

# Interpretable end-to-end Neurosymbolic Reinforcement Learning agents

Nils Grandien<sup>1</sup>

Quentin Delfosse<sup>1,2</sup>

Kristian Kersting<sup>1,3,4,5</sup>

<sup>1</sup>Computer Science Department, TU Darmstadt, Germany

<sup>2</sup>National Research Center for Applied Cybersecurity Darmstadt, Germany

<sup>3</sup>Hessian Center for Artificial Intelligence (hessian.AI), Darmstadt, Germany

<sup>4</sup>Centre for Cognitive Science, TU Darmstadt, Germany

<sup>5</sup>German Research Center for Artificial Intelligence (DFKI), Darmstadt, Germany

correspondance to [quentin.delfosse@cs.tu-darmstadt.de](mailto:quentin.delfosse@cs.tu-darmstadt.de)

## Abstract

Deep reinforcement learning (RL) agents rely on shortcut learning, preventing them from generalizing to slightly different environments [1]. To address this problem, symbolic method, that use object-centric states, have been developed. However, comparing these methods to deep agents is not fair, as these last operate from raw pixel-based states. In this work, we instantiate the symbolic Successive Concept Bottlenecks Agents (SCoBots) framework [2]. SCoBots decompose RL tasks into intermediate, interpretable representations, culminating in action decisions based on a comprehensible set of object-centric relational concepts. This architecture aids in demystifying agent decisions. By explicitly learning to extract object-centric representations from raw states, object-centric RL, and policy distillation via rule extraction, this work places itself within the neurosymbolic AI paradigm, blending the strengths of neural networks with symbolic AI. We present the first implementation of an end-to-end trained SCoBot, separately evaluate of its components, on different Atari games. The results demonstrate the framework’s potential to create interpretable and performing RL systems, and pave the way for future research directions in obtaining end-to-end interpretable RL agents.

## 1 Introduction

Despite ongoing advancements in the field, reinforcement learning (RL) continues to face numerous challenges. One such challenge is the sparsity of rewards [3], where the environment only rarely provides reward signals for the agent to learn from. A related issue is credit assignment [4; 5], which refers to the challenge of identifying the specific previous actions responsible for distant future rewards. Additionally, RL agents are susceptible to learn misaligned goals [6; 1], which occurs when the objectives optimized by the RL algorithm diverge from the intended goals of the system’s designers. The black box nature of current deep RL approaches impedes the ability to address these challenges. Even though there have been attempts of shedding light into the black box via approaches from the field of eXplainable AI (XAI) [7; 8], there is still room for improvement. The majority of the developed approaches rely on post-hoc explanations, which frequently result in a lack of faithfulness of the explanations [9; 10]. This makes it challenging to analyze an agent’s policy.

To address the lack of interpretability, we instantiate the recently proposed SCoBots framework [2]. This approach uses an architecture that achieves interpretability by design. SCoBots decompose the RL problem via concept bottleneck models [11] with intermediate interpretable representations. The

---

final action selection operates on a set of interpretable relational concepts and uses an inherently interpretable model, in our implementation a rule set policy. SCoBots facilitate human understanding of the agent and can, thereby, aid in the development and training of performing RL agents. As a side benefit, the interpretability of the model can improve trust into the RL agent, which can be crucial for deployment in the real world. Additionally, by training the RL algorithm on a set of relational concepts instead of a sequence of raw input images the complexity of the problem is being reduced, thereby improving sample efficiency [12].

By instantiating the SCoBots framework, we cover the fields of object representation learning, object-centric RL and policy distillation. Overall, this leads to a neurosymbolic AI system that combines the strengths of neural networks with symbolic AI. Neural networks are used for object representation learning and the initial RL algorithm. The utilization of structured object-centric intermediate representations and the final step of transforming a neural policy into a rule set policy also renders the approach symbolic.

Previous work has led to components that could be suitable for the steps in the SCoBots framework [13; 14; 15]. However, these components have not yet been combined and the SCoBots framework has so far only been evaluated using ground truth detection of the objects. In this work, **we introduce the first end-to-end Concept bottleneck agents that employs unsupervisedly trained components<sup>1</sup>**. As part of this, we evaluate the individual components that were presented in previous works. The experimental setting, in which we evaluate the SCoBots, are Atari games. OCAtari [16] provides access to many such games including ground truth detection of objects.

## 2 Background

### 2.1 SCoBots

In the Successive Concept Bottlenecks Agents (SCoBots) framework [2] (*cf.* Figure 1), the policy of an agent is decomposed into distinct steps with intermediate interpretable concept bottlenecks (ICBs) inspired by concept bottleneck models [11].

$$s_t \xrightarrow{\omega_{\theta_1}} \Omega_t \xrightarrow{\mu_{\mathcal{F}}} \Gamma_t \xrightarrow{\rho_{\theta_2}} a_t$$

This differs from the standard deep RL approach, in which the raw input is processed to directly derive the selected action without any structured intermediate steps.

**Object Extractor:**  $s_t \xrightarrow{\omega_{\theta_1}} \Omega_t$  The object extractor, denoted as  $\omega_{\theta_1}(\cdot)$ , extracts objects and their properties from the  $n$  most recent frames, represented by  $s_t = \{x_i\}_{i=t-(n-1)}^t$ . As a result, a collection of object representations is returned:  $\omega_{\theta_1}(s_t) = \Omega_t = \{o_i^j\}_{j=1}^{c_t}$ . Here,  $c_t$  refers to the number of detected objects. The object representations  $o_i^j$  are tensors that capture a multitude of properties of each object (*e.g.*, `position`).

**Relation Extractor:**  $\Omega_t \xrightarrow{\mu_{\mathcal{F}}} \Gamma_t$  In this step, relational concepts are derived from the previous output via the relation extractor:  $\mu_{\mathcal{F}}(\cdot)$ . Here,  $\mathcal{F}$  parameterizes a set of relational functions that includes general object relations like `distance` and `speed`. Formally, we denote this step as  $\mu_{\mathcal{F}}(\Omega_t) = \Gamma_t = \{g_k^t\}_{k=1}^{d_t}$ , where  $d_t$  quantifies the relational concepts.

**Action Selector:**  $\Gamma_t \xrightarrow{\rho_{\theta_2}} a_t$  Finally, the action selector, denoted as  $\rho_{\theta_2}$ , determines the action,  $a_t$ , from the relational concepts. In contrast to the earlier stages where ICBs provided sufficient interpretability, in this stage the action selector itself must be interpretable to enable overall interpretability (*e.g.*, by using decision tree or rule set policies).

---

<sup>1</sup>Code at <https://github.com/nlsgrndn/SCoBots/tree/dev>, [https://github.com/k4ntz/SCoBots/tree/space\\_detector](https://github.com/k4ntz/SCoBots/tree/space_detector)

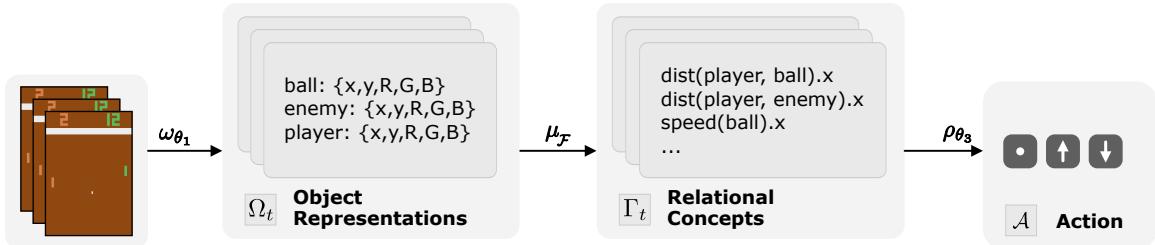


Figure 1: **Overview of the SCoBots framework.** SCoBots decompose the policy into 3 consecutive steps: object extraction, relation extraction, and action selection using intermediate Interpretable Concept Bottlenecks (ICBs). This enables external users to inspect how the SCoBot agent selects its action. Figure adapted from [2].

## 2.2 SPACE

SPACE [13] is Variational Autoencoder (VAE)-based architecture for unsupervised object-oriented scene representation learning (*cf.* Figure 2). Its latent space is designed to represent location-related information (*i.e.* the object position) and the feature-related information (*i.e.* the object visualization) of each object, and an encoding of the background information separately. SPACE is trained using a standard VAE reconstruction loss.

## 2.3 MOC

To address the insufficient performance of both object localization and representation learning of SPACE, a follow up work has added 2 loss terms within the Motion and Object Continuity (MOC) training scheme [14]. This is an approach that can be applied to any base detection model to improve the object locations (loc) and encodings (enc). The motion supervision loss utilizes additional motion information to improve localization variables (*i.e.* loc and pres). The object continuity loss is designed to gather object encodings of the same entity across successive frames, and to separate encodings of the different objects. The MOC training scheme is depicted in Figure 2.

## 2.4 ECLAIRE

ECLAIRE [15] is a rule extraction method for deep neural networks. The input for ECLAIRE is a set of unlabeled training instances  $X = \{x^{(i)} \in \mathbb{R}^m\}_{i=1}^N$  and a pre-trained neural network  $f_\theta : \mathbb{R}^m \rightarrow [0, 1]^L$ . The function  $f_\theta(x)$  outputs a probability distribution over labels in the set  $Y = \{l_1, l_2, \dots, l_L\}$ . The output is a set of IF-THEN rules, denoted as  $R_{x \rightarrow \hat{y}}$ . These rules are designed to collectively predict the outcome that corresponds to the maximum value in the output vector of the network  $f_\theta$  when subjected to a majority vote given the input  $x$ . A single rule is formalized as follows:

$$\text{IF } ((x_i > v_i) \wedge (x_j \leq v_j) \wedge \dots \wedge (x_n > v_n)) \text{ THEN } l_k$$

In this structure,  $x_i$  represents the  $i$ -th feature of an input instance  $x$ , while  $v_i$  is a threshold value determined through the learning process. These rules are composed of premises that are conjunctions of conditions like  $(x_i > v_i)$  or  $(x_i \leq v_i)$ .

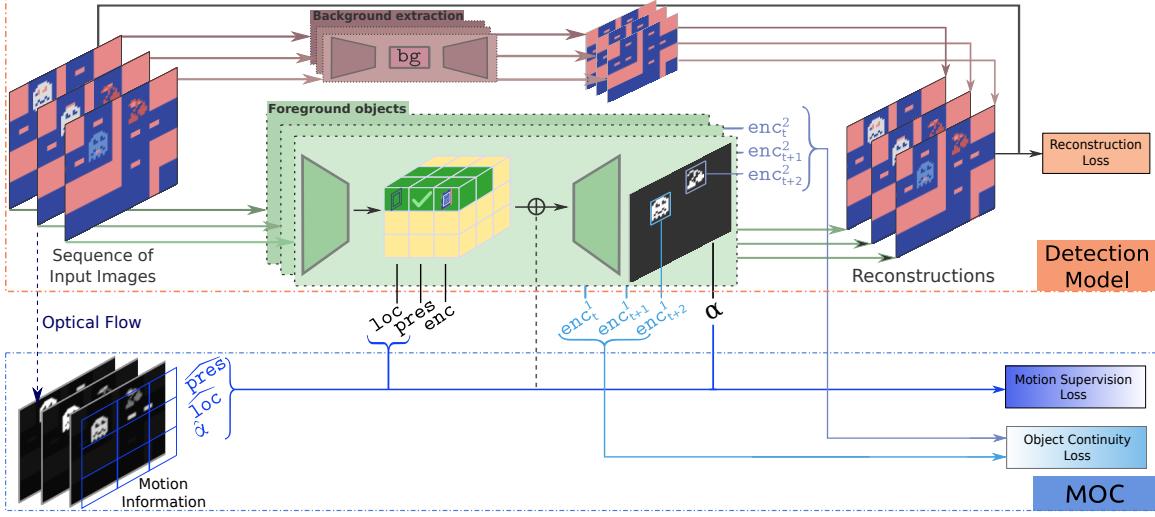


Figure 2: **Overview of MOC applied to SPACE.** SPACE learns to extract objects using a VAE architecture. The reconstruction problem is split into foreground and a background components. The foreground latent space is composed of positional and feature information about the objects in the image. The MOC training scheme adds the motion supervision loss and the object continuity loss to improve a base detection model, SPACE in this case. The motion supervision loss is designed to enhance the localization capabilities by guiding the locations (loc) with motion data of the input image. The object continuity loss is applied to feature encodings (enc) of the objects in consecutive images with the goal of ensuring consistency of the encodings representing the same entity.

### 3 Implementation

Let us introduce our DINSA method (*cf.* Figure 3). DINSA uses the SPACE [13] architecture, trained with the motion and object consistency (MOC) loss of [14]. It then use a k-mean classifier to classify each object, then uses a simple object tracking algorithm to stabilize the detection. The object-centric space is then augmented with relations, provided to an neural action selector (distilled in a set of rules). Let us now detail each component.

#### 3.1 Object Extractor

The object extractor receives the last  $n$  frames of the game as input. It returns the objects of the current frame together with their properties. These properties can be time-related, which is why a sequence of images is given as input.

##### 3.1.1 SPACE+MOC for Object Representation Learning

This component receives an image as input and produces two key outputs: a bounding box for each detected object and an encoding for each object. We use the SPACE architecture and enhance it using the MOC training scheme. The desired outputs are obtained from the latent space of the VAE architecture of SPACE.

During inference, a set of objects with bounding box and encoding information must be obtained. To this end, only the encoder parts of the foreground module of the SPACE model are required. The other components of SPACE and the MOC training scheme are only needed for training. The variables  $z^{pres}$ ,  $z^{where}$  and  $z^{what}$  are extracted from the SPACE architecture. The value of  $z^{pres}$  is thresholded, resulting in a binary variable indicating the presence of an object. For the cells in which an object is present, the  $z^{where}$  information is transformed into a bounding box and the  $z^{what}$  encoding is saved. For implementation details and hyperparameter values, see App. A.1.2.

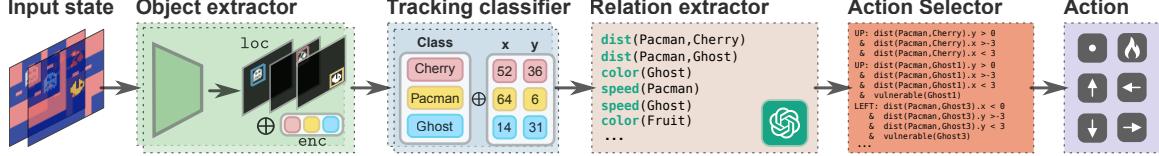


Figure 3: **Overview of the DINSA implementation** The input images are fed consecutively into the object extractor. The resulting objects represented by location and encoding are classified into a specific object class of the respective game. Also, object identity between consecutive frames is inferred via object tracking. This results in a set of objects with properties, including time-related ones. The relation extractor computes inter- or intra-object relations. Finally, the action selector decides, which action to take using an interpretable algorithm.

### 3.1.2 Object Classification

This component is designed to classify objects based on their feature encodings as inputs. In the context of the SPACE+MOC model, these encodings are provided by the  $z^{what}$  latent variable. The output of the classification component is a label assigned to each object.

The classifier should be unsupervised in order to not break the overall unsupervised setting. Similar to [17] and [18], we classify the representation of an object based on its distance to the centroids that result from applying k-means clustering to a training set of encodings. A full description of the algorithm can be found in the Appendix at A.1.3.

### 3.1.3 Object Tracking

The property **position history** is usually required as part of the object representations, as it is essential for computing time-related relational concepts such as **speed** in the downstream relation extractor. In order to include this property, the number of image frames  $n$  to be contained in the state  $s_t = \{x_i\}_{i=t-(n-1)}^t$  must be set to (at least) two. Moreover, it is necessary to determine which of the localized objects represent the same entity across the sequence of frames. In [2], this information was provided implicitly as part of the ground truth detection via OCAAtari. Our solution approach is to use a simple tracking algorithm on top of the single frame localizations by SPACE+MOC. More details are provided in the Appendix at A.1.4.

## 3.2 Relation Extractor

The relation extractor uses the extracted objects' properties from the current frame as input, and outputs a vector containing the values of the relational concepts for the detected objects.

The SCoBots framework includes a complete implementation of the relation extractor. In this implementation, the number of detectable objects per class are specified in advance, resulting in a set of unique identifiers for potentially detected objects. The relational concepts are then defined relative to these identifiers using straightforward functions, such as Euclidean distance, to generate scalar values for each concept. During inference, detected objects are mapped to an identifier by sorting them based on their proximity to a key object (e.g., the player), with excess objects discarded and missing ones assigned zero values. For a more detailed description, see App. A.2.

## 3.3 Action Selector

The action selector component receives the feature vector from the relation extractor and returns an action. The learning phase for our implementation of this component is divided into two steps. First, a neural policy is learned using standard deep RL techniques. Then, this policy is transformed into an interpretable representation that uses a set of rules. This transformation is a trade-off between maintaining the policy's similarity and finding a small, easily interpretable set of rules.

### 3.3.1 Deep Reinforcement Learning

This component learns a neural policy based on the relational concepts. This neural policy determines the actions or decisions made in a given context based on the input information. We utilize Proximal Policy Optimization (PPO) [19] to learn the neural policy. Other RL algorithms that can handle continuous state spaces and discrete action spaces could have also been used. Details on the choice of the hyperparameters are provided in App. A.3.2.

### 3.3.2 Policy Distillation via ECLAIRE

In this step, we transform the neural policy into a rule set policy using ECLAIRE [15]. Applied to our case, the training instances  $X$  in ECLAIRE are the one-dimensional vectors, which the relation extractor provides. The neural network  $f_\theta$  is the policy network learnt using the PPO algorithm.  $Y$  is the discrete action space of the respective game.

## 4 Experimental Evaluation

In our experiments, we successively evaluate the different components of our SCoBots [2] instantiation. First, the object extraction is evaluated. As the relation extractor only applies deterministic functions to object properties, it is not investigated separately. Second, the action selector is analyzed including both the preliminary neural policy and the final rule set policy.

The Pong, Boxing and Skiing environments (depicted in Figure 4) were used in the experiments for the object extractor. Only Pong and Boxing remained for the action selector experiments, as Skiing is a difficult credit assignment problem, that requires additional techniques to be solved (*cf.* App. A.1.1). The object extractor focuses on the moving objects of the games, only considering the relevant objects for playing the games (*i.e.* excluding *e.g.*, scores or the timer in Boxing). This was realized by applying a filter based on potential detection areas of the moving objects (*cf.* App. A.1.2).

### 4.1 Object Extractor

In this subsection, we present the evaluation of our object extractor. However, we only included the localization and encoding component plus the classifier, but did not include the object tracker. The scores were calculated relative to *all* ground truth objects and not only relative to the *localized* objects. This allowed us to obtain a better understanding of how the object extractor would behave for the downstream task.

Overall, the F-score was the best for Boxing (*cf.* Figure 5). Pong had a slightly lower F-score due to the poor recall for the localization of the ball object. Both are likely to be suitable for the downstream task. Skiing performed the worst, which can be explained by the poor classifier performance. The average correct detection of only four out of five ground truth objects is unlikely to be sufficient for the downstream task. In particular, when we examined the confusion matrix (*cf.* Figure 6), we observed that the detection of the player object was problematic, arguably the most important object. Many trees were classified as the player. This behavior would confuse the downstream RL algorithm.



Figure 4: **Visualization of the games used in our experimental evaluation.** The Boxing, Skiing and Pong Atari environment are shown from left to right. While Boxing and Pong were also used for RL evaluation, the difficult credit assignment game Skiing is only used for object detection.

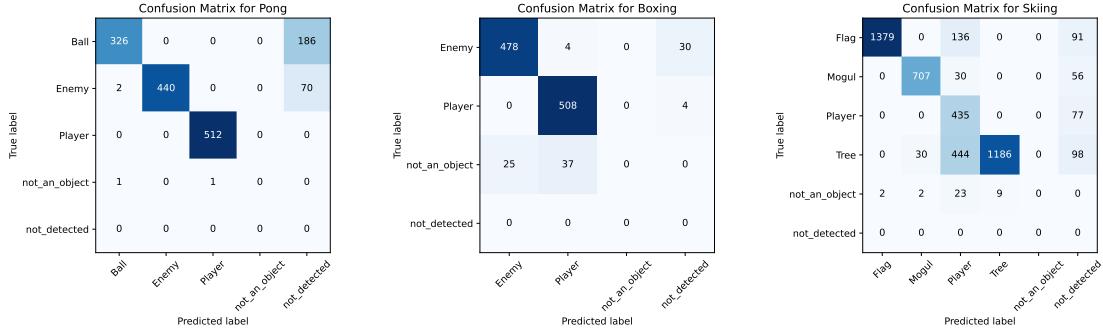


Figure 6: **Confusion matrices for each object type per game.** *not\_an\_object* refers to bounding boxes returned by SPACE+MOC that do not correspond to a ground truth object. *not\_detected* refers to objects provided by OCAtari that were not localized by SPACE+MOC.

## 4.2 Action Selector

Let us here evaluate the action selection process.

### 4.2.1 Deep Reinforcement Learning

This experiment assesses the performance of RL agents equipped with neural policies, with object-centric state input that are provided via the components of the SPACE+MOC object extractor and relation extractor.

The results for Pong reveal that an agent employing the SPACE+MOC object extractor can achieve comparable performance to an agent utilising a ground truth object extractor, provided that the two hidden layer configuration is used (*cf.* Figure 1). The outcomes of the Boxing experiment indicate that the SPACE+MOC object extractor is too inaccurate for use in competitive object-centric agents. The agents using the SPACE+MOC data achieve poor performance compared to the agents using ground data. The latter agents perform well in particular for both of the unpruned configurations. To summarize, the experiment indicates that the SCoBots framework’s modular design which enhances interpretability and allows for incremental component upgrades, can come at the cost of error accumulation.

### 4.2.2 Policy Distillation

The goal of this experiment was to ascertain the final performance score for our SCoBots implementation. Additionally, we sought to determine the extent to which performance is diminished by extracting the rule set policy from the neural policy. We also aimed to understand how the size of the neural network and of the feature vector for the relational concepts affect the performance.

The findings suggest the benefits of using the pruned and two-layer configuration for distillation via ECLAIRE, although the trends are not entirely clear (*cf.* Figure 1). Notably, a configuration for a SCoBot using SPACE+MOC input and a rule set policy was identified that achieved a respectable average reward of 14.4 in the game Pong. The best configuration for Boxing using SPACE+MOC input and a rule set policy achieved an average reward of 51.8, although using the unpruned configuration.

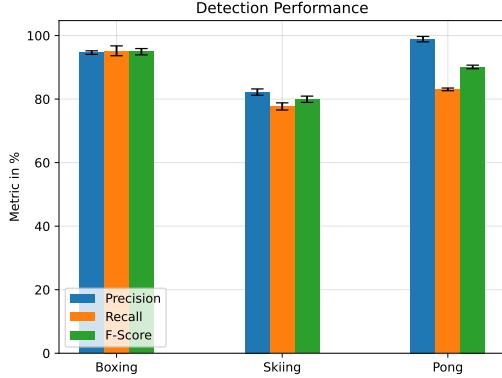


Figure 5: **Object extractors for Boxing and Pong demonstrate high performance, while Skiing shows lower performance.** Precision, recall, and F-score for the combined object extractors of SPACE+MOC and classifier are shown. Results are averages with standard deviations across five SPACE+MOC models trained with different seeds and their corresponding classifiers.

Pong	unpruned		pruned	
	2 layer	1 layer	2 layer	1 layer
neural ground truth objects	17.4 $\pm$ 1.6	17.0 $\pm$ 2.3	19.0 $\pm$ 1.9	14.6 $\pm$ 1.0
rule set ground truth objects	4.8 $\pm$ 5.7	-15.8 $\pm$ 3.1	15.0 $\pm$ 3.0	10.6 $\pm$ 2.4
rule set SPACE+MOC objects	-7.6 $\pm$ 9.2	-3.2 $\pm$ 6.3	14.4 $\pm$ 2.6	-3.8 $\pm$ 3.4
neural SPACE+MOC objects	16.8 $\pm$ 1.5	5.4 $\pm$ 3.1	16.8 $\pm$ 2.3	-1.0 $\pm$ 6.6
image data			16.4	
random			-20.7	
human			9.3	

Boxing	unpruned		pruned	
	2 layer	1 layer	2 layer	1 layer
neural ground truth objects	93.0 $\pm$ 3.9	97.0 $\pm$ 2.5	78.8 $\pm$ 4.7	47.4 $\pm$ 10.1
rule set ground truth objects	91.2 $\pm$ 7.7	77.4 $\pm$ 14.3	67.2 $\pm$ 6.7	46.0 $\pm$ 5.7
rule set SPACE+MOC objects	51.8 $\pm$ 8.2	9.2 $\pm$ 11.5	37.8 $\pm$ 16.8	38.2 $\pm$ 6.0
neural SPACE+MOC objects	65.0 $\pm$ 11.3	56.8 $\pm$ 4.4	39.8 $\pm$ 12.9	38.4 $\pm$ 9.3
image data			90.3	
random			0.1	
human			4.3	

Table 1: **Overview of PPO and rule extraction experiments results.** Results using PPO with object-centric input from either the SPACE+MOC object extractor or a ground truth object extractor based on OCAtari and results using rule set policies with input from either the SPACE+MOC object extractor or a ground truth object extractor based on OCAtari. Average rewards with standard deviations across five differently seeded evaluation episodes. PPO agents using image input [2], random agents, and human scores [20] are provided for comparison. Pong and Boxing have a maximum achievable reward of 21 and 100 respectively.

Overall, the action selector generated interpretable rule set policies with performance nearly matching neural policies under certain conditions.

## 5 Limitations

The current approach relies on strong assumptions about the training environment, such as the availability of training images showing all object variations and motion data from optical flow estimation. The first aspect can be problematic in environments where objects appear only after certain thresholds, requiring pre-trained agents to collect data. Furthermore, it is uncertain whether valuable motion data can be obtained in more complex scenarios than Atari games.

Currently, the extracted properties only concern the location and the class of the objects. More advanced properties such as the `orientation` of an object are currently not extracted, even though they can be highly relevant in some Atari games (*e.g.*, in Skiing).

The rule set representation of the policy lacks interpretability due to a large number of generated rules, complex premises with many terms and the fact that the premises of multiple rules with conflicting outputs can be satisfied for the same input data point.

## 6 Future Work

Replacing the implementation of the object extractor with unified object detection and tracking methods (*e.g.*, YOLO [21; 22]) could be promising, although this would lead to a limited set of properties. Keeping the multi-step implementation of the object extractor, alternative models to SPACE+MOC could be investigated such as SlotAttention [23] or CutLER [24]. In addition,

---

leveraging the sequential nature of the images, beyond the MOC framework, could improve the robustness and reliability of the object extractor (*e.g.*, by incorporating a Kalman filter in the object tracking step). Another avenue for further investigation is enhancing the object extractor’s capacity to identify additional properties.

Further investigation into the action selector’s interpretability, including tuning ECLAIRE’s hyperparameters for simpler rule sets, analyzing actions from a human perspective, and exploring alternative policy distillation methods, could be promising.

Expanding the SCoBots framework to a wider variety of games and three-dimensional environments could provide valuable insights, particularly by testing the effectiveness of optical flow estimation for motion supervision, and is essential for advancing toward real-world applications.

## 7 Related Work

### 7.1 Explainable and Interpretable Reinforcement Learning

Explainable Reinforcement Learning (XRL) is a prominent area within the field of XAI, focusing on giving human insights into the decision-making processes of AI agents. Key publications such as [25; 26; 27; 28; 29] have extensively reviewed the field of XRL. These works propose frameworks for categorization, highlight complexities, and emphasize ongoing issues that require resolution.

SCoBots can be categorized as an *intrinsic* approach as it directly allows humans to grasp how the model reaches its predictions without requiring any additional computation as *post-hoc* approaches would [27]. However, they are only truly an intrinsic approach if an appropriate action selector is chosen, as in our case a rule set policy. The explanations provided by a SCoBot are usually *local* as they are focused on a single input instance, in contrast to global explanations, which would enable understanding the overall input-output behavior [27]. Again, the choice of the action selector is the main determining factor. Our instantiation via a rule set policy leads to local explanations. The SCoBots framework falls under the *feature importance method* category as defined by [25], emphasizing the identification of crucial input features that influence decisions. SCoBots provide *zero-order explanations* according to the framework by [26], focusing on the agent’s immediate response to inputs. The SCoBots approach aligns with *model explaining* as outlined in [29], with the focus on elucidating the model’s rationale. According to the categorization by [30], SCoBots learn *Symbolic Representations* and use *Object-Recognition*. That categorization focuses solely on intrinsic approaches, for which the authors reserve the term *interpretable*. With regard to *Interpretable Decision-Making*, the SCoBots framework itself does not fall into a specific category. However, our use of policy distillation via rule extraction classifies as an *Indirect Approach* in the subcategory *Decision Trees and Variants*.

#### 7.1.1 Policy Distillation

Policy distillation [31; 32], a specialized form of knowledge distillation [33], involves training a highly performing teacher model and subsequently distilling this knowledge into a simpler student model. By selecting an appropriate student model architecture with sufficient constraints on complexity, an interpretable yet still performing model can be derived. Policy distillation is particularly advantageous when the desired final model architecture is challenging to create independently due to less performing optimization algorithms.

One notable implementation of this concept is VIPER [34], which utilizes imitation learning to extract decision tree policies from a neural policy and Q-function. MoËT [35] extends this approach by incorporating a mixture of expert trees. MAVIPER [36] adapts VIPER to a multi-agent setting. Furthermore, approaches from the field of rule extraction can be used for policy distillation. These approaches transform the teacher model into a set of explicit IF-THEN rules. ECLAIRE [15] exemplifies this approach and is used in our instantiation of the SCoBots framework. NUDGE [37] is an approach from the domain of Neural Logic Reinforcement Learning [38]. This approach uses a

---

neural policy to guide the search for a promising rule set. Furthermore, it can leverage the Q-function, if available, to initialize the critic in its actor-critic RL algorithm. EXPIL [39] integrates predicate discovery, to reduce the reliance on experts, and BlendRL [40] uses a mixture of deep and logic policies, to overcome the potential lack of available concepts. Another approach is INTERPRETER [41]. This method creates interpretable programs by using the concept of oblique decision trees.

## 7.2 Object Representation Learning

If the properties obtained by the object extractor only include the location and object class, as in our experiments, an object detection system suffices. In this case, models such as CutLER [24] and LOST [17] could be considered. These have demonstrated strong performance by leveraging features obtained through self-supervised learning combined with vision transformers [42]. However, perspective the goal is to extract more properties. Hence, our choice of using an approach from the field of object representation learning. SPACE [13] follows a line of work that started with AIR [43]. AIR introduced a sequential attention mechanism that iteratively attends to and infers objects in a scene, using a recurrent neural network to propose object regions and a VAE to generate object representations. SPAIR [44] modified by introducing spatial invariance, utilizing a grid-based attention mechanism to enhance computational efficiency. Other approaches that operate on the pixel level, rather than with bounding boxes, include with Tagger [45] and NEM [46]. Notably, some works have demonstrated the ability to generate representations with dimensions that can be associated with specific features of the objects (*e.g.*, color, shape) [47; 48; 49]. Recently, approaches have emerged that employ the concept of optical flow to benefit from the consecutive nature of the images [50; 51], as does MOC [14], which is used in our work together with SPACE. In supervised settings, automatic concept finding to extend the object-centric representations has been developed for explanation [52; 53; 54], or automated with lambda calculus based concepts [55]. Interpretable concept can also be revised by expert in case of misalignment [56; 2]. Related to RL, concept revision has also been used in the domain of time series [57].

## 8 Conclusion

We instantiated the SCoBots framework [2] and successfully demonstrated its application to Atari games. The SCoBots used a trained object detection component instead of the ground truth detection used in previous work. In the process, we explored several critical areas, including object representation learning, which involves simplifying a scene into an object-centric representation, and object-centric RL, which focuses on learning policies based on the representations of objects. Additionally, we covered policy distillation by applying rule extraction, which transforms a neural policy into a more interpretable rule set policy. Through this work, we hope to contribute to the improvement of interpretability in RL. We believe that by improving interpretability, RL agents can be analyzed and designed more successfully, which can facilitate addressing pervasive challenges in the field of RL. We also identified promising future research directions that could further enhance the framework’s potential for advancing the field.

---

## References

- [1] Quentin Delfosse, Jannis Blüml, Bjarne Gregori, and Kristian Kersting. Hackatari: Atari learning environments for robust and continual reinforcement learning, 2024.
- [2] Quentin Delfosse, Sebastian Sztwiertnia, Wolfgang Stammer, Mark Rothermel, and Kristian Kersting. Interpretable concept bottlenecks to align reinforcement learning agents. *Advances in Neural Information Processing Systems*, 2024.
- [3] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Joshua Tobin, P. Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Neural Information Processing Systems*, 2017.
- [4] David Raposo, Samuel Ritter, Adam Santoro, Greg Wayne, Théophane Weber, Matthew M. Botvinick, H. V. Hasselt, and Francis Song. Synthetic returns for long-term credit assignment. *ArXiv*, 2021.
- [5] Yue Wu, Yewen Fan, Paul Pu Liang, Amos Azaria, Yuanzhi Li, and Tom Mitchell. Read and reap the rewards: learning to play atari with the help of instruction manuals. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2024.
- [6] Lauro Langosco di Langosco, Jack Koch, Lee D. Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In *International Conference on Machine Learning*, 2021.
- [7] Riccardo Guidotti, Anna Monreale, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)*, 2018.
- [8] Gabrielle Ras, Ning Xie, Marcel van Gerven, and Derek Doran. Explainable deep learning: A field guide for the uninitiated. *Journal of Artificial Intelligence Research*, 2022.
- [9] Sara Hooker, D. Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Neural Information Processing Systems*, 2018.
- [10] Chun Sik Chan, Huanqi Kong, and Guanqing Liang. A comparative study of faithfulness metrics for model interpretability methods. In *Annual Meeting of the Association for Computational Linguistics*, 2022.
- [11] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.
- [12] Jaesik Yoon, Yi-Fu Wu, Heechul Bae, and Sungjin Ahn. An investigation into pre-training object-centric representations for reinforcement learning. In *International Conference on Machine Learning*, 2023.
- [13] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations*, 2020.
- [14] Quentin Delfosse, Wolfgang Stammer, Thomas Rothenbächer, Dwarak Vittal, and Kristian Kersting. Boosting object representation learning via motion and object continuity. In *Machine Learning and Knowledge Discovery in Databases: Research Track*, 2023.
- [15] Mateo Espinosa Zarlenga, Zohreh Shams, and Mateja Jamnik. Efficient decompositional rule extraction for deep neural networks. In *eXplainable AI approaches for debugging and diagnosis.*, 2021.
- [16] Quentin Delfosse, Jannis Blüml, Bjarne Gregori, Sebastian Sztwiertnia, and Kristian Kersting. Ocatari: Object-centric atari 2600 reinforcement learning environments. *ArXiv*, 2023.

- 
- [17] Oriane Siméoni, Gilles Puy, Huy V. Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, Patrick Pérez, Renaud Marlet, and Jean Ponce. Localizing objects with self-supervised transformers and no labels. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2021.
  - [18] Sandra Kara, Hejer Ammar, Florian Chabot, and Quoc-Cuong Pham. Image segmentation-based unsupervised multiple objects discovery. *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.
  - [19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, 2017.
  - [20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.
  - [21] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
  - [22] Chien-Yao Wang, I-Hau Yeh, and Hongpeng Liao. Yolov9: Learning what you want to learn using programmable gradient information. *ArXiv*, 2024.
  - [23] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, 2020.
  - [24] Xudong Wang, Rohit Girdhar, Stella X. Yu, and Ishan Misra. Cut and learn for unsupervised object detection and instance segmentation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
  - [25] Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. Explainable reinforcement learning: A survey and comparative review. *ACM Computing Surveys*, 2023.
  - [26] Richard Dazeley, Peter Vamplew, and Francisco Cruz. Explainable reinforcement learning for broad-xai: a conceptual framework and survey. *Neural Computing and Applications*, 2021.
  - [27] Agneza Krajna, Mario Brčić, Tomislav Lipić, and Juraj Doncevic. Explainability in reinforcement learning: perspective and position. *ArXiv*, 2022.
  - [28] George A. Vouros. Explainable deep reinforcement learning: State of the art and challenges. *ACM Computing Surveys*, 2022.
  - [29] Yunpeng Qing, Shunyu Liu, Jie Song, and Mingli Song. A survey on explainable reinforcement learning: Concepts, algorithms, challenges. *ArXiv*, 2022.
  - [30] Claire Glanois, Paul Weng, Matthieu Zimmer, Dong Li, Tianpei Yang, Jianye Hao, and Wulong Liu. A survey on interpretable reinforcement learning. *Mach. Learn.*, 2024.
  - [31] Andrei A. Rusu, Sergio Gomez Colmenarejo, Çağlar Gülcöhre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *CoRR*, 2015.
  - [32] Wojciech M Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *The 22nd international conference on artificial intelligence and statistics*, 2019.
  - [33] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, 2015.

- 
- [34] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Neural Information Processing Systems*, 2018.
  - [35] Marko Vasic, Andrija Petrović, Kaiyuan Wang, Mladen Nikolic, Rishabh Singh, and Sarfraz Khurshid. Moët: Interpretable and verifiable reinforcement learning via mixture of expert trees. *ArXiv*, 2019.
  - [36] Stephanie Milani, Zhicheng Zhang, Nicholay Topin, Zheyuan Ryan Shi, Charles A. Kamhoua, Evangelos E. Papalexakis, and Fei Fang. Maviper: Learning decision tree policies for interpretable multi-agent reinforcement learning. In *ECML/PKDD*, 2022.
  - [37] Quentin Delfosse, Hikaru Shindo, Devendra Dhami, and Kristian Kersting. Interpretable and explainable logical policies via neurally guided symbolic abstraction. *Advances in Neural Information Processing Systems*, 2024.
  - [38] Zhengyao Jiang and Shan Luo. Neural logic reinforcement learning. In *International Conference on Machine Learning*, 2019.
  - [39] Jingyuan Sha, Hikaru Shindo, Quentin Delfosse, Kristian Kersting, and Devendra Singh Dhami. Expil: Explanatory predicate invention for learning in games. *arXiv*, 2024.
  - [40] Hikaru Shindo, Quentin Delfosse, Devendra Singh Dhami, and Kristian Kersting. Blendrl: A framework for merging symbolic and neural policy learning. *arXiv*, 2024.
  - [41] Hector Kohler, Quentin Delfosse, Riad Akroud, Kristian Kersting, and Philippe Preux. Interpretable and editable programmatic tree policies for reinforcement learning. In *Seventeenth European Workshop on Reinforcement Learning*, 2024.
  - [42] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
  - [43] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, koray kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2016.
  - [44] Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *AAAI Conference on Artificial Intelligence*, 2019.
  - [45] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hotloo Hao, Harri Valpola, and Jürgen Schmidhuber. Tagger: Deep unsupervised perceptual grouping. In *Neural Information Processing Systems*, 2016.
  - [46] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2017.
  - [47] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loïc Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International conference on machine learning*. PMLR, 2019.
  - [48] Christopher P. Burgess, Loïc Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matthew M. Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *ArXiv*, 2019.
  - [49] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations. In *International Conference on Learning Representations (ICLR)*, 2020.

- 
- [50] Dong Lao, Zhengyang Hu, Francesco Locatello, Yanchao Yang, and Stefan O Soatto. Divided attention: Unsupervised multi-object discovery with contextually separated slots. *ArXiv*, 2023.
  - [51] Thomas Kipf, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. In *International Conference on Learning Representations*, 2022.
  - [52] Felix Friedrich, David Steinmann, and Kristian Kersting. One explanation does not fit xil. *arXiv*, 2023.
  - [53] Wolfgang Stammer, Felix Friedrich, David Steinmann, Manuel Brack, Hikaru Shindo, and Kristian Kersting. Learning by self-explaining. *arXiv*, 2023.
  - [54] Wolfgang Stammer, Antonia Wüst, David Steinmann, and Kristian Kersting. Neural concept binder. *Advances in Neural Information Processing Systems*, 2024.
  - [55] Antonia Wüst, Wolfgang Stammer, Quentin Delfosse, Devendra Singh Dhami, and Kristian Kersting. Pix2code: Learning to compose neural visual concepts as programs. *arXiv preprint arXiv:2402.08280*, 2024.
  - [56] David Steinmann, Wolfgang Stammer, Felix Friedrich, and Kristian Kersting. Learning to intervene on concept bottlenecks. *arXiv*, 2023.
  - [57] Maurice Kraus, David Steinmann, Antonia Wüst, Andre Kokozinski, and Kristian Kersting. Right on time: Revising time series models by constraining their explanations. *arXiv preprint arXiv:2402.12921*, 2024.
  - [58] Aditya M. Deshpande. Multi-object trackers in python. <https://github.com/adipandas/multi-object-tracker>, 2020.
  - [59] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, 2023.
  - [60] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents (extended abstract). In *International Joint Conference on Artificial Intelligence*, 2018.
  - [61] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

---

## A Appendix

This appendix presents supplementary information on our research experiments and results. It includes details on the data set used, model configurations, and evaluation metrics.

### A.1 Object Extractor

#### A.1.1 Data Set

The data set was generated using OCAtari [16]. For generation, a random agent was used. A total of 2048 training, 128 validation, and 128 test image sequences were collected, each comprising four consecutive frames. The reason for collecting four consecutive frames was that consecutive images are required to calculate the object continuity loss. For each frame, the ground truth object detections were also stored. A deliberate pause of at least 16 steps between sequences was implemented to ensure data diversity. The frames were downscaled to 128x128 pixels for compatibility with the SPACE [13] model.

##### Games

Pong, Boxing and Skiing were used in the experiments for the object extractor. These games cover different levels of difficulty for the object extractor. Pong and Boxing only contain a very small number of objects in each frame. The shapes of the objects in Boxing are more complex and can vary in shape, depending on whether the boxers are punching or not. Skiing contains more objects and the visual appearance of objects can vary within the same object class. An image of each game is depicted in Figure 4. In the experiments for the action selector, only Pong and Boxing were used. Skiing is too challenging even for end-to-end deep RL approaches due to its extremely delayed reward signal.

##### Optical flow

The optical flow method involved the collection of a mode image based on 100 images. A manual check was conducted to ensure that the background of the respective game was accurately represented. The optical flow and motion data were then calculated using the background subtraction approach outlined in [14].

#### A.1.2 SPACE+MOC Details

In general, the hyperparameters from [14] were reused with a few modifications (*cf.* Table 2). Overall, the hyperparameters related to SPACE are largely consistent with those presented in the original work [13].

The MOC loss was applied using the dynamic scheduling approach proposed in [14]. This approach dynamically balances the motion supervision loss and the object continuity loss depending on the ability to correctly localize the bounding boxes. For more details, please refer to [14]. In contrast to [14], no bootstrapping of the decoder with the help of the motion information was used. The training was repeated using five different seeds, and the resulting data was used to calculate the mean and standard deviation for the metrics.

##### Filtering SPACE+MOC Localizations

The evaluation of the object extractor was focused on the moving objects of the games, only considering the relevant objects for playing the games. Consequently, static objects that are also provided via OCAtari were removed from the ground truth data (*e.g.*, the clock object). Furthermore, the predicted objects by the model for localization underwent filtering based on potential detection areas of the moving objects. Practically, this was implemented by discarding localized objects that have bounding box coordinates clearly outside of the areas where moving objects can appear in the game. For the games that were considered in our experiments, this approach is sufficient because

Parameter	Value
batch size	16
gradient steps	5000
Foreground / Background lr	$3 \cdot 10^{-5} / 10^{-3}$
$z_{\text{pres}}$ prior probability	$0 \rightarrow 5000 : 0.1 \rightarrow 10^{-10}$
$z_{\text{scale}}$ prior mean	$0 \rightarrow 5000 : -2 \rightarrow -2.5$
$z_{\text{scale}}$ prior std	$0.1 \cdot \mathbf{I}$
$z_{\text{shift}} / z_{\text{what}} / z_{\text{depth}}$ priors	$\mathcal{N}(0, \mathbf{I}) / \mathcal{N}(0, \mathbf{I}) / \mathcal{N}(0, \mathbf{I})$
$\tau$ (gumbel-softmax-temperature)	2.5
Foreground / Background stds	0.2 / 0.1
Background Components	3
Grid Size	16
fixed $\alpha$ & boundary loss	removed
Motion Kind	Mode
$\eta$	0.5
$\lambda_{\alpha} / \lambda_{\text{pres}} / \lambda_{\text{where}}$	100 / 1000 / 10000
$\lambda_{\text{guid}}$	$0 \rightarrow 3000 : 1.0 \rightarrow 0.0$
$\beta_{\text{mismatch}}, \beta_{\text{underestimation}}$	0.1, 1.25
$\beta_{\text{differ}}$	5
$\lambda_M / \lambda_{OC}$	100 / 10
Deviations - Boxing:	
$z_{\text{scale}}$ prior	-1
Deviations - Pong:	
$\beta_{\text{underestimation}}$	1.5

Table 2: SPACE and MOC shared parameter values for training, SPACE base parameter values, MOC parameter values, Deviations from SPACE+MOC base parameters for Pong and Boxing as in [14]

Game	Rule
Boxing	$0.148 < y_{\text{min}} \wedge y_{\text{max}} < 0.859$
Pong	$0.164 < y_{\text{max}} \wedge 0.031 < y_{\text{min}}$
Skiing	True

Table 3: Region filters for SPACE+MOC localizations

non-moving objects can only be found outside of this area. The same approach was also used in [14]. The rule for Pong differs from the rule used in [14].

### A.1.3 Classifier Details

The creation of the classifier involves multiple steps. First, k-means clustering is performed to obtain the centroids. For each Atari game, the value for  $k$  is given by the number of object classes as specified in OCAtari [16]. Second, descriptive labels are assigned to the obtained centroids, since the k-means clustering only returns enumerated class labels. For this, a k-nearest neighbors classifier is used, where  $k$  does not refer to the same number as in k-means. The initialization is based on object encodings extracted by SPACE+MOC from a small image data set. These objects have been assigned a descriptive label based on the object names of the ground truth detections provided by OCAtari. The descriptive labels for the centroids are finally assigned via the k-nearest neighbors classifier. It is important to note that this approach is not entirely in accordance with the unsupervised setting, but the supervised data is only used for assigning names to the classes. Third, the final classifier is given by a 1-nearest neighbor classifier initialized only with the centroids.

---

In k-means clustering, the parameter  $k$  was set to the number of relevant object types, and only one run was conducted with random initialization of the centroids. For determining descriptive labels for the centroids, the k-nearest neighbors classifier was employed with  $k$  set to 24. In all three steps, the Euclidean distance was utilized as the distance metric.

The classifier was trained using latent variables extracted from a SPACE+MOC model, which was applied to the validation set images. The model used was the SPACE+MOC model from the localization experiments with the highest F-score. The training set was not used to avoid data leakage. Only the first of each sequence of consecutive images was used, as the images and extracted latents for images in a sequence would be too similar. For the purposes of testing, the test set was employed, with the remaining aspects left unchanged.

#### A.1.4 Tracking Algorithm Details

In the initial pass of the tracking algorithm through a video frame, each detected object is added to a tracking list. Each object is identified by the bounding box that encapsulates it, and at this stage, there are no previous tracks to compare against, so all detections are treated as new objects. From the second frame onwards, the algorithm calculates the distances between the centroids of the currently tracked objects and the centroids of the new detections. The matching scores, derived from the distance calculations, are used to determine which new detections correspond to which existing tracks. A detection is assigned to the closest tracked object, thereby ensuring continuity in tracking. New detections that do not closely match any current track—either because they are too far from existing tracks or are only the second-best match—are initiated as new tracks. This step accounts for new objects entering the scene. Conversely, objects that have been previously tracked but do not find a match in the new detections are removed from the tracking list. This addresses objects that leave the scene or become occluded.

The algorithm’s simplicity can result in failure when objects cross paths or even overlap for multiple consecutive frames. While the former may result in incorrect features for a single frame, the latter can lead to significant issues. The current implementation is based on [58].

#### A.1.5 Combined Object Extractor Experiments

The results were generated using the SPACE+MOC models with the highest F-score from five differently seeded runs, in conjunction with a corresponding classifier trained based on the encodings of that model.

The utilized F-score metric is a well-established and widely accepted standard. However, the evaluation of the localizations is heavily reliant on the manner, in which the localized and ground truth bounding boxes are assigned to one another. Therefore, further details on this are provided below.

##### How to match bounding boxes?

**Pairwise Matching Scores** - How to measure the similarity between a predicted and a ground truth bounding box?

The authors of [14] argue that the intersection over union (IoU) metric, which is arguably the most common metric in this context, is not ideal for assessing performance for the downstream task of RL for Atari games. This is due to the architecture of SPACE, which tends to return bounding boxes of similar sizes for a game. Consequently, in games such as Pong where the ball and player objects differ in size, at least one of the object classes will have a bounding box that is either too large or too small. This will result in low IoU scores. However, since the objects are (mostly) of constant size in the games that we consider, the downstream RL algorithm can implicitly infer the size. In fact, the object-centric input representation returned by the relation extractor only contains the center coordinates of the objects in our implementation. As an alternative to IoU, the authors of [14] introduce the center divergence metric. This metric focuses on the distance of the center coordinates

---

of the bounding boxes. For details, we refer to [14]. The center divergence metric was employed in the calculation of precision, recall, and the derived F-score.

**Matching** - How to assign predicted bounding boxes to ground truth bounding boxes based on the pairwise matching scores?

The first condition is that the matching score of a predicted bounding box to a ground truth bounding box must exceed a specified threshold. We employed a threshold of 0.5 for the matching score computed via center divergence. In addition to the first condition, further conditions are required because two predicted bounding boxes might have a sufficient matching score with the same ground truth bounding box or a single predicted bounding box might have a sufficient matching score with two ground truth bounding boxes. Due to the relatively simple nature of the images, we applied a straightforward greedy approach. The predicted bounding boxes that met the first condition were sorted by confidence and then were assigned greedily to the ground truth bounding boxes. In more complex scenarios, the use of Hungarian matching is advisable.

### Joint treatment of Localization and Classification

In the event that a localized object cannot be matched with a ground truth object, it is labeled as "no object." Conversely, if a ground truth object does not correspond to any localized object, it is designated as "not detected." This classification system allows for a more nuanced understanding of the detections.

In order to evaluate the performance of the object extractor, we calculated the precision, recall, and F-score. It is necessary to be precise with the definition of these metrics. This is due to the two-step approach in which the classifier, in the second step, can receive localized objects that do not have a ground truth object associated with them. At the same time, the classifier does not necessarily receive all of the ground truth objects. Our calculated precision score addresses the following question: What proportion of all the detected objects corresponds to a ground truth object and is correctly labeled? In other words, when looking at the confusion matrix (*cf.* Figure 6), the "not\_detected" column is removed, and then the micro precision is calculated. Analogous to the precision score, our recall score answers the following question: What proportion of all the ground truth objects has been localized and has received the correct label? This is achieved by removing the "not\_an\_object" row from the confusion matrix and then calculating the micro recall.

## A.2 Relation Extractor

### Algorithmic description

The relational concepts are specified in advance. To this end, the number of objects of each class to be considered must be specified at first. Objects of the same class are given unique identifiers by enumeration. Then, the relational concepts to be computed based on these objects are defined. A relational concept is characterized by the input objects and the relational function applied to their properties. The relational functions themselves are straightforward (e.g., calculating Euclidean distance). This approach specifies clear computational instructions that yield a single scalar value for each defined relational concept. Each relational concept is assigned a fixed position in the output vector.

During inference, the detected objects must be mapped to the enumerated objects from the predefined computational instructions. For this purpose, the detected objects are sorted in ascending order based on their distance to the player object. The player object is defined as the object controlled by the agent (e.g., the right paddle in Pong). Objects within each class are then enumerated according to this order. If the object extractor detects more objects for a class than specified in advance, those with identifiers exceeding the maximum number are discarded. Conversely, if a predefined object of a class remains unassigned due to fewer detected objects, the relational concepts dependent on this object are assigned a scalar value of zero.

### Selection of relational concepts

- 
- Full set: All relational concepts as detailed in Table 3 of [2] were included. This set considers all possible combinations of objects for n-ary relations, including symmetric ones like distance, where both directions (*e.g.*,  $d(\text{Ball1}, \text{Player1})$  and  $d(\text{Player1}, \text{Ball1})$ ) are accounted for, resulting in six combinations. For the `linear trajectory` relation, combinations with the object itself are included, *e.g.*, leading to nine combinations for three objects.
  - Pruned set: The pruned set of relational concepts only includes the subset of relational concepts that are presumed to be relevant for the game as listed in Table 4 of [2]. The selection of relational concepts is based on human understanding of the games.

Both the full set and the pruned set stem from [2].

### A.3 Action Selector

#### A.3.1 Experiment Design

We designed the experiments to encompass different levels of complexity of the model for the neural policy. This was done with the rationale that the rule set extraction would probably work more reliably the simpler the neural network (NN) structure and the policy is. Therefore, we varied the depth of NN architecture in the PPO [19] algorithm. The NNs were instantiated with either 1 or 2 hidden layers of 64 neurons. Additionally, the set of relational concepts was varied. Either the full set of relations or a pruned version was used (*cf.* App. A.2).

Different versions for the Atari games are available in OCAtari via the underlying Gymnasium library [59]. Following the recommendations of [60] and for better comparability with [2], v5 of the games was used.

#### A.3.2 Deep Reinforcement Learning Experiments

The hyperparameters for PPO [19] mostly stem from [2], with a few minor exceptions. Specifically, 10M frames were used instead of 20M, and only a single training seed was used. In contrast to [2], the policy and value networks were initialized with either 1 or 2 hidden layers of size 64. One important aspect to point out is that the advantages in the experiments were normalized as in [2]. The existing implementation of PPO in the package stable-baselines3 [61] was used.

#### A.3.3 Policy Distillation Experiments

For ECLAIRE [15], we retained the default hyperparameter settings. The main hyperparameter that could be tuned for better performance is  $\mu$ . This parameter specifies the minimum number of samples that are required for C5.0 to have before splitting a node. As mentioned in the Method chapter, the training instances must be diverse enough. For this purpose, we used the PPO policy, with a random action taken in 25% of cases. A total of 50,000 training instances were collected using this strategy, from which the rules were extracted. For inference, the conclusion of the rule with the highest confidence was chosen among all the rules whose premises are satisfied by the input data point.