ISSN 1870-4069

Relational Representation for Knowledge Transfer in Reinforcement Learning

Armando Martínez, Eduardo F. Morales,
L. Enrique Sucar

Instituto Nacional de Astrofísica, Óptica y Electrónica,
Coordinación de Ciencias Computacionales,
Mexico

{armandomtzruiz, emorales, esucar}@inaoep.mx

Abstract. In reinforcement learning, an agent learns to behave, through trial and error, in a dynamic environment. This way of learning requires a considerable amount of data and time. Relational representation allows to abstract the state space, and therefore it can help the learning process to be faster, and also, abstract representations are easier to transfer to other similar domains that share these abstractions. In this paper we investigate whether the agent learns faster using these methods, as well as verify that this abstraction is transferable to other tasks. The proposed approach was evaluated on learning to play different ATARI games with promising results.

Keywords: Reinforcement learning, relational representation, causal models, transfer learning, object detection.

1 Introduction

Reinforcement learning is a paradigm of machine learning that studies how an agent maximizes a future reward it receives from the environment through its interactions with the environment [10]. At each iteration, the agent receives a signal from its current state s and selects an action a. The action possibly changes the state and the agent receives a reward signal r. The goal is to find a policy, which is a function that maps states to actions, that maximizes the total expected reward.

One of the problems with reinforcement learning is that the interactions of an agent and the environment in which it finds itself require a considerable amount of data and time for the agent to learn, so the learning process in this approach is time consuming. In addition, reinforcement learning suffers from its poor ability to generalize learned knowledge to new, although related problems.

Based on the above, the use of relational representation can allow us to abstract certain characteristics of the environment, so that the agent's learning can be accelerated [7]; also, this more abstract representation facilitates policy transfer to related tasks. In the experiments conducted in this work, the Arcade Learning Environment (ALE) library in Python was used to model Atari games as reinforcement learning environments. Relationships representing states within the environment were defined and the Q-Learning algorithm was adapted to incorporate relational representations into the agent's decision-making process.

Table 1. Summary of the most relevant related work.

Paper	Reference	Year	Approach	Key Contributions
Deep-Q Networks	[6]	2015	Deep Q-Learning	Introduced the Deep Q-Network (DQN) for Atari games, pioneering deep RL by achieving human-level performance. Leveraged experince replay and target networks for stability.
Continuous control with deep RL	[4]	2019	Deep Q-Learning	Introduces a deep RL algorithm for continuous control, leveraging an actor critic method with neural function approximators.
Transfer Learning in RL	[1]	2020	Knowledge Graphs, Transfer Learning	Proposed knowledge graph-based transfer learning, outperforming existing methods.
Towards deep symbolic RL	[3]	2016	Deep Learning, Symbolic Reasoning	Presented a hybrid approach fusing deep neural networks with symbolic reasoning for complex tasks, surpassing pure deep and symbolic methods in a 3D maze navigation task.
Deep Hierarchical Approach to Lifelong Learning in Minecraft	[11]	2016	Deep Hierarchical RL, Minecraft	Introduced hierarchical RL approach for lifelong learning in Minecraft with the Deep Skill Module, enabling high-level skill acquisition and reuse. Demonstrated adaptability to various tasks in Minecraft, with implications for complex, open-ended environments beyond gaming.

Although several relations can become true at certain stage, only one is considered, reducing this way the state-action space required.The experimental results obtained in learning to play several Atari games showed that the proposed approach achieved results comparable to existing works (e.g., DQN) in the selected games.

In particular, an improvement in learning speed and knowledge transfer between similar games was observed. Futhermore, it was shown that the incorporation of relational representations in the agent's decision making process allowed a better understanding of the environment and effective decision making.

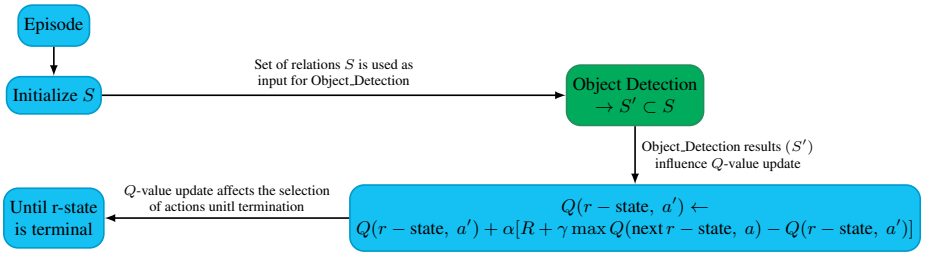

Fig. 1. Block diagram of the Q-Learning algorithm with the incorporation of the Object Detection system. The Object Detection system is used to detect and classify the agent and the objects in its environment (trophies, obstacles, and walls). By incorporating the Object Detection system into the Q-Learning algorithm, the agent can learn to associate its actions with the objects in its environment and learn an optimal policy more efficiently.

2 Background

2.1 Markov Decision Processes

A Markov decision process (MDP) models a sequential decision problem, where an agent controls a system in a dynamic environment. The solution of an MDP consists of a sequence of states that maximize the expected reward, called a policy. Formally, a Markov decision process is a stochastic process defined as follows: Let $S = \{s_1, s_2, \cdots, s_n\}$ and $A = \{a_1, a_2, \cdots, a_m\}$ be finite sets with $n, m > 0$, where S represents the set of states where the agent can be found and A represents the set of actions the agent can perform. Then, a Markov decision process is a tuple $M =< S, A, \phi, r >$, where ϕ is defined as:

$$\phi : A \times S \times S \longrightarrow [1, 0]. \tag{1}$$

Which represents the transition of the specified agent as a probability distribution, and R defined as:

$$r : S \times A \longrightarrow \mathbb{R}. \tag{2}$$

Is the reward function. A policy π is a function $\pi : S \longrightarrow A$ that determines what action $a \in A$ to perform given a state $s \in S$. We define an optimal action-value function $Q^*(s, a)$ as the maximum expected return value:

$$Q^*(s, a) = \max_{\pi} \mathbb{E}[R_t \,|\, s_t = s,\ a_t = a,\ \pi], \tag{3}$$

where $R = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ represents the future discounted return at time t.

3 Related Work

In this section, we briefly review previous work on deep RL, transfer learning and relational representations for RL.

Algorithm 1 RQ-Learning.

Algorithm parameters: $\alpha \in [0, 1]$, $\gamma \in [0, 1]$, $\epsilon > 0$ small (between 0.1 and 0.01).
Initialize $Q(s,\ a), \forall s \in S,\ a \in A$ in arbitrary way.

for each episode **do**
  Initialize S (set of relations)
  while r-state is not terminal **do**
    $S' =$ Object_Detection(frame) $\subset S$
    $r -$ state $=$ best_relation(S')
    Choose a' from s using policy derived from Q (ϵ greedy)
    Observe $R(r -$ state, $a') = R$
    $S'' \leftarrow$ Object_Detection(next frame) $\subset S$
    next_r $-$ state $=$ best_relation(S'')
    $Q(r-$state, $a') \leftarrow Q(r-$state, $a')+\alpha[R+\gamma \max_a Q($next_$r-$state, $a)-Q(r-$state, $a')]$
    $r -$ state $\leftarrow$ next_r $-$ state
  end while
end for

3.1 Deep Q-Networks

The fundamental work by Mnih et al. (2015) titled "Human-level control through deep reinforcement learning" [6] has made significant contributions to the field of deep reinforcement learning (DRL). The authors introduced the Deep Q-Network (DQN) algorithm, which is a powerful deep reinforcement learning algorithm capable of learning to play Atari games at or exceeding human-level performance. DQN is a deep neural network that predicts the expected reward for an action in a given state. This algorithm uses a strategy called experience replay, which consists in storing the agent's experiences in a replay memory and randomly samples from it during training.

An important work related to DQN is the work of Lillicrap et al. [4], where the focus is a DQN algorithm that handles a continuous action space, which uses an iterative optimization process to find the action that maximizes the action-value function in continuous-valued environments. In contrast with the work by Mnih, the approach introduced here employs an actor-critic method with neural function approximators to directly learn a policy from raw sensory input, enabling it to manage continuous and high-dimensional action spaces.

3.2 Transfer Learning in Reinforcement Learning

Rajeswaran et al. (2020) introduced a new way to improve deep reinforcement learning using prior knowledge defined as knowledge graphs [1], which are structured representations of information about a domain. These graphs include entities, relationships between entities, and attributes for each entity. This approach involves training a knowledge graph using data from a source task and then transferring this knowledge to a target task. They use the Deep Q-Network algorithm that incorporates the knowledge graph into the process of decision making by the agent. In their evaluation on different transfer learning tasks, they found that their approach performs better than existing methods.

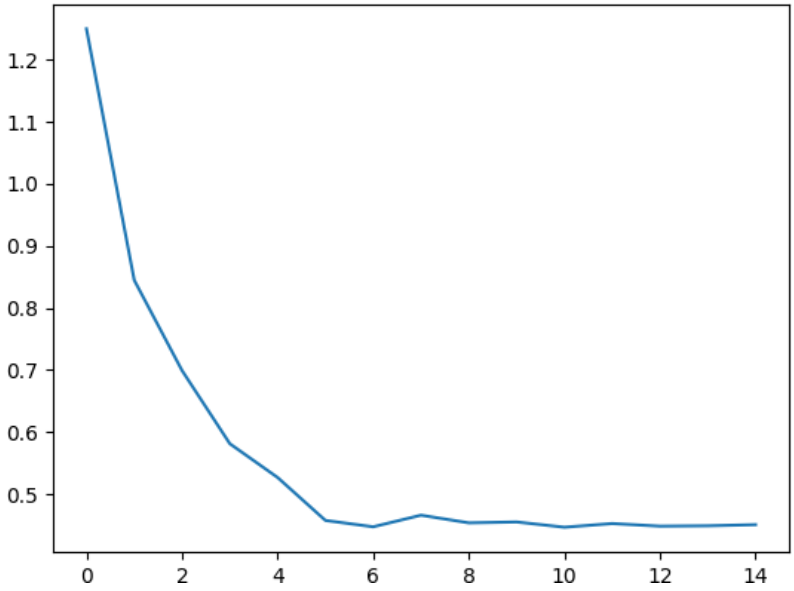

Fig. 2. Loss function.

They also show that the knowledge graph can be updated during learning, allowing the agent to adapt to changes in the environment. Tessler et al.'s paper, "A Deep Hierarchical Approach to Lifelong Learning in Minecraft," introduces a novel approach to reinforcement learning within the environment of the game Minecraft. The authors tackle the challenge of enabling an agent to acquire a diverse range of skills and adapt continuously to the dynamic Minecraft enfironment. They propose a hierarchical architecture, which uses the Deep Skill Module.

This module empowers the agent to learn, reuse, and combine high-level skills, which are generated separately by training the agent in specific tasks. The approach is grounded in a combination of reinforcement learning techniques and deep neural networks, which collectively provide a powerful framework for tackling the challenges posed by tasks in complex videogame environments, such as Minecraft.

3.3 Relational Representation in Reinforcement Learning

Garnelo et al. (2016) proposed an innovative way to tackle complex tasks in reinforcement learning [3]. They acknowledge that deep learning is powerful when it comes to learning from experience, meaning it can excel at learning patterns and representations from large amounts of data. It's particularly effective when dealing with high-dimensional inputs, such as images or raw sensor data.

However, there are certain tasks in reinforcement learning that require what they call "symbolic reasoning". Symbolic reasoning involves using logical rules and relations to understand abstract concepts and symbols. To overcome this, they present a hybrid approach that combines deep learning with a symbolic reasoning engine. This approach can be divided in two parts:

Fig. 3. Real-time object detection in a video game environment.

1. Deep learning component: This part uses a deep neural network to learn a policy from experience, where it was used a variant of the Deep Q-Network algorithm.
2. Symbolic reasoning engine: This component handles the symbolic reasoning of the agent within the environment. A first order logic engine is used to represent the actions performed by the agent in a symbolic manner.

To test their approach, the authors applied it to a 3D maze navigation task. They found that their approach outperformed both a pure deep learning approach and a pure symbolic reasoning approach. Table 1 summarizes the studies related to this work. In contrast to prior work, our approach directly employs images to extract object relations within the environment. This method may enhance the agent's learning efficiency, facilitating the transfer of the earned policy to similar games.

4 Proposed Approach

The block diagram in Figure 1 is a visual representation of the proposed approach. It is based on two main elements: (i) an object detection system that identifies and classifies the main visual elements in the game environment; and (ii) a relational representation of the states based on the detected objects and their spatial relations with the agent. Next we describe both elements.

4.1 State and Action Representation

A relational representation is a way of representing a state in the environment or knowledge about it in terms of entities and their relationships. This representation allows to model in a more expressive way the domains in such a way that some aspects of the environment can be generalized.

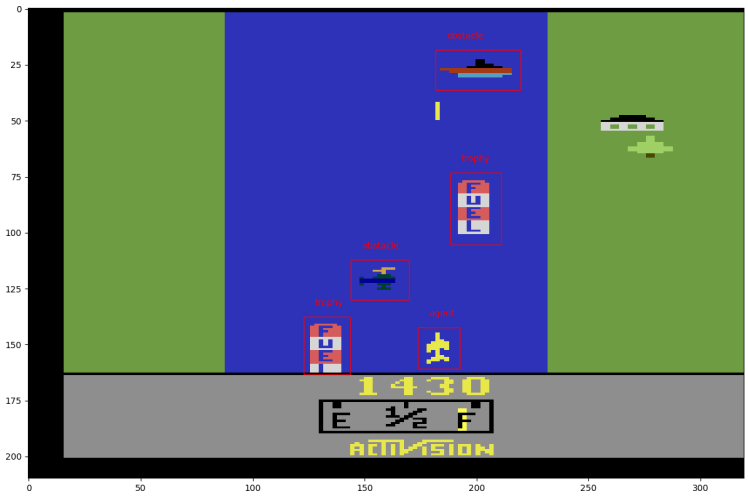

Fig. 4. Example of relational representation through object detection.

In this case, we used spatial relations to make an abstraction of the environment the agent is in. These relations where defined according to the games selected for the experiments, which are the following:

CloseToObject(Agent, Object),
FarFromObject(Agent, Object),
CloseToWall(Agent, Wall),

where Object can take the values obstacle or trophy, and Wall can take the values R, L, UR, UL, F, B to denote the right sidewall, left sidewall, right top wall, left top wall, top wall, and bottom wall respectively. Since first order logic allows to give truth value to these relations, one can represent the states of the environment with disjunctions between the relations, i.e., the states within the environment have the following possible representation:

CloseToObject(Agent, Object) $\vee$ CloseToWall(Agent, Object),
FarFromObject(Agent, Object) $\vee$ CloseToWall(Agent, Object).

Then, the set of states S was defined with 119 relations that allow to make an abstraction of the elements surrounding the agent. These relations were defined by considering possible combinations of variables (obstacle, trophy and wall), focusing solely on detectable combinations. Specifically, we considered only those relationships that can occur within the object detection process during the agent's trainnig.

In this paper, the states defined by relations are called $r-$states. The proposed approach is applied to ATARI 2600 game environments. The choice of this type of environments is due to the fact that important features can be generalized using relationships between the agent and its environment. The ALE library for Python allows to define the set of actions as integers, i.e., the set of actions is predefined according to the possible movements of the Atari control.

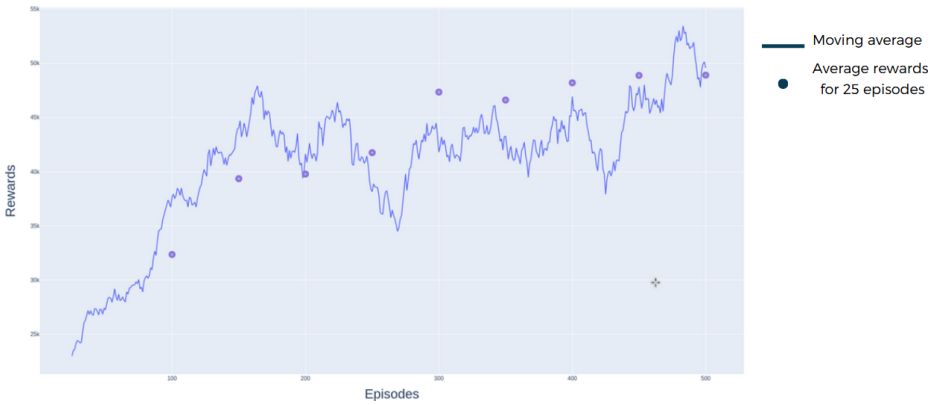

Fig. 5. Episodes vs Rewards. This graph represents the average rewards every 50 episodes to observe the increase in rewards as the episodes progress.

Since these 18 control actions (NOOP, up, down, left, right, upleft, upright, downleft, downright, fire, upfire, downfire, leftfire, rightfire, upleftfire, uprightfire, downleftfire and downrightfire) can be represented by integers in the order in which they were listed, the set of actions is then defined as follows:

$$\mathbb{A} = \{0,\ 1,\ 2,\ \cdots,\ 15,\ 16,\ 17\}. \tag{4}$$

Having the set of states represented as a set of relations and the set of actions defined as a set of integers representing each possible action that can be executed with the Atari controller, the reward function was defined as follows, given $k, m, n \in \mathbb{R}$:

$$R(r - \text{state},\ \text{action}) = \begin{cases} k & \text{if the agent is near a trophy,} \\ m & \text{if the agent is near an obstacle,} \\ n & \text{if the agent is near a wall.} \end{cases} \tag{5}$$

In this work, the decision was made to assign a higher reward to the agent for collecting trophies compared to the reward for moving away from an obstacle. In addition, it was defined that the highest reward the agent can receive is for moving away from the walls. To achieve this, the following value relation was established: $m < k < n$.

4.2 Object Detection

The incorporation of these elements to the Q-Learning algorithm is based on object detection, since through it the current state of the agent can be defined. The object detection system is implemented using the Python Detecto library, based on the Faster R-CNN architecture. This architecture consists of two modules: a deep convolutional network for region processing (the Region Proposal Network or RPN) and a Fast R-CNN detector using theses proposed regions.

Table 2. Comparison of average game scores between DQN and RQ-Learning approaches in Atari game environments.

Game	DQN	Frames	Q-Learning + OBD	Frames
River Raid	3166.2 (125.2)	10M	2667.1 (761.7)	8.4M

These regions define environmental objects, enabling us to identify spatial relations between the agent and its surroundings. Moreover, since object detection results in a subset S' of the set of states S, it is necessary to define a state selection criterion. In each state there can be relations with several objects. As the number of objects increases, the possible number of states grows exponentially (combinations of all the relationships of all the objects).

One way to avoid this and simplify the problem is to look only at the relationship with the object that yields the best value in the Q function. For this purpose, it was decided by choosing the relation or $r-$state with the largest Q-value, and in case of ties, a relation is chosen at random. Formally, the choice of the $r-$state can be represented by the following function:

$$r - state = \begin{cases} R_i & \text{if } \arg\max Q(S', a) = R_i,\ \forall a \in \mathbb{A},\ i = 1,\ 2,\ \cdots,\ |S'|, \\ \mathcal{R} & \text{otherwise,} \end{cases} \tag{6}$$

where $\mathcal{R}$ is a random variable with probability function:

$$P(\mathcal{R} = R_i) = \frac{1}{|S|},\ i = 1,\ 2,\ \cdots,\ |S'|. \tag{7}$$

Therefore, the proposed Q-Learning algorithm, which includes the incorporation of the object detection system is described in Algorithm 1. Notice that the function best_relation is equivalent to the defined selection of r-states based on the subset of relations found by the object detection system.

It is important to mention that the object detection is based on the images extracted in real time from the game during the episode. In addition, in order to compare the results obtained in this work with other similar ones, we chose to process every 4 frames the image of the game to extract the relationships of the agent with the environment.

5 Experimental Results

5.1 Arcade Learning Environment

For this work, the Arcade Learning Environment (ALE) library for Python was used to model Atari games as reinforcement learning environments. In particular, games were selected from the shoot'em up category, where the main objective is to survive waves of enemy attacks while trying to destroy them or evade obstacles and enemies. These games provide fast-paced gameplay challenges, with intense action and levels that increase in difficulty as the player progresses through the game. Video games serve as ideal environments for reinforcement learning for several reasons.

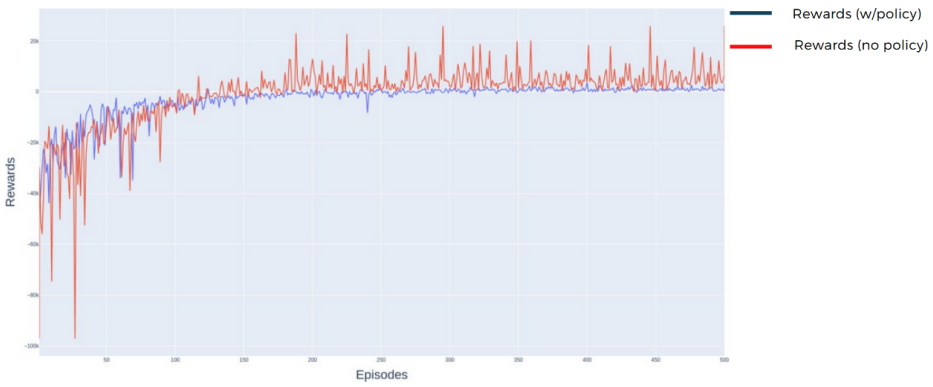

Fig. 6. Comparison of the performance of the agent in the game Chopper Command with a previously learned policy. The x-axis of the graph represents the number of episodes, while the y-axis represents the average reward obtained by the agent over the last 1000 episodes. The red line represents the performance of the agent when learning from scratch, while the blue line represents the performance of the agent when using a policy learned in a different game (River Raid) as a starting point.

Firstly, they offer well-defined settings with clear objectives and specific actions, enabling the direct observation and precise measurement of an agent's actions and outcomes. Secondly, games can simulate intricate scenarios, often impossible to replicate in reality, facilitating the exploration of reinforcement learning aspects such as decision-making, strategic planning, and adaptation to dynamic environments.

Thirdly, their scalability allows to assess an agent's ability to learn and adapt to increasingly complex tasks. Additionally, games generate substantial data, including state information, actions, and rewards, which prove invaluable for training and evaluating reinforcement learning algorithms. Lastly, the repetitive nature of games enables consistent task repetition, simplifying experimental replication and result comparisons among different approaches, making them a favored domain for machine learning research.

5.2 Object Detection

The object detection model was trained using a dataset comprising 65 images, aiming to accurately detect and classify both the agent and objects within its environment, including trophies, obstacles, and walls. Figures 2 and 3 illustrate the loss function graph for the detection model and provide an example of object detection within an ATARI 2600 game environment.

This detection process is conducted periodically during the agent's training via the ALE library. It involves extracting the image at a specific time t and subsequently processing it through the trained object detection system model. The model generates a set of tensors that represent the coordinates of the bounding boxes' extreme points for objects detected within the given frame.

The relations between the agent and surrounding objects are defined based on object detection. Based on Figure 4, one can define the relation of the agent and the nearest object (obstacle) using first-order logic [9] as follows:

$$\begin{gathered}\text{CloseToObject}(a,\ o), \text{ which is true, if agent } a \text{ is near object } o,\\ \text{CloseToWall}(a,\ o), \text{ which is true, if agent } a \text{ is near wall } w,\\ \text{FarFromObject}(a,\ o), \text{ which is true, if agent } a \text{ is far from object } o.\end{gathered}$$

For the objects, the relations were defined using the euclidean distance between the centroid of the agent and the centroid of the object. Let $m,\ n \in \mathbb{R}$ different than 0 with $m < n$, if $(x_a,\ y_a)$ represents the centroid of the agent and $(x_o,\ y_o)$ represents the centroid of the obstacle, then the relation CloseToObject$(a,\ o)$ is true if:

$$d((x_a,\ y_a),\ (x_o,\ y_o)) = \sqrt{(x_a - x_o)^2 + (y_a - y_o)^2} < m. \tag{8}$$

And the relation FarFromObject(a, o) is true if:

$$m < d((x_a, y_a), (x_o, y_o)) < n. \tag{9}$$

In the case of walls, the CloseToWall relation was defined using the standard Euclidean distance between two real numbers, along with constraints imposed on the position of the agent with respect to the position of the wall. So the first-order logic expression of the relation is:

$$\begin{gathered}\exists a[\text{Rel}(a,\ \text{obstacle},\ \text{left},\ \text{up}, \emptyset) \iff\\ \text{CloseToObject}(a,\ \text{obstacle}) \vee \text{CloseToWall}(a,\ w) \wedge x = \text{left} \wedge y = \text{up} \wedge w = \emptyset],\\ \text{obstacle} \in \hat{O},\ x \in \hat{X},\ y \in \hat{Y},\ w \in \hat{W},\end{gathered} \tag{10}$$

where $\hat{O} = \{\emptyset,\ \text{trophy},\ \text{obstacle}\}$, $\hat{X} = \{\emptyset,\ \text{left},\ \text{right}\}$, $\hat{Y} = \{\emptyset,\ \text{up},\ \text{down}\}$, $\hat{W} = \{\emptyset,\ R,\ L,\ UR,\ UL,\ F,\ B\}$ are the domains of o, x, y and w respectively. Notice that the relation FarFromObject$(a,\ \text{trophy})$ is also true for some adequate values for m and n. Utilizing these relations, a collection of states was established to determine the agent's current state based on its interactions with the elements within the environment. It's worth noting that these relations were initially formulated to enable the agent to make decisions using straightforward spatial directions, thus simplifying the set of states.

5.3 Learning to Play River Raid

The agent underwent training in Atari's River Raid game utilizing the proposed algorithm across 500 episodes with with parameter settings of $\epsilon = 0.1$ and $\gamma = 0.6$, with periodic testing conducted every 50 episodes to assess the effectiveness of the policy generated up to that point. The outcome of this experiment is depicted in Figure 5. Notably, the experimental results exhibit an upward trend as the episodes progress, displaying a positive linear trajectory. The selection of the hyperparameters ϵ and γ was guided by Machado et al.'s research [5], which compiles data from experiments conducted across various Atari games using the DQN algorithm, selected as a benchmark for comparison in this study.

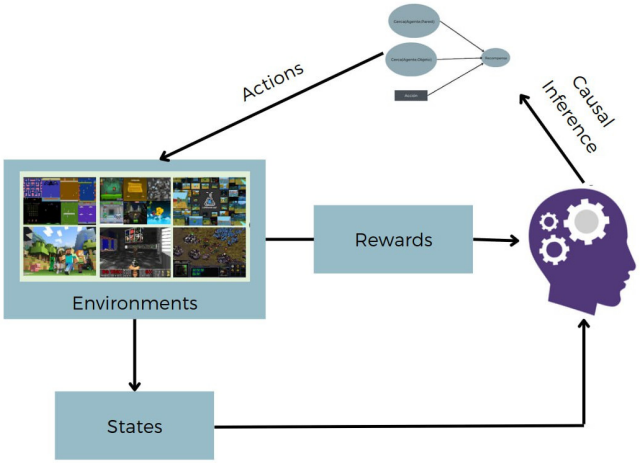

Fig. 7. Incorporation of causal models based on the relationships obtained from object detection for different similar tasks.

Table 2 presents the average scores obtained from 100 episodes using the proposed approach and a known algorithm used in Atari game environments [5]. Values in parentheses indicate the corresponding standard deviations. It can be seen that the RQ-Learning approach obtained comparable results with DQN in a smaller number of frames. While the outcomes between the suggested approach and DQN might exhibit similarities in certain scenarios, it's crucial to acknowledge that due to the stochastic nature of video games, these results can exhibit substantial variations. Additionally, it's important to consider the necessity of pre-training the object detection model when aiming for knowledge transfer, as this model must be adapted with information from the new environment before agent training can commence.

5.4 Transfer Learning

Building upon the previous results, the subsequent experiment involved applying the policy derived from 1000 episodes of gameplay in River Raid to another game. Chopper Command was selected for this task due to its shared objectives and similarities with River Raid. Initially, the agent attempted independent policy learning, followed by the application of a pre-existing policy to evaluate the agent's capacity to leverage this acquired knowledge effectively. This policy is structured as a Q-table and is deployed by the agent during the algorithmic step where it selects the most suitable relation during object detection.

Given that the relationships within the set of states remain consistent across both games, the transference of this environmental abstraction is feasible. From the results we can see in Figure 6, the agent starts with higher reward values when transfer is made. On the other hand, learning is more stable. These results lead us to conjecture that knowledge transfer using a policy learned in one game to another is possible if the environments have enough similar characteristics that can be represented with relations.

5.5 Discussion

The results obtained so far show us that the object detection system works adequately under the conditions of the reinforcement learning environment. This is due to the object detection system's ability to not only classify objects within a given image but also provide valuable geometric information about them. By leveraging this spatial information, it becomes possible to establish spatial relations between these objects and define a set of states for the agent.

Furthermore, in the case of the experiment in learning to play River Raid, it can be observed that the agent reaches a new maximum reward approximately every 100 - 150 episodes, so it can be conjectured that given more training time, the agent will find an optimal policy as the number of episodes increases. Also, the preliminary results in transfer learning suggest that knowledge transfer between games is possible when they share similar characteristics represented by relations.

6 Future Directions

Based on the relationships identified by the detection system, the proposed approach suggests using a causal model derived from these relationships to represent the immediate effect of an action on a given set of relationships as illustrated in Figure 7. The idea is that the agent learns a causal model at the same time it is learning a policy, based on this abstract relational representation. Once it has a partial model [8], it can use it in selecting the appropiate actions (for instance, those that maximize the reward), and thus accelerate even more the learning process. This is left as future work.

7 Conclusions

The use of relational representation and causal models can improve an agent's ability to understand complex environments, resulting in more efficient decision-making. This work proposes a way to incorporate an external sensing system into reinforcement learning environments, particularly in Atari games, and provides some initial findings on training the agent from relations.

It is expected that these results will lead to a way of transferring the knowledge acquired by the agent to similar tasks, thus defining an approach that combines elements of computational vision and relational representation with reinforcement learning. The results obtained from the proposed approach showed that it achieved results comparable to existing works (e.g., DQN) in the selected games. In particular, an improvement in learning speed and knowledge transfer between similar games was observed.

The proposed approach suggests using a causal model derived from the relationships identified by the detection system to represent the immediate effect of an action on a given set of relationships and the immediate reward. Through this study, we can speculate that this approach might enhance the agent's environmental understanding, resulting in more effective decision-making and knowledge transfer to tasks sharing a similar causal structure, facilitated by abstract environmental relationships.

References

1. Ammanabrolu, P., Riedl, M.: Transfer in deep reinforcement learning using knowledge graphs. In: Proceedings of the 13th Workshop on Graph-Based Methods for Natural Language Processing, pp. 1–10 (2019) doi: 10.18653/v1/d19-5301
2. Bellemare, M. G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, vol. 47, pp. 253–279 (2013) doi: 10.1613/jair.3912
3. Garnelo, M., Arulkumaran, K., Shanahan, M.: Towards deep symbolic reinforcement learning. pp. 1–13 (2016) doi: 10.48550/ARXIV.1609.05518
4. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: Proceedings of the International Conference on Learning Representations, pp. 1–14 (2016) doi: 10.48550/ARXIV.1509.02971
5. Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., Bowling, M.: Revisiting the arcade learning environment: evaluation protocols and open problems for general agents. Journal of Artificial Intelligence Research, vol. 61, no. 1, pp. 523–562 (2017) doi: 10.48550/ARXIV.1709.06009
6. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature, vol. 518, no. 7540, pp. 529–533 (2015) doi: 10.1038/nature14236
7. Morales, E. F.: Scaling up reinforcement learning with a relational representation. In: Proceedings of the Workshop on Adaptability in Multi-Agent Systems, pp. 15-26 (2003)
8. Méndez-Molina, A., Feliciano-Avelino, I., Morales, E. F., Sucar, L. E.: Causal based Q-learning. Research in Computing Science, vol. 149, no. 3, pp. 95–104 (2020)
9. Raedt, L. D., Kersting, K., Natarajan, S., Poole, D.: Statistical relational artificial intelligence: Logic, probability, and computation. Synthesis Lectures on Artificial Intelligence and Machine Learning, 1st Edition (2016) doi: 10.1007/978-3-031-01574-8
10. Sutton, R. S., Barto, A. G.: Reinforcement learning: An introduction. The MIT Press (2020)
11. Tessler, C., Givony, S., Zahavy, T., Mankowitz, D., Mannor, S.: A deep hierarchical approach to lifelong learning in minecraft. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, no. 1, pp. 1553–1561 (2017) doi: 10.1609/aaai.v31i1.10744