Article

Boosting Deep Reinforcement Learning with Semantic Knowledge for Robotic Manipulators

Lucía Güitta-López [*] , Vincenzo Suriani , Jaime Boal , Álvaro J. López-López , Daniele Nardi

Posted Date: 30 May 2025

doi: 10.20944/preprints202505.2416.v1

Keywords: deep reinforcement learning; semantic knowledge; robotics; sample efficiency

Article

Boosting Deep Reinforcement Learning with Semantic Knowledge for Robotic Manipulators

Lucía Güitta-López [1,†,*], Vincenzo Suriani [2,†], Jaime Boal [1], Álvaro J. López-López [1], and Daniele Nardi[3]

[1] Institute for Research in Technology (IIT), ICAI School of Engineering, Comillas Pontifical University, Rey Francisco, 4, 28008, Madrid, Spain
[2] University of Basilicata, via dell'Ateneo Lucano, 10, 85100, Potenza, Italy
[3] Sapienza University of Rome, Department of Computer, Control and Management Engineering, via Ariosto, 25, 00185, Rome, Italy
[*] Correspondence: lucia.guitta@iit.comillas.edu
[†] These authors contributed equally to this work.

Abstract: Deep Reinforcement Learning (DRL) is a powerful solution for complex sequential decision problems, especially in robotic control tasks. However, DRL's effectiveness is often restrained by the extensive experience required for learning, leading to high computational and time costs. We present a novel integration of DRL and semantic knowledge through Knowledge Graphs Embeddings (KGEs), to enhance robotic control tasks by incorporating contextual information. Our approach leverages semantic knowledge from pre-built knowledge graphs to provide additional context to the learning agent, effectively reducing the sample complexity and improving learning efficiency. We propose a DRL agent architecture that concatenates KGEs with visual observations, enabling the agent to use those inputs and environmental knowledge. We validate our method through experiments with robotic manipulators in environments with fixed and randomized target attributes, demonstrating significant improvements in task performance, learning speed, and accuracy compared to baseline DRL agents without contextual information. This integration leads to a reduction in the learning time and, therefore, an improvement in sampling efficiency, up to 60%, and to an improvement in the agents' accuracy of approximately 15 percentage points, without increasing the training time and computational complexity.

Keywords: deep reinforcement learning; semantic knowledge; robotics; sample efficiency

1. Introduction

Deep Reinforcement Learning (DRL) [1,2] is now consolidating as one of the most promising solutions for solving complex sequential decision problems where an agent interacts with its environment. While previous Reinforcement Learning (RL) [3] is limited by its memory and computational capabilities, DRL overcomes these limitations by leveraging the potential of deep learning for function approximation and representation learning in high-dimensional state spaces [4,5].

The application of DRL to tasks performed by robotic manipulators has broadened the scope from mostly repetitive jobs in fixed scenarios to tasks in changing environments with potentially unknown dynamics [6–9]. One of the major disadvantages of DRL is the extensive amount of experience, i.e., interactions between the agent and the environment, required for learning. It should also be noted that the greater the complexity of the environment, the more experience is needed. Virtual environments can be used to reduce the time and economic cost of overruns. By modeling assets in virtual scenarios, the *sample efficiency* problem is mitigated, as experience is acquired more quickly in simulators. However, the issue of the sheer amount of experience required remains.

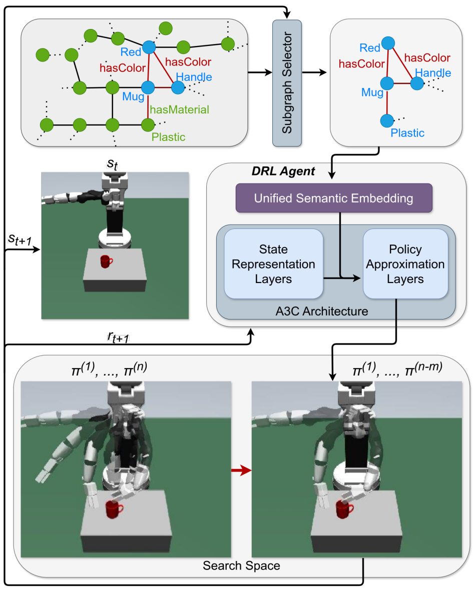
Red
hasColor
hasColor
Handle
Mug
hasMaterial
Plastic
Subgraph Selector
DRL Agent
Unified Semantic Embedding
State Representation Layers
Policy Approximation Layers
A3C Architecture
s_t
s_{t+1}
r_{t+1}
$\pi^{(1)}, ..., \pi^{(n)}$
$\pi^{(1)}, ..., \pi^{(n-m)}$
Search Space

Figure 1. Proposed framework diagram. A subgraph selector is used to extract the important entities and relationships based on the environment from a complete graph. This subgraph is embedded and concatenated in the layer prior to the policy approximation blocks of the DRL agent architecture. The result is an improvement in learning times and accuracy and a reduction of the required exploration.

As Figure 1 presents, we propose integrating contextual information about the environment within the agent's learning architecture to address the sample efficiency issue and enhance the agent's performance. Our main motivation is that any learning process, especially in DRL setups, should improve if the agent has knowledge about the environment that complements its observations. As already shown in [10], providing some basic semantic structure to the agent can speed up its learning. Ultimately, we argue that this approach introduces *commonsense* as semantic knowledge without additional complexity cost, with only a slight increase in the computational time. The environment's semantic information is obtained from a pre-built knowledge graph, where we extract the embeddings. After the representation learning layers, these embeddings are concatenated as inputs in the agent's architecture. To sum up, our main contributions and findings are:

- An original DRL agent architecture that successfully integrates Knowledge Graph Embeddings (KGEs) during the learning process. We modify the baseline layers to allow for concatenating the KGEs with the hidden activations from the "visual layers".
- A methodology to quantify the improvement of the DRL agents' performance, in terms of time and accuracy, through the use of KGEs with respect to a baseline agent with no semantic information of the environment and an agent with only partial information.
- An analysis of the possible improvement of the agents' exploration and exploitation capabilities by examining the distribution of each joint orientation during the evaluation episodes.
- A quantitative and qualitative evaluation of the embedding's influence in different environments and with various robot manipulators.

The paper is structured as follows: Section 2 presents the related work, Section 3 defines the experiments done, the setup employed, and the MDP designed. In Section 4, we describe the methodology followed for the knowledge graph and embedding obtention, as well as the deep reinforcement learning framework, that is, the algorithm used and our architecture, besides the corresponding training and post-training evaluation procedures. Section 5 gathers the results discussed in Section 6. Section 7 focuses on describing shortcomings and open issues related to our approach, and finally, Section 8 sums up the most relevant insights obtained with this research.

2. Related Work

To leverage the capability of robots to interact with the physical world, robot learning has emerged as a major research area [11]. Progress in deep learning has contributed significantly to advancements in robot learning, particularly when state representation includes images [12]. A variety of algorithms have been developed for robot learning, with reinforcement learning and imitation learning remaining the dominant approaches [13,14]. To facilitate the comparison of different reinforcement learning algorithms, numerous benchmarks have been introduced. For instance, some benchmarks focus on specific tasks such as door opening, furniture assembly, and in-hand dexterous manipulation [15]. Others, like those in [16] and [17], offer diverse environments but lack long-horizon tasks and lack of background knowledge. Background knowledge has been successfully deployed when dealing with natural language action space, to reduce the search space ([18,19]), but not extensively applied in the field of robotics.

Recently, commonsense has been integrated into the RL pipelines to enrich the representation of the environments. With the advent of the LLMs, a taxonomy of the combination of the two fields of study has been proposed in [20], where three classes are defined based on the way that the two model types interact with each other. An LLM can be used to supplement the training of an RL model that performs a general task that is not inherently related to natural language, as *LLM4RL* states.

Here, the semantic knowledge of Large Language Models (LLMs) is leveraged to enhance the performance of Reinforcement Learning (RL) agents. This enhancement can occur in two primary ways: by grounding the agent's environment to improve its performance [21,22], or by improving the training process of the RL agent. Training an RL agent is often computationally intensive and requires significant resources and large amounts of data. Moreover, RL training can suffer from inefficient sampling, particularly for complex, long-term tasks. Consequently, several LLM4RL frameworks aim to enhance training efficiency, ensuring the successful execution of target tasks during testing. They achieve this by facilitating exploration [23,24], enabling policy transfer of trained models [25], and implementing effective planning to reduce data requirements.

The role of a large-scale model in helping an RL agent, is discussed in [26], where the authors investigate how to combine these complementary abilities in a single system consisting of three parts: a Planner, an Actor, and a Reporter.

In the learning process of a grasping task, it has been demonstrated how generating natural human grasps needs to consider not only the object geometry, but also semantic information. In fact, in [27], a semantic-based grasp generation method, termed SemGrasp, generates a static human grasp pose by incorporating semantic information into the grasp representation.

If LLMs offer an invaluable opportunity to provide semantic information, it is also true that they can be demanding in many applications. To this end, we adopted a lighter architecture, using graphs. Semantic knowledge can successfully be stored in graph representations and manipulated using knowledge graph embedding techniques and pre-trained models. Some pioneering work has employed research-rich reinforcement learning (RL) techniques to address graph mining tasks. In [28], a unified graph reinforcement learning (GRL) formulation is proposed. However, most existing formulations struggle to effectively integrate groups of semantically related nodes within the RL paradigm. On the other hand, far from the RL field, a semantic representation framework based on a knowledge graph has been demonstrated to be suitable as a method to extract manipulation knowledge from multiple sources of information [29] and guide the agent to generate semantic representations of entities and relations in the knowledge base.

We propose to leverage the semantic knowledge of a lightweight graph representation to improve sample efficiency, and agent performance in terms of time and accuracy. Our architecture provides additional contextual information about the environment, which complements the agent's observations, leading to faster learning and higher accuracy, compared to baseline models without knowledge graphs.

3. Environment Setup

3.1. Experiment description

The task addressed involves using a robotic arm to approach three targets (e.g., a mug, a bottle, and a cereal box) that move randomly in the workspace area between episodes. The robot's end-effector should reach the grasping point within a distance of 5 cm during training and 10 cm during the post-training evaluation, with a specific orientation. The training orientation threshold is $\pm 15^o$, whereas in the post-training evaluation, it is $\pm 20^o$. The placement of the reachable point and the gripper approach orientation depend on the target. The environment observation is a $64\times64\times3$ RGB image without additional proprioception information. We performed the experiments in simulation to enable a thorough analysis of the proposed approach and to allow for total control of all the external drivers that might affect the agent's performance. In this way, we ensure total control of all the external drivers that might decrease the agents' performance.

3.2. Setup description

Two robots have been used in this research. On the one hand, the TIAGo from PAL Robotics [30], a 7-DoF mobile manipulator equipped with a two-finger gripper and mainly designed for human-robot interaction. On the other hand, the IRB120 [31] from ABB, a 6-DoF industrial robotic arm also with a two-finger gripper end-effector. The virtualization of both assets has been carried out through the MuJoCo [32] platform. The arena is the same for both robots. Due to the morphology of the TIAGo arm and to facilitate the reach, it was necessary to add a table that allowed us to place the objects at a greater height in the robot's workspace. Figure 2 displays the virtual environment for the TIAGo and IRB120 models.

The targets are represented by three graspable objects: a mug, a bottle, and a cereal box. Figure 2 shows the reachable spots for the three targets. We present the grasping point as a black sphere for better understanding of the reader, but in the observation received by the agent, it is not shown. Besides, Figure 2 illustrates the orientations of the grasping point axes that the grippers must align with, where the x coordinate is red, the y coordinate is green, and the z coordinate is blue. In the case of the mug, the grasping point is placed on its handle within an inclined orientation with respect to the base plane. For the bottle, we placed the grasping point on the stopper perpendicular to the base plane. Lastly, in the cereal box, the grasping point is set on the right-hand side edge, parallel to the base plane, above half its height. For the experiments without domain randomization (DR) applied to the targets' color, the mug is red (RGB code (1.0, 0.0, 0.0)), the bottle is yellow (RGB code (1.0, 1.0, 0.0)), and the cereal box is brown (RGB code (0.55, 0.27, 0.07)). When we experiment with DR applied to the targets' color, the mug can be red or blue (RGB code (0.0, 0.0, 1.0)), the bottle yellow or purple (RGB code (0.4, 0.0, 0.9)) and the cereal box brown or light-blue (RGB code (0.5, 0.7, 0.9)).

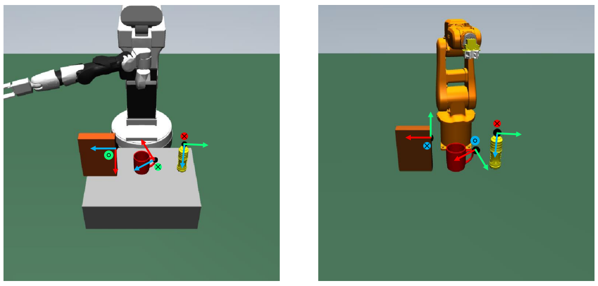
Figure 2. Environment observation with the three targets for the TIAGo and IRB120 scenarios. Note that we have added a black sphere on each grasping point for convenience. During training, the resolution is 64×64, and the sphere does not appear. The site axes correspond to the x coordinate in red, the y coordinate in green, and the z coordinate in blue.

3.3. MDP Definition

DRL problems can be defined as Markov Decision Processes (MDPs), which are characterized by a set of finite states, actions, a state transition probability matrix, a reward function, and the discount factor. Table 1 gathers the MDP definition of our problem. The state is a 64×64×3 RGB image captured at the beginning of each step. We use an episodic setup where each episode lasts 50 steps or less if the robot's gripper reaches the target within the rewarding distance and orientation. The reward given to the agent depends on the relative distance and orientation between the gripper and the grasping point. If the target is not reached, the agent receives a negative reward, whereas if it is, it receives a high positive value, as shown in Table 1, and the episode ends. The action set is formed by seven possible actions that scale the maximum joint movement range (Maximum Position Increment - MPI). At the beginning of the episode, the first and second joint orientations are set randomly within ±15% of their working range. Additionally, before starting the episode, one of the three targets is selected and placed randomly following a uniform distribution within the workspace area, which is a 20×25 cm rectangle for the TIAGo environment and a 60×20 cm rectangle for the IRB120 setup.

Table 1. MDP definition. rel_{dist} and rel_{deg} represent the relative distance and degrees between the gripper and the grasping point. MPI is the Maximum Position Increment. WRLL and WRUL are the Working Range Lower Limit and the Working Range Upper Limit of the robots. The target position is selected according to a uniform distribution.

MDP Definition	
Ep. Length	50 steps maximum
Reward configuration	$-2 * \text{rel}_{\text{dist}}^2 - \text{rel}_{\text{deg}}/70$ 100 if grasping point reached
Reward constraints	$\text{rel}_{\text{dist}} < 5\,\text{cm}$ $\text{rel}_{\text{deg}} < 15^o$
Action set	0 ±MPI ±MPI/10 ±MPI/100
Initial robot config	$U(15\%\text{WRLL}, -15\%\text{WRUL})$ for joints 1 and 2
Initial target config	$\text{x} \sim U(x_{min}, x_{max})$ $\text{y} \sim U(y_{min}, y_{max})$

4. Methodology

4.1. Knowledge Graph and Embeddings

To provide external knowledge to the learner, we extract contextual information of the scene from a Knowledge Graph $\mathcal{G}$. Then, we embedded the information in a latent space in the form of Knowledge Graph Embedding (KGE).

The objective of the KGE is to learn a continuous vector representation of a Knowledge Graph $\mathcal{G}$, which encodes vertices that represent entities $\mathcal{E}$ as a set of vectors $v_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times d_{\mathcal{E}}}$ (where $d_{\mathcal{E}}$ is the dimension of the vector of entities $\mathcal{E}$), and as a set of edges which represent relations $\mathcal{R}$ as mappings between vectors $W_{\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}| \times d_{\mathcal{R}}}$, where $d_{\mathcal{R}}$ is the dimension of the vector of relations. The knowledge graph $\mathcal{G}$ is composed by triples (h, r, t), where $h, t \in \mathcal{E}$ are the head and tail of the relations, while $r \in \mathcal{R}$ is the relation itself. One example of such a triple is (*mug*, *hasColor*, *yellow*).

We initially used ANALOGY as a direct embedding model. Since we need to have a semantic representation of the scene of the agent, we defined a unified semantic embedding that is extracted by introducing an operator Γ that, given a graph $\mathcal{G}'$, select the subgraph $\mathcal{S}$ from the larger knowledge graph $\mathcal{G}$ ($\mathcal{S} \subseteq \mathcal{G}$). This subgraph $\mathcal{S}$ contains all the nodes representing the objects that the agent is perceiving and the node at distance 1 from them. resulting in all relevant entities and their relationships pertinent to the specific scene the agent is interacting with. For instance, if the scene includes a mug, a bottle, and a cereal box, the subgraph $\mathcal{S}$ will contain nodes representing these objects and edges denoting their relationships with other nodes (e.g., (*mug, hasColor, red*), *mug, isConnectedTo, handle*), (*bottle, hasShape, cylindrical*), (*cereal box, isMadeOf, cardboard*)). From the subgraph, a textual representation is obtained by concatenating the labels of the nodes and the edges in a single sentence. The sentence is then converted into a single embedding representing the scene. To effectively, capture the semantic nuances of the scene, we replaced ANALOGY with a pre-trained embedding model, GloVe[33]. With GloVe, we generated a single embedding for the scene. The use of a pre-train model on a large corpus, ensures that the resulting vectors capture a wide range of semantic nuances. The dimension of these embeddings is set to 10×4 per word, allowing for a rich representation of the subgraph's content.

The resulting GloVe-based embeddings are then integrated into the learner's architecture. In our proposed setup, the embeddings are concatenated with the hidden activations from the visual observation layers before the policy approximation block. This strategic placement allows the DRL agent to utilize both the raw visual data and the rich contextual information provided by the embeddings, enhancing its ability to understand and interact with the environment.

4.2. Deep Reinforcement Learning Framework

4.2.1. Asynchronous Advantage Actor Critic (A3C)

For the sake of clarity, we first recap the concepts and terminology used to describe DRL. Model-free DRL includes value-based and policy-based approaches. The former are methods that learn either the estimation of the expected discounted cumulative reward, represented by the state-value function, $V^{\pi}(s)$, or the estimation of the expected discounted cumulative reward for taking action a in state s at step t, defined with the action-state value function, $Q^{\pi}(s,a)$ whose definitions are:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_t = s\right] \tag{1}$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_t = s, a_t = a\right] \tag{2}$$

where $\mathbb{E}_{\pi}$ is the expectation following policy π, γ is the discount factor, r_{t+1} is the reward received at the time step $t+1$, and s_t is the current state.

The latter are algorithms seeking to maximize the discounted cumulative reward, but directly learning and optimizing the parametrized policy $\pi(s|a;\theta)$ [3].

A3C [34] is a policy-based method, with the particularity that the *actor* optimizes the policy, and, at the same time, the *critic* estimates the state-value function. A3C allows for the deployment of several agents that run in parallel, each with a unique environment instance, and asynchronously, they update a shared global network. This guarantees stable and efficient learning, reducing the correlation between the experience gathered by the instantiated agents. As shown in [35,36], and in [37], A3C is a good alternative for discrete action spaces and high-dimensional state spaces.

Regarding the learning process, A3C uses the *Advantage Function*, $A(s,a) = Q(s,a) - V(s)$, to measure the goodness of an action. This function can be approximated by estimating the value of the current state, the reward, and the discounted value of the next state. This is the *Temporal Difference error*, or TD error at t:

$$\delta^{(t)} = r^{(t)} + \gamma V(s_{t+1};\theta) - V(s_t;\theta) \tag{3}$$

where $V(s_t;\theta)$ is the estimated value of s_t given the policy parameters θ. To smooth this estimation, we use the *General Advantage Estimator (GAE)*:

$$A_{GAE}^{(t)} = \gamma\lambda A_{GAE}^{(t+1)} + \delta^{(t)} \tag{4}$$

where $A_{GAE}^{(t)}$ is the GAE at time step t, and λ is the trace decay factor that controls the bias-variance trade-off. On the other hand, the loss is defined as the sum of the *Policy Loss* and the *Value Loss*, defined as:

$$\mathrm{P_{loss}} = -\sum_i \Big(\log \pi(a_{ti}|s_{ti};\theta) \cdot A_{\mathrm{GAE}}^{(t)} - \beta H(\pi(s_t;\theta))\Big) \tag{5}$$

where i iterates over actions at t, $\pi(a_{ti}|s_{ti};\theta)$ is the probability of taking a_{ti}, β is the entropy regularization weight, and $H(\pi(s_t;\theta))$ is the policy entropy.

$$\mathrm{V_{loss}} = \frac{1}{2}\sum_t (R_t - V(s_t;\theta))^2 \tag{6}$$

where R_t is the n-step return at t.

4.2.2. Proposed Architecture

Figure 3 shows the proposed architecture. The state representation layers are defined by a convolutional layer CNN$_1$ with 3×3 kernel and stride 4 and another convolutional layer CNN$_2$ with 5×5 kernel and stride 2. Both layers have at the end a non-linear f implemented through the ReLU function, $f(x) = max(0,x)$. The policy approximation block is formed by a 1152×128 fully connected layer (FC), a Long-Short Term Memory (LSTM), whose capacity depends on the size of the integrated KGE. Since the agent commands the orientation of each robot joint, the A3C actor layers are n FC layers, where n is the number of robot joints, i.e., 7 for the TIAGo and 6 for the IRB120. The dimension of the n FC blocks is determined by the discrete set of actions, which has seven actions per robot joint (Table 1). We use the softmax function to convert the activations into probabilities. Therefore, the actor outputs are n 7-dimension arrays that contain the probability associated with each action. Finally, the A3C critic is a single FC layer that estimates the value of $V(s)$. We will refer to the Baseline Agent or Baseline Model (BM) as the agent trained under this architecture with just the RGB image. It will serve as a reference to quantify the improvement obtained by integrating the KGE.

The originality of our proposal lies in the integration of embeddings coming from a graph containing some environment semantic information. This new input, which does not change throughout the training, might be placed next to the policy approximation blocks, since it should complement the state representation from the "vision" layers. Therefore, the most appropriate place to integrate the KGE is before the LSTM layer, linking the embeddings to the activations coming from the FC. The FC layer receives the representation coming from the CNNs and provides a higher abstraction level. The size of the embeddings and, consequently, the size of the LSTM will depend on the case study. In general, the embedding concatenated is a 150-dimension array, except for the one used when DR is applied to the targets' color, which is a 300-dimension array. As a result, the LSTM size will be 128, when no KGE is integrated, 428 when the KGE includes all the information from the randomized targets, which is an assumable size for our architecture, and 278 otherwise.

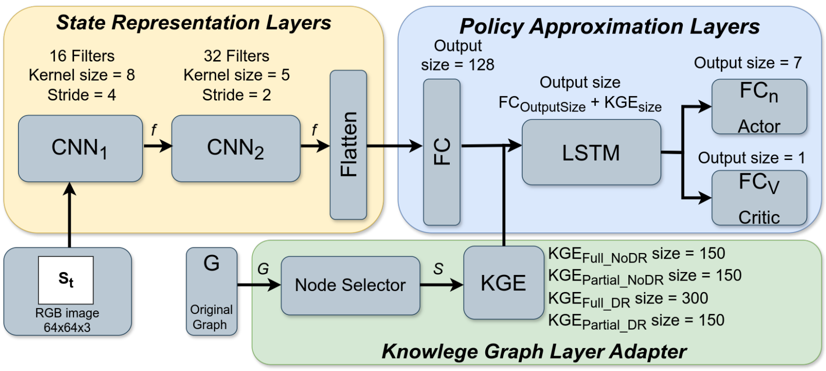
State Representation Layers
16 Filters Kernel size = 8 Stride = 4
32 Filters Kernel size = 5 Stride = 2
CNN1
CNN2
f
Flatten
Policy Approximation Layers
Output size = 128
FC
Output size FCOutputSize + KGEsize
LSTM
Output size = 7
FCn Actor
Output size = 1
FCV Critic
St
RGB image 64x64x3
G
Original Graph
Node Selector
S
KGE
KGEFull_NoDR size = 150
KGEPartial_NoDR size = 150
KGEFull_DR size = 300
KGEPartial_DR size = 150
Knowlege Graph Layer Adapter

Figure 3. Proposed A3C architecture to integrate KGEs in the DRL agent's learning process. The KGE size depends on the number of features encoded. The output is divided into seven actors for the TIAGo and six for the IRB120, plus the critic value.

4.2.3. Training Procedure

We train the models for 70 million steps using the A3C algorithm. We use a random orthogonal initialization for all the layers. Based on the computational resources available, the number of asynchronous agents instantiated is 17 plus one for the interim evaluation. This interim evaluation happens every 50,000 steps during training, and it assesses the global shared model for 40 episodes. These episodes keep the same 40 initial configurations, i.e., robot and target poses, across all the interim evaluations. At the beginning of each episode, a target is selected and placed randomly within the workspace area. The non-selected targets do not appear on the image. The rewarding distance is 5 cm, and the allowed orientation deviation is $\pm 15^o$.

The optimizer selected was the Root Mean Square Propagation (RMSprop) [38]. The values of the most relevant hyperparameters are learning rate $1e^{-4}$, discount factor (γ) 0.99, entropy weight (β) 0.01, trace decay (λ) 1, and RMSprop decay 0.99.

4.2.4. Post-Training Evaluation Procedure

After the training process, we select the best model according to the average return obtained in the interim evaluations. Then, we perform a post-training evaluation across 1,000 episodes with a different seed and initial robot and target poses from the ones used during training. The rewarding distance and allowed orientation deviation in the post-training evaluation are 10 cm and $\pm 17^o$, respectively. We modify these training constraints because we believe that training in more restrictive circumstances yields better post-training results. In the assessment, we calculate the mean and standard deviation (Std) return, the mean and Std episode length, the mean, Std, and maximum failure distance, and the accuracy. In our context, we understand accuracy as the percentage of successful episodes where the robot reached the target within the required distance. We formulate the accuracy as:

$$\text{Accuracy (\%)} = \frac{\sum_{i=1}^{N} \mathbb{I}(\text{dist}_i \leq 10\,\text{cm})}{N} \times 100 \tag{7}$$

where N is the total number of evaluated episodes, $dist_i$ is the relative distance between the gripper and the target in the i^{th} episode, and $\mathbb{I}$ is the indicator function.

4.2.5. Experiment Description

To analyze the influence of KGE in the learning process of a DRL agent, we present two sets of experiments. We first train three agents in an environment where each target has one possible color. The difference between these agents is the information we provide through the KGE. The BM agent has no KGE integrated. Then, there is an agent with only partial knowledge about the targets, just the object type (partial KGE). Finally, there is an agent with a complete description of the target type and color (full KGE).

In the second set of experiments, we decided to increase the complexity of the problem by introducing DR to the target color. Each object has two possible colors (Section 3) selected randomly at the episode beginning after the target choice and its placement. As in the first set, we train three different agents. A BM agent with no KGE, an agent with the partial KGE that contains only the object type, and an agent with the full KGE that has the object type and the possible colors per object.

We complement both groups of experiments with a quantitative and qualitative analysis of the joints' angles distribution to research how the exploration and exploitation capabilities of the agents might be affected by the KGEs.

5. Results

5.1. Experiments without DR

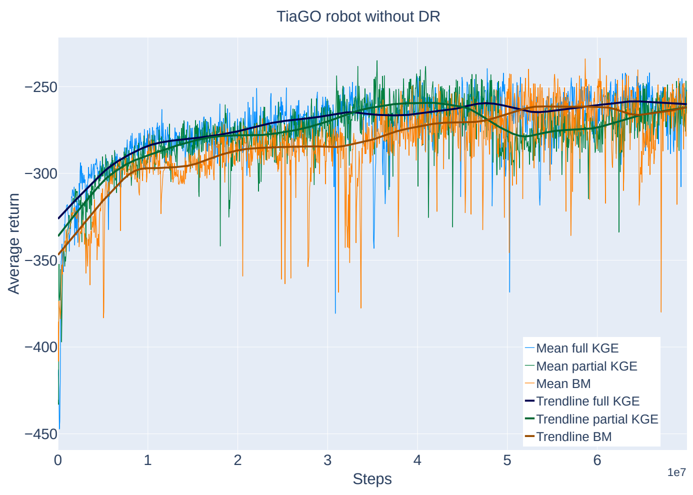

Figure 4. Training curves of the agents trained with the TIAGo and fixed targets' colors. The blue curve is the full KGE model, the green is the partial KGE, and the orange is the BM. The training process lasts for 70 M steps.

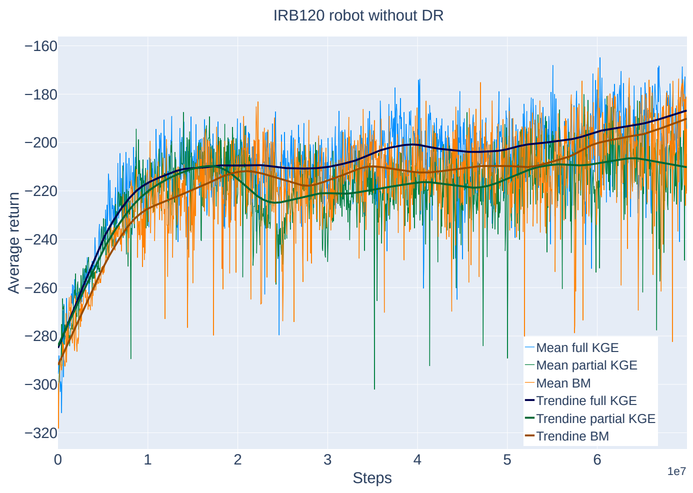

Figure 5. Training curves of the agents trained with the IRB120 and fixed targets' colors. The blue curve is the full KGE model, the green is the partial KGE, and the orange is the BM. The training process lasts for 70 M steps.

Figure 4 and Figure 5 show the training results in the environment with invariant features for the TIAGo and IRB120 robots, respectively. In the case of TIAGo, Figure 4 shows that the agent trained with the complete KGE learns faster and remains with a higher return than the agent trained with the partial KGE. In the first case, the best model is achieved at 48 M steps and has 72 % accuracy, whereas in the second, it is at 36 M steps and has 70 % accuracy. On the other hand, the differences with the BM in terms of learning speed and accuracy are higher. Although both agents reach the same steady regime, the best BM is at 60 M steps, and its accuracy is 60 %. Hence, the improvement gain using KGE under a setup with invariant features is 12 percentual points.

Regarding the IRB120 robot, Figure 5 shows that, although there is a larger difference between the full KGE and the partial KGE trendlines, the accuracy of the best models are 92 % and 91 %, respectively. They are obtained in the 60 M step and the 59 M step. On the other hand, comparing the results with the BM, we notice that its learning is slower than the agent with the full KGE, which stands above the BM during the process. The performance of the BM best agent, which occurs in the 60 M step, reaches 80 % accuracy. Therefore, the accuracy improvement in this setup without DR is also around 12 % when using the KGE as an additional input to the model.

Lastly, the plots in the first column of Figure 6 show the joints' angles distribution for the TIAGo, while the Figure 6 second column graphs represent the IRB120 joints. We compare the distributions of the full KGE agents, represented in blue, with the BMs, shown in orange. The selected joints for the TIAGo are joints 3 and 4, and for the IRB120, they are joints 2 and 3. These joints were selected because they are the ones that present meaningful differences in the distribution of their values, which is interpreted as an improvement in the agent's ability to learn more straightforward the right policy. TIAGo's joint 3 experienced a slight reduction in the standard deviation of 0.01 points, but a significant mean variation. TIAGo's joint 4, however, has a large standard deviation decrease of 0.21 points and a mean shift. Besides, the distribution changes to a bimodal shape, which might indicate some change in the agent's policy. In the case of IRB120, joint 2 has a standard deviation decrease of 0.02 points, as well as joint 3, which also presents a relevant mean difference in the distributions.

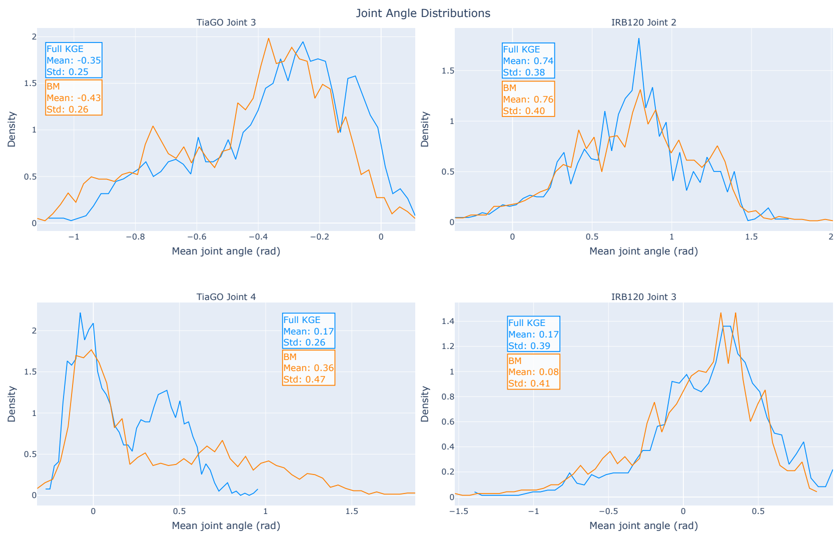

Figure 6. Joint angle distributions for the best full KGE agent (blue), and the best BM agent (orange), without DR. The first column corresponds to the TIAGo and the second to the IRB120. The selected joints for TIAGo are joints 3 and 4, and for the IRB120, they are joints 2 and 3.

5.2. Experiments with DR

Figure 7 and Figure 8 present the training curves for the three models trained in the TIAGo and IRB120 setups when the targets' color change along the training episodes. For the first scenario, we spot that the best model of the agent with the full KGE achieves 79 % accuracy, higher than the 62 % obtained by the agent with the partial KGE, and the 63 % of the BM. Additionally, we can see that the learning is faster, and as a result, the best model is obtained at the 24 M step, earlier than in the partial KGE model, which happens at the 59 M step and in the BM at the 57 M step. Therefore, we see that the overall improvement of using a complete KGE with the target type and color in a setup where this last feature can change is around 16 % with respect to the BM, and it is achieved in half of the training time.

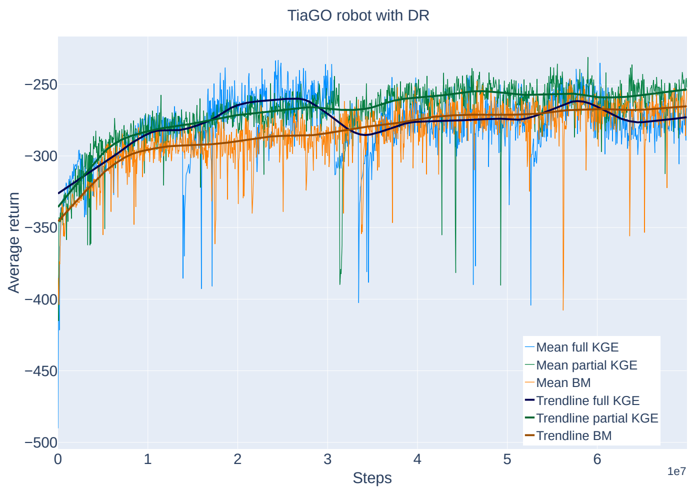

Figure 7. Training curves of the agents trained with the TIAGo and DR on the targets' colors. The blue curve is the full KGE model, the green is the partial KGE, and the orange is the BM. The training process lasts for 70 M steps.

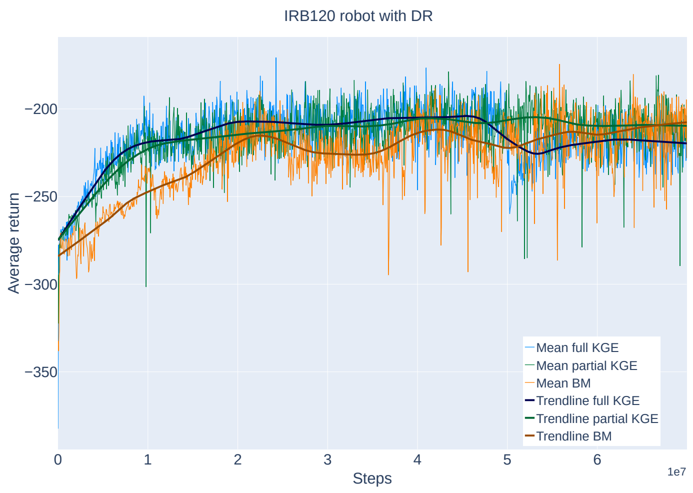

Figure 8. Training curves of the agents trained with the IRB120 and DR on the targets' colors. The blue curve is the full KGE model, the green is the partial KGE, and the orange is the BM. The training process lasts for 70 M steps.

In the scenario with the IRB120 robot, the agent with the complete KGE learns faster and yields higher than the BM. Whereas the first one reaches 86 % accuracy in the 24 M step, the latter attains 76 % in the 56 M step. If we compare the results with the model with partial KGE, it obtains 87 % accuracy in the 43 M step, a comparable success rate. Table 2 collects the accuracy results of the trained agents.

Table 2. Experiments made to analyze the KGE influence in the DRL agent learning process, using TIAGo and IRB120 robotic arms. There are two sets: without DR and with DR. Each set includes three types of agents: BM, Partial KGE, and Full KGE.

Experiment Set	Robots	Agent Type	Accuracy %	Best Model Step
Without DR	TIAGo	BM	60 %	60 M
		Partial KGE	70 %	36 M
		Full KGE	72 %	48 M
	IRB120	BM	80 %	60 M
		Partial KGE	91 %	59 M
		Full KGE	92 %	60 M
With DR	TIAGo	BM	63 %	57 M
		Partial KGE	62 %	59 M
		Full KGE	79 %	24 M
	IRB120	BM	76 %	56 M
		Partial KGE	87 %	43 M
		Full KGE	86 %	24 M

Figure 9 exhibits the joints' angle distributions for the TIAGo and IRB120 models with DR. For the TIAGo, the most significant results appear on joints 2, 4, and 5. They are displayed in Figure 9 first column. In all of them, the full KGE agent, the blue curve, presents lower standard deviation values on the joints' distributions than the BM, the orange curve. In the case of joint 2, it decreases 0.03 points, 0.04 points in joint 4, and 0.11 points in joint 5. For the IRB120, we analyze joints 1, 2, and 4, shown in Figure 9 second column. As in the TIAGo, the agent with the complete KGE, the blue plot, has less standard deviation than the BM, the orange curve. This reduction is 0.05 points in joint 1, 0.07 points in joint 2, and 0.12 points in joint 4. We decided not to include the result of the other joints because it does not have a significant change.

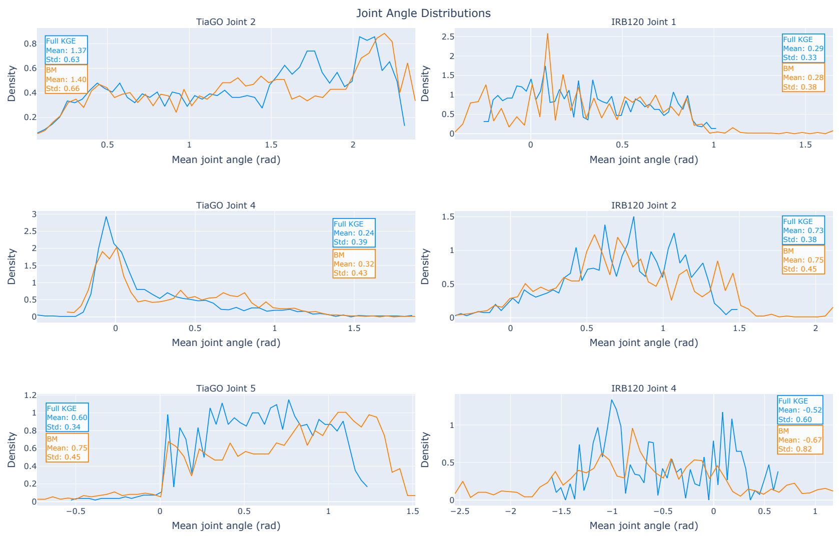

Figure 9. Joint angle distributions for the best full KGE agent (blue) and the best BM agent (orange), with DR. The first column corresponds to the TIAGo and the second to the IRB120. The selected joints for TIAGo are joints 2, 4, and 5, and for the IRB120, they are joints 1, 2, and 4.

6. Discussion

According to the results presented, we can say that, in general, the TIAGo's agents achieve less accuracy than IRB120's because their environment is more challenging. We argue that the morphology of the arm, the fact that it has one more DoF than the IRB120, its reach, and its color are factors that increase the complexity of the problem, although the task is the same.

In the experiments carried out with the TIAGo and with the IRB120 without DR we consistently observed an improvement in the transitory regime and a decrease in the learning process time. For the

TIAGo, there is an improvement of 25 % in the learning time between the full KGE agent and the BM. In the IRB120, although the best models are achieved at the same time, the full KGE trendline is always above the BM, and its transitory is much faster. With respect to the success rates, in both setups, there is almost no difference between the full and the partial KGE. This might prove that, since the color is an invariant feature in these experiments, the presence of this specification on the embeddings does not change the agent performance significantly. However, we do obtain 12 % accuracy improvement with respect to the BM when we evaluate the full KGE agent in the TIAGo and IRB120.

Regarding the analysis of the joints' angle distribution, we studied whether the use of KGEs in the learning process might involve an improvement in the exploration ability and, in consequence, better exploitation and more efficient learning. We see that in the TIAGo setup, there is a significant drop in the standard deviation value for joint 4 and a big change in the joint 3 mean. Conversely, in the IRB120 scenario, we did not obtain great standard deviation variations, but differences in the distribution means, as we see in joint 2 and, especially, in joint 3. This absence of meaningful changes might be in concordance with the fact that the BM is already close to mastering the task, so the differences in the exploration are not that relevant. These outcomes show that under the setup without DR, the exploration capability might present an enhancement, helping the KGE agents to achieve better policies sooner than the BM agents and, hence, learn faster.

When we apply DR to the targets' color, we see that for the TIAGo and the IRB120, the agents with the complete KGE are three times faster than the other agents, reducing the learning time needed by almost 60 %. Besides, their transitory regime is consistently better, as it happens in the experiments without DR. This might confirm that in a situation with an increase in the problem complexity resulting from varying features, adding contextual information helps in the learning process and the final performance. For the TIAGo, the full KGE agent performs 16 % better than the partial KGE agent and the BM. The TIAGo's and IRB120's trendline analysis shows that the agent with the full KGE decreases its performance around the 30 M step and 50 M step, respectively, although later, they narrow the distance again with the other models. We argue that this might have happened due to a high learning rate and the fact that we do not use a learning rate decay mechanism.

For the IRB120, there is 10 % improvement between the full KGE agent and the BM. With this robot, it is noticeable that there is not a significant improvement between the full KGE and partial KGE success rate as it is in the TIAGo. We suggest that the color information does not involve an accuracy increase because of the scenario colors. Since the robot is orange, the RGB filters corresponding to the green and, especially, the blue channels do not have the same relevance as the red ones. Hence, this unbalance in the visual learned features might implicate an exclusion of part of the color information encoded in the KGE. This does not happen with the TIAGo because its arm is white, so it should equally use the filters for the three RGB channels.

Finally, the outcomes obtained by analyzing the distribution of the joints' orientation, we noted a significant decrease in the standard deviation of the distribution for three of the TIAGo and IRB120 joints when KGEs are used. As in the previous set of experiments, this suggests that the agents can exploit the correct policies early thanks to the information encoded in the embeddings. This is consistent with the fact that these agents learn faster than the BM agents. This drop is larger and occurs in more joints than in the scenarios trained without DR, which might be because of the increase in the problem's complexity.

7. Limitations

While our experiments have shown improved learning efficiency in predefined environments, the reliance on predefined knowledge graphs may limit adaptability in dynamic and unstructured environments, where the robot must perform tasks that cannot benefit from semantic conditioning. The effectiveness of our approach depends on the completeness of the knowledge graph. If the graph lacks critical semantic relationships or contains incorrect relationships, the resulting embeddings may mislead the DRL agent, potentially degrading its performance. Our experiments were conducted in simulated environments to ensure controlled evaluations. However, since our observation is represented by an RGB image, transferring the learned policies to real-world robotic platforms would need an additional image-to-image translation method in order to efficiently bridge the simulation-to-reality gap.

8. Conclusions

In this paper, we demonstrate how adding contextual information about the environment, encoded in embeddings, helps DRL agents learn faster and more efficiently. These descriptions of the environment are obtained through the use of embeddings obtained by the selection of semantically relevant nodes in a Knowledge Graph. We have shown that adding these inputs to the DRL model does not significantly increase the computational time, and the impact on the model's architecture complexity is minimal. We propose an original architecture in which the embeddings are concatenated to the hidden activations of the FC layer that follows the representation learning blocks, thus complementing the visual information.

Overall, the results show an improvement in the agents' performance when KGEs are introduced. If the targets' colors are fixed, we noted a faster learning transitory regime, a decrease in the learning time, up to 25 % in the TIAGo setup, and an increase in the agent's accuracy by 12 % for both robots. If we apply DR to the targets' colors, the behavior of the transitory period remains better than the baseline, the decrease in the learning time rises up to 60 % in both robots, and the agent's accuracy is increased up to a 20 % for TIAGo and 10 % for IRB120. Additionally, we investigated the effect of KGEs on the joints' angle distribution to determine if there is an improvement in the agents' ability to exploit sooner the correct policy between the full KGE agent and the baseline.

For the experiments without DR, there are changes in the mean and standard deviation for some joints, but these are higher, and more joints are affected in the scenarios with DR. This suggests that, first, the use of KGEs enhances the agents' learning process and therefore, the learning time is reduced, the exploitation of the best policies happens earlier, and higher accuracies are achieved; and second, that providing the embedded information in a more complex problem leads to a more significant improvement than in simpler scenarios.

In conclusion, we have validated the proposed architecture that integrates contextual information about the environment, given as KGEs. We have proven that if DRL agents can leverage semantic knowledge, there is an improvement in their learning speed and success rate without significant increases in computational and architectural complexity.

Beyond the proposed application, the presented methodology shows promising capabilities in any scenario where DRL agents can leverage semantic knowledge sources. We leave for future work the analysis of how the agents' performance is affected depending on where the KGE input is placed and the sim-to-real transfer for both setups.

Author Contributions: Conceptualization, L.G-L., V.S., J.B., A.L-L. and D.N.; methodology, L.G-L, V.S., J.B., A.L-L. and D.N.; software, L.G-L and V.S.; validation, L.G-L and V.S.; formal analysis, L.G-L and V.S.; investigation, L.G-L and V.S.; resources, L.G-L and V.S.; data curation, L.G-L and V.S.; writing—original draft preparation, L.G-L and V.S.; writing—review and editing, L.G-L, V.S., J.B., A.L-L. and D.N.; visualization, L.G-L and V.S.; supervision, J.B., A.L-L. and D.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data will be available upon request.

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

References

1. Lazaridis, A. Deep Reinforcement Learning: A State-of-the-Art Walkthrough. *Journal of Artificial Intelligence Research* **2020**, *69*, 1421–1471. https://doi.org/10.1613/jair.1.12412.

2. François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning* **2018**, *11*, 219–354. https://doi.org/10.1561/2200000071.
3. Sutton, R.S.; Barto, A.G. *Reinforcement learning: an introduction*; MIT Press, 2018.
4. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: -, 2016.
5. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. https://doi.org/10.1038/nature14539.
6. Panzer, M.; Bender, B. Deep reinforcement learning in production systems: a systematic literature review. *International Journal of Production Research* **2022**, *60*, 4316–4341. https://doi.org/10.1080/00207543.2021.1973138.
7. Rupprecht, T.; Wang, Y. A survey for deep reinforcement learning in markovian cyber–physical systems: Common problems and solutions. *Neural Networks* **2022**, *153*, 13–36. https://doi.org/10.1016/j.neunet.2022.05.013.
8. Singh, B.; Kumar, R.; Singh, V.P. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review* **2022**, *55*, 945–990. https://doi.org/10.1007/s10462-021-09997-9.
9. Qureshi, A.H.; Nakamura, Y.; Yoshikawa, Y.; Ishiguro, H. Intrinsically motivated reinforcement learning for human–robot interaction in the real-world. *Neural Networks* **2018**, *107*, 23–33. https://doi.org/10.1016/j.neunet.2018.03.014.
10. Davidson, G.; Lake, B.M. Investigating Simple Object Representations in Model-Free Deep Reinforcement Learning. *Computing Research Repository (CoRR)* **2020**. https://doi.org/10.48550/arXiv.2002.06703.
11. Kroemer, O.; Niekum, S.; Konidaris, G. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of machine learning research* **2021**, *22*, 1–82.
12. Cabi, S.; Colmenarejo, S.G.; Novikov, A.; Konyushkova, K.; Reed, S.E.; Jeong, R.; Zolna, K.; Aytar, Y.; Budden, D.; Vecerík, M.; et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *Robotics: Science and Systems XVI* **2019**. https://doi.org/10.15607/rss.2020.xvi.076.
13. Wang, D.; Walters, R.; Zhu, X.; Platt, R. Equivariant *q* learning in spatial action spaces. In Proceedings of the 5th Conference on Robot Learning (CoRL). PMLR, 2022, pp. 1713–1723.
14. 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. *The International Journal of Robotics Research* **2022**, *41*, 690–705. https://doi.org/10.1177/0278364919868017.
15. Lee, Y.; Hu, E.S.; Lim, J.J. IKEA Furniture Assembly Environment for Long-Horizon Complex Manipulation Tasks. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 6343–6349. https://doi.org/10.1109/ICRA48506.2021.9560986.
16. Zhu, Y.; Wong, J.; Mandlekar, A.; Martín-Martín, R.; Joshi, A.; Nasiriany, S.; Zhu, Y. robosuite: A modular simulation framework and benchmark for robot learning. *Computing Research Repository (CoRR)* **2020**. https://doi.org/10.48550/arXiv.2009.12293.
17. Delhaisse, B.; Rozo, L.; Caldwell, D.G. PyRoboLearn: A Python Framework for Robot Learning Practitioners. In Proceedings of the Proceedings of the Conference on Robot Learning; Kaelbling, L.P.; Kragic, D.; Sugiura, K., Eds. PMLR, 30 Oct–01 Nov 2020, Vol. 100, *Proceedings of Machine Learning Research*, pp. 1348–1358.
18. Ammanabrolu, P.; Hausknecht, M. Graph constrained reinforcement learning for natural language action spaces. *Computing Research Repository (CoRR)* **2020**. https://doi.org/10.48550/arXiv.2001.08837.
19. Dambekodi, S.; Frazier, S.; Ammanabrolu, P.; Riedl, M.O. Playing text-based games with common sense. *Computing Research Repository (CoRR)* **2020**. https://doi.org/10.48550/arXiv.2012.02757.
20. Pternea, M.; Singh, P.; Chakraborty, A.; Oruganti, Y.; Milletari, M.; Bapat, S.; Jiang, K. The RL/LLM Taxonomy Tree: Reviewing Synergies Between Reinforcement Learning and Large Language Models. *J. Artif. Int. Res.* **2024**, *80*. https://doi.org/10.1613/jair.1.15960.
21. Xie, T.; Zhao, S.; Wu, C.H.; Liu, Y.; Luo, Q.; Zhong, V.; Yang, Y.; Yu, T. Text2reward: Automated dense reward function generation for reinforcement learning. *Computing Research Repository (CoRR)* **2023**. https://doi.org/10.48550/arXiv.2309.11489.
22. Carta, T.; Romac, C.; Wolf, T.; Lamprier, S.; Sigaud, O.; Oudeyer, P.Y. Grounding large language models in interactive environments with online reinforcement learning. In Proceedings of the 40th International Conference on Machine Learning. PMLR, 2023, pp. 3676–3713.
23. Quartey, B.; Shah, A.; Konidaris, G. Exploiting Contextual Structure to Generate Useful Auxiliary Tasks. *Computing Research Repository (CoRR)* **2023**. https://doi.org/10.48550/arXiv.2303.05038.

24. Du, Y.; Watkins, O.; Wang, Z.; Colas, C.; Darrell, T.; Abbeel, P.; Gupta, A.; Andreas, J. Guiding Pretraining in Reinforcement Learning with Large Language Models. In Proceedings of the Proceedings of the 40th International Conference on Machine Learning; Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; Scarlett, J., Eds. PMLR, 23–29 Jul 2023, Vol. 202, *Proceedings of Machine Learning Research*, pp. 8657–8677.
25. Reid, M.; Yamada, Y.; Gu, S.S. Can wikipedia help offline reinforcement learning? *Computing Research Repository (CoRR)* **2022**. https://doi.org/10.48550/arXiv.2201.12122.
26. Dasgupta, I.; Kaeser-Chen, C.; Marino, K.; Ahuja, A.; Babayan, S.; Hill, F.; Fergus, R. Collaborating with language models for embodied reasoning. *Computing Research Repository (CoRR)* **2023**. https://doi.org/10.48550/arXiv.2302.00763.
27. Li, K.; Wang, J.; Yang, L.; Lu, C.; Dai, B. SemGrasp: Semantic grasp generation via language aligned discretization. In Proceedings of the Computer Vision – ECCV 2024, Cham, 2025; pp. 109–127. https://doi.org/10.1007/978-3-031-72627-9_7.
28. Nie, M.; Chen, D.; Wang, D. Reinforcement learning on graphs: A survey. *IEEE Transactions on Emerging Topics in Computational Intelligence* **2023**, *7*, 1065–1082. https://doi.org/10.1109/TETCI.2022.3222545.
29. Miao, R.; Jia, Q.; Sun, F.; Chen, G.; Huang, H.; Miao, S. Semantic Representation of Robot Manipulation with Knowledge Graph. *Entropy* **2023**, *25*. https://doi.org/10.3390/e25040657.
30. Pagès, J.; Marchionni, L.; Ferro, F. TIAGo: the modular robot that adapts to different research needs.
31. ABB. IRB 120: Product specification, 2023. Accessed: 2024-06-28.
32. Todorov, E.; Erez, T.; Tassa, Y. MuJoCo: A physics engine for model-based control. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, 2012, pp. 5026–5033. https://doi.org/10.1109/IROS.2012.6386109.
33. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global vectors for word representation. In Proceedings of the Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543. https://doi.org/10.3115/v1/D14-1162.
34. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Harley, T.; Lillicrap, T.P.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48. JMLR.org, 2016, ICML'16, p. 1928–1937. https://doi.org/10.5555/3045390.3045594.
35. Gu, Z.; Jia, Z.; Choset, H. Adversary A3C for Robust Reinforcement Learning. In Proceedings of the International Conference on Learning Representations (ICLR), 12 2018. https://doi.org/10.48550/arXiv.1912.00330.
36. Grondman, I.; Busoniu, L.; Lopes, G.A.; Babuška, R. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* **2012**, *42*, 1291–1307. https://doi.org/10.1109/TSMCC.2012.2218595.
37. Babaeizadeh, M.; Frosio, I.; Tyree, S.; Clemons, J.; Kautz, J. Reinforcement Learning through Asynchronous Advantage Actor-Critic on a GPU. In Proceedings of the International Conference on Learning Representations (ICLR), 11 2017. https://doi.org/10.48550/arXiv.1611.06256.
38. Tieleman, T.; Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* **2012**, *4*, 26–31.