Usually, large volumes of data are required in deep learning to achieve maximum efficiency and optimal results. Because it deals with large volumes of data, it requires substantial computational power. Deep models can be seen as artificial neural networks involving deep structures. The idea of the artificial neural network can be traced back to the early 1940s [35]. Since then, several turning points and landmarks have been achieved, with the introduction of perceptron, backpropagation method, rectified linear unit, max-pooling, dropout, batch normalization, etc. This series of algorithmic advances, the availability of large-scale data, and the emergence of highly efficient parallel computing entities such as Graphics Processing Units (GPUs) have contributed to the success of deep learning today. The first major achievement in the field of deep learning was achieved by a convolutional neural network (CNN) performing classification after training in 2012 [36]. It was trained on hundreds of thousands of labelled images, using loss computation and backpropagation to learn the parameters. Since then, this method has seen many improvements and can be considered the most popular deep learning approach; however, it is not well-suited for learning robotic manipulations. The primary reason for its unsuitability for learning robotic manipulation is its very high time requirement, which necessitates a large number of labelled images of joint angles for training. Nevertheless, researchers are exploring alternatives [37], and deep reinforcement learning, which will be discussed later, appears to be more efficient and promising.

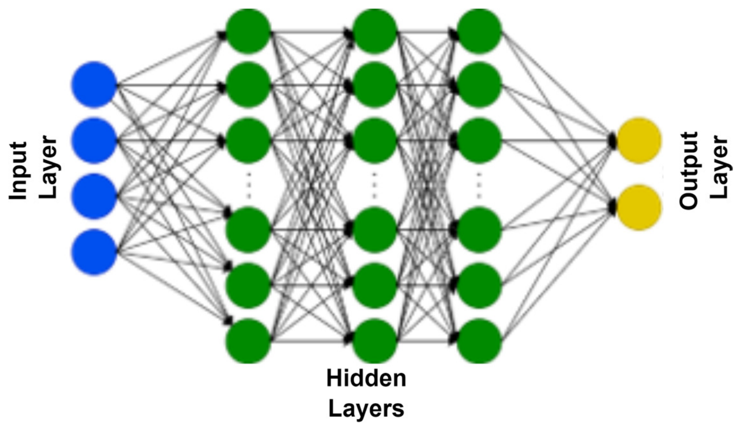

Figure 2. The flow diagram of a deep neural network.

3.2. Reinforcement Learning

Reinforcement learning is one of the subdomains of the machine-learning field [38]. It focuses on finding behaviors that maximize rewards through a process of trial and error. This mirrors how humans and animals learn: by perceiving and processing information, storing it as knowledge, and adapting their behavior in response to changing circumstances. This self-supervised technique is being deployed in several fields such as statistics, information theory, system optimization, recommendations systems, operations research, game theory, and control theory [39]. Natural reinforcement learning can be seen in living beings through punishment or reward, thus reinforcing positive behavior and suppressing negative behavior. In the same manner, an agent starts in a state, takes an action according to its current policy, receives a reward from the environment, ends up in a new state, chooses a new action, and continues iterating through this procedure. For a better understanding, this cycle is shown in Figure 3. Simply put, it is like playing a video game in which a character performs actions to maximize their score. Reinforcement learning is not like supervised learning, where labeled datasets are available for training the agent. When deploying reinforcement learning in interactive use cases, such as robotic control problems, labeled data or depictions of all potential actions are often scarce. Therefore, the concept of rewards is introduced. Reinforcement learning is also not like unsupervised learning, where we identify patterns or structures in unlabeled datasets. Instead of finding patterns and structures, the agent gradually learns to behave in the environment based on future rewards. Therefore, it can be termed as the third paradigm of machine learning, next to supervised and unsupervised learning [40].

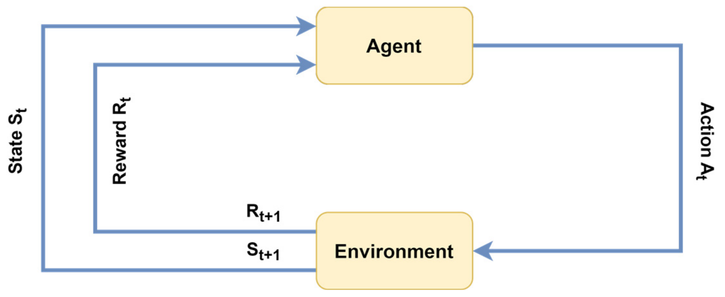

Figure 3. The working of a reinforcement-learning-based agent.

3.3. Deep Reinforcment Learning

As evident from the name, deep reinforcement learning is an amalgamation of reinforcement learning and deep learning. From reinforcement learning, it borrows the technique of awarding rewards to agents based on the actions they chose to perform, and from deep learning, it uses the idea of learning features through the deployment of neural networks. While traditional reinforcement learning is suited for environments with simple state representations, deep reinforcement learning enables agents to learn from unstructured and multi-dimensional inputs using neural networks. A general outlook of a deep reinforcement learning-based approach is shown in Figure 4. In the last few years, as deep learning has gained popularity, deep reinforcement learning has also made significant progress in various domains. Deep reinforcement learning gained much of its popularity through its role in games research. In 2013, DeepMind produced significant results in the Atari game without relying on hand-coded features. Later, in 2016, AlphaGo made headlines by defeating a human expert in the ancient Chinese game Go. Its improved version, AlphaZero, produced in 2017, demonstrated proficiency in learning features in shogi and chess as well. In 2019, Pluribus and OpenAI Five outperformed top professionals in multiplayer poker and defeated the defending world champion in Dota 2, respectively. Beyond games, deep reinforcement learning holds potential in other domains such as robotics, computer vision, natural language processing, finance, healthcare, and transportation. Many tech giants, academic labs, and non-profit research organizations have focused on applying deep reinforcement learning and have published several breakthroughs. The widely used deep reinforcement learning algorithms can generally be classified into three categories: value-based, policy gradient, and model-based methods. In the first category, value-based methods, the policy is defined by a value function based on an algorithm using the Bellman equation, such as Q-learning or its variant, fitted Q-learning. The successful implementation of the deep Q-network (DQN) and its various extensions, including double DQN and distributional DQN, are examples of this category. The performance of these methods, in terms of final results and data efficiency, is benchmarked against the Atari 2600 environment in [41]. Deep Q-network (DQN) approaches handle problems with discrete action spaces and deterministic policies. On the other hand, policy gradient methods address problems with continuous action spaces and stochastic policies by finding a policy based on neural network parameters to increase the expected cumulative reward. Policy gradient methods require an estimate of a value function for the current policy, like other policy methods, and deploy actor–critic architectures through a sample-efficient approach, enabling the use of off-policy data. An example of such a method is the Deep Deterministic Policy Gradient (DDPG). Research on combining Q-learning and policy gradient methods is also ongoing. Both value-based and policy-based methods face limitations in sample efficiency because they do not utilize the environmental model, making them model-free approaches. In contrast, model-based methods learn an environmental model through experience, using approximation functions or pre-supplied models, in conjunction with a planning algorithm. Studies have also explored combining model-based and model-free methods to leverage the strengths of both approaches.

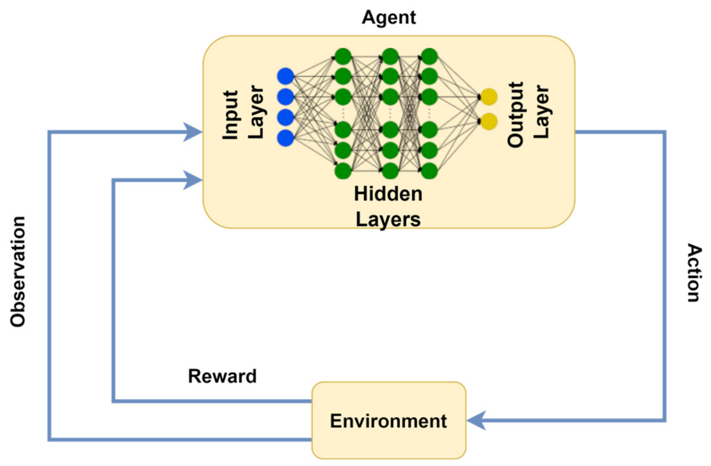

Figure 4. The block diagram represents the working of a deep reinforcement learning-based agent.

4. Methodology

In this research study, we develop a polydexterous robotic pick-and-place framework. The reason behind terming it a polydexterous framework is that "poly" means many and "dexterous" means skill with the hand. Thus, a polydexterous robotic arm can be seen as a robotic arm performing prehensile and non-prehensile manipulations while involving more than one robotic gripper, which is the case in this proposed approach.

In this research paper, we design a pick-and-place framework for variously shaped objects, including regular and irregular ones. The task is modeled as a deep reinforcement learning problem, where the agent tries to learn the skill of the pick-and-place of regular- and irregular-shaped objects from a conveyor belt through a continuous sequential combination of prehensile and non-prehensile robotic manipulations with the help of two robotic grippers. We train an agent through reinforcement learning to learn the pick-and-place task. An agent here means an abstract entity that iteratively learns a policy to maximize its expected reward, thereby mastering the task. As the task is to be resolved as a reinforcement learning problem, we formulate a Markov Decision Process (MDP), which is a quintuple process comprising states, actions, transition function, rewards, and a discount factor. Thus, according to the MDP flow, an agent (the robotic arm in our case) at any time $\boldsymbol{t}$, residing in the initial state $\boldsymbol{s_t}$, takes an action $\boldsymbol{a_t}$ while following the policy $\boldsymbol{\pi}$, and ends up in the next state $\boldsymbol{s_{t+1}}$. For this transition from the initial state to the next state, it also earns some reward, denoted by $\mathbf{r_{at}(s_t, s_{t+1})}$. Thus, in terms of MDP, the actual goal here for the agent is to find the optimal policy $\boldsymbol{\pi^*}$ which can lead the agent to reach the goal of maximum reward, depicted by $\boldsymbol{R_t} = \boldsymbol{\Sigma_{i=t}^{\infty}} \gamma \boldsymbol{r_{ai}}(s_i, s_{i+1})$. Here, the γ represents the discount factor, valued between 0 and 1, which helps the agent to learn with more speed.

This approach is essentially a self-learning, data-driven scheme where the agent (the UR5 robotic arm we are deploying) tries to predict the most suitable pixel or coordinate for the relevant robotic manipulation to be performed. As soon as the agent becomes successful in predicting the accurate pixel for the manipulation, the overall pick-and-place process by becoming proficient in the sequential synchronization of the prehensile and non-prehensile manipulations also becomes successful, as evident from the results. In this approach, we deployed the off-policy, model-free Q-learning algorithm to enable the agent to learn the pick-and-place task through prehensile and non-prehensile robotic manipulations. The Q-learning agent tends to learn the optimal policy in a greedy manner, always choosing the action with the maximum action-value. These action-values are calculated through the Q-function, which can be depicted as $\boldsymbol{Q_\pi(s_t, a_t)}$. The working of a Q-function can be summed up as in Equation (1).

$$\boldsymbol{Q}(\boldsymbol{s_t}, \boldsymbol{a_t}) \leftarrow \boldsymbol{Q}(\boldsymbol{s_t}, \boldsymbol{a_t}) + \boldsymbol{\alpha}\left[\boldsymbol{R_{at}}(\boldsymbol{s_t}, \boldsymbol{s_{t+1}}) + \gamma \boldsymbol{Q}\left(\boldsymbol{s_{t+1}}, \boldsymbol{argmax}\left(\boldsymbol{Q}\left(\boldsymbol{s_{t+1}}, \boldsymbol{a'_t}\right)\right) - \boldsymbol{Q}(\boldsymbol{s_t}, \boldsymbol{a_t})\right)\right] \quad (1)$$

where the $\boldsymbol{\alpha}$ and $\boldsymbol{a'_t}$ represent the learning rate and the directory of all the action options available at the time.

To understand the flow of the approach, the main three elements of the MDP are described as follows:

- **States:** A state ($\mathbf{s} \in \mathbf{S}$) is the current snapshot of the agent's environment. In our case, states are represented graphically. Each state is depicted as an RGB-D heightmap, which shows the graphical view of the agent's environment. As the UR5 robotic arm is our agent, the conveyor belt containing the regular and irregular objects is seen in its surrounding environment at any given time. This surrounding environment can also be called the workspace of our agent within which it can observe and perform actions. Since the state is designed graphically, vision sensors are needed to capture the current state of the UR5 robotic arm's workspace. We installed two kinds of vision sensors: an orthographic vision sensor and a perspective vision sensor. These sensors have different fields of view: the orthographic vision sensor's field of view is rectangular, while the perspective vision sensor's field of view is trapezoidal. The reason for using these two different vision sensors is to install them at fixed positions, as shown in

Figure 5, and then merge their field of views to collect the maximum amount of RGB and depth information from the environment. Both vision sensors capture their own individual RGB-D images, which are later amalgamated into one image. As the name suggests, the RGB-D image contains the color information (R, G, and B channels) as well as the depth information (D channel). The depth information represents the distance of objects' surfaces from a viewpoint. Since our robotic manipulation is from the top angle, the depth data can be considered elevation or height-from-bottom data. As stated above, the state is represented in RGB-D heightmap format. The RGB-D image is translated to the corresponding RGB-D heightmap using the same scheme as we used before in [9]. As shown in Figure 6, after generating and back-projecting the 3D point cloud from the input RGB-D data in an orthographic manner, the RGB-D heightmap is procured. The size of the generated RGB-D heightmap corresponds to the actual workspace of the agent, which is equivalent to 224^2 pixels in our case. By this design, each of the 50,176 pixels in the RGB-D heightmap corresponds to a distinct 3D location in the physical workspace of the agent designed in the simulator.

- **Actions**: An action ($a \in A$) is a step chosen from the available list performed by the agent after observing the environment, following the policy to maximize its expected future reward. In our case, the actions are the prehensile and non-prehensile robotic manipulations performed by the UR5 robotic arm agent to pick-and-place regular and irregular objects in clutter. We consider four different robotic manipulations: grasp, inward-slide, outward-slide, suction-grip. These actions cover both prehensile and non-prehensile robotic manipulation. In our setup, two robotic grippers are involved: the jaw gripper and the suction cup. The grasp is the prehensile manipulation of the jaw gripper, while the inward-slide and outward-slide are its non-prehensile manipulations, as shown in Figure 7. The suction-grip is the non-prehensile robotic manipulation performed by the suction cup, as shown in Figure 7. In summary, there is one prehensile manipulation and three non-prehensile manipulations that the agent can learn to synchronize such that each robotic manipulation increases the chances of successfully picking and placing objects.
  At any given time, the agent can select a pixel from the 224^2 sized RGB-D heightmap. As discussed before, each pixel corresponds to a particular 3D location on the robotic arm's workspace on the conveyor belt represented as $pixel \rightarrow 3D\ location \in S_t$. This 3D location on the workspace has different roles depending on the action. For instance, in the case of grasping action, it will be considered the mid-point of the grasp. In the inward- or outward-slide, it will be the start point of the linear push. In the suction-grip, it is the contact point. According to the design setup, some static modifications are made to the runtime when performing the action in the given 3D location. For instance, because we have vision sensors installed at top angles and the pick-and-place is performed from the top angle, the z-coordinate of the predicted 3D location on the workspace is incremented by around 2 cm in the cases of grasp and suction-grip. The increment ensures that the object can be grasped from center point, and in the case of suction gripping it helps to create vacuum suction pressure inside the suction cup, leading to a stronger grip. In non-prehensile manipulations, the direction of the slide is towards the robotic arm base in the inward-slide and away from the robotic arm base in the outward-slide. For these sliding non-prehensile manipulations, the straight linear push limit is kept at 8 cm.
- **Rewards**: The reward ($r \in R$) in the reinforcement learning is the incentive that the agent receives when it transitions from one state to another by choosing the appropriate action. The goal of the agent is to maximize its expected future reward. In our case, we have a reward scheme consisting of discrete reward values. For a successful grasp and placement, the agent is awarded the full reward, which is 1. If the grasp is successful but the placement fails, meaning the object slips from the gripper before placement, the reward is reduced to 0.8. To determine whether the object has slipped from the jaw gripper, a ray-type infrared proximity sensor is installed between the jaws of

the gripper. For successful inward- or outward-slide actions, the reward is 0.5, with success measured by comparing the workspace before and after sliding. If changes are detected, the action is considered successful, and a reward is awarded. The suction-grip action has the highest chance of success due to the lower probability of slipping, and therefore it is rewarded lightly at 0.2. In any scenario other than the ones stated above, the agent is rewarded 0. This reward scheme encourages the agent to prioritize actions that lead to the successful grasping and placing of objects while also providing feedback for partial success and less optimal actions.

- **Q-Function**: In reinforcement learning, the Q-function generates action-values representing the expected future reward an agent will receive if it takes a specific action in a given state and transitions to the next state. In simple problems, this can be maintained as a Q-table, but in more complex problems, such as ours, neural networks are used for function approximation. Since we are deploying an off-policy model-free Q-learning algorithm and using deep neural networks, this approach is referred to as deep Q-networks (DQNs) for transition function approximation [42,43]. In our approach, we design the Q-function using multiple fully convolutional networks (FCN) [44]. The goal is to enable the agent to learn the synchronization among the four distinct robotic manipulations to enhance the success rate of the picking and placing variously shaped objects in clutter. To achieve this, we deploy a distinct FCN for each robotic manipulation, allowing the agent to learn the important features of all the robotic prehensile and non-prehensile manipulations in substantial depth. We designed four FCNs: $\text{FCN}_{\text{Grasp}}$, $\text{FCN}_{\text{Inward-Slide}}$, $\text{FCN}_{\text{Outward-Slide}}$, and $\text{FCN}_{\text{Suction-Grip}}$. Each of these FCNs receives the state as input, represented as an RGB-D heightmap of the workspace, sized as 224^2 pixels. All four FCNs are fed the same RGB-D heightmap simultaneously at any given time. This approach uses the pixelwise-parameterization technique, where processing is performed at the pixel level. Consequently, the output generated by each FCN is a 224^2-sized pixelwise map of values. The input and output are the same size because of this technique. Thus, we supply the same RGB-D heightmap to all four FCNs and receive four distinct pixelwise maps of the same size. The values in these pixelwise maps are the action-values on which the agent bases its decision for the current state. Each value in the pixelwise map predicts the expected future reward if the agent performs the particular action at the corresponding 3D location of the predicted pixel in the workspace. This setup allows the agent to evaluate and choose actions that maximize its expected reward, improving its effectiveness in performing pick-and-place tasks.

All four distinct FCNs share the same neural architecture, specifically DenseNet-121 [45]. The reason for adopting the DenseNet architecture is due to the connections between all layers. In a traditional FCN with n number of layers, each layer is only connected to its next layer, resulting in n number of connections. However, in DenseNet, a network with n layers has $n(n+1)/2$ total connections, meaning that each layer receives features from all preceding layers and transmits features to all succeeding layers. This enhances feature reuse and propagation, reduces the vanishing gradient problem, and minimizes the number of parameters. Despite these advantages, the increased connectivity and depth can lead to an exponential quadratic growth of the features, causing a bottleneck. To address this vulnerability, we use a memory efficient implementation of DenseNet which will be detailed in the next section. DenseNet-121 was trained and tested in two settings: from scratch and pretrained on ImageNet [36]. The DenseNet-121 architecture was extended by adding two pointwise convolution layers and batch normalization operations, with ReLU as the activation function. Our approach employs a multimodal architecture as shown in Figure 8. Each FCN has two separate trunks: one for color data (RGB) and one for depth data (DDD). The depth channel, originally a single, is cloned to match the three color channels using the method used in [9]. Later, these trunks merge, passing their learned features into the extended pointwise convolutional layers.

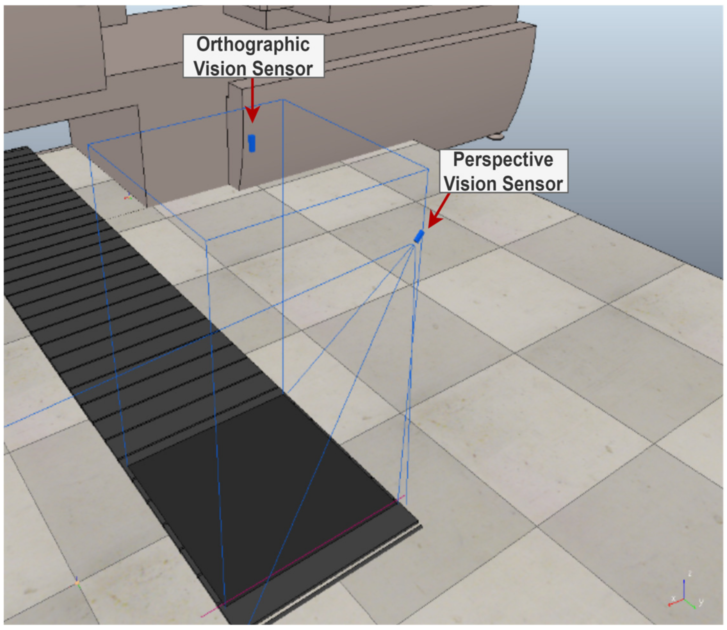

Figure 5. Orthographic and perspective vision sensors' field of view.

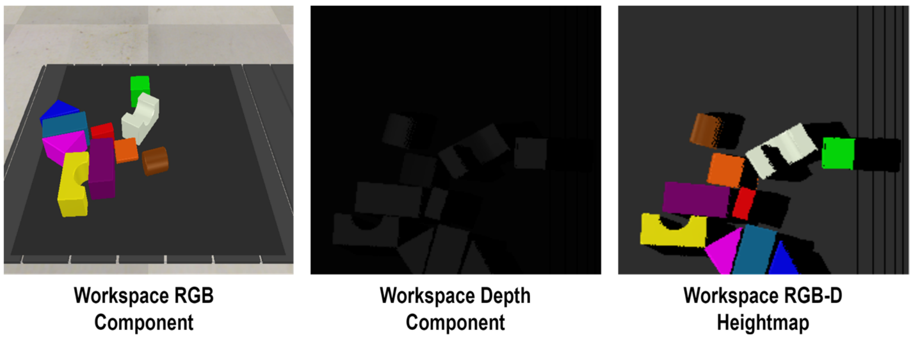

Figure 6. Generation of RGB-D heightmap from the RGB and depth components.

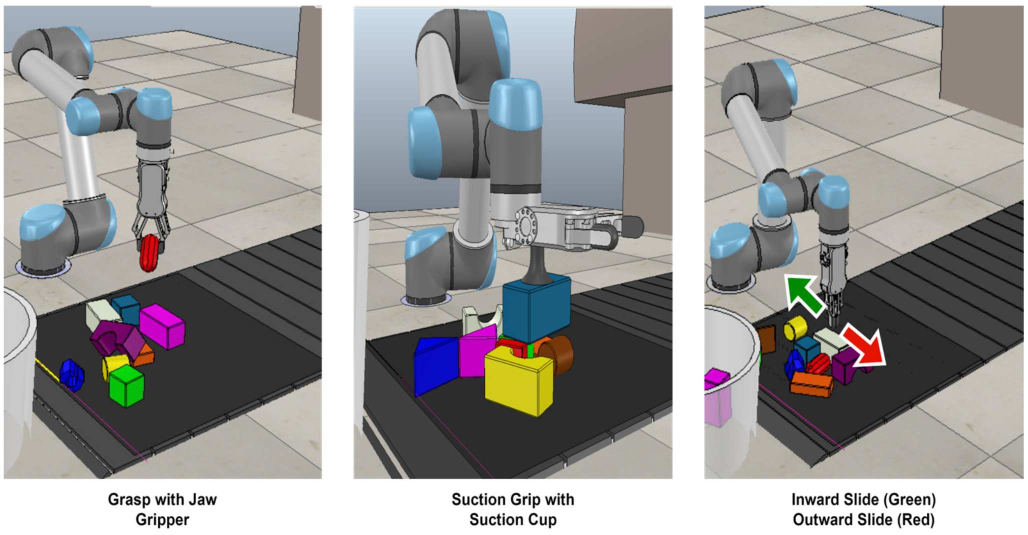

Figure 7. Examples of robotic arm performing grasping, suction-gripping, inward-slide, and outward-slide.

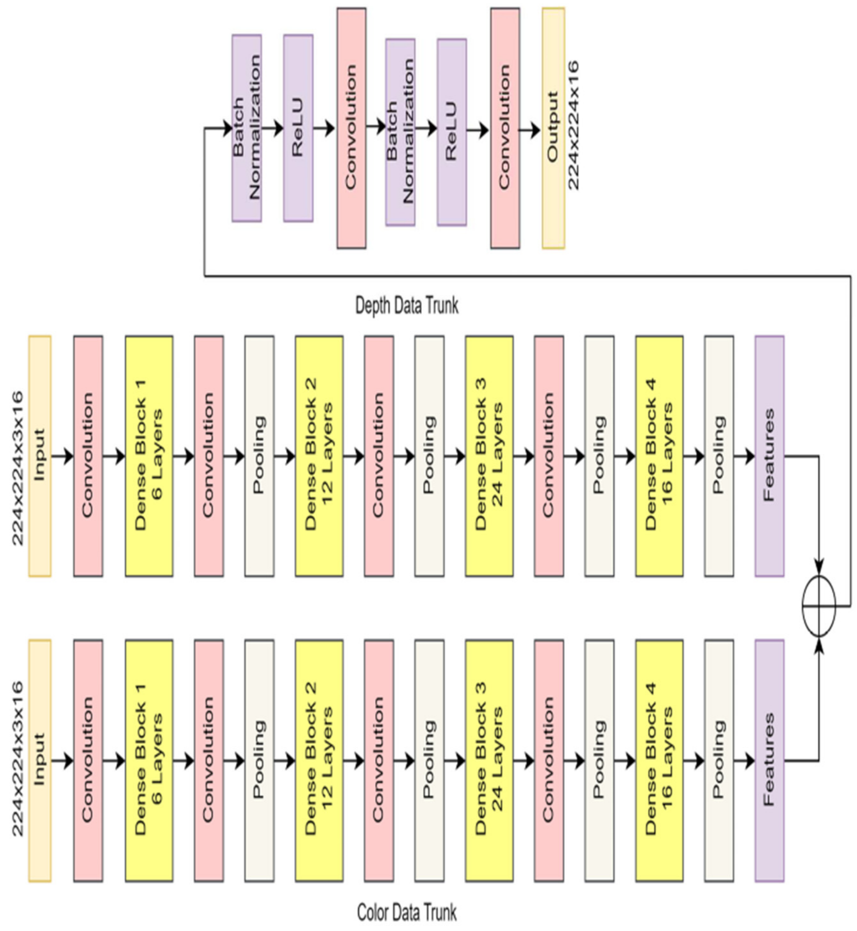

Figure 8. The block diagram of the multimodal extended DenseNet-121.

In our previous work, we have employed a scheme of rotations of the current state view to enhance feature learning. We apply the same rotation technique in this case too. To compare the performance, we rotated the RGB-D heightmap by 90°, 45°, and 22.5° as shown in Figure 9. These rotation settings generate 4, 8, and 16 rotations, respectively, for each RGB-D heightmap state. Similar to our previous results, the maximum feature learning was noted at the rotation setting of 22.5°. Adopting the 16-rotation setting means that each of the FCNs is supplied with 16 rotated versions of the RGB-D image, resulting in 16 pixelwise maps generated by each FCN. In one single iteration of the FCNs, 64 pixelwise maps of action-values are produced, with 16 pixelwise maps for each action: grasp from the $\text{FCN}_{\text{Grasp}}$, inward-slide from the $\text{FCN}_{\text{Inward-Slide}}$, outward-slide from the $\text{FCN}_{\text{Outward-Slide}}$, and the suction-grip from the $\text{FCN}_{\text{Suction-Grip}}$. Each pixelwise map contains 50,176 (224^2) action-values, resulting in a total of 802,816 (50,176 $\times$ 16) action-values for each action. The decision of which action to perform out of four is made by the agent by selecting the maximum action-value from all 3,211,264 (50,176 $\times$ 16 $\times$ 4) action-values combined. In other words, the corresponding action is performed at the 3D location corresponding to the pixel predicted with the maximum action-value/Q-value out of all the 64 pixelwise Q-values maps, as expressed in Equation (2).

$$\mathbf{max}\boldsymbol{Q}(\boldsymbol{s_t},\boldsymbol{a_t}) = \mathbf{max}\big(\boldsymbol{FCN_{grasp}}(\boldsymbol{s_t}),\ \boldsymbol{FCN_{Inward-Slide}}(\boldsymbol{s_t}), \boldsymbol{FCN_{Outward-Slide}}(\boldsymbol{s_t}), \boldsymbol{FCN_{Suction-Grip}}(\boldsymbol{s_t})\big) \tag{2}$$

With the help of the pixelwise-parameterization technique along with the rotation scheme described above, our approach enables the agent to learn at an optimal learning rate with a minimal amount of training data required.

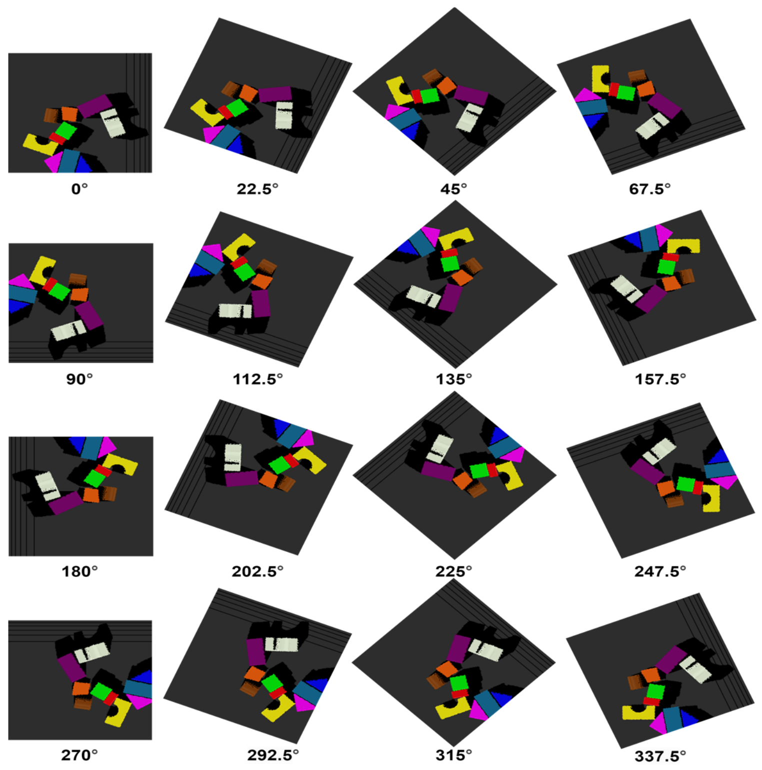

Figure 9. RGB-D heightmap rotations by 22.5°.

4.1. Training Specifications

In this section, we describe the process step by step to provide a better view of the training cycle, as shown in Figure 10. The cycle starts with regular and irregular objects entering the workspace on the conveyor belt. A ray-type infrared proximity sensor installed at the workspace detects the objects, and the vision sensors capture the current situation of the workspace in the form of RGB-D images, which are processed to generate the RGB-D heightmap. This RGB-D heightmap is passed to the rotation scheme function, which takes the RGB-D heightmap as the input, performs rotation at 22.5°, and produces 16 rotations of the RGB-D heightmap. These 16 rotations are fed to all four FCNs, and they generate the 16 pixelwise maps of action-values/Q-values for the agent. To represent these 3,211,264 (50,176 $\times$ 16 $\times$ 4) action-values, we use a heatmap representation as shown in Figure 11, which can compactly sum up the huge amount of data in less space and highlight the higher Q-values by showing them as hot regions in orange–red shades. In Figure 10, the approach is shown as a top-down tree structure. The RGB-D heightmaps from color and depth data act as the root of the tree, and the next level shows four nodes of the tree representing the 16 pixelwise maps of Q-values presented in heatmap format. The leaves of this tree show the 3D locations corresponding to the predicted pixels with the highest Q-values on the workspace (represented by the red dots). Each red dot represents the highest Q-value 3D location for the corresponding action to be performed. At this stage, the Q-values of all these four 3D locations/pixels are compared, and the one with the maximum value is chosen to perform the corresponding action.

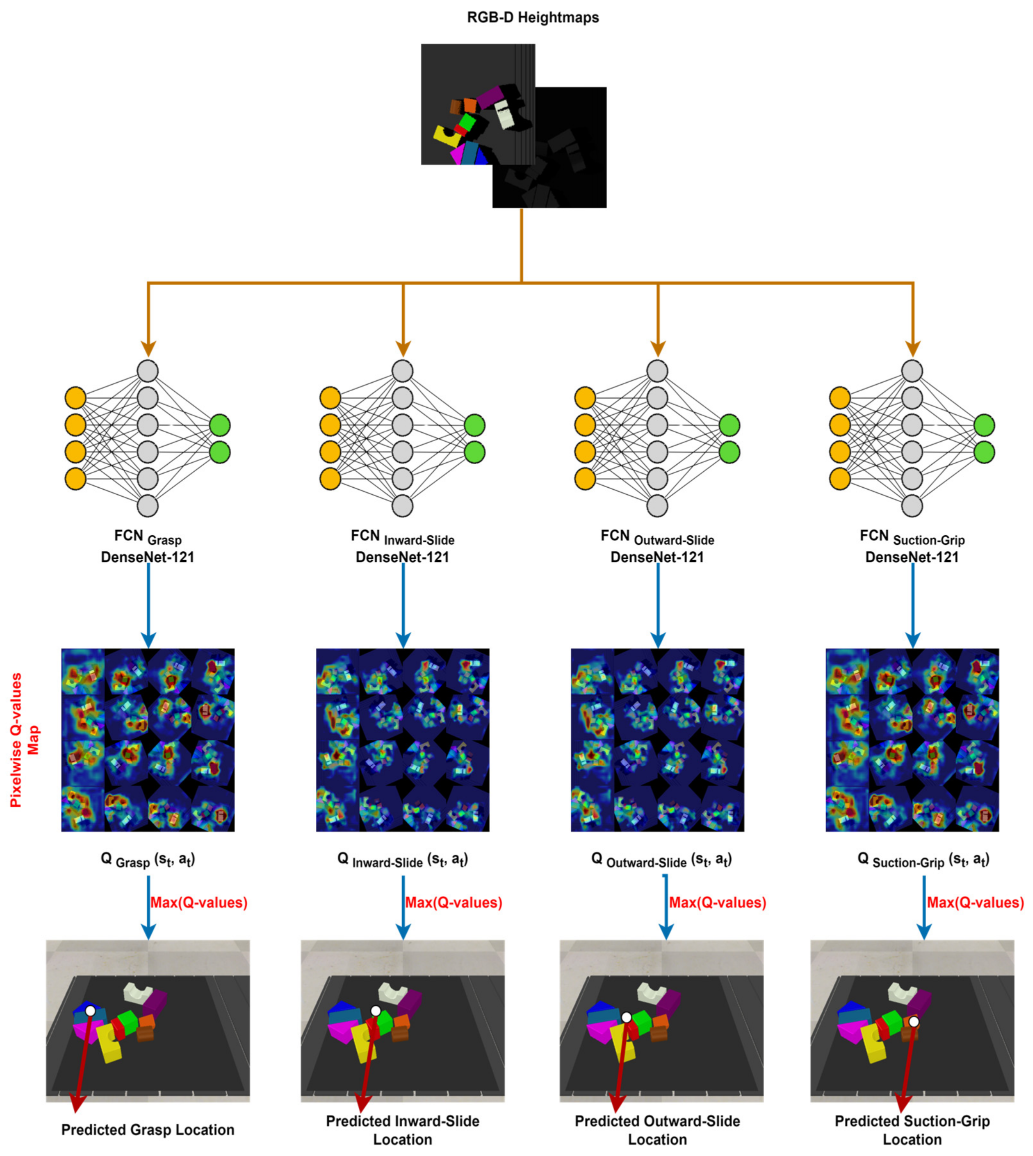

Figure 10. The flowchart diagram of the proposed approach.

As we discussed for the forward pass, backpropagation is also crucial for the weight tuning. To leverage the benefits of two widely used loss functions, i.e., mean square error (MSE) and mean absolute error (MAE), we deployed the Huber Loss function [46]. During the backpropagation, gradient passing is performed only for the FCN that predicts the pixel with the maximum Q-value. For the remaining pixels, zero loss is backpropagated. For the optimizer, the stochastic gradient descent with momentum (SGDM) variant [47] is used in this approach instead of basic SGD. The reason behind choosing SGDM over SGD is its ability to provide more stable values, being closer to the actual loss due to exponential weighted averages. This helps the agent to obtain faster convergence. In the training hyperparameters, the momentum is kept at 0.9, the weight decay is set to 1×10^{-3}, and the learning rate is set to 1×10^{-2}. An epsilon value is defined to address the exploration vs. exploitation dilemma and it is kept between 0.1 and 0.4, with a gradual decrease. In terms of hardware specifications, our system is equipped with an Intel® Core™ i7-9700K processor, 16 GB DDR3 RAM, and a GeForce RTX 2080 GPU with 8GB memory capacity. For the development, training, and testing of the FCN models, we used PyTorch version 2.2.1.

As discussed before, the reason behind selecting the DenseNet architecture is its design of maximum connections between layers, which leads to maximum feature reuse and propagation while diminishing the vanishing gradient problem. However, this design can make the system vulnerable to exponential quadratic growth of the features. This potentially causes a bottleneck situation that chokes the network. To remedy this vulnerability, we implement a memory-efficient variant of DenseNet-121 [48] instead of the base variant. In this variant of DenseNet-121, a memory sharing scheme is employed for all operations such as batch normalization, concatenation, gradients, etc. Temporary storage buffers are introduced to accommodate the outputs generated after these operations in the network, instead of being allocated new memory every time.

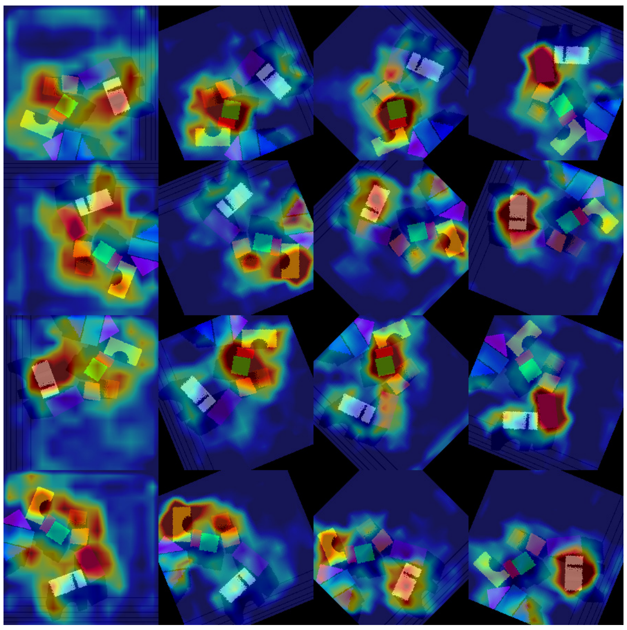

Figure 11. Visualization of 16 pixelwise Q-values maps through heatmap representation.

In the training phase, our agents learn through self-supervision. In each training iteration, the workspace has a certain number (=<10) of regular- and irregular-shaped objects placed in a random clutter pattern, as shown in Figure 12. The agent's task is to clear the workspace by picking and placing all the objects. The iteration is marked complete as soon as the workspace is cleared, and then the next iteration starts with another random clutter of regular- and irregular-shaped objects in the workspace, and the cycle continues.

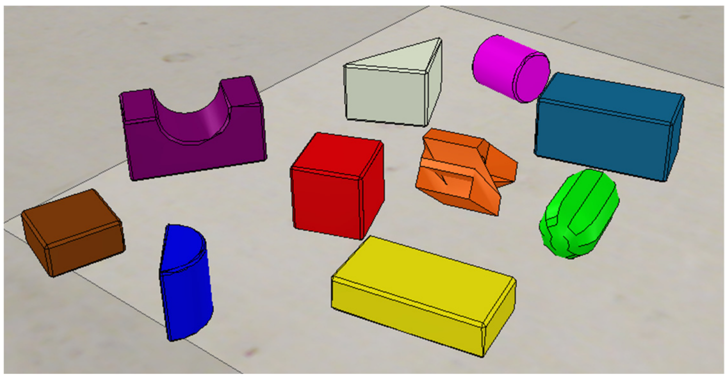

Figure 12. Regular- and irregular-shaped 3D objects.

4.2. Simulation Setup

For the experimental setup design, we chose the virtual robot experimentation (V-REP) robotic simulator. This simulator not only provides coding options but is also equipped with physics engines that simulate real-world physics effects, such as ODE, Bullet, etc. Multiple different programming platforms, such as Python and C++, can be used with the simulator from outside through threaded and non-threaded Lua scripts inside the simulator. In our case, we implemented the DRL side in PyTorch in Python externally and then communicated with the simulated setup inside the simulator using Lua scripting. Moreover, in addition to the programming support, it also includes various planning modules, such as motion planning and path planning. It features both forward and inverse kinematics modules. The difference between the two is that the forward kinematics module takes all joint angles as the input and provides the end-effector coordinates as the output, while the inverse kinematics module does the opposite. It takes the end-effector position as the input and calculates all the joint angles. In our scenario, we deployed the inverse kinematics module. Another crucial module provided by the simulator is the collision detection and avoidance module, which highlights if any part of the robotic arm collides with the environment. In our designed simulation, we used the popular 6-degree-of-freedom cobot UR5 with two grippers: the RG2 jaw gripper and suction-cup gripper, as shown in Figures 13 and 14. The RG2 jaw gripper handles the grasp, inward-slide, and outward-slide manipulations, while the suction-grip manipulation is performed by the suction-cup gripper. As shown in Figure 15, our simulated experimental setup includes a conveyor belt, on which regular- and irregular-shaped object clutter moves and enters the workspace. The objects are detected by the vision sensors and a proximity infrared sensor, initiating the pick-and-place process. Once the workspace is cleared, the next clutter reaches the workspace, and the cycle continues.

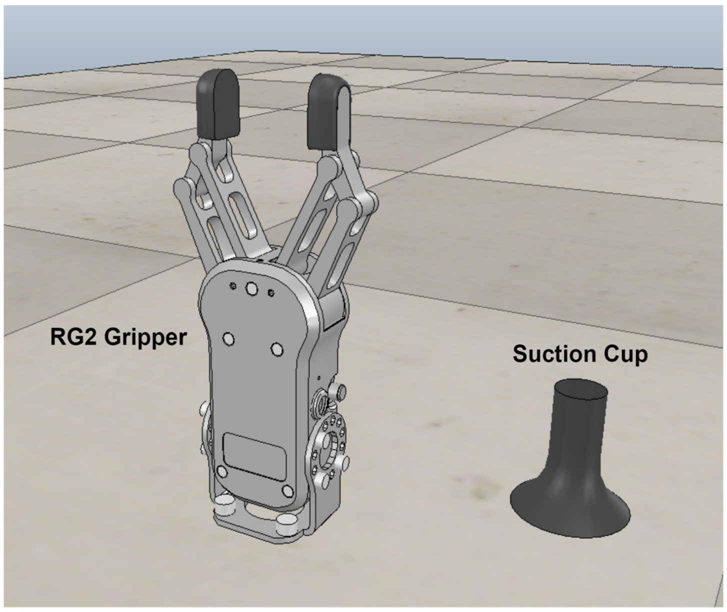

Figure 13. Jaw gripper and suction cup.

To implement the motion planning scheme for the UR5 robotic arm for the pick-and-place task, we explored various options available in the literature, such as OpenRave [49] and Trajpot [50], and finally settled on the Open Motion Planning Library (OMPL) [51]. The reason for this selection was the flexibility it offers for customization and control. OMPL provides several options for control-based and geometry-based planners, and many widely used planning state-spaces are also available. In our work, we deployed a single-query planner known as RRT-Connect [52], which is a bidirectional variant of the traditional RRT. RRT-Connect outperforms traditional RRT because, unlike the base variant, it works with two trees: one at the start and another at the end, which are later connected.

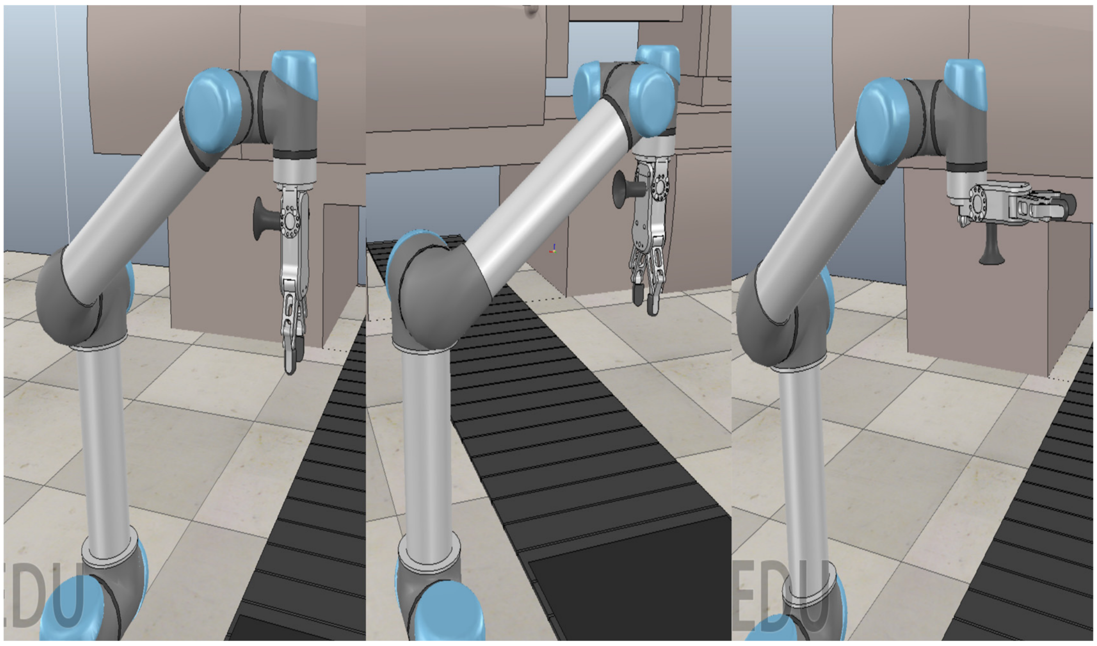

Figure 14. RG2 gripper and suction-cup installation.

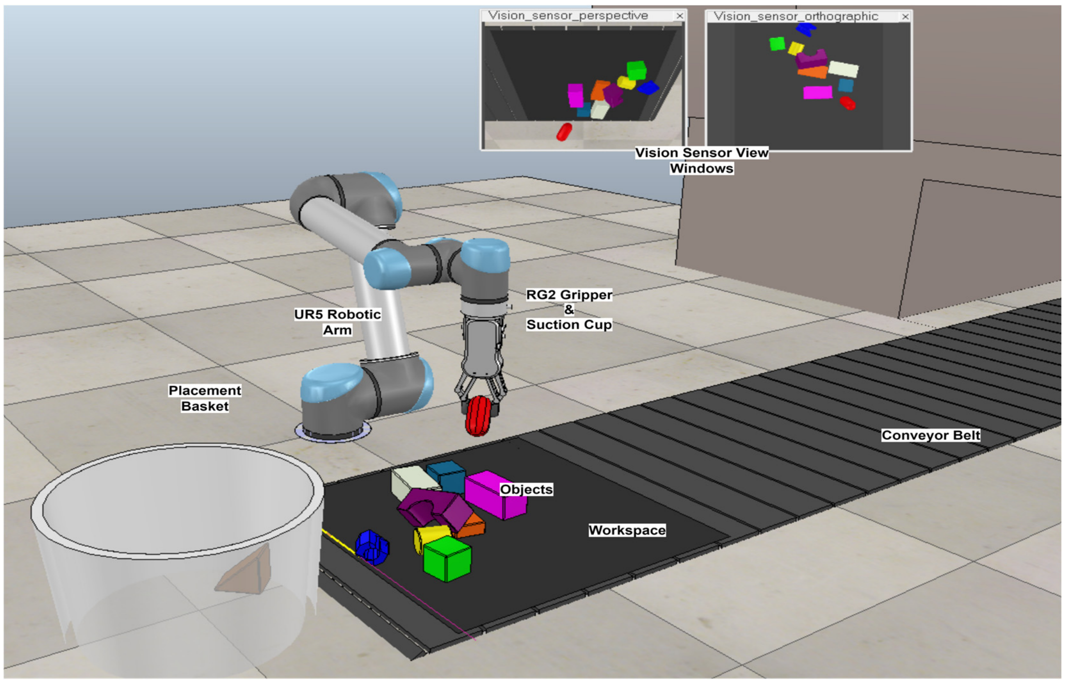
Vision_sensor_perspective
Vision_sensor_orthographic
Vision Sensor View Windows
UR5 Robotic Arm
RG2 Gripper & Suction Cup
Placement Basket
Conveyor Belt
Objects
Workspace

Figure 15. Simulation testbed design.

5. Results and Discussion

This section comprises a series of experiments designed to assess the performance of the developed agents within the simulated environment. The purpose is to evaluate the agents' ability to learn the synchronization of prehensile and non-prehensile manipulations for the pick-and-place task. These experimental results also highlight the role of non-prehensile manipulations in facilitating the prehensile manipulations.

For assessing the performance of our designed agent in terms of pick-and-place, we compare its performance with two different baselines. The first baseline is a variant of the proposed approach that is also based on the pixelwise-parameterization scheme. The major difference is that this baseline lacks the reinforcement learning element, being purely designed as a deep-learning-based binary classification approach. The structural design is the same as the proposed approach, with the representation of states, the design of actions, and the number and architecture of FCNs remaining consistent. While both use the same input and output pixelwise maps of action-values, the difference lies in the agent's training method. In the proposed approach, the agent is trained in self-supervision mode through Q-learning, whereas in this baseline, the FCNs are trained in supervised mode using binary classification. Labels are produced at runtime: a successful action is labeled as 1, and an unsuccessful one as 0. The backpropagation approach is also the same as in the proposed approach. This baseline approach has been previously implemented in [12] with a different architecture and in [9] with the same DenseNet-121 architecture. The comparison of the performance of the proposed approach and the binary classification baseline variant is shown in Figure 16. In this experiment, a series of 3000 episodes is conducted where each episode involves random clutter of ten regular- and irregular-shaped objects being introduced to the workspace. The episode is considered complete when the workspace is cleared or when 10 consecutive failed actions occur. In Figure 16, the orange line represents the proposed approach, while the green line represents the binary classification baseline. The plot shows that the proposed approach outperforms the binary classification, achieving up to an 89% success rate, while the baseline approach does not exceed 70%. The success rate is defined as the number of successfully picked-and-placed objects divided by the total number of actions performed. The lower success rate of the binary classification baseline is likely due to its greedy policy, which prevents it from recognizing long-term strategies that lead to higher expected rewards, unlike the proposed approach.

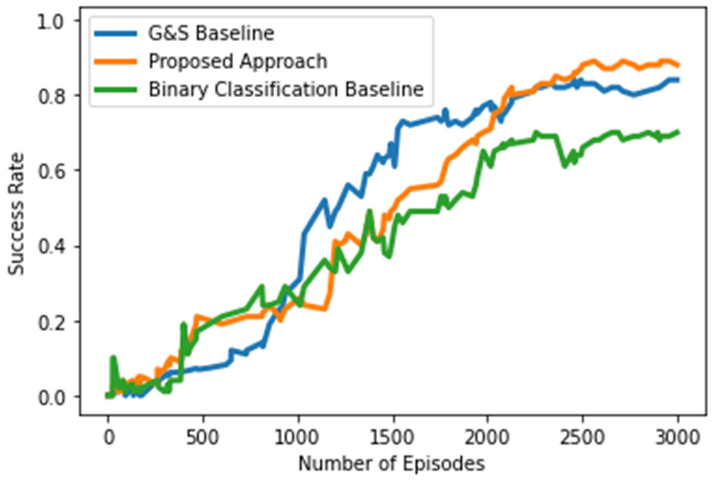

Figure 16. Performance comparison of the proposed approach and the baselines.

The second baseline is deployed from [9], where the agent has only three designed actions: grasping, left-slide, and right-slide. The major difference between the proposed approach and the G&S baseline from [9] is that the latter involves only one robotic gripper and lacks the suction gripping non-prehensile manipulation. The blue line in Figure 16 demonstrates the performance of the G&S baseline against the performance of the proposed approach, represented by the orange line. The addition of the second gripper and the suction gripping action, along with bidirectional sliding and grasping, leads to an improvement of around 5% in the success rate compared to the G&S baseline. This improvement is due to the role of suction gripping in facilitating the other three actions—grasping, inward-slide, and outward-slide—leading to successful pick-and-place operations in the clutter.

To investigate the role of FCNs architecture in the learning of the agent, we designed another variant of the proposed approach. Since we are specifically targeting the role of the neural network architecture, the only difference between the proposed approach and this variant is the architecture of the deployed FCNs. The proposed approach comprised FCNs designed using the DenseNet-121 architecture, but in this variant, the DenseNet-

121 architecture is replaced by the pretrained ResNet-101 [53] architecture. The ResNet architecture consists of residual blocks; hence, it is called a residual network. Any deep neural network has a certain threshold of its depth. As the layers keep on adding and that threshold is breached, the performance deterioration becomes evident because of the widely known vanishing gradient problem. The vanishing gradient problem is basically the extreme shrinking of the gradient while being backpropagated through the depths due to repetitive multiplications. ResNet was presented as the remedy to the vanishing gradient problem. The cure for the vanishing gradient problem in the ResNet architecture lies in the connections known as skip connections. As is evident from the name, the skip connections are named as such because they skip a few layers in gradient transmission, resulting in reduced repetitive multiplications, which leads to no vanishing gradient problem. Figure 17 presents the performance comparison of the proposed approach (in blue) and the ResNet101-based variant (in red). The graph clearly shows that the ResNet101-based variant is beaten by the proposed approach by around a 17% margin. The most probable reason behind this result is the difference in feature maps' handling between the two architectures. The ResNet implements feature map summation, whereas the DenseNet performs feature map concatenation. This results in better feature reusability in DenseNet. Moreover, where the skip connections in the ResNet limit the connectivity among the layers comparatively, the $\boldsymbol{n(n+1)/2}$ connections in the DenseNet enable higher feature propagation too.

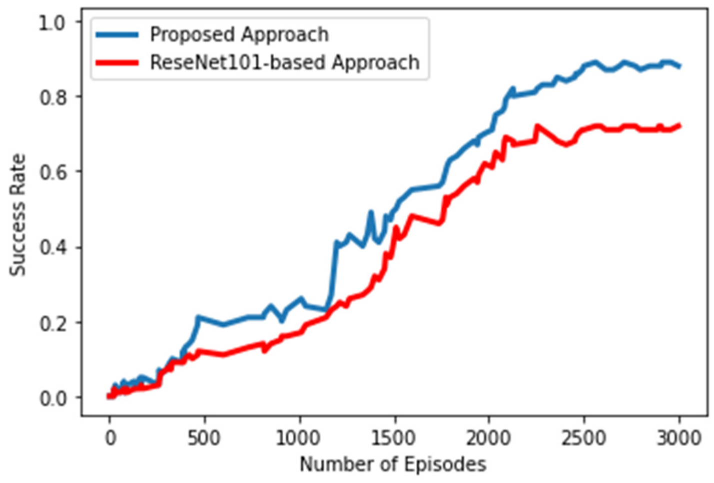

Figure 17. Performance comparison of the proposed approach and the ResNet101-based variant.

In this research study, we explore the role of other non-prehensile manipulations behind the prehensile manipulation. The idea is to evaluate the performance of these non-prehensile manipulations in the convergence of the policy, leading to a high success rate for the prehensile manipulations. To understand the role of non-prehensile manipulations—inward-slide, outward-slide, and suction-grip—we developed a variant of the proposed approach which awards zero rewards for all three non-prehensile manipulations. Rewards were only given for successful prehensile manipulation, i.e., grasping. If the whole pick-and-place was successful, the reward given was 1. If the placement failed after successful grasp, the reward given was 0.8. Figure 18 presents the performance of the no sliding-suction rewards variant (in magenta) against the proposed approach (in blue). The graph shows a fall of around 26% in the performance of the variant compared to the proposed approach, depicting the vital role of non-prehensile manipulations in the convergence of the policy. The delayed rise of the magenta line clearly shows the slower pace of learning of the agent due to the absence of the non-prehensile manipulations' rewards, leading to a grasping-based policy learning. It is possible that the agent is learning an indirect pattern of non-prehensile manipulations leading to prehensile manipulation rewards, but not at the desirable rate.

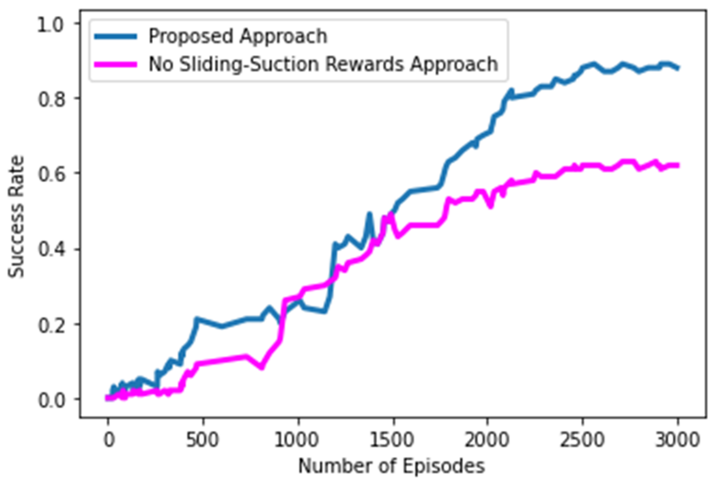

Figure 18. Performance comparison of the proposed approach and the no sliding-suction rewards variant.

In the proposed approach, we deployed the DenseNet-121 architecture pretrained on the ImageNet dataset, which contains a large number of images across approximately 1000 object classes. The general reason behind using pretrained networks is to achieve a better accuracy and reduce training time. In this series of experiments, we also explore the role of pretrained weights in the agent's learning of synchronization between prehensile and non-prehensile manipulations. A variant of the proposed approach was developed with a difference in weights. While the proposed approach uses pretrained weights from the ImageNet dataset, the variant uses no pretrained weights and is trained from scratch. Figure 19 shows the performance of the no pretrained weights variant against the proposed approach. The former is represented by the purple line, whereas the latter is shown by the blue line. According to these results, there is not much difference between the two in terms of performance and success rate. The pretrained weights do not play a significant role in the learning process of the agent. The probable cause behind the lack of benefit from pretrained weights in our case could be the difference in pixel patterns between the RGB-D heightmaps and the RGB images used in the ImageNet dataset. Not only do the pretrained weights seem to have minimal impact on the agent's learning, but the early rise of the purple line also highlights a faster pace of learning. In the case of the proposed approach with pretrained weights, the agent likely needs to first "clean the slate" of the previously learned weights to exit local optima and then start fresh learning, resulting in a delayed rise in the learning curve.

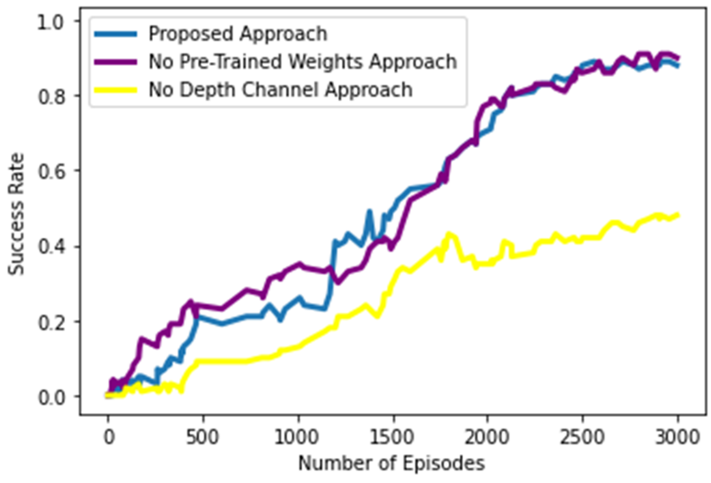

Figure 19. Performance comparison of the proposed approach and the no pretrained weights and no depth channel variants.

We also explore the role of the depth data fed to the FCNs in training the agent. As discussed earlier, in addition to the RGB channels, we utilize the cloned DDD depth channels. To evaluate the role of the depth channel, we designed a variant of the proposed approach that lacks depth channel information. As previously mentioned, in the proposed approach each FCN has two trunks: one is supplied with color information, and the other one is given depth information. In this variant, the major difference is that there is no second trunk designed for depth information. Therefore, feature learning is carried out solely on the color information maps. Figure 19 presents the performance of the no depth channel variant (in yellow) against the performance of the proposed approach (in blue). The results clearly show a significant degradation in performance in the absence of feature learning from the depth data. The lack of height-from-bottom information reduces the performance by approximately 41% compared to the proposed approach. This result underscores the importance of depth data in the agent's learning for the pick-and-place task and its convergence to the optimal policy.

In our training phase, we conducted a total of 3000 episodes, with the number of regular and irregular objects in the random clutter kept at 10. To evaluate the trained model, we used varying clutter sizes. We designed three clutter sizes as shown in Figure 20. The first is the minimum-size clutter, which consists of 6 to 10 randomly distributed regular and irregular objects. The second is called medium-size clutter, containing 15 to 20 regularly and irregularly shaped objects randomly distributed in the workspace. The last is named as maximum-size clutter, where the object count ranges from 25 to 30. The average performance results of the trained agent for these clutter densities are presented in Table 1.

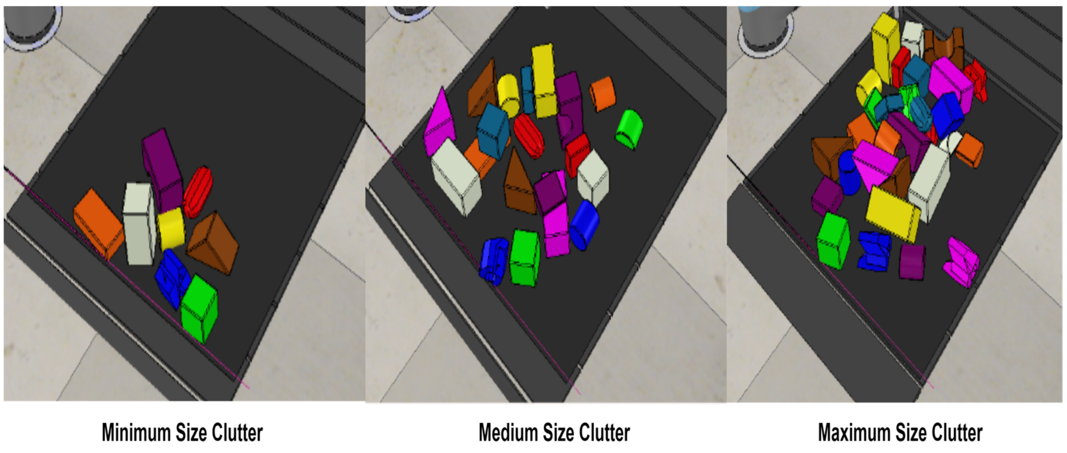

Figure 20. Clutter size categorization [11].

Table 1. Average testing performance.

Clutter Size	Success Rate	Grasp Success	Suction-Grip Success
Minimum-Sized	89%	97%	100%
Medium-Sized	87%	96%	100%
Maximum-Sized	82%	85%	99%

Table 1 presents the average success rate, grasp success, and the suction-grip success of testing across varying densities of clutter. The success rate is defined as the number of successfully picked-and-placed objects divided by the total number of actions performed. Grasp success is calculated as the number of successful grasps in an episode divided by the total number of attempted grasps in that episode. The suction-grip success rate is calculated in the same manner. The overall results indicate that performance is lowest in the maximum-sized clutter. The primary reason for this lower performance in the maximum-sized clutter is the random and weak arrangement of regular and irregular objects. With a higher number of objects, they become loosely stacked in the workspace and start falling or displacing/moving as soon as they are touched by the gripper, leading to failed prehensile and non-prehensile manipulations. Despite these complications, the overall results demonstrate that the agent has achieved a substantial level of learning and convergence to the optimal policy for the pick-and-place task involving regular- and irregular-shaped objects through a sequential combination of prehensile and non-prehensile robotic manipulations.

Discussion

In the experimental trials, we trained and tested our proposed approach agent along with its various variants. A comparative analysis was performed on the results of the proposed approach, its designed variants, and previously presented state-of-the-art methods. The results showing that the proposed approach outperforms the previously published state-of-the-art method by around 5% clearly highlights the positive role of the non-prehensile suction-grip manipulation in the pick-and-place task involving regular and irregular objects in clutter. These results not only demonstrate the agent's success in learning and converging to the optimal policy by understanding the patterns between prehensile and non-prehensile manipulations, but also emphasize the utility of deploying different robotic grippers together. The better performance of the proposed approach compared to the previously presented work underscores the effectiveness of suction gripping and bidirectional sliding manipulations in achieving successful pick-and-place operations. The role of reward-based learning was also explored by designing a binary classification-based agent. The importance of an efficient rewarding scheme and optimal policy convergence through the off-policy model-free Q-learning becomes evident when the binary classification-based variant is restricted to a success rate of under 70%. This variant lacks the reward-based learning ability, which apparently limits its feature learning compared to the proposed approach, resulting in a lower success rate in the pick-and-place task. Another important factor explored was the architecture of the four jointly trained FCNs used for Q-function approximation. It was crucial to determine the role of the FCN architecture in the convergence of the proposed agent to the optimal policy, in order to understand whether the architecture influences Q-function approximation. For this reason, a variant was designed by replacing the DenseNet-121 architecture with the ResNet-101 architecture. Both architectures address the vanishing gradient problem effectively and were chosen for this reason. The role of the FCN architecture in Q-function approximation was confirmed when the proposed approach with DenseNet-121 architecture outperformed the ResNet-101 architecture variant by around 17%. The most likely reason for this outperformance is the difference in feature map handling between the two architectures. DenseNet-121 uses feature concatenation, resulting in higher feature learning and reusability due to its $\boldsymbol{n(n+1)/2}$ connections among layers. In contrast, the ResNet-101 architecture performs feature summation and relies on skip connections. To further investigate the role of non-prehensile manipulations, a variant was designed with zero rewards for the suction-grip, inward-slide, and outward-slide actions. This variant exhibited a slower learning compared to the proposed approach and underperformed in terms of success rate by approximately 26%. This slower learning rate clearly indicates the agent's difficulty in learning synchronization between the prehensile and non-prehensile manipulations. Without rewards encouraging the use of non-prehensile manipulations to maximize future expected rewards, the success rate was lower. This finding supports the earlier results comparing the proposed approach to the previously presented state-of-the-art method. Additionally, the role of depth data was examined by designing a variant that used only RGB channels. As anticipated, performance deteriorated by approximately 41% when only RGB data were provided. This performance decline underscores the importance of depth data in predicting action-values during Q-function approximation. The absence of depth data hindered efficient feature learning by eliminating the concept of distance and spatial relationships. This finding highlights the depth data as a critical element of the proposed approach. As mentioned in previous sections, both pretrained and untrained models of DenseNet-121 architecture were tested.

Surprisingly, the untrained variant performed better in terms of learning pace compared to the pretrained one. This was evident from the earlier rise in the curve of the untrained variant compared to the pretrained one. The likely cause for this could be the difference in pixel patterns between our simulation data and the ImageNet dataset. Thus, the pretrained variant had to first escape the local maxima represented by pretrained weights before it could relearn new weights. Overall, the testing based on clutter density further supports the earlier findings that the agent successfully learned and converged to the optimal policy by sequentially synchronizing non-prehensile and prehensile manipulations, leading to a higher success rate.

6. Conclusions

This research study developed a deep reinforcement learning-based polydexterous framework that enables agents to learn industrial pick-and-place tasks in a self-supervised manner using cobots. The proposed framework allows agents to learn the sequential combination of prehensile and non-prehensile robotic manipulations using different robotic grippers. This learning process facilitates the achievement of optimal policy convergence, leading to increased efficiency and throughput for the pick-and-place task. The framework incorporates four distinct robotic manipulations: grasp, inward-slide, outward-slide, and suction-grip, using two robotic grippers: a jaw gripper and a suction cup. A key contribution of this research is that the framework integrates the manipulations of two different grippers, unlike most existing literature, where a single gripper typically performs all manipulations. Another significant contribution is that the framework operates without requiring prior domain-specific knowledge about objects or the environment, unlike previous approaches. The off-policy model-free Q-learning algorithm is utilized, and the pixelwise-parameterization technique is employed for Q-function approximation. Pixelwise maps of action-values are obtained through pixelwise-parameterization, and the 3D location corresponding to the pixel with the maximum action-value is selected for the relevant robotic manipulation. The reward scheme is designed accordingly, allowing the agent to learn in a completely self-supervised manner by recognizing patterns in non-prehensile manipulations, which increases the likelihood of successful prehensile manipulations in the near future. The Results Section demonstrates that the agents achieve optimal policy convergence, as evidenced by comparisons with baselines, various variants, and density-based testing clutter.

There are some limitations in the presented approach. For instance, the agents are trained on a fixed number of regular- and irregular-shaped objects. This could be extended in future research by incorporating real-life 3D object shapes such as balls, cups, etc. Similarly, transitioning from the simulated environment to the physical world could be achieved by redesigning the approach using the Robot Operating System (ROS/ROS2), which would enable the robotic arm to experiment with real-life objects. Additionally, the sequential combination learning of the agent could be expanded to include parallel combinations of robotic manipulations, incorporating tasks such as rotating and stacking, and exploring their impact on efficiency and throughput.

Author Contributions: M.B.I. conceived and implemented the idea and wrote the paper; Y.Q. and B.L. reviewed and supervised the work. All authors have read and agreed to the published version of the manuscript.

Funding: This publication emanated from research conducted with the support of Science Foundation Ireland (SFI) under Grant Number SFI 16/RC/3918, co-funded by the European Regional Development Fund. Additionally, this work was supported by the North–South Research Programme, which is delivered by the Higher Education Authority (HEA) on behalf of the Department of Further and Higher Education, Research, Innovation, and Science (DFHERIS), and the Shared Island Unit at the Department of the Taoiseach.

Data Availability Statement: Data are available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Vaidya, S.; Ambad, P.; Bhosle, S. Industry 4.0—A Glimpse. *Procedia Manuf.* **2018**, *20*, 233–238. [CrossRef]
2. Friedman, T.L. At Lunch, Donald Trump Gives Critics Hope. *New York Times*, 22 November 2016. Available online: https://www.nytimes.com/2016/11/22/opinion/at-lunch-donald-trump-gives-critics-hope.html (accessed on 1 July 2024).
3. Siciliano, B.; Khatib, O. *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2007; ISBN 3-540-23957-X.
4. Automation—Computer Process Control | Britannica. Available online: https://www.britannica.com/technology/automation/Robots-in-manufacturing (accessed on 26 January 2022).
5. Bogue, R. Growth in e-commerce boosts innovation in the warehouse robot market. *Ind. Robot Int. J.* **2016**, *43*, 583–587. [CrossRef]
6. Lamiraux, F.; Mirabel, J. Prehensile Manipulation Planning: Modeling, Algorithms and Implementation. *IEEE Trans. Robot.* **2022**, *38*, 2370–2388. [CrossRef]
7. Imtiaz, M.B.; Qiao, Y.; Lee, B. Prehensile Robotic pick-and-place in clutter with Deep Reinforcement Learning. In Proceedings of the 2022 International Conference on Electrical, Computer and Energy Technologies (ICECET), Prague, Czech Republic, 20–22 July 2022; pp. 1–6. [CrossRef]
8. Dogar, M.R.; Srinivasa, S. *A Planning Framework for Non-Prehensile Manipulation under Clutter and Uncertainty*; Carnegie Mellon University: Pittsburgh, PA, USA, 2018. [CrossRef]
9. Imtiaz, M.B.; Qiao, Y.; Lee, B. Prehensile and Non-Prehensile Robotic Pick-and-Place of Objects in Clutter Using Deep Reinforcement Learning. *Sensors* **2023**, *23*, 1513. [CrossRef]
10. Hogan, F.R.; Rodriguez, A. Feedback Control of the Pusher-Slider System: A Story of Hybrid and Underactuated Contact Dynamics. In *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*; Goldberg, K., Abbeel, P., Bekris, K., Miller, L., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 800–815, ISBN 978-3-030-43089-4.
11. Clavera, I.; Held, D.; Abbeel, P. Policy transfer via modularity and reward guiding. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1537–1544. [CrossRef]
12. Zeng, A.; Song, S.; Yu, K.-T.; Donlon, E.; Hogan, F.R.; Bauza, M.; Ma, D.; Taylor, O.; Liu, M.; Romo, E.; et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *Int. J. Robot. Res.* **2022**, *41*, 690–705. [CrossRef]
13. Mahler, J.; Liang, J.; Niyaz, S.; Laskey, M.; Doan, R.; Liu, X.; Ojea, J.A.; Goldberg, K. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. In Proceedings of the Robotics: Science and Systems XIII, Cambridge, MA, USA, 12–16 July 2017; Amato, N.M., Srinivasa, S.S., Ayanian, N., Kuindersma, S., Eds.; Massachusetts Institute of Technology: Cambridge, MA, USA, 2017. [CrossRef]
14. Koivikko, A.; Drotlef, D.-M.; Sitti, M.; Sariola, V. Magnetically switchable soft suction grippers. *Extreme Mech. Lett.* **2021**, *44*, 101263. [CrossRef]
15. Rodriguez, A.; Mason, M.T.; Ferry, S. From caging to grasping. *Int. J. Robot. Res.* **2012**, *31*, 886–900. [CrossRef]
16. Singh, T.; Ambike, S. A soft-contact and wrench based approach to study grasp planning and execution. *J. Biomech.* **2015**, *48*, 3961–3967. [CrossRef]
17. Zhou, J.; Liu, Y.; Liu, J.; Xie, Q.; Zhang, Y.; Zhu, X.; Ding, X. BOLD3D: A 3D BOLD descriptor for 6Dof pose estimation. *Comput. Graph.* **2020**, *89*, 94–104. [CrossRef]
18. Goldfeder, C.; Ciocarlie, M.; Dang, H.; Allen, P.K. The Columbia grasp database. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 1710–1716.
19. Pinto, L.; Gupta, A. Learning to push by grasping: Using multiple tasks for effective learning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 2161–2168. [CrossRef]
20. Lynch, K.M. Estimating the friction parameters of pushed objects. In Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93), Yokohama, Japan, 26–30 July 1993; Volume 1, pp. 186–193. [CrossRef]
21. Bauza, M.; Rodriguez, A. A probabilistic data-driven model for planar pushing. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3008–3015. [CrossRef]
22. Zhou, J.; Mason, M.T.; Paolini, R.; Bagnell, D. A convex polynomial model for planar sliding mechanics: Theory, application, and experimental validation. *Int. J. Robot. Res.* **2018**, *37*, 249–265. [CrossRef]
23. Owen-Hill, A. Why Vacuum Grippers Are Really the Best Option for Robot Palletizing. Available online: https://blog.robotiq.com/why-vacuum-grippers-are-really-the-best-option-for-robot-palletizing (accessed on 2 July 2024).
24. How Does a Vacuum Gripper Work? Available online: https://www.unboxindustry.com/blog/7-how-does-a-vacuum-gripper-work (accessed on 3 July 2024).
25. Eppner, C.; Höfer, S.; Jonschkowski, R.; Martín-Martín, R.; Sieverling, A.; Wall, V.; Brock, O. Lessons from the Amazon picking challenge: Four aspects of building robotic systems. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, Melbourne, Australia, 19–25 August 2017; AAAI Press: Melbourne, Australia, 2017; pp. 4831–4835.
26. Hernandez, C.; Bharatheesha, M.; Ko, W.; Gaiser, H.; Tan, J.; van Deurzen, K.; de Vries, M.; Van Mil, B.; van Egmond, J.; Burger, R.; et al. Team Delft's Robot Winner of the Amazon Picking Challenge 2016. In Proceedings of the RoboCup 2016: Robot World Cup XX; Behnke, S., Sheh, R., Sarıel, S., Lee, D.D., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 613–624.

27. Schwarz, M.; Milan, A.; Lenz, C.; Muñoz, A.; Periyasamy, A.S.; Schreiber, M.; Schüller, S.; Behnke, S. NimbRo picking: Versatile part handling for warehouse automation. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3032–3039. [CrossRef]
28. Domae, Y.; Okuda, H.; Taguchi, Y.; Sumi, K.; Hirai, T. Fast graspability evaluation on single depth maps for bin picking with general grippers. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 1997–2004. [CrossRef]
29. Mahler, J.; Matl, M.; Liu, X.; Li, A.; Gealy, D.; Goldberg, K. Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds using a New Analytic Model and Deep Learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018.
30. Liu, W.; Pan, Z.; Liu, W.; Shao, Q.; Hu, J.; Wang, W.; Ma, J.; Qi, J.; Zhao, W.; Du, S. Deep learning for picking point detection in dense cluster. In Proceedings of the 2017 11th Asian Control Conference (ASCC), Gold Coast, QLD, Australia, 17–20 December 2017; pp. 1644–1649. [CrossRef]
31. Boularias, A.; Bagnell, J.; Stentz, A. Learning to Manipulate Unknown Objects in Clutter by Reinforcement. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; Volume 29. [CrossRef]
32. Sarantopoulos, I.; Kiatos, M.; Doulgeri, Z.; Malassiotis, S. Total Singulation with Modular Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4117–4124. [CrossRef]
33. Imtiaz, M.B.; Qiao, Y.; Lee, B. Implementing Robotic Pick and Place with Non-visual Sensing Using Reinforcement Learning. In Proceedings of the 2022 6th International Conference on Robotics, Control and Automation (ICRCA), Xiamen, China, 26–28 February 2022; pp. 23–28. [CrossRef]
34. Imtiaz, M.B.; Qiao, Y.; Lee, B. Comparison of Two Reinforcement Learning Algorithms for Robotic Pick and Place with Non-Visual Sensing. *Int. J. Mech. Eng. Robot. Res.* **2021**, *10*, 526–535. [CrossRef]
35. Pitts, W.; McCulloch, W.S. How we know universals the perception of auditory and visual forms. *Bull. Math. Biophys.* **1947**, *9*, 127–147. [CrossRef]
36. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems—Volume 1, NIPS'12, Lake Tahoe, NV, USA, 3–6 December 2012; Curran Associates Inc.: Red Hook, NY, USA, 2012; pp. 1097–1105.
37. Mohammed, M.Q.; Kwek, L.C.; Chua, S.C.; Al-Dhaqm, A.; Nahavandi, S.; Eisa, T.A.E.; Miskon, M.F.; Al-Mhiqani, M.N.; Ali, A.; Abaker, M.; et al. Review of learning-based robotic manipulation in cluttered environments. *Sensors* **2022**, *22*, 7938. [CrossRef]
38. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement Learning: A Survey. *J. Artif. Int. Res.* **1996**, *4*, 237–285. [CrossRef]
39. Kober, J.; Bagnell, J.A.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
40. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018; p. 352.
41. Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.G.; Silver, D. Rainbow: Combining Improvements in Deep Reinforcement Learning. *arXiv* **2017**, arXiv:1710.02298. [CrossRef]
42. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M.A. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
43. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
44. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; IEEE Computer Society: Los Alamitos, CA, USA, 2015; pp. 3431–3440. [CrossRef]
45. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. *arXiv* **2018**, arXiv:1608.06993.
46. Huber, P.J. Robust Estimation of a Location Parameter. *Ann. Math. Stat.* **1964**, *35*, 492–518. [CrossRef]
47. Liu, Y.; Gao, Y.; Yin, W. An Improved Analysis of Stochastic Gradient Descent with Momentum. *arXiv* **2020**, arXiv:2007.07989.
48. Pleiss, G.; Chen, D.; Huang, G.; Li, T.; van der Maaten, L.; Weinberger, K.Q. Memory-Efficient Implementation of DenseNets. *arXiv* **2017**, arXiv:1707.06990.
49. Diankov, R.; Kuffner, J.J. *OpenRAVE: A Planning Architecture for Autonomous Robotics*; Robotics Institute: Pittsburgh, PA, USA, 2008.
50. Schulman, J.; Ho, J.; Lee, A.X.; Awwal, I.; Bradlow, H.; Abbeel, P. Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization. In Proceedings of the Robotics: Science and Systems IX, Berlin, Germany, 24–28 June 2013.
51. Şucan, I.A.; Moll, M.; Kavraki, L.E. The Open Motion Planning Library. *IEEE Robot. Autom. Mag.* **2012**, *19*, 72–82. [CrossRef]
52. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the 2000 ICRA. Millennium Conference. In Proceedings of the IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*; San Francisco, CA, USA, 24–28 April 2000, Volume 2, pp. 995–1001. [CrossRef]
53. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2020**, arXiv:1512.03385.