EXPLAINABILITY OF DEEP REINFORCEMENT LEARNING ALGORITHMS IN ROBOTIC DOMAINS BY USING LAYER-WISE RELEVANCE PROPAGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A key component to the recent success of reinforcement learning is the introduction of neural networks for representation learning. Doing so allows for solving challenging problems in several domains, one of which is robotics. However, a major criticism of deep reinforcement learning (DRL) algorithms is their lack of explainability and interpretability. This problem is even exacerbated in robotics as they oftentimes cohabitate space with humans, making it imperative to be able to reason about their behaviour. In this paper, we propose to analyze the learned representation in a robotic setting by utilizing graph neural networks. Using the graphical neural networks and Layer-wise Relevance Propagation (LRP), we represent the observations as an entity-relationship to allow us to interpret the learned policy. We evaluate our approach in two environments in MuJoCo. These two environments were delicately designed to effectively measure the value of knowledge gained by our approach to analyzing learned representations. This approach allows us to analyze not only how different parts of the observation space contribute to the decision-making process but also differentiate between policies and their differences in performance. This difference in performance also allows for reasoning about the agent's recovery from faults. These insights are key contributions to explainable deep reinforcement learning in robotic settings.

1 INTRODUCTION

Possible flow: One of the goals in AI research is coexistence of AI and Humans While DRL has seen more and more success recently One of the main limiting factors in using it in the real world is its limited explainability What did others do to solve this problem? How are these approaches transferable to DRL, if they are? What are the main open questions? How are we addressing one or possibly multiple of these questions? What is the approach we are suggesting? How does our work exactly answer the open questions? Main contribution! Remainder of this manuscript

2 BACKGROUND

2.1 GRAPH NETWORK

Graphs are tools that are used to represent structured data. A graph contains multiple entities, each having relationships with others. Entities and their relationships are shown using nodes and edges of the graph, respectively. This gives us a flexibility in designing representation architectures of arbitrary shape. Furthermore, this way of knowledge representation emphasizes the location of each entity relative to other entities.

Graph Networks(GN) are neural networks that operate on graph inputs. Graph networks impose constrains on relationships and interactions among entities while finding the optimal solution. In other words, they emphasize on the *relational inductive bias*. The graph network architecture and operations in this work are based on the work proposed by Battaglia et al. (2018). In an input graph, there are three kinds of features: 1) *node features* that include a vector for each node showing the state of each entity, 2) *edge features* that include a vector for each edge illustrating the state of each relation, and 3) *global features* that contain a vector for the whole graph, showing the state of the system. $E = \{e_k\}$ and $V = \{v_i\}$ denote the set of edge and node feature vectors respectively and u denotes the global-level feature vector. A graph network has three kinds of operations for updating node, edge, and global-level features. Equation 1 shows the operations in a graph network.

$$\begin{aligned} \mathbf{e}'_k &= \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}) & \overline{\mathbf{e}}'_i &= \rho^{e\to v}(E'_i) \\ \mathbf{v}'_i &= \phi^v(\overline{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}) & \overline{\mathbf{e}}' &= \rho^{e\to u}(E') \\ \mathbf{u}' &= \phi^u(\overline{\mathbf{e}}', \overline{\mathbf{v}}', \mathbf{u}) & \overline{\mathbf{v}}' &= \rho^{v\to u}(V') \end{aligned} \tag{1}$$

where $E'_i = \{(\mathbf{e'_k}, r_k, s_k)\}_{r_k=i,k=1:N^e}$ is the updated set of edges whose receiver is node i, $V' = \{\mathbf{v'_i}\}_{i=1}^{N^v}$ is the set of updated nodes, $E' = \bigcup_i E'_i = \{(\mathbf{e'_k}, r_k, s_k)\}_{k=1}^{N^e}$ is the set of all the updated edges. ϕ^e updates edge features which incorporates surrounding nodes and global-level features to update edge features. ϕ^v updates node features, which incorporates aggregated features of the surrounding edges of a node and global-level features. Finally, ϕ^u updates global-level features given the graph's aggregated node and edge features (Battaglia et al., 2018). A summary of the graph network operations can be seen in Appendix A.

2.2 LAYER-WISE RELEVANCE PROPAGATION

Originally, the LRP has been used as a visualization tool highlighting the contribution of each input component to the classification by neural networks. The LRP decomposes the output probability given to a specific class and back-propagates this score to the input components so that the sum of scores in each layer is equal across layers:

$$f(x) = \cdots = \sum_{d\in(l+1)} R_d^{l+1} = \sum_{d\in l} R_d^{(l)} = \cdots = \sum_d R_d^{(1)} \tag{2}$$

where R_d^l is the relevance score given to unit d of layer l and $f(x)$ is the score of the output layer. Bach et al. (2015) proposed two methods for calculating these relevance scores and preventing them from taking unbounded values. These two methods are the $\alpha\beta$-rule and the ε-stabilized rule. As discussed by Baldassarre & Azizpour (2019), the latter rule is more robust and simple, which is why we also use the second rule. For the details about LRP, please refer to Appendix B.

Intuitively, during the forward pass, the neural network emphasizes on some parts of the input that are contributing more to the output by giving them higher weights and activating them. LRP uses these weights and activations in each layer to propagate the output back through the network up until the input layer. Hence, it can highlight the most contributing parts of the input.

2.3 DEEP REINFORCEMENT LEARNING

Our method can explain all the policies with neural networks as function approximators regardless of the type of DRL algorithm. We selected Soft Actor-Critic (SAC), popular and known to perform well in robotic environments (Haarnoja et al., 2018a). SAC is trained through an off-policy process, meaning that the policy being evaluated and updated is separate from the policy used for generating

trajectories. SAC's objective function maximizes not only the expected cumulative reward but also an entropy term. Therefore, the optimal policy looks as follows:

$$\pi^* = \arg\max_{\pi} \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha\mathcal{H}(\pi(.|s_t))\right] \tag{3}$$

where α is the temperature parameter that determines the importance of entropy, and $\mathcal{H}(\pi(.|s_t))$ is the entropy term. This entropy term facilitates the optimization process by smoothing out the objective function to avoid getting stuck in local optima. Empirically, this entropy term incentivizes the agent to explore the environment more widely. For the details, please refer to the original paper (Haarnoja et al., 2018b). We also use automatic temperature parameter tuning, as discussed in the paper.

3 Proposed Method

A high-level sketch of the problem is to identify the contribution of each component of the robot to the decision-making process. Our solution is to behave each component of the robot as an entity with relationships among them, then learn a policy on the environment using this decomposition. After convergence to the optimal policy, the LRP tries to highlight the most contributing parts to the decision-making. Hence, the first step in our method is to decompose the robot's observation from a vector of features to a graph of entities and relations. The observation would be in the form of:

$$s_t = G(V_t, E_t, u_t) \tag{4}$$

where V_t and E_t are the set of node and edge features at time step t respectively and u_t is the global-level feature vector. To use graph representation of the environment by the DRL algorithm, we need to replace fully-connected networks as function approximators with graph networks. There are 5 networks in SAC as follows: policy network ψ, Q-networks θ_1 and θ_2, and target networks $\bar{\theta_1}$ and $\bar{\theta_2}$ as explained in the algorithm (Haarnoja et al., 2018b). Wherever we have states as input, we use a graph network architecture to extract features from the state.

For graph structures and graph network operations, we use the GitHub repository [1] developed by Baldassarre & Azizpour (2019) for their work.

After training the agent until convergence to the optimal policy, we proceed with the explanation phase. In this phase, there is no further update to the policy, and we want to analyze the learned behavior. At each time-step t, given the state of the environment, the policy network outputs the action $a_t = \pi_\psi(s_t)$ corresponding to state s_t. This action, which is the mean of the Gaussian distribution, would look like the following:

$$a_t = [a_t^{(1)}, a_t^{(2)}, \ldots, a_t^{(h)}] \tag{5}$$

where h is the number of action elements. To calculate the relevance of each action $a_t^{(i)}$ where $i \in \{0, \ldots, h-1\}$, to the input graph components, we zero out all the elements in the action vector except the element at index i, which forms the relevance score of the action i at time step t, $r_t^{(i)}$ in the output graph's global features. If $e_i \in \mathbb{R}^h$ denotes one hot vector whose elements are zero except the one at index i (which equals 1), then

$$r_t^{(i)} = a_t \cdot e_i \tag{6}$$

This relevance score is set to the global features of the output graph, then back-propagated to the input. If we denote the layer-wise relevance propagation operation on a neural network with $LRP()$, then

$$R_{c,t}^{(i)} = LRP(r_t^{(i)}) \quad \text{for } c \text{ in } \{0, \ldots, C-1\} \tag{7}$$

Where C is the number of components of the input graph, $R_{c,t}^{(i)}$ is the relevance of action i to component c of the input graph at time-step t given by the LRP. The $LRP()$ back-propagates the vector $r_t^{(i)}$, which is the global features of the output graph, to the input graph's components. Then, the relevance of each action to a corresponding component of the input is averaged across time steps. The evaluation phase is summarized in algorithm 1.

[1] https://github.com/baldassarreFe/graph-network-explainability.git

Algorithm 1 Calculating relevance scores for the components in the observation space.

let $R_c^{(i)} = 0$ for $c \in \{0, \ldots, C-1\}$ and $i \in \{0, \ldots, h-1\}$
let N denote the number of episodes
for each episode n in $\{0, \ldots, N-1\}$ **do**
  for each time-step t in $\{0, \ldots, T-1\}$ **do**
    Sample current state s_t of the environment
    $a_t = \pi_\phi(s_t)$ ▷ a_t equals the mean of the policy distribution
    for each element i of the action vector **do**
      $r_t^{(i)} = a_t \cdot e_i$
      $R_c^{(i)} = R_c^{(i)} + LRP(r_t^{(i)})$
    end for
  end for
end for
return $\forall c, i : R_c^{(i)}/(T \times N)$ ▷ Average of the relevance across time-steps

4 EXPERIMENTS

We have two phases of experiments. In the first phase, we will train the agent until convergence using graphs as input observations and graph networks as function approximators for the DRL algorithm. At the end of this phase, we apply the LRP algorithm to provide a heatmap showing the relevance between each action element and each entity of the observation. Then, by using the LRP scores, we identify the most important entities and joints in the observation and action spaces respectively. In the second phase, the importance scores calculated in the first phase are evaluated. Each joint plays two roles in decision-making. First, its features are represented to the agent as state. Second, an action in the form of torque is applied to that joint create motion. Therefore, there are two kinds of information extracted by the LRP:

1. **Entity Importance in the Observation Space:** Each action element gives a relevance score to each entity of the observation space. These relevance scores are averaged for each entity across actions to yield the importance of each entity to the whole decision-making process.
2. **Joint Importance in the Action Space:** The relevance scores given by each action to entities of the observation space are averaged for each action element across input entities to yield the importance of each joint in the action space.

Note that for elements of the observation space, we use the term "Entity", but for the elements of the action space, we use the term "Joint". The reason is that not all the elements of the observation space are "Joints", unlike the elements of the action space.

For each environment, the evaluation targets the two kinds of information provided above. For the importance of the entities in the observation space, we **occlude** the features of that entity in the observation space and then rerun the experiments in the new (partially observable) environment. The occlusion process corresponds to a sensor failure, in which the entity's state is not given. Based on the amount of drop in the performance, we can evaluate the correctness of the importance scores. We expect that the amount of drop is proportional to the importance score given to the entity. For the importance of the joints in the action space, we **block** each joint so that no torque (motion) can be applied to it. Again, based on the amount of drop in the performance, we can measure the correctness of the importance scores for each joint in the action space. We expect that the amount of drop is proportional to the importance score given to the joint.

The experiments are run across two simulated robotic environments in MuJoCo Todorov et al. (2012) OpenAI Gym OpenAI (2021): `Walker2D-v2` and `FetchReach-v1` . The observation space is converted to a graph with nodes and edges. Nodes are links of the robot and edges are joints. The features of nodes do not contribute in decision making, therefore we only have edge features and global features.

The walker is a two-dimensional two-legged figure that consists of four main body parts – a single torso at the top (with the two legs splitting after the torso), two thighs in the middle below the torso,

two legs in the bottom below the thighs, and two feet attached to the legs on which the entire body rests. The goal is to make coordinate both sets of feet, legs, and thighs to move in the forward (right) direction by applying torques on the six hinges connecting the six body parts OpenAI (2022). The fetch robot is a robot arm consisting of 7 joints ordered in a line, where the `shoulder pan` is the joint closest to the base and `wrist roll` is located at the other end, connected to the end-effector. The goal is to move the end-effector to a specific 3D position given by the environment – the `goal` . The action for both environments are torques applied to joints.

For both `Walker2D-v2` and `FetchReach-v1` , the edge feature is the position and velocity of the joint. In `Walker2D-v2` , the `torso` in the model is a joint connecting the robot to the world, showing its position and velocity. In `FetchReach-v1` , the global feature is the position of the `goal` .

4.1 Train and Explanation Phase

This phase starts by training the DRL agent until convergence on the two environments, followed by applying LRP to the learned policy to generate a heat map. This heat map represents the relevance of each action element to each observation entity, as in figure 1. For each environment, the relevance scores are calculated and averaged across 10 seeds, each run for 20 episodes. Furthermore, by averaging LRP scores across each observation entity and each action element separately [2], we can calculate importance scores for them. The *entity importance* score is on the top-left and the *action importance* score is on the bottom-left of Figures 2 and 3 for `Walker2D-v2` and `FetchReach-v1` , respectively.

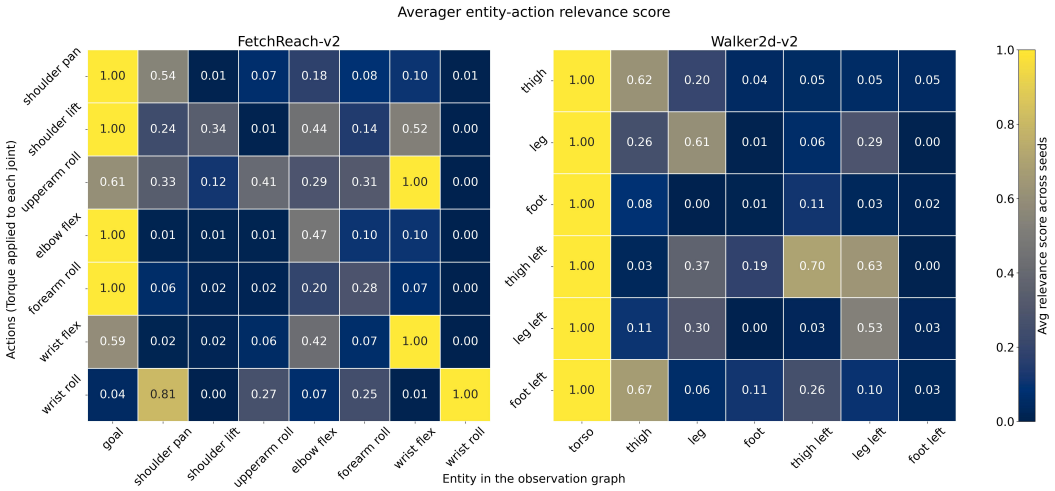

Figure 1: The LRP heat map for action-entity relevance score. The y-axis and x-axis show elements of the action space and entities of the observation space respectively. Since the range of motion for each joint is different, hence the amount of torque applicable to each, the actions have different ranges. Therefore, we normalize the relevance scores for each action across all the observation entities to be between zero and one.

4.1.1 Walker2D-v2

Figure 1 right represents the heat map produced for the `Walker2D-v2` environment by the LRP. As the heatmap indicates, scores given to the `torso` entity are the highest across all the elements of the action space. The reason is that the goal of the environment is to increase the speed of the torso. Another thing that we expect to observe is the high relevance of the action elements to their corresponding joint entities in the observation space. This dependence can be nicely seen on

[2]The importance scores are not the average scores on the heat map because the scores on the heat map are normalized for each action.

the heat map in which the relevance scores around the diagonal are relatively high (by diagonal, consider removing the torso column of the heatmap and then look at it). For example, for the action (torque) applied to the thigh joint, the relevance score given to the thigh entity is high relative to other joints (its relevance score is 0.62). The same is true for leg , thigh left , and leg left . For the thigh left action, in addition to thigh left entity, the relevance score given to the leg left entity is also high. This case proves one of the reasons we selected the graph structure: not only do we take into account the effect of features of each entity on the decision-making process, but we also consider their position in the structure. In other words, the vicinity of the two entities in the robot is the cause of the high score given to the leg left entity by thigh left action.

According to the *entity importance* plot (top-left bar plot in figure 2, the most important entity to the policy is torso – the goal of the environment. The importance score for the other entities is pretty small; however, the score of leg , thigh left , and leg left are higher than the remaining. Moreover, the *action importance* plot (bottom-left bar plot in figure 2 indicates that the critical action elements are foot and foot left joints, respectively. The correctness of these claims and the intuition is discussed in Section 4.2.

4.1.2 FETCHREACH-V1

Figure 1 left shows the heat map generated by the LRP for the FetchReach-v1 environment. The position of the goal received a high score across all actions except wrist roll . This is because the torques should be adjusted in a way to position the end-effector at the goal . Ignoring the goal column, similar to Walker2D-v2 's heat map, we can see a diagonal pattern. The highest scores for shoulder pan , elbow flex , forearm roll , wrist flex , and wrist roll actions belong to their corresponding joint entities in the observation space as expected. Although for shoulder lift and upperarm roll actions we can see unexpected results, still the scores given to their corresponding joint entities in the observation space are relatively high. This unexpected result might be explained using the graph structure and the vicinity of joints: for shoulder lift and upperarm roll actions, the relevance scores are distributed across multiple entities in the observation space.

According to the *entity importance* plot (top-left bar plot in Figure 3), the most important entity to the policy is goal . Other than goal , shoulder pan , forearm roll , wrist roll , and upperarm roll have relatively high importance respectively. Other entities have pretty small importance scores. Moreover, the *action importance* plot (bottom-left bar plot in Figure 3) indicates that the critical joints for positioning the robot's end-effector to the goal are elbow flex , wrist flex , and forearm roll joints, respectively. The correctness of these claims and the intuition is discussed in Section 4.2.

4.2 EXPLANATION EVALUATION PHASE

To ensure that the results provided by the LRP are neural network architecture independent and only depend on the policy learned by the DRL algorithm, we use fully-connected networks in this phase of experiments. The purpose of this section is to evaluate the correctness of the information gained by the LRP. As discussed, the evaluation targets observation entity importance and action's joint importance. The fist one is fulfilled by occluding the entity in the observation space and the second one by blocking the joint. In each case, based on the amount of drop in performance, their importance scores are validated. In figures 2 and 3, the upper and lower rows reflect the entity importance and action importance evaluation results, respectively. In these two figures, all the importance scores and the performance bars have been normalized

4.2.1 WALKER2D-V2

First we discuss the observation entity importance evaluation in the upper-row plots of figure 2. The top-left bar chart says that the most important entity in the observation space is the torso . The scores for entities other than torso are pretty low and close to each other, among which, leg , thigh left , and leg left have the highest scores. As expected, by occluding the torso from the observation space, the agent could not learn a policy at all. Therefore, it has the most significant drop in performance compared to the standard case in the upper-middle plot

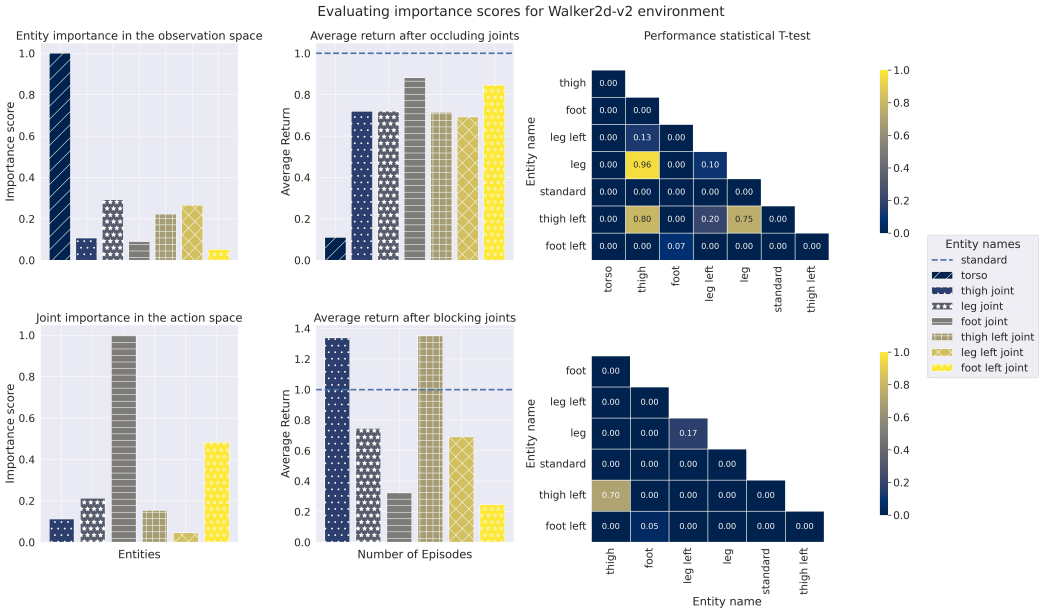

Figure 2: Evaluating explanation for the `Walker2D-v2` . Upper-left: entity importance in the observation, upper-middle: final behavior performance after occluding each entity, upper-right: significancy test for the final behavior after occlusion, lower-left: joint importance in the action, lower-middle: final behavior performance after blocking each joint, lower-right: significancy test for the final behaviors after blocking. The significancy tests are t-test with 95% confidence interval, which means that for $p \leq 0.05$ the performances are significantly different. Also, the results for performance bars are averaged across 30 seeds.

of figure 2. The performances of the `thigh` , `leg` , `thigh left` and `leg left` are equal as can be seen in their performance bar and their significancy test on the upper-left heat map of the figure 2. The `foot` and the `foot left` have the least drop in performance compared to the standard setting as shown in the performance bar, which can be deduced from their importance score as well. Nevertheless, the importance score for the `thigh` is unexpected because, based on its performance bar, we expect its score to be noticeable, similar to `leg` , `thigh left` , and `leg left` .

The joint importance in the action space evaluation is reflected in the lower-row plots of figure 2. As indicated by the bottom-left bar chart, `foot` and `foot left` joints are the most important ones. Their performance bar in the bottom-middle figure proves the correctness of this claim by showing that the amount of drop in performance after blocking `foot left` and `foot` joints is substantial compared to the standard setting. For `leg` and `leg left` the amount of drop in performance is close to each other. Both `leg` and `leg left` 's importance scores explain this drop, however the `leg left` 's importance is a little higher, which is unexpected. One interesting thing is that after blocking `thigh` and `thigh left` joints, the performance became even better than the standard setting. Although LRP gives a pretty low importance score to these two entities, it cannot explain why the performance improves compared to the standard setting.

4.2.2 FETCHREACH-V1

First we focus on analyzing the observation entity importance in the upper-row plots of figure 3. As indicated by the upper-left bar plot, the most important entity is the `goal` . As expected, when occluding the `goal` in the observation space, you can see that the performance dropped significantly compared to the standard setting, which is clear in its performance bar in the upper-middle plot. After `goal` , `shoulder pan` and `forearm roll` entities have the highest scores. Their corresponding performance bars show a proportional amount of drop in performance after occlusion. We expect that the `wrist roll` joint receives a relatively low importance score based on its per-

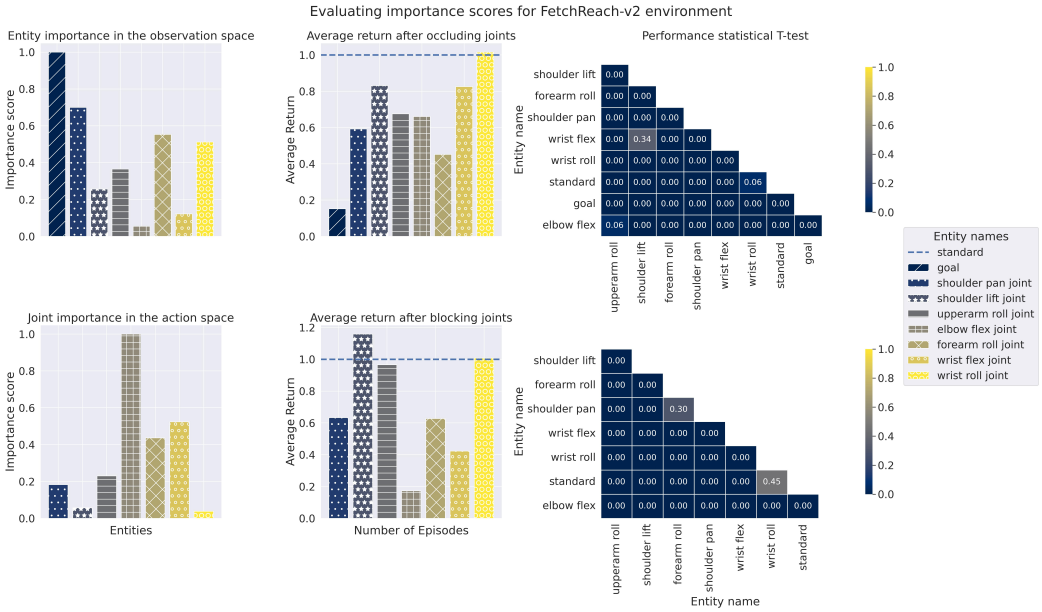

Figure 3: Evaluating explanation for the `FetchReach-v1` . Upper-left: entity importance in the observation, upper-middle: final behavior performance after occluding each entity, upper-right: significancy test for the final behavior after occlusion, lower-left: joint importance in the action, lower-middle: final behavior performance after blocking each joint, lower-right: significancy test for the final behaviors after blocking. The significancy tests are t-test with 95% confidence interval, which means that for $p \leq 0.05$ the performances are significantly different. Also, the results for performance bars are averaged across 10 seeds.

formance bar. Although the `wrist roll` joint seemed to be critical according to its importance score, if we look at the heat map of the `FetchReach-v1` in figure 1, the action applied to this joint gave a relatively low score to the `goal` . Thus, we can conclude that this joint is not contributing to reaching the goal at all. That's why after occluding it, the performance did not change. The importance score of the `upperarm roll` entity can be verified by its performance bar as well. However, since the `upperarm roll` and `elbow flex` preformance bars are approximately equal, we expect a high score for the `elbow flex` as well, unlike its current importance score. It remains `shoulder lift` and `wrist flex` entities that as clear from their performance bars, the amount of drop in their performance can imply their importance score.

The joint importance in the action space evaluation is reflected in the lower-row plots of figure 2. The most critical joints to the actions indicated on the lower-left bar plot are `elbow flex` , `forearm roll` , and `wrist flex` joints, respectively. The noticeable drop in the performance after blocking these three joints, shown in their performance bar in the lower-middle plot, is an implication of their importance. The `wrist roll` does not cooperate in reaching the goal, therefore neither its occlusion nor its blocking affect the performance, because, as discussed above, its action gives a low relevance score to the goal on the heat map. For `upperarm roll` and `shoulder lift` joints, the drop in their performance bar compared to the standard setting can be explained by their importance score. Nevertheless, the LRP fails to explain why the performance became better than the standard case after `shoulder lift` joint's blockage. Also, we expect that LRP gives an importance score approximately equal to the `forearm roll` joint, because their performance is nearly the same.

5 Conclusion and Future Work

In this paper, we propose a novel technique for interpreting the deep reinforcement learning algorithms in robotic environments using graph networks and Layer-wise Relevance Propagation. This

method identifies the contribution of robot's components to the decision making process, allowing us to analyze the learned policy. The experimental results prove that our method can successfully highlight these contribution. Future directions include analyzing how the agent adapts to new dynamics by analyzing the change in importance scores. We can also differentiate the performance of two policies trained through different DRL algorithms. Another future approach is to analyze the transferability of a policy to different dynamics based on the importance of the part whose dynamic has changed. One final future direction is to consider saliency methods other than LRP and examine their performance in the same problem.

REFERENCES

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*, 2019.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

OpenAI. Getting started with gym. `https://gym.openai.com/docs/`, 2021. Accessed: 2022-01-29.

OpenAI. Walker2d. `https://www.gymlibrary.ml/environments/mujoco/walker2d/`, 2022. Accessed: 2022-08-16.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

A COMPUTATION STEPS IN A GRAPH NETWORK

B LAYER-WISE RELEVANCE PROPAGATION METHOD

The LRP method is explained according to the work by Bach et al. (2015). the relevance scores from higher layers are introduced as messages sent from those layers. Therefore, the relevance of a neuron i at layer l (except the last layer) is computed as follows:

$$R_i^{(l)} = \sum_{k:\ i \text{ is input for neuron } k} R_{i \leftarrow k}^{(l,l+1)} \tag{8}$$

where $R_{i \leftarrow k}^{(l,l+1)}$ shows the relevance coming from neuron k in layer $l+1$ to neuron i in layer l. The relevance of the last layer is defined as the classification score $f(x)$. Equation 8 checks the sum of relevance scores with respect to the output neurons from the input neuron. We can consider the other way and check the sum of relevance scores of the input neurons for the output neuron:

$$R_k^{(l+1)} = \sum_{i:\ i \text{ is input for neuron } k} R_{i \leftarrow k}^{(l,l+1)} \tag{9}$$

Algorithm 2 Steps of computation in a GN block

function GRAPH NETWORK($E, V, \mathbf{u}$)
  for $k \in \{1 \dots N^e\}$ **do**
    $\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u})$ ▷ Compute updated edge attributes
  end for
  for $i \in \{1 \dots N^v\}$ **do**
    let $E'_i = \{(\mathbf{e'_k}, r_k, s_k)\}_{r_k=i, k=1:N^e}$
    $\overline{\mathbf{e}}'_i = \rho^{e \to v}(E'_i)$ ▷ Aggregate edge attributes per node
    $\mathbf{v}'_i = \phi^v(\overline{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$ ▷ Compute updated node attributes
  end for
  let $V' = \{\mathbf{v}'_i\}_{i=1}^{N^v}$
  let $E' = \{(\mathbf{e'_k}, r_k, s_k)\}_{k=1}^{N^e}$
  $\overline{\mathbf{e}}' = \rho^{e \to u}(E')$ ▷ Aggregate edge attributes globally
  $\overline{\mathbf{v}}' = \rho^{v \to u}(V')$ ▷ Aggregate node attributes globally
  $\mathbf{u}' = \phi^u(\overline{\mathbf{e}}', \overline{\mathbf{v}}', \mathbf{u})$ ▷ Compute updated global attribute
  return $(E', V', \mathbf{u}')$
end function

Equations 8 and 9 are the main constrains of defining Layer-wise Relevance Propagation. Multi-layer networks are commonly built as a set of interconnected neurons organized layer-wise. We denote neurons from layer l by x_i and neurons form layer $l+1$ by x_j. In the same manner, the summation over all neurons of layers l and $l+1$ are denoted by $\sum_i$ and $\sum_j$ respectively. A common mapping from one layer to the next one consists of a linear projection followed by a non-linear function:

$$z_{ij} = x_i w_{ij} \ , \tag{10}$$

$$z_j = \sum_i z_{ij} + b_j \ , \tag{11}$$

$$x_j = g(z_j) \tag{12}$$

where w_{ij} is the weight connecting neuron x_i to neuron x_j, b_j is the bias term, and g is a non-linear activation function. Common non-linear functions can be Rectified Linear Unit (ReLU) or hyperbolic tangent (tanh). One possible choice of relevance decomposition for messages from layer j to layer i is as follows:

$$R_{i \leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j}.R_j^{(l+1)} \tag{13}$$

This type of formalization guarantees the conservation properties of equation 2. One drawback of the equation 13 is that for small values z_j, $R_{i \leftarrow j}$ can take unbounded values. Two solutions provided to overcome this drawback are ε-stabilizer and $\alpha\beta$-stabilizer.

For the ε-stabilizer, let $\varepsilon \geq 0$, then the relevance scores would be as follows:

$$R_{i \leftarrow j}^{(l,l+1)} = \begin{cases} \frac{z_{ij}}{z_j + \varepsilon}.R_j^{(l+1)} & z_j \geq 0 \\ \frac{z_{ij}}{z_j - \varepsilon}.R_j^{(l+1)} & z_j < 0 \end{cases} \tag{14}$$

One problem with this method is that the relevance can be fully absorbed if the stabilizer ε becomes very large. For this case, we use the alternative $\alpha\beta$-stabilizer, which treats negative and positive pre-activations separately. Let $z_j^+ = \sum_i z_{ij}^+ + b_j^+$ and $z_j^- = \sum_i z_{ij}^- + b_j^-$ be negative and positive part of pre-activation respectively, where "$+$" and "$-$" denote the negative and positive values of z_{ij} and b_j. The relevance scores are then calculated as follows:

$$R_{i \leftarrow j}^{(l,l+1)} = R_j^{(l+1)} \left(\alpha.\frac{z_{ij}^+}{z_j^+} + \beta.\frac{z_{ij}^-}{z_j^-} \right). \tag{15}$$

where $\alpha + \beta = 1$. This method can also control the importance of positive and negative evidence by changing the values of α and β. The complete layer-wise relevance propagation procedure for neural networks is summarized in algorithm 3.

C APPLICATIONS OF OUR METHOD

Algorithm 3 Layer-wise relevance propagation for neural networks

1: **let** $R^{(L)} = f(x)$
2: **for** $l \in \{L-1 \dots 1\}$ **do**
3: $\quad R_{i \leftarrow j}^{(l,l+1)}$ as in equation 14 or 15
4: $\quad R_i^{(l)} = \sum_j R_{i \leftarrow j}^{(l,l+1)}$
5: **end for**
6: **return** $\forall d : R_d^{(l)}$