Towards Explainable Embodied AI

Masters Thesis

Submitted in partial fulfillment of the requirements of
for the degree of Master of Science in Robotics

By

Vidhi Jain

CMU-RI-TR-21-31

Thesis committee

Prof. Katia Sycara, CMU RI
Prof. Yonatan Bisk, CMU LTI & RI
Prof. David Held, CMU RI
Wenhao Luo, CMU RI

The Robotics Institute
CARNEGIE MELLON UNIVERSITY
Pittsburgh, Pennsylvania, 15213
August 2021

Copyright © 2021 Vidhi Jain. All rights reserved.

CARNEGIE MELLON UNIVERSITY

Masters of Science in Robotics

Abstract

Towards Explainable Embodied AI

by Vidhi Jain

The performance of autonomous agents has improved with the advancements in learning and planning algorithms, but the applicability of such agents in the human-inhabited world is limited. One of the factors is that humans find it difficult to interpret the model's decision-making and thus, do not trust it as a teammate. The goal of explainable embodied AI is to provide predictions with explanations that clarify the internal logic and are human understandable. In this work, we approach explainability in AI agent's policies by taking inspiration from how humans explain. First, humans are known to associate a few dominant factors while explaining their decision. We visualize such important features of the trained policies for control and navigation tasks by gradient-based attribution methods. Second, having shared knowledge for representing concepts often helps to understand and explain the decision-making. We utilize language priors in navigation algorithms for robots assisting in simulated urban households and search-and-rescue settings. Third, explainability comes with structured reasoning. To bring explainability in architecture design, we learn modular and hierarchical navigation policies for the task of maximizing area coverage in unseen environments. We conclude that embodied AI policies can be understood with feature attributions to explain how input state features influence the predicted actions. But feature attributions are not human intelligible in all cases, and attributions for the same policy is sensitive to the design choice of 'reference' or 'baseline'. A complementary direction is to develop inherently explainable policies by incorporating common knowledge priors and modular hierarchical components, that allow humans to understand high-level information flow and influence AI's decisions. We hope that the proposed explainability methods for embodied AI facilitate the analysis of policy failure cases in different out-of-distribution scenarios.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0036. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s). They do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

I want to thank my research advisor, Professor Katia Sycara, for her guidance, insightful suggestions, and patience. Her vision in human-robot teaming has deeply influenced me - to develop the ability of robots to understand and act like humans. She has been kind and supportive in my difficult times. Our discussions in lab alongside Dr Dana Hughes and Professor Michael Lewis, have driven me to think deeply at the intersection of robot learning and human social psychology.

I would also express my gratitude to Professor Yonatan Bisk, who has helped me extend the study of explainability in visual language navigation models. His enthusiasm for research to enable natural language understanding in robots for planning and control has inspired me.

Professor David Held's seminar course on *DRL with Robotics* covered various skill-centric robot learning, which inspired me to investigate deep reinforcement learning algorithms. I am thankful for the opportunity to dive into modular policy learning as part of the course project.

I would give special thanks to my collaborators – Huao, Rohit, Tejus for predicting navigation strategy for the human rescuer in simulated search and rescue scenario; Aviral, Akshay, Siddharth for visual dialogue in search and rescue scenario; Prakhar and Shishir, for object embeddings in urban household settings; and Ganesh, for goal-conditioned RL for exploration. I would also thank Swami, Sophie, Ini, and Wenhao, for brainstorming discussions and valuable feedback. I am thankful to Barbara Jean (BJ) and Prof. George Kantor for always promptly supporting and managing the MS in Robotics program.

I could not have become the researcher I aspire to be without my family, who have stood firm and encouraging amidst all the uncertainty. I thank them for providing care and mental strength for this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We, as human beings, have always striven to create intelligent systems that increase our efficiency and reduce monotonous work. Developing intelligence in robots to act as our teammates and enable success on complex tasks is a long-standing challenge for AI.

From the early days of AI, algorithms consisted of explicit instructions written in human-readable symbols and logical statements, colloquially referred to as Software 1.0. The high-level language provided step-by-step interpretable behavior for prediction and inference. Early AI explanation methods focused on rule-based insight [53] into the AI system's internal working. These systems were expensive to build, maintain and extend to the complexity of the real world. Customized the AI architecture for military mission simulation such as [31, 23, 29] provided domain-specific explanation and .

Recently, machine learning algorithms provided an indirect way to specify how to learn from the data, often termed as Software 2.0. Such code optimizes a high-level desired goal like balancing the inverted pendulum or following natural language instructions during navigation. The resulting behavior after data-driven learning is usually incomprehensible to humans, such as the weights of a neural network.

With increasing success in machine learning today, we have autonomous systems that can perceive visual, language, and sensory signals to act in our environment. The success of machine learning algorithms is evaluated based on higher prediction accuracy than other methods. But the underlying reasons for prediction are hard to assess [30, 41]. Often machine learning models are considered "black-box" in terms of how the implicit rules and patterns predict the output. The rapid adoption of such data-driven AI methods has necessitated better explainable AI.

1.1 Explaining "Explainability"

Explainability in AI aims to enable humans to understand the workings and trust the results of machine learning and data-driven algorithms. An explanation should clarify the internal logic and be human intelligible.

While programming explicit instructions (assuming not a halting problem code), the reasoning for the outcome can be tractably deduced and thus, explained. For example, it is possible to explain the result of a nested if-else algorithm or optimal planning like Dijkstra's or A* search in terms of the logical path of statements executed.

For data-driven models that are rule-based, like Decision Tree, the logical path taken for a particular prediction or outcome has been used to explain it. Recent approaches like [34] discuss game-theoretic feature attribution approaches to explain ensembles of decision tree algorithms.

Deep neural network (DNN) models are considered universal approximators. Unlike the family of decision tree algorithms, DNNs can predict images, text, and other complex formats of input-output data. The prediction rules are implicitly encoded in the form of weights and biases. For a linear model, the coefficients of each input feature indicate their relative importance for prediction. The equivalent of coefficients is taking gradient with respect to the input features. This work will discuss some of the explainability techniques on DRL models, where the actor neural network (also known as the policy) predicts actions.

1.2 Why Explainability?

Explainable AI is essential for human-robot collaboration so that the human counterparts can understand the working and trust the prediction models. While working with a machine learning model making predictions or deciding navigation action, it is imperative to know when the model will likely succeed or fail and why the model makes a particular decision.

Explainability by attribution is also helpful in detecting bias in training dataset or optimization and thereby mitigate the situation. For embodied AI, ensuring safety before deployment is crucial. Due to distribution shift, it may not be possible to validate in simulation alone. Further, even if simulation and real-world accuracy metrics are at par, failure cases in the tail distribution of the training data can go undetected.

1.3 Research Questions

In this thesis, we attempt to answer the following questions:

1. Can feature attribution methods help to explain deep reinforcement learning and sequential prediction methods? (Chapter 3, 4)

When we, humans, are asked to explain decision-making, we often attempt to describe what aspects we found salient. Similarly, we can use feature attribution methods (2.1) to identify what input features did the model gave positive or negative importance to in the decision-making process. We find that such methods can be applied to explain input features of deep reinforcement learning (2.2) models and decision trees to explain the predicted outcome.

2. Can language priors be incorporated in policies for object search tasks? (Chapter 5, 6)

While we actively gain and have accumulated knowledge of the world, this is not true for machine learning algorithms. Language (2.3) provides a way for humans to interpret the world and explain abstract concepts. Incorporating language into the algorithm design creates a common shared knowledge set and brings built-in interpretability for the embodied AI policy. We incorporate knowledge priors in policy design and apply in the tasks to search objects in urban households and to mitigate disaster rescue missions.

3. Can hierarchical learning outperform heuristic path planning for efficiently exploring the environment? (Chapter 7)

The structure of navigation policies becomes inherently interpretable with modular, hierarchical learning (2.4), where each module has a specific input-output scheme. The modular structure also improves sample complexity and mimics the human way of thinking. We consider the task of exploring the environment in minimum steps for model-free learning algorithm and compare the performance to heuristic path planning.

1.4 Contributions

We present explainability techniques for designing embodied AI policies for navigation. Our work is one of the first steps towards developing social-cognitive skills in AI agents that involves the ability to think about mental states, both your own and those of others, to eventually aid in human-robot trust and collaboration.

In the first half, we explore the applicability of input feature attribution methods for explainability. These models include decision making from a first-person perspective (ego-agent: how to control or navigate to the goal by learning from experience) and from a third-person perspective (observer-agent: how will the other ‘agent’ behave or what is their strategy given the behavior of other agents on this task).

In Chapter 3, we show that gradient-based feature attribution methods can explain the actor neural network’s predictions in terms of the relative importance given to the input state features for action prediction. We analyzed classic control tasks for cart pole and a lunar lander with SHAP Deep Explainer. We also investigate these methods on higher-dimensional inputs like pixels in ALFRED with XRAI attributions.

In Chapter 4, we design interpretable state features from human trajectories to train gradient boosted decision trees that predict the victim triage strategy and prior knowledge condition in simulated search and rescue task. To understand what features are considered salient by the model for predictions, we visualize the feature attributions on unseen trajectories. The top attributed features are then processed to select the most likely natural language explanation for the prediction.

While input feature attribution is a powerful tool for ML and DRL models, such explanations are only expressed in terms of the input features to the model. It does not cover other aspects, like the common knowledge and language priors. In the second half, we design embodied AI policies that integrate domain knowledge and planning for built-in explainable architecture.

In Chapter 5, we learn and contrast representations to capture spatial semantics. We demonstrate that a momentum inspired, similarity-based greedy navigation technique results in success rate $> 90\%$. We show that using spatial semantic prior knowledge provides an intuitive way to search for objects in common households without compromising the navigation performance.

In Chapter 6, we propose a system for collaboration between an autonomous agent in a disaster site for search and rescue mission, and a human located in a safe remote environment. Just like the human first-person responders would communicate with their teammates situated remotely, we build a system that enables an autonomous agent to parse a human’s natural language message and either (1) generate a natural language response for the query, or (2) interact in the environment to execute the instruction/-command. We focus on the qualitative analysis to demonstrate the capability of the latter module across four simulated disaster scenarios for natural language interaction commands.

In Chapter 7, we develop a customized version of the classic `FourRooms` environment, with a realistic human-like field of view, and variation in clutter density γ for evaluating the navigation policies for the task of maximizing area coverage in unseen environments. We observe that as the clutter density increases, the performance of modular learning based agent improves as compared to the heuristic of nearest frontier based exploration.

In Chapter 8, we discuss the takeaways from these research questions as well as a variety of other ways to explain. We conclude that embodied AI policies can be understood with feature attributions to explain how input state features influence the predicted actions. But feature attributions are not human intelligible in all cases, and attributions for the same policy is sensitive to the design choice of 'reference' or 'baseline'. A complementary direction is to develop inherently explainable policies by incorporating common knowledge priors and modular hierarchical components, that allow humans to understand high-level information flow and influence AI's decisions. We hope that the proposed explainability methods for embodied AI facilitate the analysis of policy failure cases in different out-of-distribution scenarios.

Chapter 2

Background

2.1 Feature Attributions

Consider a linear model $f(x) = w_1x_1 + w_2x_2$. To explain which feature is more important for predicting the value of f(x), we can compare their coefficients. If $w_1 = 1000$ and $w_2 = 0.01$, we can say that x_1 would be weighed more than x_2. This type of explanation assumes that the values of x_1 and x_2 are of the same order. This is true in the case of most inputs to the neural network models, for example image pixels. Gradients are the general way of discussing the coefficient with respect to a particular feature to discuss its importance. Feature importance or attribution is an approximation of how important features are in the data.

One of the ways to visualize the feature attributions at every instance is by force plots. **Force plot** [33] show which features and by how much contribute positively (denoted by arrows in red to right) or negatively (denoted by arrows in blue to left) to the output. The sum of the attributions is equal to the difference between the output value $f(x)$ and the base value. The longer the arrow, the more dominant the feature is for the predicted outcome.

Shapley values The original concept was proposed by Lloyd Shapley in 1950s [44] to describe the payoffs to the players in a coalitional game, where players collaborate to generate some value. The objective is to determine what each participating player should receive from the total reward obtained by the team based on their 'contribution' to the team. Shapley values consider the average payoff based on the sequence of arrival of the features, but this is prohibitively expensive to compute when the number of features is large. SHAP provides a fast implementation for tree-based models for Shapley values. TreeSHAP [34] can fairly distribute the importance to each feature contrastive explanations that compare the prediction with the average prediction. In machine learning, the objective is the same, except the 'players' are the features of the ML model, and the total reward corresponds to the predicted output.

Integrated Gradients While gradient with respect to input is a starting point for attribution, these are sensitive to noise. [48] proposed ways to visually sharpen these vanilla gradient-based attributions [46] applying Gaussian noise perturbations over averaged over a sufficient number of samples. IG [52] and path methods have been studied as a cost-sharing method called Aumann-Shapley. Attribution based on IG preserves axiomatic properties like *sensitivity* and *implementation invariance*.

While IG aggregate the gradients on sampling inputs on a straight line between the baseline and the input, there are several paths possible in higher dimensional spaces and corresponding different attribution. Recent works build on IG to obtain more visually intuitive attributions, like in SHAP Deep Explainer [51], Blur IG [57], Guided IG [25] and XRAI [24]. More details on various attribution methods are in Appendix A.

Some explanations use the activations in layer l produced by input examples in the concept set versus random samples. [26] define a 'concept activation vector' (or CAV) as the normal to a hyperplane separating examples without a concept and examples with a concept in the model's activations.

2.2 Reinforcement Learning

Notation Deep reinforcement learning algorithms have been successful in discrete and low-dimensional action spaces. The standard RL setting has an agent which interacts with the environment over periods of time according to the policy π. At each time step t, the agent receives the state observation $s_t \in \mathbb{R}^|\mathcal{S}|$ from the environment. The agent then samples an action $a_t \sim \pi(s_t)$ where $a_t \in \mathcal{A}$, and acts in the environment to transition into the next state observation s_{t+1} and receive reward $r_t \sim R(s_t, a_t)$. The objective for the agent is to predict actions given state observation such that the discounted reward is maximized in expectation.

Actor-Critic paradigm [38] 'Actor' is responsible for taking action given the current state, and a 'Critic' estimates the "value", a scalar estimate of importance, of the given state. Actor and critic can be any function approximators like neural networks. Often, both the models have a common trunk for both to process the current state and separate heads for their respective predictions. Different algorithms leverage the information learned by the critic based on the reward and use it to inform the actor for learning optimal policy. We use Deep Q-learning [39] and policy gradient algorithms like PPO [43] for the experiments in lunar lander and cartpole respectively in chapter 3.

2.3 Language based navigation

Language-driven navigation often involves a speaker describing navigation instructions to a listener to reach from one point to another [27, 55]. Interactive question answering by Gordon et al. [19] considers cases that require navigation and answer questions only about counting, spatial relationships and existence questions. Instead, we consider disambiguating the natural language message into parts that need question answering about the environment, or that require navigation to a location.

More complex long-horizon task descriptions are also recently introduced [45], ALFRED Dataset, provides the similar task of human-agent collaboration in simulated indoor house environments requiring challenging long-horizon planning. We visualize the feature attribution on the visual and language inputs for models trained on ALFRED dataset in chapter 3 section 3.3.3.

Synthetic Language Learning BabyAI, a 2D grid setup with synthetic language learning [14, 12] proposed efficient testing of sample complexity of reinforcement learning algorithms at language-driven navigation. Synthetic generated language has been preferred in prior works [1, 35], not only for its cost-effectiveness but also to mitigate the bias introduced with data collection. In chapter 6 section **??**), we use template-based language generation to create dataset for search and rescue task and demonstrate natural language based navigation.

Object Embeddings Classical navigation techniques in robotics generalize to unseen environments by mapping, localization, and path planning. However, most of these approaches fail to leverage the environment's semantic structure, like certain objects are mutually close to each other. Multi-relational knowledge-base based embeddings such as RoboCSE [16] have demonstrated an agent's ability to predict object affordances and materials in the real world. The RoboCSE embeddings were trained using the object metadata from AI2Thor simulator. While these embeddings were used for the prediction of object affordances by an immovable agent, we differ in our approach of utilizing these embeddings by evaluating how well they capture the spatial semantics for query search and navigation task for a mobile agent.

Recent work in object navigation [9] shows a modular approach to map the environment and predict sub-goal based on the semantic object map. In chapter 5, we do not address the vision-based mapping as addressed. Instead, our approach is a modular component that can be integrated with classic robotics paradigm of mapping and planning. We focus on high-level navigation decisions aligned to the spatial semantics of objects.

2.4 Modular Hierarchical Learning

Hierarchical Reinforcement Learning (HRL) We consider decomposing the actual reinforcement learning problem down into sub-problems such that solving them leads to a more efficient or powerful solution to the original problem. With the advent of Deep Learning, hierarchical representations are considered as an interaction between multiple policy networks. Such representations have been useful in solving all kinds of tasks. [28] was the first to introduce a hierarchical DQN (H-DQN) structures operating at different time scales to solve the sparse reward problem of Montezuma's revenge. [2] also propose a hierarchical PPO formulation for the gym-minigrid [13] environments. More recently, hierarchical policies have also been used for embodied question answering. [17] decomposes the problem of navigation for question-answering into a planner and controller framework, with the controller pre-trained by imitation learning based method.

Robot Navigation The concept of frontiers for exploration of unknown spaces was first introduced in [58] by Yamauchi, in which regions on the boundary between known and unknown regions are considered valuable targets to explore to increase information about the environment. Traditionally, frontier based exploration is focused on geometric methods for navigation using SLAM, as demonstrated in works like [49], [15], and more recently deep learning based methods like [5].

More recently, active learning methods for navigation have been proposed as a solution to downstream robot control tasks, that propose to navigate end-to-end with implicit mapping. In many of these works, the selection of frontiers is implicit, and is attributed to semantic priors [60], or geometric priors [11]. Some of the work that comes close to our work regarding the explicit use of frontiers is [50], where the expected cost value for each state action pair is calculated by observing trajectories of an optimistic planner.

In the context of Visual Language Navigation on ALFRED benchmark [45], recent state-of-the-art agents often have modular approach to train the policy to imitate the expert's trajectory. MOCA [47] decouples the visual perception and decision to act. For selecting the object to interact with, it first predicts target class, followed by the instance associated pixel-level mask. This explicit modeling improves model explanability as well as the performance. HiTUT [61] leverages pretrained object detector for the visual input and converts them to token for the word embeddings to process later with BERT architecture.

Chapter 3

Attributions for Actor-Critic Neural Networks

Explanability in machine learning is obscured by the implicit rules and patterns learned from the training data. Gradient-based feature attribution method address this the feature importance of any input instance, given access to the training data and model. While several of these methods have been proposed and studied for image and text classification tasks, it is not yet explored in reinforcement learning and general sequence modeling.

3.1 Key Contributions

We show that gradient-based feature attribution methods can explain the actor neural network's predictions regarding the relative importance given to the input state features. We analyzed classic control tasks for cart pole and a lunar lander with SHAP Deep Explainer. Given the low-dimensional state input to the controller, we found that the attributions on the randomly initialized network have small magnitudes and relative difference, which increases with the training of the policy. We also investigate these methods on higher-dimensional inputs like pixels in ALFRED with XRAI attributions. We demonstrate visualizations of the feature attribution methods on policies trained with PPO, DQN, and imitation learning. Our analysis of feature attribution of a policy provides insights on where the model is likely to make mistakes and to which features variations the model will be robust.

Algorithm 1: Attributions for DRL policy over a trajectory given the initial state s_0

```
Result: list of attributions $\mathbf{V}$
time instance $i$;
environment env;
state $s_i \in \mathbb{R}^S$;
action $a_i \in \mathbb{R}^A$;
policy $\pi_\theta : s_i \rightarrow a_i$;
buffer $\mathbf{B}$;
max steps $T$;
number of roll-outs for background data $N$;
// collect rollout state observations for baseline
for $j \in \{1, 2, \cdots N\}$ do
    while not done or $t < T$ do
        $a_t = \pi_\theta(s_t)$;
        $\mathbf{B} \xleftarrow{append} s_t$;
        $s_{t+1}, r_t,$ done $=$ env.step$(a_t)$;
    end
end
// compute attributions over the trajectory
while not done or $i < T$ do
    $a_{i+1} = \pi_\theta(s_i)$;
    $v_i =$ Attributions$(s_i, \pi_\theta, \mathbf{B})$;
    $s_{i+1}, r_i,$ done, info $=$ env.step$(a_i)$;
    $\mathbf{V} \xleftarrow{append} v_i$;
end
return $\mathbf{V}$, a list of $v_i$, each of size $S$ (same as dimensions of $s_i$).
```

3.2 Method

We visualize the actor neural network's attribution in terms of certain action as the actor network is only used during deployment and the critic network is discarded. We chose action based on the models' prediction or expert's action at that timestep.

SHAP Deep Explainer or DeepSHAP [10] is a In DRL, we propose to sample states from the rollout of the trained policy to use as baseline.

The attributions for the actor policy is computed as shown in Algorithm 1. We train the agent with model-free DRL algorithms like DQN, PPO[1]. We extract the pipeline of the Actor neural network for gradient based saliency methods, and leave out the critic network. While visualizing the critic network would give us insight on the salient features for determining the value of the state, it is harder to comprehend as compared to a tangible action taken.

[1] We utilize exisiting RL frameworks like `rl-starter-files` and `stable-baselines3` to train our agents.

We often need background data for methods such as [52]. Often, in image and text classification, this is the training data. However, the training data in deep reinforcement learning depends on how trained the policy is. In the start of the training, the training data can be biased with large amounts of initial state configurations. To collect suitable background data, we rollout the trained policy for N episodes to create a buffer $\mathcal{B}$ of state observations.

For a required trajectory to be explained, we then compute attribution for the state's input features to the actor neural network for either the most likely action taken or the expert's action at that state, if available. The attribution function takes input: the actor or policy, background data buffer and the current state observation, and outputs the importance for each feature in the input current state. This importance or attribution is in regards to the predicted action probability. This answers which features contributed positively to prediction of the most likely action and which features contributed negatively.

3.3 Experiments

In the following set of experiments, we assume that the equations of motion of the system are not known apriori. The control policy needs to learn how to apply force to prevent failure conditions. Table 3.1 shows the input and the output of the policy for the actor in each environment.

Table 3.1: Input and output space for agent's policy in different environments

Environment	**Input**	**Output**
Cartpole	Cart's position x	Push the cart to the left
	Cart's velocity x˙	Push the cart to the right
	Pole's angle	
	Pole's angular velocity	
Lunar Lander	Horizontal coordinate	No-op
	Vertical coordinate	Fire Left engine
	Horizontal speed	Fire Main engine
	Vertical speed	Fire Right engine
	Angle	
	Angular speed	
	1 if first leg has contact, else 0	
	1 if second leg has contact, else 0	

3.3.1 Cartpole

Cartpole is a classic benchmark problem in control theory and reinforcement learning, described by Barto, Sutton, and Anderson [3]. It is also known as the inverted pendulum.

Figure 3.1 shows that the system consists of a cart free to move horizontally with a pole hinged to its top. The pole is allowed to rotate about this joint in 2D, under the influence of gravity. The input and output of the control policy is shown in Table 3.1. Failure conditions are (a) pole falls beyond 12 degrees, (b) car hits the boundary at 2.4 units from the center. For every time instance, the controller gets a reward of +1 until 500 steps.

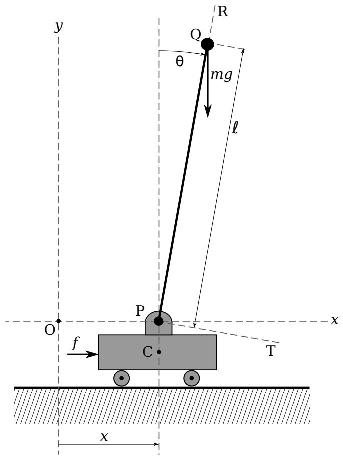


FIGURE 3.1: Cartpole system design

We train the cart-pole controller with Proximal Policy Optimization (PPO) [43] on a two-layer feed-forward neural network [2]. For a trained policy for cart-pole controller, we analyzed the feature attribution. These attributions were calculated for the actor neural network over 1000 observations sampled from buffer of size 50000.

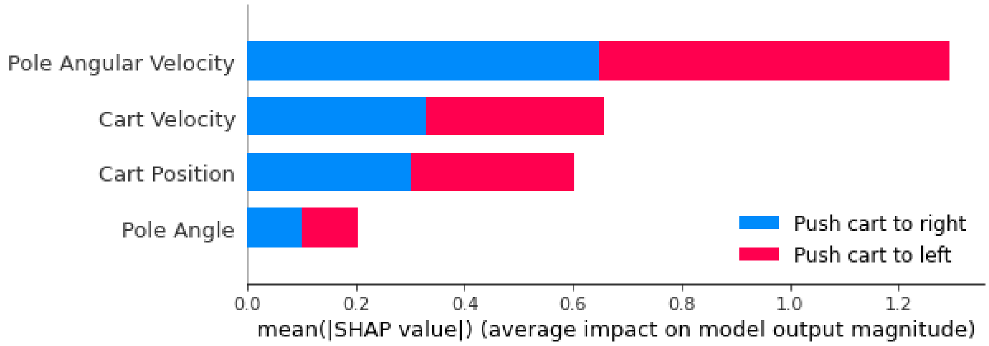


FIGURE 3.2: Summary plot of feature importance for a cartpole agent showing that Pole angular velocity is given the most importance in deciding the direction to push the cart.

Fig 3.2 shows the summary plot of feature importance of a trained policy over a rollout of 10000 timesteps. Pole's angular velocity is an important feature in influencing most of the actions, while Pole's Angle and Cart's position have relatively lower impact on the action predicted.

To visualize feature attributions for prediction at each timestep, we use force plots. **Force plot** [33] show which features and by how much contribute positively (denoted by arrows

[2] Default MlpPolicy in stable-baselines3; Code: cartpole ipynb, SHAP cartpole ipynb)

in red to right) or negatively (denoted by arrows in blue to left) to the output. The sum of the attributions is equal to the difference between the output value $f(x)$ and the base value. The longer the arrow, the more dominant the feature is for the predicted outcome.

The force plot shown in Fig 3.5 for time 0 when the trained policy predicts to take action “Push the cart to left”. Note that the function value $f(x)$ is lower than the base value for the chosen action. This is because at time 0, the cartpole is initialized with random values uniformly between -0.05 to 0.05 and this configuration happens rarely only at the start of the task. As the base value is the average function value in the training data, this indicates that model is not confident on the action to take at this timestep. Cart’s position, which is overall a less impactful feature, appears to be more dominant than pole’s angular velocity at this instance.

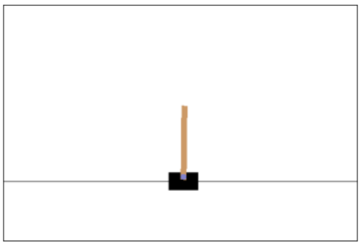


Figure 3.3: CartPole at **time 0**. Cart’s Position: +0.0427, Cart’s Velocity: +0.0169, Pole’s angle: +0.0116, Pole’s angular velocity: +0.0079. Action taken: **Push the cart to left**.

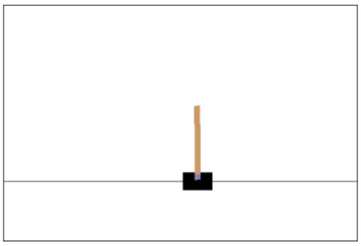


Figure 3.4: CartPole at **time 49**. Cart’s position: +0.2352, Cart’s velocity: +0.5805, Pole’s angle: -0.001, Pole’a angular velocity: -0.3915. Action taken: **Push the cart to right**.

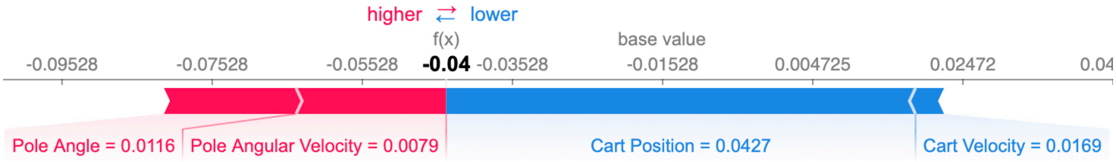


Figure 3.5: Force plot for CartPole at **time 0** for action **Push the cart to left** Cart’s position is to the right and is a dominant feature is deciding this; Pole’s angular velocity is opposing this decision. The pole is already tilting towards right. Pushing it towards left will make the pole fall faster.

Let us visualize and analyze attribution for an instance later in the trajectory. For example, at timestep 49, we observe for predicted action “Push cart to left”, the most positively attributed feature is pole’s angular velocity as shown in Fig 3.6.

The feature attribution along the trajectory can be visualized in Fig 3.7 where the positive $f(x)$ value corresponds to the action “Push the cart to the right”.

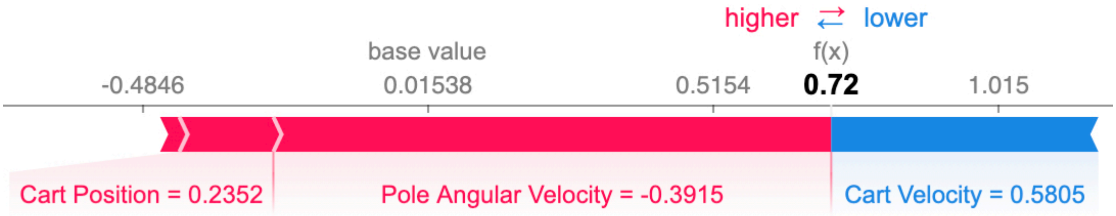


FIGURE 3.6: Force plot for CartPole at **time 49** for action **Push the cart to right**. Pole's angular velocity plays a dominant role in deciding the action. Cart's velocity is opposing this action. Cart is moving towards right and pushing to the right would drive the cart further to the right, which can terminate the episode if cart goes beyond the frame.

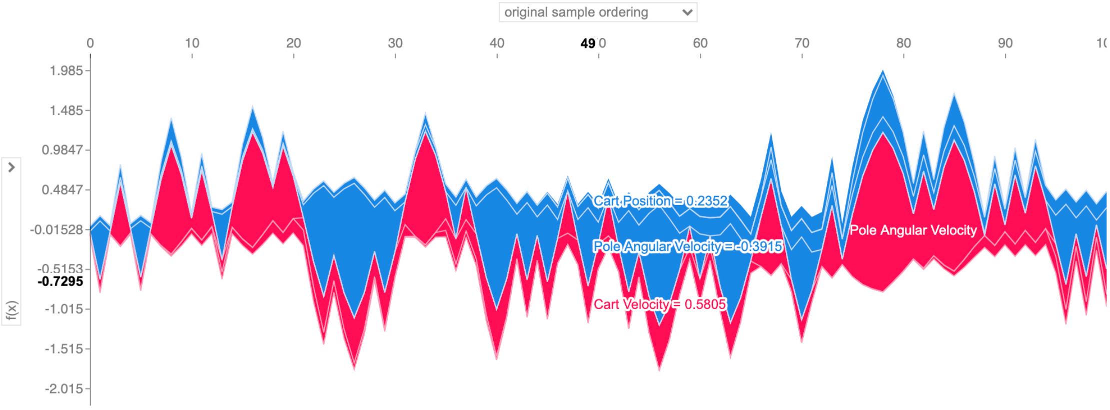


FIGURE 3.7: Aggregated Force Plot for a rollout over 100 timesteps. y-axis represents the output value for action to **push the cart to right**. x-axis represents the timesteps. The force plot shows what features positively or negatively attribute to the predicted action over time

Pole angle's effects correlate with the value of the pole angle. For higher positive values of angle, the attribution to push the cart to right increases, and vice versa for higher negative values of angle, the attribution to push to left increases, as shown in figure 3.8.

Cart position in this trajectory remains always positive, implying that the cart is always to the right of the origin. In figure 3.9, we can see that the effect of cart's position on the predicted action is always to push the cart to the left.

Cart's velocity and pole's angular velocity can take any value in $(-\infty, \infty)$. These values change faster than corresponding positions, and we see frequent fluctuations in the attributions. While usually their signs correlate with their attributed action, there are opposite effects near the transitions.

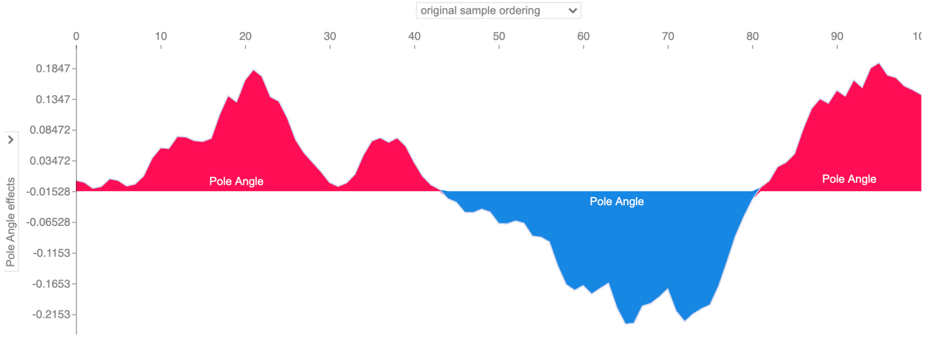


FIGURE 3.8: Trajectory Force Plot for **Pole Angle**. Pole angle's effects correlates with the value of the pole angle. For higher positive values of angle, the attribution to push the cart to right increases, and vice versa.

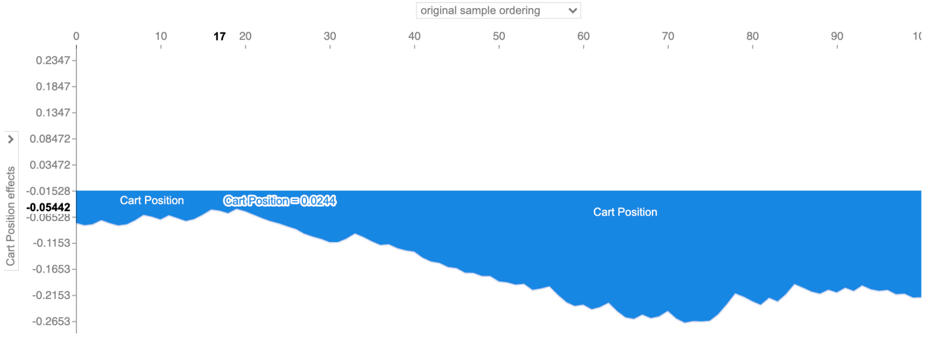


FIGURE 3.9: Trajectory Force Plot for **Cart Position**. Cart position in this trajectory remains always positive, implying that the cart is always to the right of the origin. The effect of cart's position on the predicted action is always to push the cart to the left.

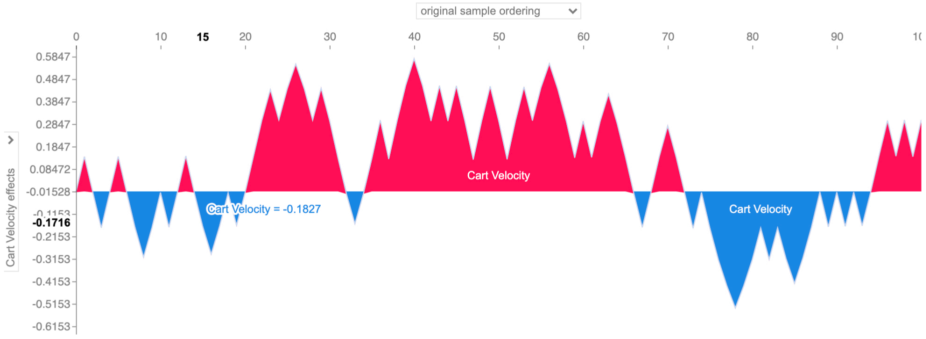


FIGURE 3.10: Trajectory Force Plot for **Cart Velocity**. The effect of Cart's velocity almost correlates with but may not be proportional to the value of the velocity. For higher positive values of cart's velocity, the attribution to push the cart to right increases, and vice versa.

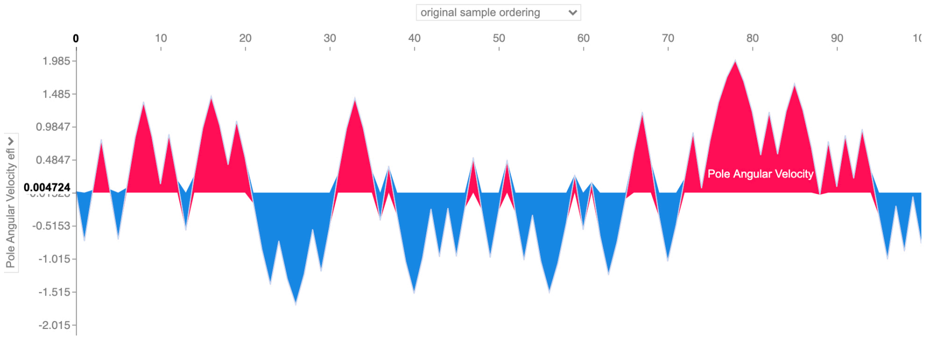


FIGURE 3.11: Trajectory Force Plot for **Pole Angular Velocity**. The effect of Pole's angular velocity almost correlates (except at transitions) with the value of the velocity. At higher positive values of pole angular's velocity, the attribution to push the cart to right is high, and lower values it may switch its effects.

3.3.2 Lunar Lander

Another classic control problem is optimizing rocket's trajectory for landing. The input observation and the expected output of the controller is described in Table 3.1.

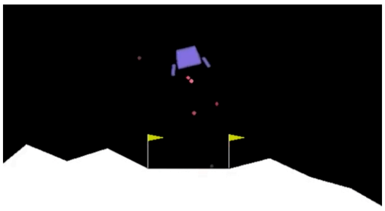


FIGURE 3.12: Lunar Lander environment

The objective of the controller is to land the spacecraft safely at coordinates (0,0). Let d_t is the distance from the landing coordinate (0,0), v_t is the lander's velocity, θ_t is the angle from the vertical direction, *hasLanded, mainEngineFired, sideEngineFired* are boolean values of the system at time t. To shape the reward, points are rewarded or deducted based on the system state as follows.

$$\begin{aligned}\text{Reward}(s_t) = &- 100 * (d_t - d_{t-1}) - 100 * (v_t - v_{t-1}) - 100 * (\theta_t - \theta_{t-1}) \\ &+ 10 \times (\mathit{hasLanded}(s_t) - \mathit{hasLanded}(s_{t-1})) \\ &- 0.3 \times \mathit{mainEngineFired} - 0.03 \times \mathit{sideEngineFired}\end{aligned}$$

The episode finishes if the lander crashes or comes to rest, receiving an additional -100 or +100 points. We train the controller policy by Deep Q-learning with three layer feed-forward network, with 128, 64, 32 units per layer. We use learning rate = 0.001, batch size = 64, relu activation, and final epsilon for exploration = 0.01.

Our trained policy obtains mean reward of 111.23 ± 66.58 over 13 episodes with minimum 23.32 and maximum 250.03. We visualize the attributions for 200 sampled states. We first sample 1000 states from a policy rollout of 10000 steps and use remaining 800 of them as baseline.

Fig 3.13 shows the mean attributions of the policy. The vertical coordinate is identified as a dominant feature. It likely so because the decreasing vertical coordinate to 0 at each timestep is positive reward. Focusing on the contact to the land is second most important feature as it determines another +100 or -100 penalty for the controller. Vertical speed of descent is equally important if the lander is not in contact yet.

The horizontal coordinate, horizontal speed, angle and angular speed become important if the spacecraft deviates from the normal.

Fig 3.14 shows the feature attributions at an instance where the action to fire the main engine has been taken. The non-zero value of angular speed and angle, as well as positive value of the vertical coordinate of the spacecraft contribute to this action. On the other

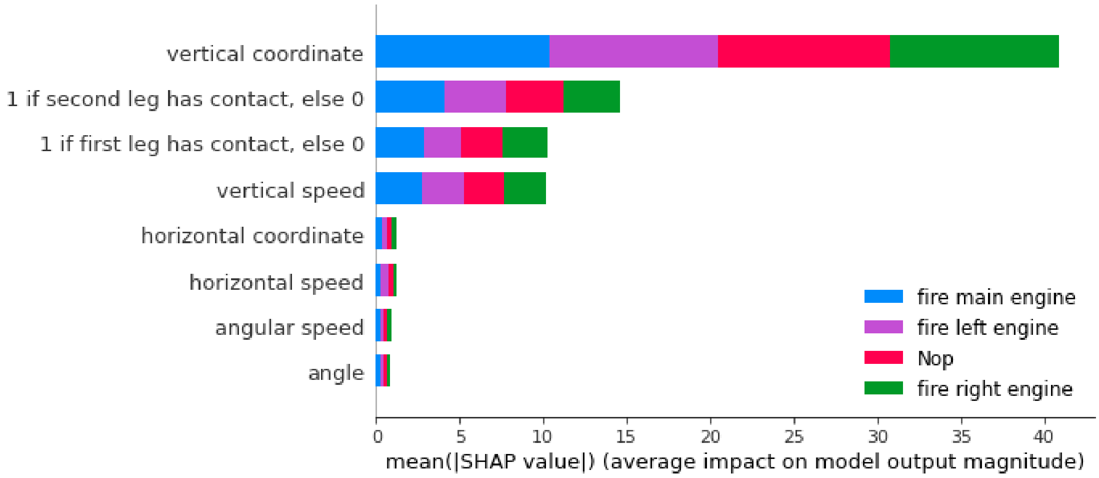


FIGURE 3.13: Summary plot visualized for DQN actor policy in **Lunar Lander** showing that `vertical coordinate` is the most dominant feature for the policy's predicted action.

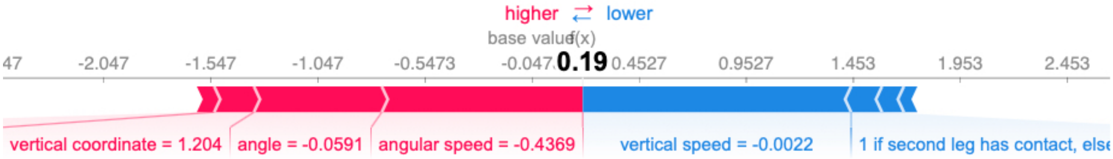


FIGURE 3.14: Force plot for the DQN actor policy in Lunar Lander; **Action taken:** Fire Main Engine; **Observations:** horizontal coordinate: 0.10422049, (+), vertical coordinate: 1.2217543, (-), horizontal speed: 0.20320427, (-), vertical speed: -0.06951332, (-), angle: 0.020212715, (+), angular speed: 0.17456493, (+), 1 if first leg has contact, else 0: 0.0, (+), 1 if second leg has contact, else 0: 0.0, (-); **Reward:** 0.7576; **Total reward** at the end of episode: 33.81; Angular speed is contributing positively whereas the downward vertical speed is opposing the decision to Fire the Main Engine.

hand, the negative vertical speed suggests that the spacecraft is already descending and therefore, it negatively attributing to the decision to fire the main engine.

3.3.3 Visual Language Navigation

Consider an agent who understands natural language instructions and can navigate and interact with objects in a typical house environment. The objective is to predict the actions that would complete the task, given natural language goal, instructions and egocentric vision. Recent works evaluate these agents on the overall task goal completion success rate (SR) and that weighted by expert's path length (PLWSR) over seen and unseen tasks, and have reported a huge gap in the performance of learning algorithms and humans at these tasks.

To understand why ALFRED benchmark is hard, we compute and visualize the attributions of the policy network for the most likely predicted action, given the expert's

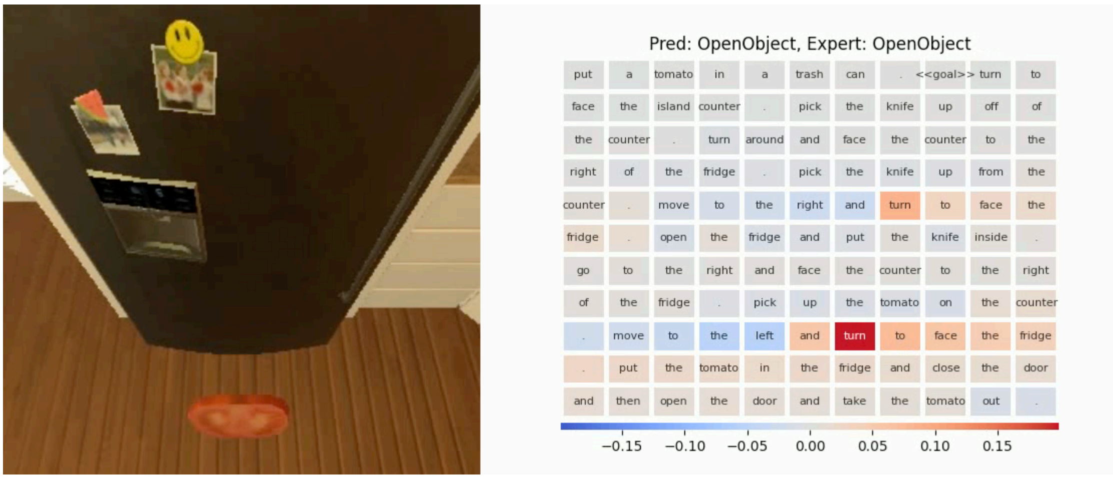


FIGURE 3.15: Language Attribution for the task to place the tomato slice in fridge. The words 'turn to face the fridge' have positive importance to the action of OPENOBJECT; whereas other words have neutral or negative attribution to the action taken.

trajectory inputs at current timestep. Expert's action is provided at each instance based on PDDL planning in the simulated environment.

The input to the policy is word embedding of goal and instructions, the current frame, the previous action and the latent state representation. The output of the agent's policy is one of 13 possible actions[3] with 5 navigation and 7 interaction based actions as shown in Appendix A.2.1.

Language Attributions To visualize the language related feature attribution, we computed the gradient of the predicted output with respect to embeddings of goal and instruction word tokens. As an embedding is n-dimensional vector, the gradient is also computed with respect to each dimension. One of the ways to reduce the attribution $\boldsymbol{a}_w$ of embedding size d, where w is a word token, to a scalar is by choosing the value which has the maximum absolute value as the attribution. This value can be positive or negative attribution. $\max abs(a_w)$

Figure 3.15 shows the gradient based language attribution. The goal and instruction tokens at an instance in the trajectory. The agent's predicted action (and the expert's action) is to `OpenObject`, which in this case is to open the fridge. The policy shows positive attribution to "turn to face the fridge" tokens, while negative or neutral attribution to tokens from previous sub-goals.

Visual Attributions

We visualize where the policy is attributing importance in the current frame with XRAI [24]. Figure 3.16 shows the attributions for the predicted most likely action to `MoveAhead`, instead of expert's action `RotateRight`. The leftmost figure shows the

[3] Note that two of –pad– and –seg– in action vocabulary are dummy values.

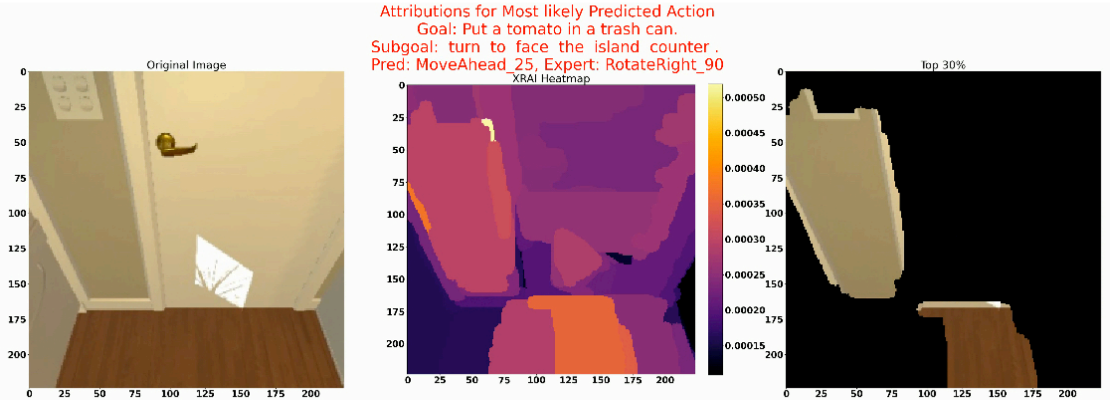


FIGURE 3.16: XRAI attributions on current image frame for predicted action: `MoveAhead_25`. The leftmost figure shows the original frame pixels input to the trained policy. The middle figure shows the attribution heatmap. The righmost figure shows the areas with top 30% attributions.

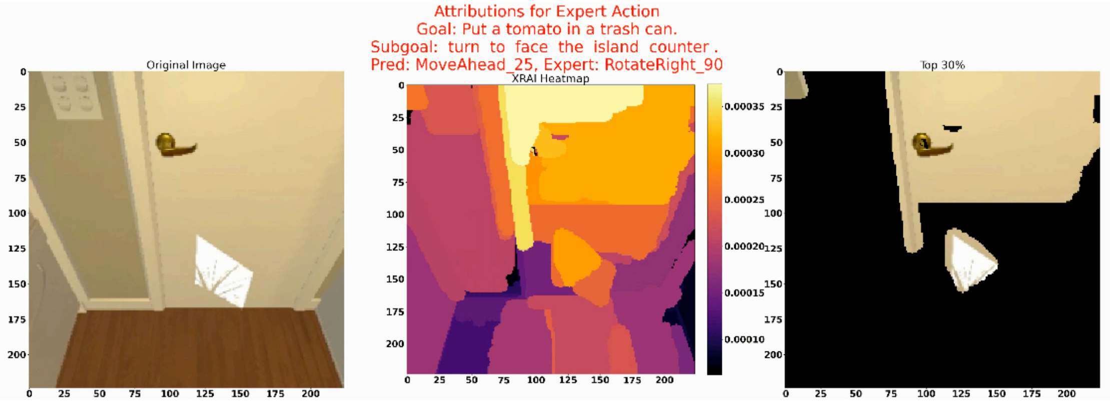


FIGURE 3.17: XRAI attributions on current image frame for expert's action: `RotateRight_90`

original frame pixels input to the trained policy. The middle figure shows the heatmap of attributions, where the light yellow corresponds to high positive attributions and dark purple corresponds to neutral or negative attributions. The rightmost figure shows the top 30% of the attributions. At this instance, it seems that the policy is focusing on the empty wooden flooring in front of it to predict the action of moving forward.

In Figure 3.17, the attributions at this instance is visualized with respect to that of the expert's action. Often, the correct action cannot be judged straight-away. For example, rotating left or right could be equally good options if this is first frame that the agent sees and does not know the structure of the environment in advance.

Another example of where the policy's prediction and expert's action differ are shown in Figure 3.18. Here we see that the policy's attributions to action 'OpenObject' are quite driven by the presence of object 'Fridge' in the current frame. The policy clearly fails to focus on the subgoal which requires it to go right side of the fridge, and does not take into account its interaction with the fridge in the previous timesteps.

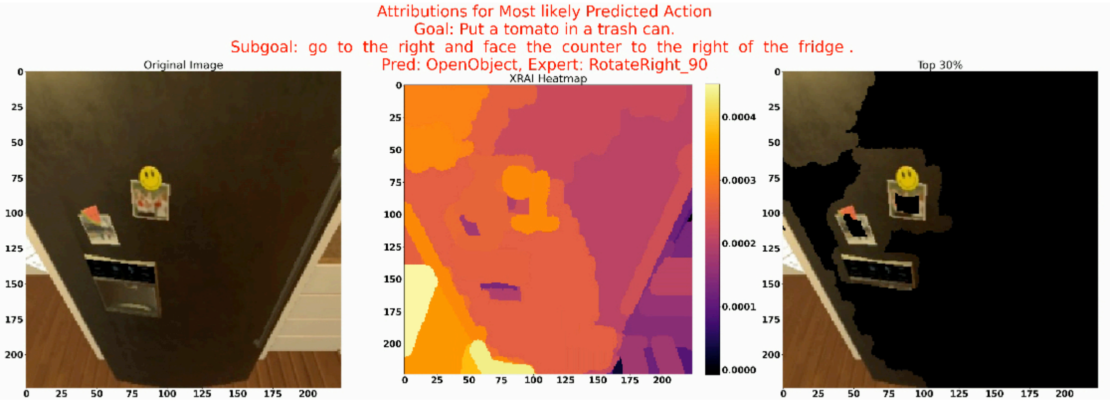


FIGURE 3.18: XRAI attributions on current image frame for predicted action: `OpenObject`

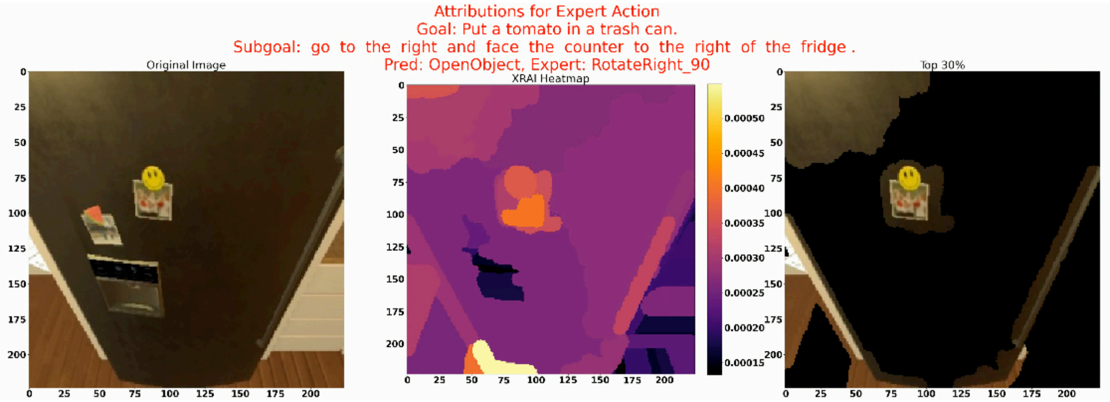


FIGURE 3.19: XRAI attributions on current image frame for expert's action: `RotateRight_90`

Figure 3.19 shows the low visual attribution scores for the action to rotate right. Interestingly, we see the lowest attribution near the handle/water dispenser of the fridge, which is an area that received high attribution for the predicted action to open the fridge.

If we had some data of where an expert's attribution should be, we could compare and supervise the model to match the groundtruth attributions. This will enable policy training to not just mimic the correct actions, but also attribute the decision to the correct reasons.

Chapter 4

Attributions for Sequential Prediction with Decision Trees

Consider a scenario where an office building gets damaged during an earthquake and has victims who need to be rescued. Victims would have varying degrees of injury, and therefore human rescuers have to develop appropriate search, navigation and victim triaging strategies so as to save the largest number of victims in limited time. The earthquake may have caused serious structural perturbations, such as blockages that limit the ability of the rescuers to reach certain areas.

In this work, we provide initial results for feature attributions of computational agent, ASIST [4] (**A**rtificial **S**ocial **I**ntelligence for **S**upporting **T**eams) to observe and predict strategies of a single person and three people rescue teams in a simulation environment. ASIST observes the environment and the behavior of each rescuer and predicts future actions of the rescuer.

Our specific scenario considers a 3D simulation of a realistic office building with several rooms and corridors where the disaster has taken place. For simplicity, victims are denoted as blocks in our environment. Some critically injured victims (denoted as yellow blocks) take more time to triage and may expire sooner than other victims (denoted as green blocks). Before starting the mission, we provide the rescuer with the building's original floorplan so that they can plan their route. During the mission, they may encounter several changes due to the damage, such as wall collapse or opening/holes in the wall. We incentivize the rescuers to prioritize the critically injured victims by rewarding them more points for saving them.

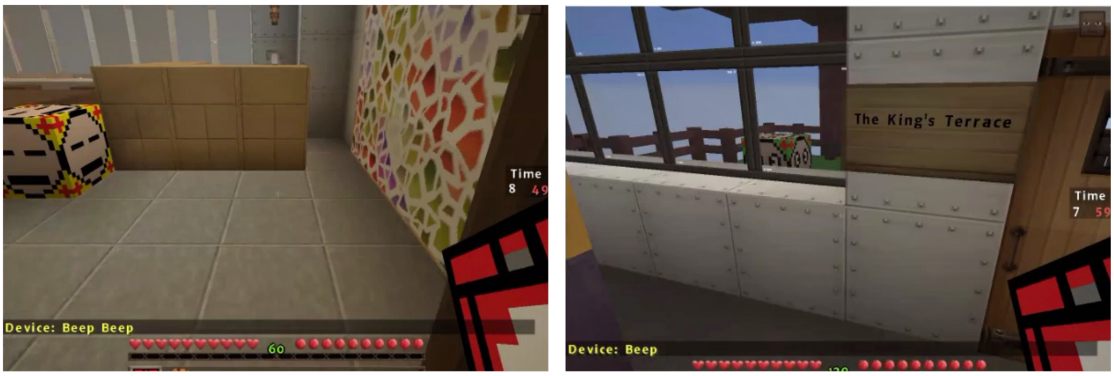


FIGURE 4.1: Victim signalling device messages near critical victim is `Beep Beep` and near less-critical victim is `Beep`

4.1 Key Contributions

We design interpretable state features from human trajectory data per timestep to predict the victim triage strategy and prior knowledge condition. We utilize TreeSHAP [34], a feature attribution method based on Shapley values optimized for family of decision tree algorithms to infer the salient features that positively or negatively contributed to the predicted outcome. The top attributed features are then processed to select the most likely natural language explanation for the prediction.

4.2 Task Design

The scenario in the Minecraft environment represents a structurally damaged office building after a disaster, with areas of building layout perturbed with collapsed rubble, wall openings, and fires. Initially, it contains 26 area segments consisting of corridors, rooms, and elevators. The current building layout and segment connectivity were changed by perturbations such as collapses, wall openings, and sporadic fires.

Both victims depend on the first responders' help to stabilize and evacuate out of the building. The rescuer's also carry a signalling device for victim proximity. It signals `beep` to indicate presence of non-critical victim and `beep beep` for the critical ones as shown in Fig 4.1. The signal messages appear on the Minecraft chat window of the human participant.

The goal of the human rescuer is to earn as many points as possible within 10 minutes.

It is known that there are 34 victims trapped in the building. The victims are of the following two types:

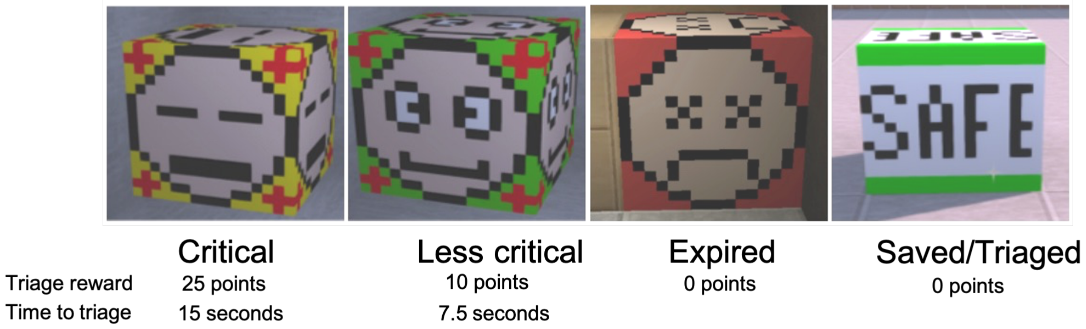


Figure 4.2: Types of victims are Critical (in yellow), Less-Critical (in green), Expired (in red) and Saved/Triaged (in white with 'SAFE') and corresponding triage tradeoffs

1. **Critical victims**: (also referred as yellow victims) 10 of them, will expire by five minutes; Each takes 15 seconds to triage and are worth 25 points.

2. **Less critical or Regular victims**: (also referred as green victims) 24 of them, will expire by ten minutes. Each takes 7.5 seconds to triage and are worth 10 points.

The victims and their possible transformed types are shown in Fig 4.2

75 trajectories were collected from human participants [22] in simulated search and rescue task setup with Minecraft game engine. Prior to each mission, human rescuers were given information on the task and original building layout to plan their exploration strategy. However, two-thirds of them were not told about meaning of the beep signal. About a third were not told about the cost-benefit tradeoffs between the critical vs non-critical victims in terms of time to triage and points rewarded to save. It was observed that the human participants learned and inferred these while solving their mission trials.

The human rescuers can have one of the three prior knowledge conditions:

1. **Triage, Signal** : full knowledge of the cost-benefit trade-offs, and understand the messages of victim proximity signal.

2. **Triage, No Signal** : knowledge of the cost-benefit trade-offs, but does not know meaning of the victim proximity signal messages

3. **No Triage, No Signal** : no knowledge of either the cost-benefit trade-offs, does not know meaning of the victim proximity signal messages

4.3 Method

We formulate the sequential prediction problems into supervised learning task as shown in Figure 4.3. Sequential data about human trajectories is hard to model and learn over

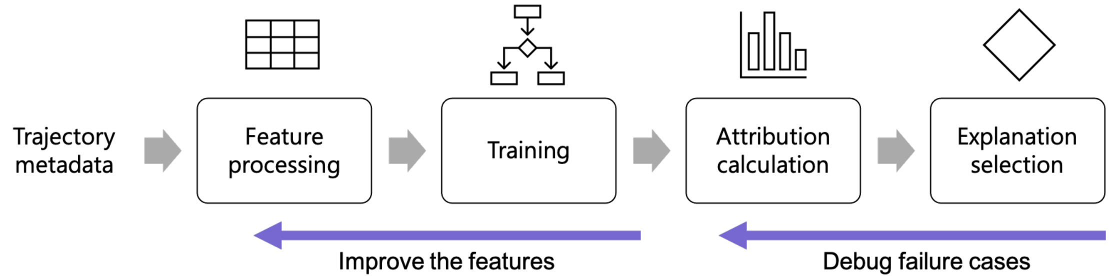


FIGURE 4.3: Overview of the process to visualize attributions and improve features for sequential prediction

with limited data. We engineer features to form a state representation at any given timestep such that Markov property is ensured. This means that the information required for any required prediction is contained within the current representation of the state. Another aspect of these features is human interpretability. Feature attribution methods explain importance of each input feature with respect to the selected output. So if a feature is given high importance by the model, it should humanly possible to explain why it is so.

We created high-level features of the trajectory data in terms of which we would like to have in our explanations. Most of these features are count-based features which is inspired from the count and hashing vectorization in the natural language processing where the count value is increased every time the word is encountered in the text corpus. Here we increase the count of the domain-specific features concerning the triage strategy and events indicating the knowledge of the rescuer. Our approach differs as we take mission time as part of the feature vector to be trained on, whereas there is no explicit notion of time required in the vector representation for language modeling.

Gradient boosted tree models have been shown to be more accurate than neural networks and more interpretable than linear models [34]. We formulate the task into classification problem and use `xgboost` library to train the models. XGBoost is an optimized gradient boosting algorithm through parallel processing, handling missing values, automated tree-pruning and other regularization to avoid overfitting.

We use TreeSHAP [34] in our study. Unlike the prior work, we construct features for the sequential prediction task such that the Markov property of the input vector at each timestep is approximately ensured for prediction. The visualization used to explain instance level predictions are called force plots [33]. Force plot show the features contributing for the prediction in red and against the prediction in blue. The sum of attributions of each feature add up to the difference in the model's output for the given input and the baseline.

We construct a list of possible sentences that describe potential reasons of a dominant positive or negative attribution for an outcome per feature. This is used to select the sentence based explanation for the dominant SHAP values for a prediction at any timestep. It provides a concise natural language way to convey the logic that the model most heavily relied upon.

4.4 Experiments and Results

Feature extraction

We construct features from human trajectory metadata to represent state at the current timestep. to sequentially predict their strategy and knowledge condition. The general mission statistic are extracted in Table A.1 which denotes the spatial-temporal features associated with the mission. Table A.3 describes some features related to triage events.

Human strategy and behavior is significantly influenced by the visual cues. Given the low bandwidth requirements to transfer human trajectory data, we do not assume access to streaming video from the first person view of the rescuer. Instead, we assume access to semantic information processed by object detectors locally. Table A.2 describes features that denote the count of salient objects in current field of view of the human rescuer.

Victim detection signals appear on the human rescuer's screen throughout the mission. Given the human rescuer's behavior, one of the objective is to predict their prior knowledge condition, including whether they understand the meaning of signal or not. Table A.4 lists the features corresponding to the signal message events. In addition, door opening events provide features in Table A.5 to learn the behavior of humans who have full knowledge of the signal's message or do not. Constructing complex feature from victim detection signal message with door opening event's in terms of their location and time provides useful features (as shown in Table A.6, especially to reason on the prior knowledge condition.

Training

We utilize all these features discussed above for training gradient boosted decision trees. The predictions are made at around 10 Hz.

For next victim triage prediction, we report the model's performance in Table 4.1. The prediction in start of the mission, that is, before 150 seconds, is biased towards critical victim triaged, due to data skew. In the next interval before critical victims expire, the accuracy improves and the confidence is more balanced. For the remaining time when

Table 4.1: Table showing results for next victim triage prediction with gradient boosted decision trees, where t denotes the mission timer.

Next Victim Triage	0<t<150		150<t<300		300<t<600	
	Acc	Conf	Acc	Conf	Acc	Conf
Critical Victim	93.4%	0.72	94.4%	0.68	-	-
Less-Critical Victim	32.8%	0.58	70.8%	0.61	99.2%	0.85

Table 4.2: Table showing results for knowledge condition prediction with gradient boosted decision trees, where t denotes the mission timer

Knowledge condition	0 <t <150		150 <t <300		300 <t <600	
	Acc	Conf	Acc	Conf	Acc	Conf
1 (TriageSignal)	41.54 %	0.17	57.23 %	0.23	67.21 %	0.34
2 (TriageNoSignal)	49.12 %	0.13	53.92 %	0.21	64.39 %	0.32
3 (NoTriageNoSignal)	53.23 %	0.14	60.47 %	0.43	69.83 %	0.44

the critical victims expire, we get almost perfect accuracy for next green victims. The model does not know the fact that critical victims expire at half time apriori and has learnt this fact from training set of trajectories.

To compare with the human observers, we evaluate our model's prediction at the same decision points as those annotated by the human observers. Our model achieves prediction accuracy of 94.8% over triage decision points over 14 trials, while the human observers are only accurate up to N%.

We report the performance on predicting prior knowledge condition in Table 4.2. The evaluation is done at three intervals: before 2.5 minutes, between 2.5 to 5 minutes and after 5 minutes. The average accuracy and confidence of predictions improve for later intervals as the model gains more information to distinguish the knowledge condition.

Human players' behavior are not always reflective of the prior knowledge that they possessed. As they navigate in the mission, they acquire knowledge about the victim triage trade-offs and the beep signal. From the observed behavior, it is hard to tell whether the player knows the triage tradeoffs or are just prioritizing the critical victims first because they expire soon. On the other hand, players with full knowledge condition have shown to not demonstrate the knowledge of triage tradeoffs. Some reasons reported in the surveys indicated intent to 'maximize the number of victims saved and not the points', and showed limited perceived control 'to find and save all the critical victims in the given time'.

To compare with the human observers, Mturkers were asked to observe the human rescuer's video and predict the next triage and knowledge condition at certain selected decision points in test set of trajectories, about 10 per mission. The model achieves accuracy of 96% for knowledge prediction and 98% for next victim triage at the sampled decision points. Human observers' achieve much perform close to random baseline on average at 56%, with top 10 observers accuracy upto 79% only.

Explanations

The purpose of explanations is to make the model more predictable to humans. This notion is defined as simulatablity and has been evaluated with human subjects in [21]. To visualize the model's feature importances, we can evaluate it on input features by black box methods like SHAP values.

4.4.1 Victim Saving Strategy

We analyze the next victim triage prediction with a random forest classifier using SHAP values. Fig 4.4 shows the overall feature importance values for the model. We find that the average feature importance values are highest for mission timer and map coverage. This aligns with the human intuition that the player will generally triage yellow victims initially before they expire. To analyze decision at each timestep, we explain it with two examples.

Fig 4.5 and 4.6 show the force plots explaining the next victim type to be triaged prediction at two given time instances. The direction indicating higher refers to increase in likelihood of the next victim triaged to be green and the lower direction indicates that the next victim to be yellow. The base value is 0.6182 which is the average prediction probability if no features were present - which is aligned to the proportion of the number of green victims in triaged in the training data.

More detailed list of model's features, and associated importance value are discussed in Appendix A.1.1.

We provide explain the model's predictions in terms of input features and their values as shown below.

Each model feature can contribute positively or negatively to an output prediction. For example, `'mission_timer'` is an input feature whose value can positively or negatively contribute to the prediction whether the next victim triaged will be green?

We prepare a list of expressions to explain how a feature positively or negatively contributes to this prediction. For an instance at test time, our model makes a prediction with the given input features and TreeSHAP computes the feature importance values associated with each feature. Depending on the feature importance values, we select the expressions to explain the model's predictions in terms of the input features which

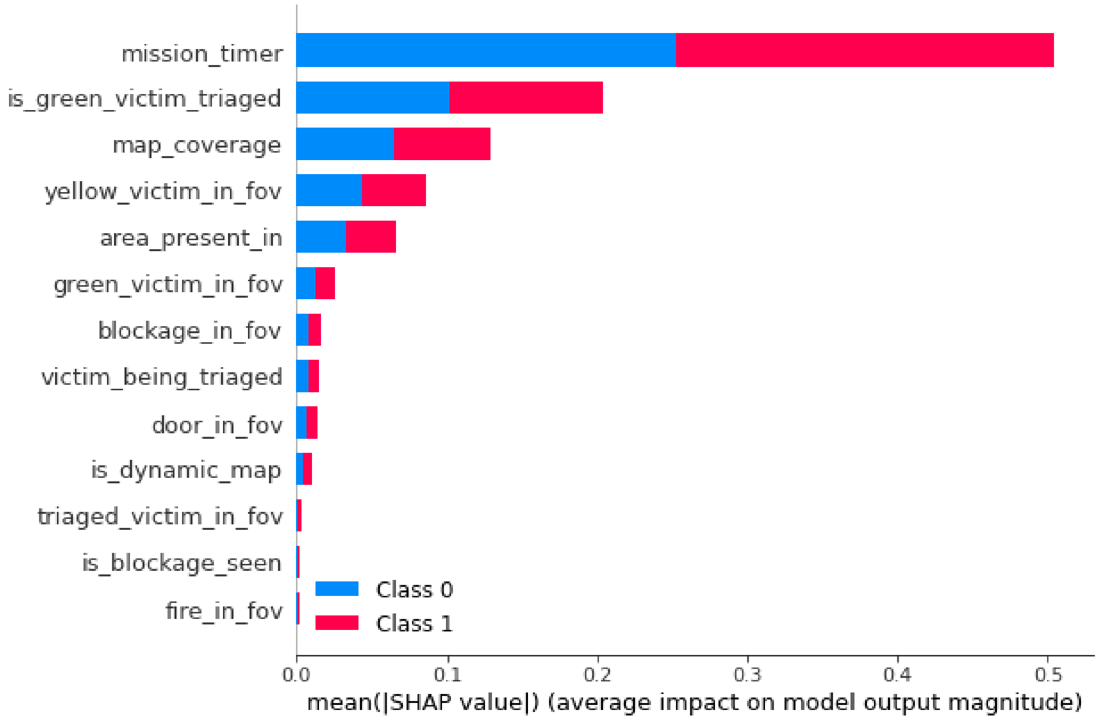


Figure 4.4: Mean feature importances by SHAP values for prediction **next victim triaged type**; where class 0 indicates the prediction for critical (yellow) triaged next, and class 1 indicates the prediction for less critical (green) victim triaged next

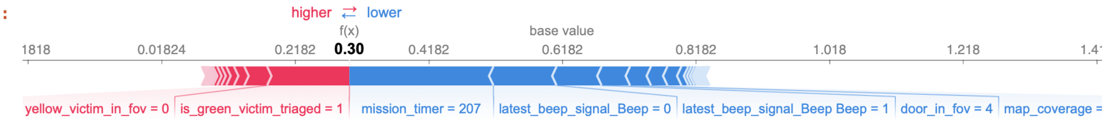


Figure 4.5: SHAP force plot showing that probability of next victim triage is green only 0.3 which makes it likely that the next victim triaged is **not green** i.e. yellow. Features such as mission timer approaching yellow victim's death, having the latest signal as 'Beep Beep' and not 'Beep', having doors in field of view decrease the likelihood that a green victim will be triaged next (implying that the rescuer will triage a yellow victim). But since the rescuer has triaged a green victim before and there is no yellow victim in field of view, these features slightly increase the likelihood that a green victim might be triaged next.

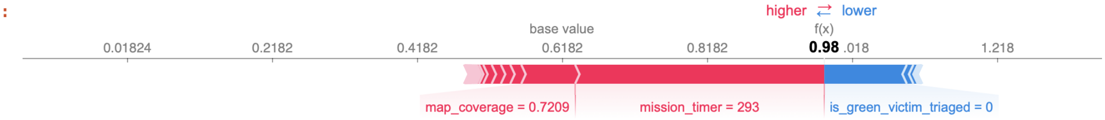
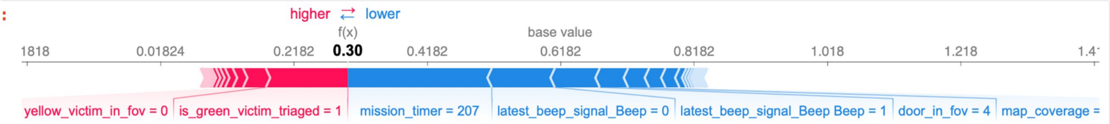


Figure 4.6: SHAP force plot showing that next victim triage is **green** is with probability 0.98; Features such as the mission timer close to the yellow victims death, and that the map coverage is high greatly influence the model's output for green victim next; But as the rescuer has not triaged any green victim uptil this point, this feature decreases the probability of that the next victim to be triaged is green



Evidence	SHAP value
Mission_timer = 207	-0.23123
Explanation: "As the yellow victims will expire soon, the player's intent is to triage as many yellow victims as possible."	
Is_green_victim_triaged = True	+0.123012
Explanation: "The player has already triaged green victims, so player could triage whoever comes first".	
Latest_beep_signal_Beep= False	-0.110957
Explanation: "No 'Beep' signal received lately, so it is unlikely that the player will find a green victim nearby".	
Latest_beep_signal_BeepBeep = True	-0.097816
Explanation: "Received a 'Beep Beep' lately. Player's belief changes to expect yellow victim will be found nearby and triaged."	
Yellow_victim_in_fov = False	+0.05498
Explanation: "No yellow victim in field of view yet. Player does not perceive yellow victim, hence slightly unlikely that the yellow victim will be triaged next".	

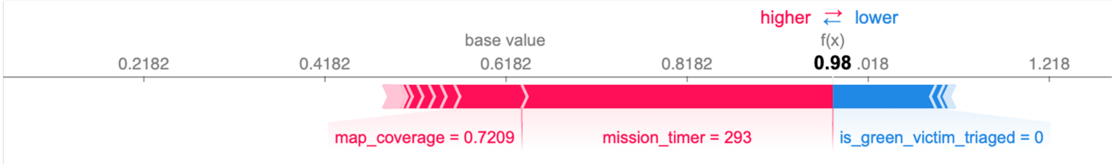
FIGURE 4.7: Explaining when hypothesis: next green victim triage is not supported

are based on the observations of the human's trajectory so far. In Fig 4.7, SHAP force plot shows that probability of next victim triage is green only 0.3 which makes it likely that the next victim triaged is **not green** i.e. yellow. Features such as mission timer approaching yellow victim's death, having the latest signal as 'Beep Beep' and not 'Beep', having doors in field of view decrease the likelihood that a green victim will be triaged next (implying that the rescuer will triage a yellow victim). But since the rescuer has triaged a green victim before and there is no yellow victim in field of view, these features slightly increase the likelihood that a green victim might be triaged next.

In Fig 4.8, SHAP force plot shows that next victim triage is **green** is with probability 0.98; Features such as the mission timer close to the yellow victims death, and that the map coverage is high greatly influence the model's output for green victim next; But as the rescuer has not triaged any green victim uptil this point, this feature decreases the probability of that the next victim to be triaged is green.

The beeswarm summary plot shown in Fig 4.9 indicates how the values of each feature affects the prediction. High values for mission timer indicate triaging of green victims as it is perceived that finding yellow victims will be hard close to the time they are expected to expire. If the human rescuer has saved few green victims saved per doors opened, they probably understand the beep signal and do not open doors unnecessarily. So, lower value of the `less_critical_saved_per_doors_opened` feature is attributed to triaging critical victim first, and vice versa.

`Critical_saved_per_doors_opened` is similarly important feature which did not get clear SHAP attribution in this model. As a variation, we trained another model for triage prediction, with all the features except the `mission_timer`.



Evidence	SHAP Value
Mission_timer = 293	+0.447922
Explanation: "As the yellow victims will expire soon, the player believes that he/she will not be able to triage them in time. So, based on this belief, it is likely that player will triage green victim next."	
Is_green_victim_triaged= False	-0.203012
Explanation: "The player has triaged no green victims yet, so could be prioritizing yellow first".	
Map_coverage = 0.72	+0.1700957
Explanation: "The player has covered significant portion of the map, so she/he believes it is unlikely to find more yellow victims. Based on this belief player is likely to triage green victims next."	

FIGURE 4.8: Explaining when hypothesis: next green victim triage is supported

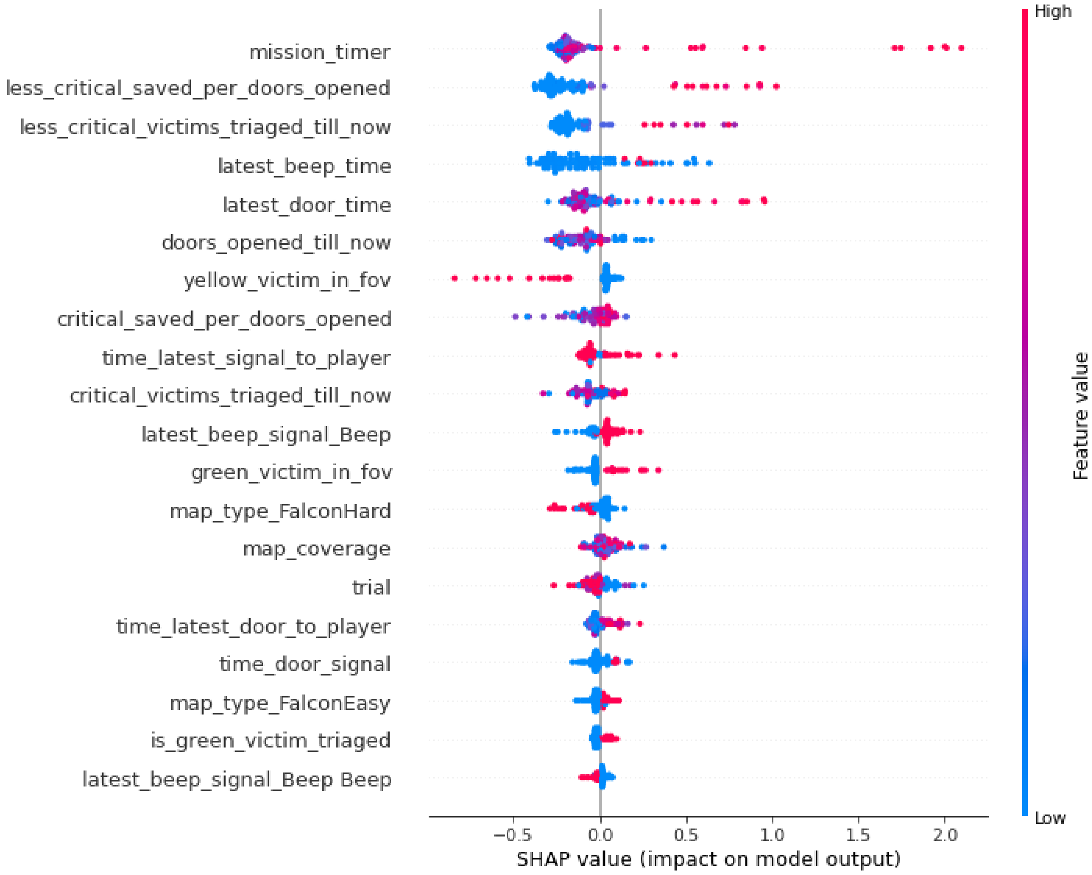


FIGURE 4.9: SHAP beeswarm summary plot for next triage with mission timer as most attributed feature for prediction

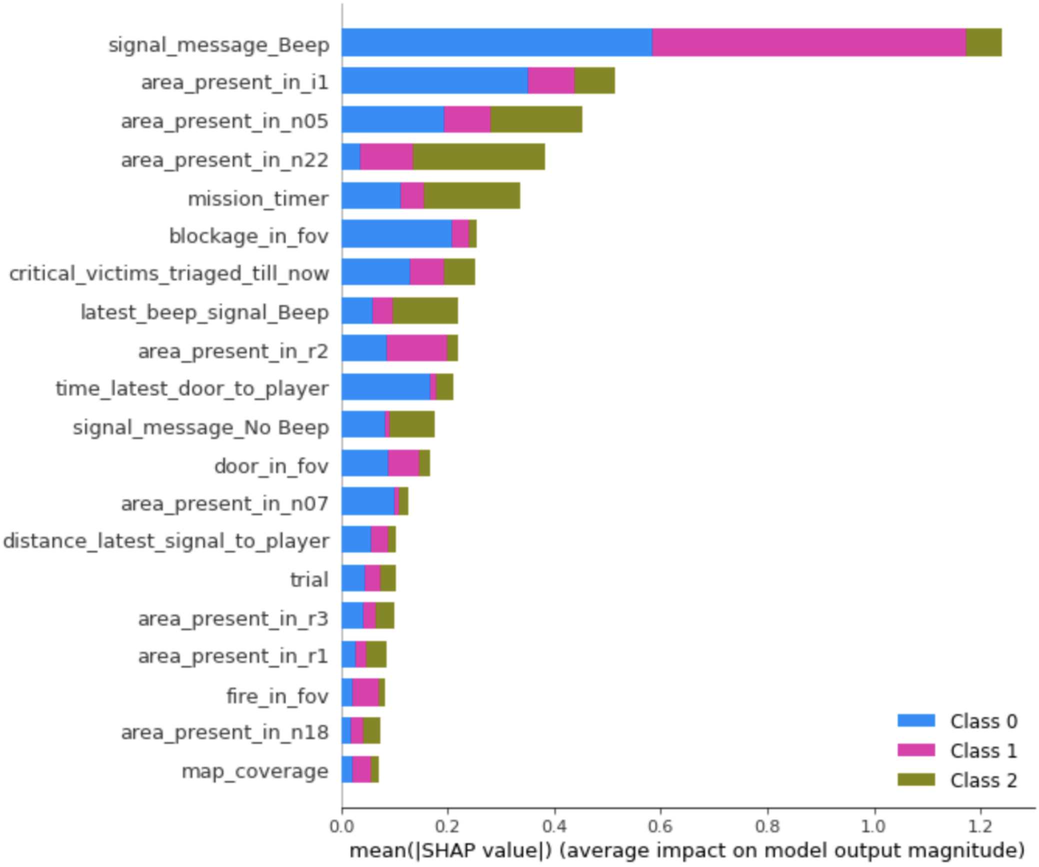


FIGURE 4.10: Mean SHAP values for **Knowledge condition** prediction; where class 0 refers to TriageSignal, class 1 refers to TriageNoSignal, and class 2 refers to NoTriageNoSignal

4.4.2 Knowledge Condition

We train two xgboost models for knowledge condition prediction, one with a set of indicator features and another with a set of count-based features.

Fig 4.10 shows that the overall feature importance values for the gradient boosted decision tree model for knowledge condition prediction. We observe that the feature indicating 'Beep' signal has been received is the given the most importance. This is because the knowledge condition can be distinguished based on the human's actions when a beep signal is received.

The area segment that the human rescuer is present in is also important indication in conjunction with features like mission timer. This is because typically a person in full knowledge condition will skip the rooms where there is no beep or a single beep signal received, and would reach certain areas at earlier (mission timer) than those who do not have the knowledge of the beep signal.

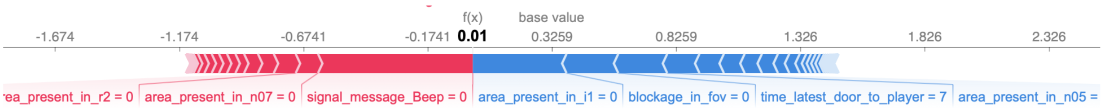


FIGURE 4.11: Force plot for **output**: *TriageSignal*, **true**: *NoTriageNoSignal*; The model's output logit is **0.01**, which is lower than baseline at **0.33**. The higher direction indicates having full knowledge and the lower direction indicates the other two knowledge conditions. The features supporting the likelihood of full knowledge condition are non-intuitive such as no signal message `Beep`. The factors such as the area segment that the rescuer is present in (as compared to those in training instances) are considered important on both sides to determine model's output for full knowledge hypothesis.

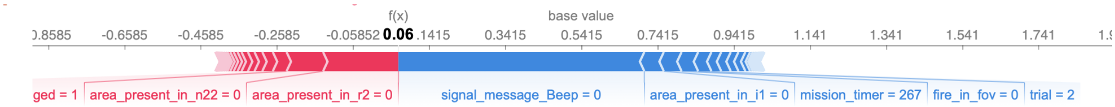


FIGURE 4.12: Force plot for **output**: *TriageNoSignal*, **true**: *NoTriageNoSignal*; The model's output logit is **0.06**, which is lower than baseline at **0.54**. The higher direction indicates having partial knowledge of triage tradeoffs but not of the signaling device, and the lower direction indicates the other two knowledge conditions. The features increasing the likelihood of partial knowledge condition arenon-intuitive such as no signal message `Beep`. The factors such as the area segment that the rescuer is present in (as compared to those in training instances) are considered important on both sides to determine model's output. Mission timer has slight attribution to decrease the likelihood of partial knowledge hypothesis.

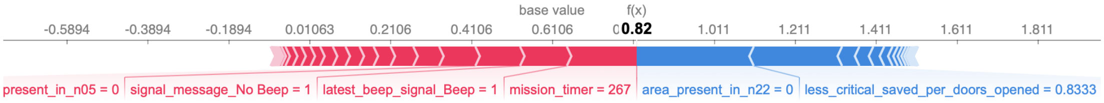
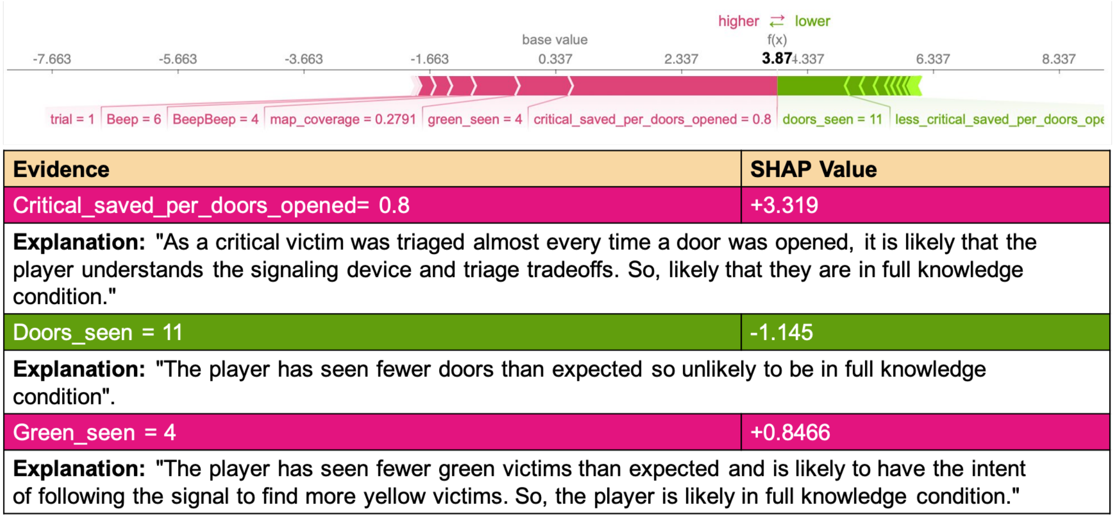


FIGURE 4.13: Force plot showing model's probability and feature importance for **output**: *NoTriageNoSignal*; **true**: *NoTriageNoSignal*; The model's output logit is **0.82**, which way above the baseline at **0.61**. The higher direction indicates having no knowledge and the lower direction indicates the other two knowledge conditions. The features increasing the likelihood of no knowledge condition are `mission timer` being close to the yellow victims death and that latest signal received was a `Beep`. The factors such as the area segment that the rescuer is present in (as compared to those in training instances) and the ratio of less critical (green) victims saved per number of doors opened contribute to slightly decrease the likelihood of being in no knowledge condition.

To explain the predictions at timestep 267 seconds into the mission, Fig 4.11, 4.12 and 4.13 show the SHAP force plots for the knowledge condition prediction on one instance where the rescuer is in knowledge condition 3 i.e. NoTriageNoSignal condition. The base value indicates the value that would be predicted if we did not know any features for the output prediction. The base value and $f(x)$ are expressed in logits, as the model's output in XGBoost is log-odds by default.



Evidence	SHAP Value
Critical_saved_per_doors_opened= 0.8	+3.319
Explanation: "As a critical victim was triaged almost every time a door was opened, it is likely that the player understands the signaling device and triage tradeoffs. So, likely that they are in full knowledge condition."	
Doors_seen = 11	-1.145
Explanation: "The player has seen fewer doors than expected so unlikely to be in full knowledge condition".	
Green_seen = 4	+0.8466
Explanation: "The player has seen fewer green victims than expected and is likely to have the intent of following the signal to find more yellow victims. So, the player is likely in full knowledge condition."	

Figure 4.14: Explaining the Full Knowledge condition hypothesis

Table 4.3: Results of knowledge condition prediction with new concise set of features

Mission timer (t)	0 <t <150		150 <= t <300		300 <= t <600	
Knowledge Condition	Acc	Conf	Acc	Conf	Acc	Conf
1 (TriageSignal)	51.54	0.45	70.98	0.70	**76.93**	0.86
2 (TriageNoSignal)	54.27	0.34	**58.22**	0.45	54.89	0.59
3 (NoTriageNoSignal)	31.79	0.46	31.66	0.65	**43.77**	0.63

Iterative Model Improvement

After analyzing the model performance and feature importance, we iterated over the features and created a new subset as described in Table A.7. The results of gradient boosted decision trees trained with these features is shown in Table 4.3. The accuracy and the confidence of prediction generally improves with more evidence of the trajectory's observations as the mission timer increases. We observe that the new model on fewer features is more confident on predicting full knowledge condition than the other two.

Feature importances based on this model with new concise set of features as input are shown in Figure 4.15. We see that more number of features are attributed importance in this set, notably the number of green and yellow victims seen is attributed importance in predicting the human rescuer's prior knowledge condition.

Also, the signal counts of `Beep Beep` is a prominent distinguishing feature when combined with mission timer value. If the human rescuer have no knowledge of the signalling messages, it is unlikely that they will hear a lot of signal for critical victim nearby, as compared to one who understands and leverages the signal for exploration.

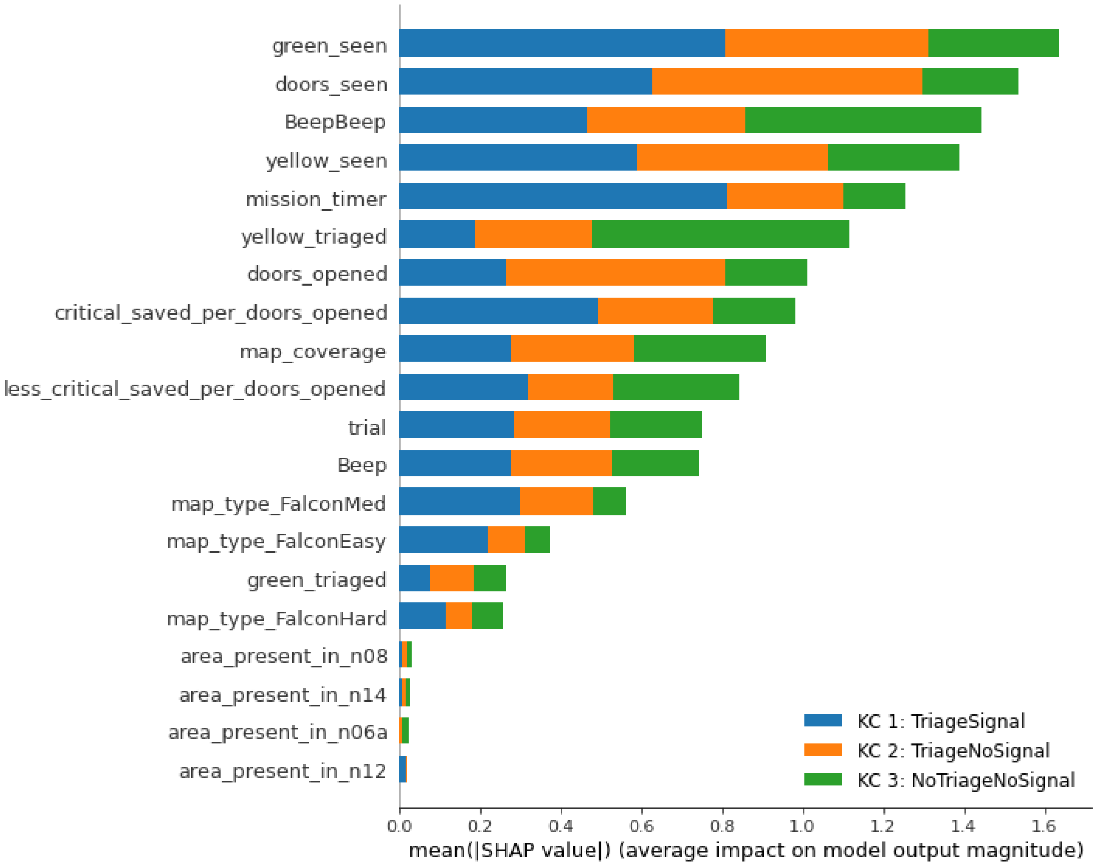


FIGURE 4.15: Mean SHAP values for **Knowledge condition** prediction with gradient boosted decision trees with temporal count vectors as inputs;

Chapter 5

Embeddings for Spatial Semantics

Language provides a way for humans interpret the world and explain abstract concepts. Incorporating language into the algorithm design for navigation provides explainability. In this chapter, we discuss algorithms for object search in urban house environments based on language and spatial priors. In the following chapter, we discuss modular architecture for visual language navigation of a robot during search and rescue task.

Consider an example of finding a key-chain in a living room. A key-chain is found either on a coffee table, inside a drawer, or on a side-table. When tasked with finding the key-chain, a human would first scan the area coarsely and then navigate to likely locations where a key-chain could be found, for example, a coffee table. On getting closer, they would then examine the area (top of the table) closely. Following this, if the key-chain is not found, they would try to navigate to the next closest place where the key-chain could be found. In all these scenarios, the presence (or absence) of a co-located objects would boost (or dampen) their confidence in finding the object along the chosen trajectory. This would, in turn, influence the next step (action) that they would take.

Our goal is to enable embodied AI agents to navigate based on such object-based spatial semantic awareness. To do this, we focus on the following problems: (a) training object embeddings that semantically represent spatial proximity, and (b) evaluating these embeddings on semantic search and navigation tasks. We aim to learn embeddings that capture the following - (a) learn about larger objects around which the smaller items could be found (e.g., key-chains are likely to be found on / near tables), and (b) learn about objects that are found mutually close to each other, e.g., (key-chains and credit-card). By learning such embeddings, we want to capture the semantic relations in terms of distance.

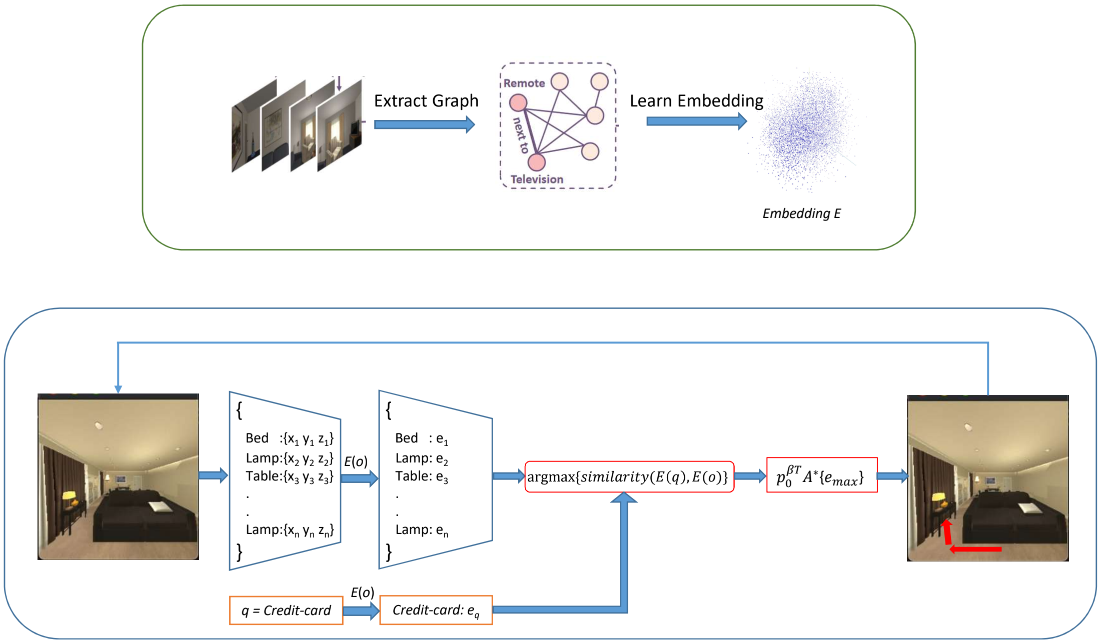


FIGURE 5.1: **Top**: Training the embeddings, **Bottom**: Using embeddings for navigation

5.1 Key Contributions

We learn and contrast representations to capture spatial semantics. We demonstrate that a momentum inspired, similarity-based greedy navigation technique results in success rate $> 90\%$. We show that using spatial semantic prior knowledge can significantly improve the navigation performance.

We train embeddings that capture the spatial semantics using a multi-relational knowledge graph. Further, we formulate an algorithm to compare the performance in terms of *Success Rate* (SR) and *Success weighted by Path Length* (SPL) across different kinds of embeddings. Interestingly, embeddings trained with general-purpose text corpora like Word2Vec [37], FastText [6] provide a strong baseline for our task. We demonstrate our techniques on the AI2Thor [18] environment.

Graph convolutional networks have been used to learn neural-network based policy in [54] and [60] for object search and navigation. These are data-driven approaches and hence, require several interactions in the environment to learn the policy. We propose a sample-efficient approach based on pre-trained embedding in section 5.3. While previous approaches have attempted to learn an end-to-end navigation policy from scratch, we demonstrate the promise of using embeddings for navigation, and their transfer to object search in 3D simulations of urban household.

Algorithm 2: Navigation towards query for the objects in the field of view

```
Result: Plan to the query object
embedding module E;
visited = [];
while query q not found do
    for all objects o in (360 degree) field of view do
        if object o not in visited then
            scores = similarity($E(q), E(o)$);
            Add $o$ to potential_sub_goals;
        end
    end
    sub-goal =potential_sub_goals[ scores];
    proposed plan $p_0^T$ := $A^*$ search to sub-goal;
    move according to initial part of the proposed plan $p_0^{\beta T}$ where $\beta < 1$;
end
```

5.2 Task Design

Our goal is to navigate an agent from a randomly initialized location in a scene to a specified query object. The task is a success if the agent can view the object of interest, and is within a small distance (~ 1 units $= 5 \times 0.25$) of the query object.

Consider the agent is tasked to find a *credit-card* (query). The agent is initialized randomly in a living room and records all the objects in its 360 field of view. By design, we enforce that the agent can only see "large" objects. Imagine you enter a living room and see a sofa, table, TV, etc. easily. However, depending on your distance, objects such as the credit card might be occluded or not be visible. We consider large objects that occupy more than 1% of the total screen size of the camera's image. This threshold is analogous to a real camera's effects as detection of objects in the frame would depend on the camera's resolution.

The agent now finds a sub-goal. The sub-goal is the most likely (large) object in its field of view, which we expect is close to the query object, or could lead the agent towards a viewpoint that brings relevant sub-goals in the agent's field of view. This is done by finding the pairwise similarity of the query object's embedding with the embeddings of all the sufficiently large objects in sight. For example, out of visible objects like table, lamp, and bed, we would expect embeddings of a table to have the highest similarity with the embeddings of a credit-card.

Once we choose a sub-goal, we calculate the potential plan $p_0^{T_i}$ where T_i is the total steps required to reach the sub-goal i. The agent executes $p_0^{\beta T}$ that is some fraction of the initial part of the potential plan. The agent then looks for any new object that is now visible and re-ranks the similarity of all objects visible to it to find the new sub-goal. This process repeats until the query object is found. Consider that in a 1D space, the agent's location is at 0, and the query object's location is at 10. With $\beta = 0.5$, the agent would first navigate to point 5, then to 7, and finally to 9 before reaching 10. This approach has two advantages: a) improves the performance of the system by exponentially reducing the amount of compute per navigation (scan, similarity computation, A^* path computation). b) helps overcome oscillation. Oscillations in navigation happen when the agent takes a step towards sub-goal A, and finds B to be having a higher similarity. On shifting the sub-goal, and taking a step towards B, B becomes occluded, and the sub-goal rotates back to A (Appendix A.2.2)

5.3 Method

Our novel approach of finding the semantic similarity between the given query and visible objects is formulated based on the distribution of distance between a pair of objects. Additionally, we identified two broad kinds of embeddings for our analysis, as discussed below. Each of these embeddings are evaluated for navigation as outlined in Algorithm 2.

Pre-trained word embeddings Language embeddings capture the word-level semantics in their metric space. For example, FastText embeddings have shown promising results for semantic navigation in procedurally generated environments [54]. To understand if these embeddings can be transferred in terms of object semantics for the downstream task of navigation , we test our algorithm on the Word2Vec [37] and FastText [6] embeddings.

Knowledge base embeddings Multi-relational embeddings encode abstract knowledge that could be obtained by the agent from its sensors or an external knowledge graph. RoboCSE [16] uses ANALOGY[32], a semantic matching method, to learn multi-relational embeddings processed from the AI2Thor environment data, where the nodes represent the objects and edges denote the relation between them. Further, we also learn a Graph Embedding by treating all relations as being equivalent and using DeepWalk [40] to learn embedding for the resulting undirected graph.

5.4 Experiments and Results

We use the AI2Thor environment to extract pairwise object relations to train the embedding network. We make two interesting design choices that we think will be valuable to the community. First, since we assume perfect object detection if the object is sufficiently in the field of view, we determine sufficiency as a parameter with respect to the percentage of pixels occupied by the object in the field of view. Second, we adaptively

			SPL by shortest path length l		
Room Type	Embedding Type	SR/SPL	l¡ 10	10 ¡l¡ 20	l ¿= 20
Kitchen	Graph Embedding	**0.992 / 0.639**	0.644	0.642	0.644
	RoboCSE	0.960 / 0.624	**0.650**	0.643	**0.867**
	FastText	0.983 / 0.615	0.626	0.624	0.573
	Word2Vec	0.984 / 0.626	0.633	**0.649**	0.581
Living Room	Graph Embedding	0.919 / 0.692	0.777	0.698	**0.693**
	RoboCSE	**0.942 / 0.686**	0.766	0.692	0.614
	FastText	0.908 / 0.682	0.774	**0.727**	0.619
	Word2Vec	0.908 / 0.666	**0.793**	0.708	0.596
Bed Room	Graph Embedding	0.954 / 0.678	**0.739**	0.631	0.659
	RoboCSE	**0.966 / 0.681**	0.731	0.628	**0.690**
	FastText	0.960 / 0.686	0.738	**0.657**	0.544
	Word2Vec	0.956 / 0.662	0.720	0.624	0.576
Bath Room	Graph Embedding	0.997 / 0.694	0.692	0.733	-
	RoboCSE	0.994 / 0.692	0.693	0.716	
	FastText	0.994 / 0.688	0.692	0.690	
	Word2Vec	**0.998 / 0.701**	**0.697**	**0.743**	

Table 5.1: **Results for navigation using Algorithm 2.** Success Rate (SR) and Success weighted normalized inverse path length (SPL) for different floor plans in AI2thor environment; l denotes the shortest path length to query object.

choose step-sizes to navigate towards the sub-goal. This allows the agent efficiently to look for other objects in the field of view on the way. For more information refer Appendix A.2.2.

In Table 5.1, we show the performance of different embeddings for navigation task in terms of Success weighted by normalized inverse path length (SPL) metric. Further, we also report SPL computed by grouping cases where the shortest path length to the query object is within a particular range. For example, the credit card may be initialized (randomly) 15m away from the agent, including it in the $10 < l < 20$ bucket.

We observe that the **graph-based embeddings (RoboCSE and Graph Embeddings) outperform the baselines in cases where the optimal path to the target object is higher** $(10 < l < 20, l >= 20)$ but perform comparably to language-based embeddings (Word2Vec, FastText) when optimal paths are shorter $(l < 10)$. Our hypothesis is that, for a query object close to the agent's initialisation $(l < 10)$, any action that the agent takes has a higher probability of taking it closer to the query object. However, for objects that are further away, the agent benefits from determining the most likely path to get to the query object.

Chapter 6

Language to Goal Location

Intelligent agents can assist us in environments where human navigation is risky or infeasible. For example, in hazardous situations such as poisonous gas or nuclear radiation leak, autonomous agents can assist in scouting for damages and search for possible victims. For effective collaboration with humans, we propose a modular system design for autonomous agents that processes raw sensory inputs like vision and language messages to take high-level decisions.

We describe a simulated search and rescue task in Minecraft, where the agent's goal is to aptly respond to natural language messages inquiring about the surroundings or instructing it to navigate to a target in the environment. These messages mimic a remotely situated human during the mission who needs to gain situational awareness of the disaster site and set the target for the agent to act. Our modular architecture enables the AI agent to parse a human's natural language message and either (1) generate a natural language response for the query or (2) interact in the environment to execute the instruction/command.

While we could look into tele-operating the agent for search and rescue missions, this has implications on network bandwidth requirements. The entire visual feed needs to be received by the human in order to tele-operate the agent and lag in the frames received could led to potentially fatal actions. On the other hand, developing ways for the agent to learn a policy for its actions would reduce the response latency and enable learning from demonstrations and mistakes. This also necessitates the agent to understand the vision and language aspects for its decision making process.

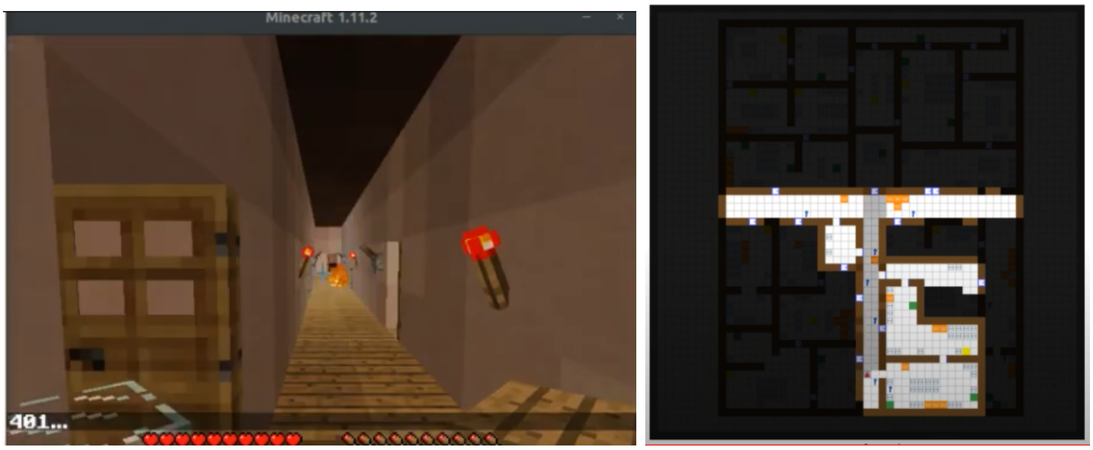


FIGURE 6.1: **Left:** First person view of the agent in Minecraft, **Right:** Semantic 2D map for bird's eye view to provide context to the human commander

6.1 Key Contributions

This work proposes a learning driven system for collaboration between an autonomous agent in search and rescue mission and a human located in a safe remote environment. Just like the human first-person responders would communicate with their teammates situated remotely, we build a system that enables an autonomous agent to parse a human's natural language message and either (1) generate a natural language response for the query, or (2) interact in the environment to execute the instruction/command. In this chapter, we focus on the qualitative analysis to demonstrate the capability of the latter module across four simulated disaster scenarios.

6.2 Task Design

As a first step, we design a system for an autonomous agent to collaborate with humans to solve simple simulated search and rescue task in Minecraft. Our specific scenario considers a 3D simulation of a realistic office building with several rooms and corridors where the disaster has taken place. For simplicity, victims are denoted as blocks in our environment. We consider key objects such as shovel, rubble, extinguisher, fire, first-aid kit and obstacles. A human may pose questions, requests or commands to the agent such as 'Do you see any fire in the corridor?' or 'Save the victims before extinguishing the fire in the room'. Example of such dialogue conversation is shown in Appendix A.1.2.

We adopted a synthetic data generation process to generate data for training the learning based modules and to evaluate them. This is inspired from prior works such as Emobodied QA [56] and DAQUAR [35]. We utilize hand crafted templates to generate questions that a human might ask while interacting with the system for the purpose of navigation and scene description. Some examples of the types of templates used for generating questions and answers are show in Tables 6.1.

TABLE 6.1: Instruction templates for navigation

search:	’Search for a _’
search + color:	’Find a *#color* _’
search + history:	’Go back to the previous _’
search + relation:	’Find a _ near a _’

6.3 Method

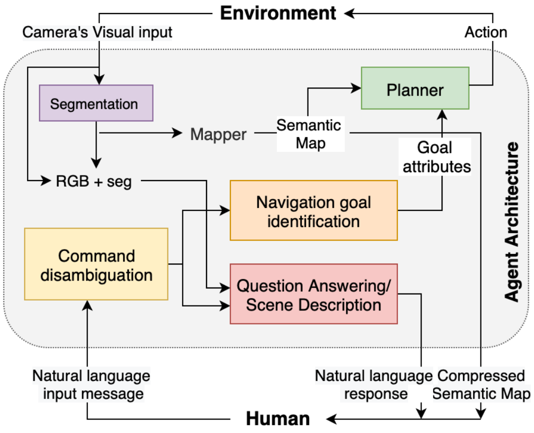


FIGURE 6.2: Proposed architecture for visual dialogue system

Fig 6.2 provides an overview of the agent, with the highlighted modules serve a specific role in enabling the agent to process the natural language query input by a remote user, understand visual input from the robot’s environment and respond appropriately by either generating a response in natural language or navigating to search for the requested object.

If the human’s instruction requires the agent to navigate, the agent needs to infer the intended goal location from the natural language message, map its surrounding and plan to reach the goal. To construct any plan, algorithms such as Djisktra’s or A^* search require a world model with the agent’s and the target’s location. Natural language messages may contain information about the object type, its properties such color, any indicates time or spatial relation to the agent. As shown in Fig 6.3, navigation goal identification module consists of a fine-tuned DistilBERT [42] model that infers these goal attributes and provide it to the planner submodule.

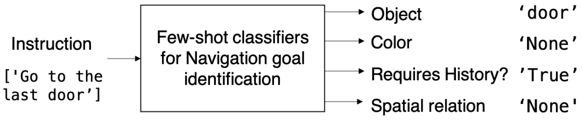


FIGURE 6.3: Navigation Goal Identification Module

To search and navigate in an unknown environment, the exact coordinates of these location are often unknown. Mapping the unknown environment is necessary to plan for navigation. Several approaches for object-centric simultaneous localization and mapping (SLAM) have been proposed [59, 7], which can

TABLE 6.2: Comparing accuracy for predicting goal attributes

Category	Acc for Zero-shot	Acc for Fine-tuned
Object	76%	**97%**
Color	22%	**95%**
Requires History?	81%	**93%**
Spatial Relation	40%	**94%**

TABLE 6.3: Evaluating SR and SPL of navigation module on 4 maps

Map Name	Valid tests	SR	SPL
argon	28/35	75.86%	0.6463
neon	33/35	78.79%	0.6902
xenon	34/35	79.41%	0.6287
ozone	32/35	78.13%	0.6674

utilized with our system for real-world transfer. In our scope, we utilize the Minecraft simulator to mimic such systems and provide field-of-view based maps incrementally.

Most path planning algorithms require some discretization of the environment to plan. Given that the mapper to provides a voxel-grid semantic representation of the environment, we utilize classical planning algorithms such as Dijkstra's or A^* search [20]. The goal location is determined based on the *not-yet-visited*, *closest* location to the agent that matches the identified goal attributes. If no such location exists in the visible map, then one of the nearest *frontiers* [58] is selected as the goal. After a threshold number of steps, the status (success/failure) is reported back to the human commander.

6.4 Experiments and Results

To understand the goal attributes in a navigation command, we fine-tune DistilBERT model on 200 examples to predict the goal object or frontier, color, time relation (i.e. whether it requires history or not), and spatial relation (left, right, in front of or back). We compared the performance of model to zero-shot classification pipeline based on BART-NLI. Table 6.2 shows the accuracy and the class-wise proportion on 100 held-out samples of navigation instructions. Our fine-tuned models achieve accuracy $> 90\%$ for predicting each of the four goal describing attributes.

Once the goal attributes are estimated from the natural language message, it is used to estimate the goal location for planning. If the unvisited goal location is present in the observed semantic map, the agent navigates to those coordinates. Otherwise, nearest frontier coordinate is selected. Our navigation strategy achieves $78.15 \pm 1.9\%$ average success rate (SR). When compared with an expert with full observability of the map, we find that 0.6578 ± 0.28 average success weighted by (normalized inverse) path length

(SPL)[1]. These results are calculated based on independent experiments on 4 different maps of size 100 × 50 with 35 navigation instructions. Refer Table 6.3.

[1] SPL metric requires to compare the agent's path length to that of the expert. While the expert planner always knows the nearest goal location from the groundtruth attributes of the navigation command, there could be multiple possible paths for our agent with partial observability. Due to the sequential nature of these navigation commands, the agent often starts with a different location and history. The expert and our agent's position could be very different with time. For fair comparison, we reinitialize the expert planner to start with the observed state and visited goals as our agent at each instruction. A navigation test is considered valid is the expert is able to reach the goal location from a random starting in a threshold number of steps.

Chapter 7

Explainable Exploration

Learning end-to-end policy for navigation with a focus on intelligent exploration has been found to be a challenging task in embodied AI. While methods like soft-Q learning and ensembles of policies have demonstrated navigation behaviors in completely observed maps, we currently do not have ways of extending these policies to unexplored or partially explored environments. To this end, we propose a modular hierarchical formulation by decomposing the navigation task into two sub-problems: selecting the next best goal in the visible space, followed by efficiently navigating to this space in the partial map setting.

When navigating in unknown environments, humans often invoke a decision criteria. This could be the next sub-goal, and the decision could either be based on some semantic context, or structural priors. Learning how to select the next sub-goal for higher level task such as coverage of the map is a challenging open question. We consider frontiers, which are the points at the boundary of known regions of a map, as points to facilitate exploration in a map. Therefore, we define such frontiers as sub-goals, and learn a policy that can select such frontiers on partial grid maps.

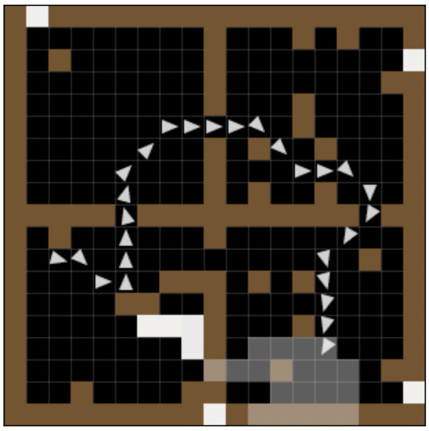


FIGURE 7.1: FourRooms environment with clutter and 2D human-like field of view (in translucent grey)

The concept of frontiers for exploration of unknown spaces was first introduced in [58] by Yamauchi. Related work in embodied AI literature has focused on the object and instance based navigation in 3D world environments. Our work is very close to [8] which demonstrated such a hierarchical policy for point- goal navigation tasks in 3D environments. Our approach differs in terms of being a study on performance of reinforcement learning

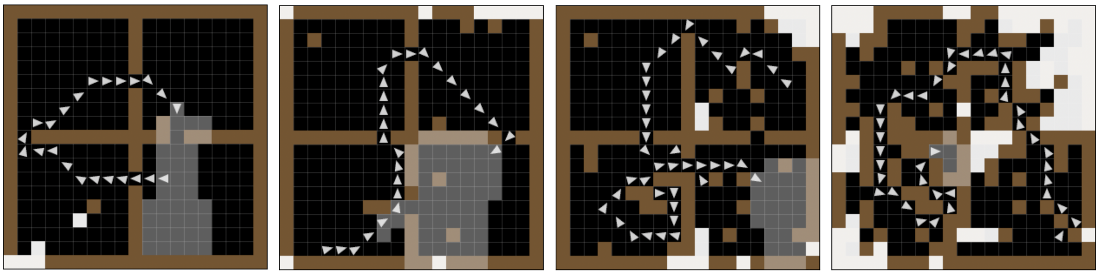


Figure 7.2: Clutter density γ in FourRooms environment (a) $\gamma = 0.01$, (b) $\gamma = 0.03$, (c) $\gamma = 0.1$, (d) $\gamma = 0.3$

algorithms to predict frontiers and contrast classical vs learning approaches for navigation with systematically varying the clutter density in the environment.

7.1 Key Contributions

We demonstrate our approach in customized version of the classic FourRooms environment in gym-minigrid [13], with a realistic human-like field of view (shown in grey area in Fig 7.1), and variation in clutter density γ for evaluating the navigation policies. We observe that as the clutter density increases, the performance of learning based agent improves as compared to the heuristic of nearest frontier based exploration.

7.2 Task Design

Gym-minigrid FourRooms environment has been a classic task for reinforcement learning with full observability. We modified the current environment to include tunable parameter γ to control the clutter density in the environment. Clutter density is the probability of grid cell being impassable, in addition to the walls of the four rooms. In Figure 7.2, the variations of different parameters for clutter density in the Fourrooms environment are shown. Note that the increasing clutter density hinders the vision of the agent. The agent's trajectory is shown with white triangles. These snapshots are recorded for NFBE with A^* planner at 100 steps.

7.3 Method

We propose a global policy network that learns the selection of the next best goal('frontier') in the observable space, followed by a local policy that learns low-level navigation conditioned on different sub-goal embeddings in known environments, as shown in Fig 7.3. Our local policy can be considered as a short term goal driven planner but trained

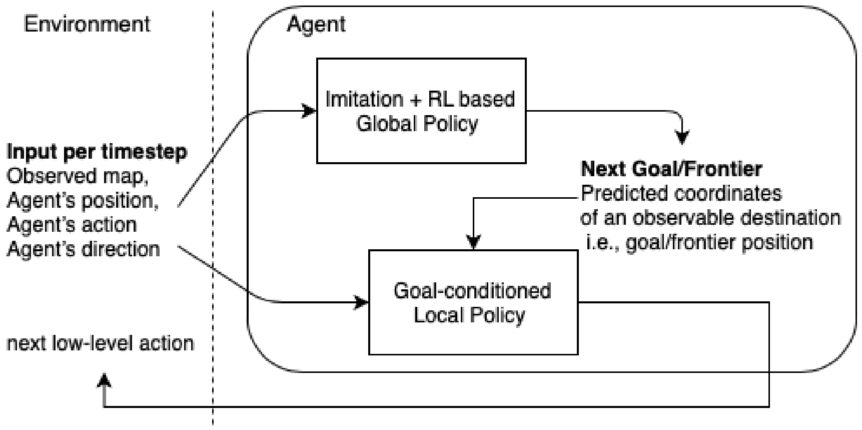

Environment
Agent
Imitation + RL based Global Policy
Input per timestep
Observed map,
Agent's position,
Agent's action
Agent's direction
Next Goal/Frontier
Predicted coordinates of an observable destination i.e., goal/frontier position
Goal-conditioned Local Policy
next low-level action

FIGURE 7.3: Proposed Architecture for modular hierarchical model-free learning

as goal-conditioned RL, and our global policy utilizes an A* star planner for learning high-level policy.

The global policy is trained on a continuous action space to predict the next sub-goal, and the local policy learns to take discrete actions like turn 15 degrees left/right and move forward. The input to both the policies is the belief over the observed map so far. Our local policy can be considered as a short term goal driven planner but trained as goal-conditioned RL. The combined use of planning and RL aligns with our global policy where we learn high-level goal selection policy by A^* planning to find the sequence of states in the path. We have trained both the policies with PPO, where the reward is based on unique area covered at each timestep for the global policy.

Our local policy, while similar in spirit to a controller focused on grid-like environments, is trained by reinforcement learning in randomized grid maps to enable generalization. Our work is very close to [8] which demonstrated such a hierarchical policy for pointgoal navigation tasks in 3D environments. Our method is different in two respects. Firstly, our local and global policies can be trained in a disjoint manner and combined to solve the task. Secondly, our global policy model can be decomposed to account for a soft representation, such that we can provide different exploration strategies for complete map exploration. Further, it is easy to see how our proposed global frontiers could also become pointgoals for navigation, although that is not the main focus of this work.

7.4 Experiments and Results

We compare our method with nearest frontier based exploration (NFBE) and end-to-end trained policies for coverage task. Although classic approaches perform well in the original FourRooms environments, our research question is to compare the learning and traditional navigation approaches when the clutter density in the rooms increases. The

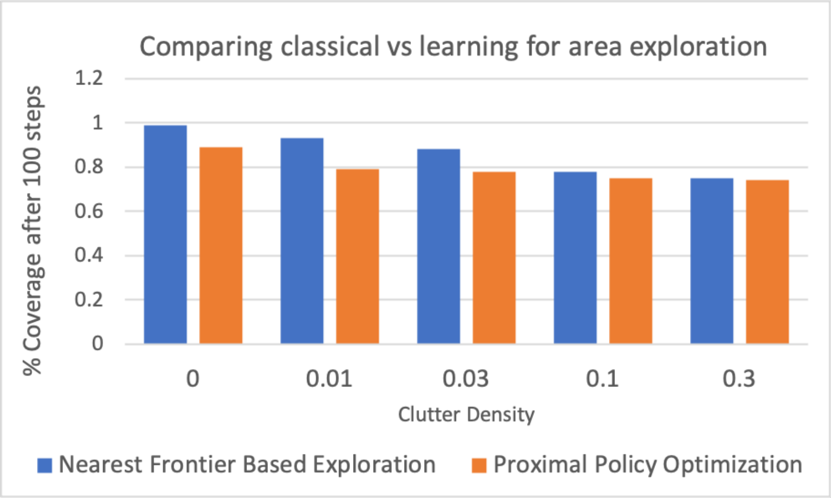


FIGURE 7.4: Coverage % based on clutter density. In less clutter, the heuristic path planning outperforms the learning based approach. With increasing clutter, the coverage of both kind of policies reduces and the gap in performance also reduces.

notion of clutter density is analogous to the real world case, where a lot of frontiers detected may occur in the small enclosed spaces, and requires learning of the structural priors for intelligent navigation. In Fig 7.4, we observe that the gap in coverage of nearest frontier based exploration (NFBE) and global policy trained with proximal policy optimization (PPO) for higher clutter is less than 1%. This suggests that learning based methods can perform at par with classical navigation techniques when the nearest frontier may not be the optimal one to maximize coverage in cluttered environments.

Increasing the clutter density leads to impassable areas in the map and it becomes close to the traditional maze puzzle without much learnable structural information. So, we cannot arbitrarily increase the density in this comparison. The next step is to realistically organize clutter and incorporate movable clutter in the 3D environments, for example with indoor house simulators.

We note that learning based approaches are sensitive in terms of performance with the change in environments and show a large variance in the coverage percentage. End-to-end training of policy gradients algorithm is unstable due to non-stationary rewards based on coverage. Decomposing the low-level navigation to local policy while using the high-level structure information with the global policy allowed us to train the networks for comparison.

Chapter 8

Conclusions

Explaining something can be tedious-both for the speaker and the listener. Assuming the listener is as attentive as possible, the speaker should adapt their explanations to help the listener understand better. From day-to-day experience, the explanations should be starting with information of what the listener already understand [**ted-dominic**]. People can only take certain amount of information at a given time. The explanations should be more clear to get the point across. Technically, the feature attribution based explanations only give a part of the story, but it is approximately correct based on the model's attribution.

In Chapter 3, our work shows how explanation methods can be utilized for DRL models. We demonstrate the use of visualizing feature attributions to investigate instances the policy succeeds/fails. Visualizing attributions is just the first step to understand the issues in the trained policy. We can further extend it to analyze why policy fails in different distribution shifts of the environment settings.

Gradient with respect to the inputs connects attributions to adversarial attack/defenses methods. Utilizing attributions to create more semantically guided variations in the sensor input for training robust policies is open direction for future work.

In Chapter 4, we engineered features to explain search strategy and prior knowledge condition from sequential human trajectory data. The input gradient-based explanations heavily dependent on feature vectors that are interpretable and represent the Markov state at each timestep. We utilized count based features for objects such as victims, doors and rubble in field-of-view, signalling device and its use and general spatio-temporal mission statistics. Gradient-based interpretability methods can be applied to other sequential models such as LSTM and Transformers. While these models will reduce the need for feature engineering to process sequential data into approximate Markov states, these models often require a lot of demonstrations to generalize.

However, feature attribution methods do not provide a full explanation for an action or set of actions taken. The world view is limited in terms of the inputs to the model, but we fail to take their frequency and scale into consideration, which is especially important for credit assignment in sequential prediction and decision making.

In Chapter 5, we experimented with graph-based embeddings (RoboCSE and Graph Embeddings) and found them to outperform the baselines in cases where the optimal path to the target object is higher. These, however, perform comparably to language-based embeddings (Word2Vec, FastText) when optimal paths are shorter ($l < 10$).

When the agent is tasked with navigating to $n, (n > 1)$ objects, there are two possible scenarios, a) all n objects are given as queries at the start, b) objects are queried sequentially one-by-one. For addressing (a), Algorithm 2 can be modified to jointly optimize for all the n objects. For addressing (b), it involves handling consecutive searches based on the information gained during navigation for the first $(n - 1)$ objects to better navigate the n^{th} object. Both of these are avenues for future work.

In Chapter 6, we present a modular approach for visual language understanding with path planning in simulated search and rescue missions. The proposed system is trained to reason over which human instructions require navigation. By utilizing language and semantic map to provide context for the remote human commander instead of streaming first-person video, our system is more suitable for limited communication bandwidth scenarios. As the focus of this work is to enable efficient communication between humans and agents, we do not address the vision-to-mapping challenge. Existing object-detection and SLAM methods can be incorporated for generating required semantic maps for planning in real-world environments.

Our system design can be extended to understand other kinds of human instructions and set up specialized pipeline to process them. Each model's output probability scores can provide cues for dialog with the human. Evaluation for such dialog settings is challenging as it requires real-time interaction. Scalable solutions for human-in-the-loop evaluation, especially of the embodied AI dialog systems is an active research direction. In terms of broader impact, our system can serve as a digital companion to assist a visually-impaired person or help a maintenance worker after training these models with context-specific interaction and visual-language grounding data.

In Chapter 7, we compare a modular learning-based approach to explore coverage with heuristic nearest frontier search in varying clutter densities. Increasing the clutter density leads to impassable areas in the map and it becomes close to the traditional maze puzzle without much learnable structural information. So, we cannot arbitrarily increase the density in this comparison. The next step is to realistically organize clutter and incorporate movable clutter in the 3D environments, for example with indoor house simulators.

Learning based approaches are sensitive in terms of performance with the change in environments and show a large variance in the coverage percentage. End-to-end training of policy gradients algorithm is unstable due to non-stationary rewards based on coverage. Decomposing the low-level navigation to local policy while using the high-level structure information with the global policy allowed us to train the networks for comparison.

All of the experimental results are based on synthetic simulations and serve as proof of concepts for some possible interesting scenarios for explainability in embodied AI. Extending these insights to either more complex scenarios where different phenomena coexist or to real-world applications is an interesting direction for future research.

Appendix A

Appendix Environments

A.1 Minecraft for Search and Rescue setting

A.1.1 Human Trajectory Data

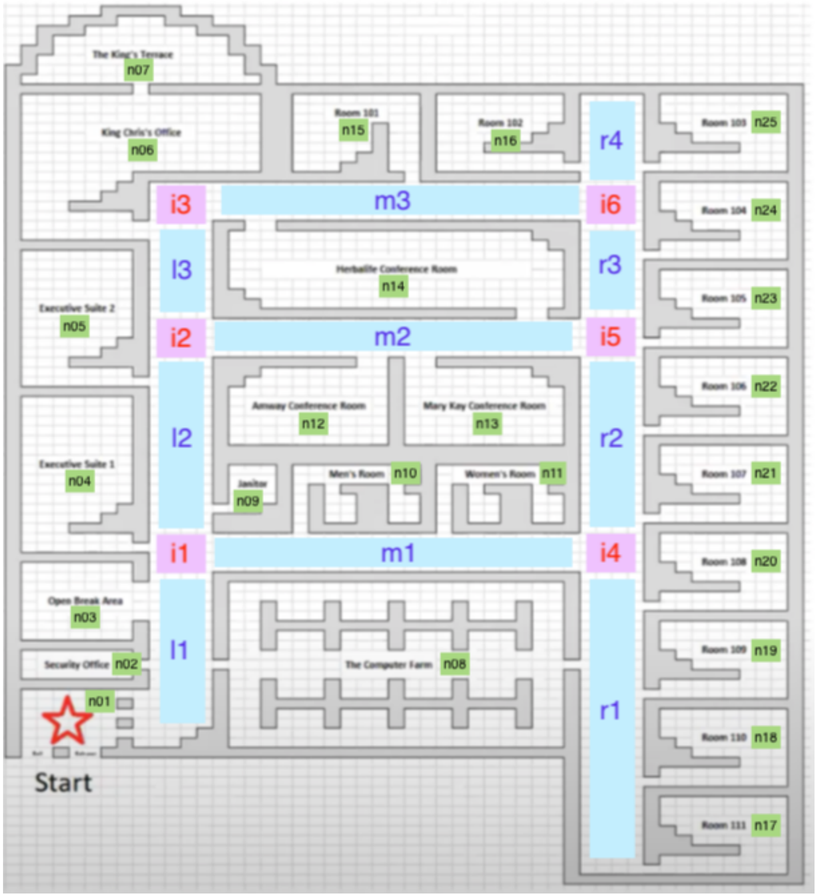


Figure A.1: Area segments corresponding to the Falcon map

Hyperparameters for xgboost classifier

```
{'criterion': 'gini',
 'max_depth': 7,
 'max_features': 'sqrt',
 'n_estimators': 500}
```

Table A.1: Feature extracted from general mission statistics per timestep

Feature	Type	Description
mission_timer	continuous	Time in seconds from the mission start
area_present_in	categorical	Which area segment player is present in? (shown in Fig A.1)
map_coverage	continuous	Percentage of area covered in the map
trial	categorical	Player's trial number : 1 - 2 - 3
map_type	categorical	Map variant: easy - medium - hard

Table A.2: Feature extracted from player's field of view events

Feature	Type	Description
door_in_fov	count	Number of door in view at current time
green_victim_in_fov	count	Number of green victim in view at current time
yellow_victim_in_fov	count	Number of yellow victim in view at current time
triaged_victim_in_fov	count	Number of triaged victim in view at current time
blockage_in_fov	count	Number of blockage/rubble in view at current time
fire_in_fov	count	Number of fire segments in view at current time
is_blockage_seen	binary	1 if any blockage seen till now; else 0

Table A.3: Feature extracted from victim triaging/saving events

Feature	Type	Description
victim_being_triaged	binary	1 if a victim is being triaged; else 0
critical_victims _triaged_till_now	count	Number of yellow victims saved till current time
less_critical_victims _triaged_till_now	count	Number of green victims saved till current time
is_green_victim_triaged	binary	1 if any green victim has been saved till now; else 0

Table A.4: Feature extracted from victim detection signal events

Feature	Type	Description
signal_message	categorical	No beep - beep - beep beep received at current second
latest_beep_signal	categorical	Last beep signal received: no beep - beep - beep beep
latest_beep_time	continuous	Time in seconds since last device signal was received
latest_beep_x	continuous	X coordinate where the most recent signal was received
latest_beep_z	continuous	Z coordinate where the most recent signal was received
distance_latest_signal _to_player	continuous	Distance between player's current location and where last detection signal was received
time_latest_signal _to_player	continuous	Time passed in seconds since last detection signal was received

Table A.5: Feature extracted from player's door related events

Feature	Type	Description
door_entered	binary	1 if door entered at current second; else 0
doors_opened_till_now	count	Number of doors opened till current time
less_critical_saved _per_doors_opened	continuous	Ratio of regular or less critical victims saved to number of doors opened till current time
critical_saved _per_doors_opened	continuous	Ratio of critical victims saved to number of doors opened till current time
latest_door_time	continuous	Time in seconds since last door opening event
latest_door_x	continuous	X coordinate where the most recent door was opened
latest_door_z	continuous	Z coordinate where the most recent door was opened
distance_latest_door _to_player	continuous	Distance between the last door opened and player's current location
time_latest_door _to_player	continuous	Time passed in seconds since last door was opened

Table A.6: Feature extracted from player's door opening and detection signal events

Feature	Type	Description
distance_door_signal	continuous	Distance between the locations where last signal was received and where last door was opened
time_door_signal	continuous	Time between the events when last signal was received and when last door was opened

Table A.7: Count-based features from Human Trajectory Data

Feature	Type	Description
mission_timer	continuous	Time in seconds from the mission start
player_position_x	continuous	Player's x coordinate in [0, W]
player_position_z	continuous	Player's z coordinate in [0, H]
area_present_in	categorical	Area segment player is present in as shown in Fig \ref{}
map_coverage	continuous	Percentage of area covered in the map
green_seen	count	Number of regular or less critical victims seen till current time
yellow_seen	count	Number of critical victims seen till current time
doors_seen	count	Number of doors seen till current time
yellow_triaged	count	Number of critical victims saved till current time
green_triaged	count	Number of regular or less critical victims saved till current time
doors_opened	count	Number of doors opened by the player till current time
less_critical_saved _per_doors_opened	continuous	Ratio of regular or less critical victims saved to number of doors opened
critical_saved _per_doors_opened	continuous	Ratio of critical victims saved to number of doors opened till current time
Beep	count	Number of beep (for regular victim nearby) signals till current time
BeepBeep	count	Number of beep beep (for critical victim nearby) signals till current time
map_type	categorical	Map variants: Easy - Medium - Hard

Features input at a timestep for Knowledge condition prediction

At 18 seconds in the trajectory, the rescuer is predicted in knowledge condition 3 with high probability **0.6396**. The probability for being in *TriageSignal* and *TriageNoSignal* is relatively much lower (0.1723, and 0.1881 respectively).

The inputs given to the model at this instance are:

```
mission_timer                                       18
area_present_in                                    n01
map_coverage                                 0.0465116
door_in_fov                                          0
green_victim_in_fov                                  0
yellow_victim_in_fov                                 0
triaged_victim_in_fov                                0
blockage_in_fov                                      0
fire_in_fov                                          0
critical_victims_triaged_till_now                    0
less_critical_victims_triaged_till_now               0
is_green_victim_triaged                          False
is_blockage_seen                                 False
signal_message                                 No Beep
door_entered                                     False
doors_opened_till_now                                1
less_critical_saved_per_doors_opened                 0
critical_saved_per_doors_opened                      0
latest_door_time                                     5
latest_door_x                                    −2096
latest_door_z                                      147
latest_beep_signal                             No Beep
latest_beep_time                                 10000
latest_beep_x                                    10000
latest_beep_z                                    10000
distance_door_signal                             10000
time_door_signal                                 10000
distance_latest_door_to_player                 3.89658
time_latest_door_to_player                          13
distance_latest_signal_to_player               15597.3
time_latest_signal_to_player                     −9982
trial                                                2
victim_being_triaged                             False
```

The force plot in Figure A.2 is an instance from the trajectory of trial 179 in *NoTriageNoSignal* condition for the not-so likely output *TriageSignal*. The predicted logits for *TriageSignal* are close to the base value at 0.43; The higher direction indicates having full knowledge and the lower direction indicates the other two knowledge conditions. Features contributing to increase the likelihood of rescuer being in full knowledge condition

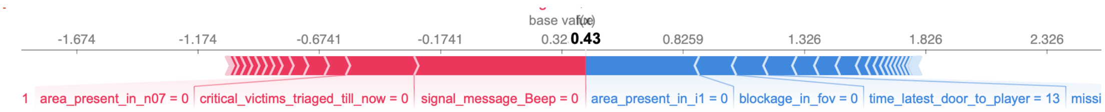


FIGURE A.2: Force plot showing model's probability and feature importance for *TriageSignal* prediction at timestep of 18 seconds; **True Knowledge Condition**:*NoTriageNoSignal*;

are the number of critical victims triaged and beep signal message. The fact that the rescuer is not present in the first intersection (i1) and has opened a door 13 seconds, hints that the rescuer might not be acting by the beep signal and thereby, decrease the likelihood of them being in full knowledge condition.

Attribution-based Sentence Selection

We prepare a list of expressions to explain how a feature positively or negatively contributes to this prediction. For an instance at test time, our model makes a prediction with the given input features and TreeSHAP computes the feature importance values associated with each feature. Depending on the feature importance values, we select the expressions to explain the model's predictions in terms of the input features which are based on the observations of the human's trajectory so far.

`explanations.csv` contains positive and negative explanatory sentence for most of the attributes used for prediction. Below is the code showing how explanatory sentences are selected

```python
explainer = shap.TreeExplainer(model)

# Calculate Shap values
shap_values = explainer.shap_values(single_pred)

reasons = []
for i in range(x_test.columns.shape[0]):
    reasons.append([x_test.columns[i], x_test.iloc[index, i],
     ↪  shap_values[1][i]])

sorted_reasons = sorted(reasons, key = lambda x: abs(x[2]),
 ↪  reverse=True)

df = pd.read_csv('explanations.csv')

for values in sorted_reasons:
    if values[0] in features[:,0]:
        print(values[0], ' = ', values[1])
        if values[2] > 0:
            print(df[df['Feature'] == values[0]]['Positive'].values)
        else:
            print(df[df['Feature'] == values[0]]['Negative'].values)
```

A.1.2 VLN by Goal Identification and Path Planning

Table A.8: Example of Dialogue between the Human and the Agent

Human :	Find a victim
Agent :	... reached near victim!
Human :	Search for a nearby switch
Agent :	... reached near switch!
Human :	Is victim near the switch?
Agent :	yes, victim seen near a switch.
Human :	Search for fire nearby
Agent :	... reached near fire!
Human :	locate more fire
Agent :	... reached near fire!
Human :	is a bookshelf visible?
Agent :	yes, bookshelf can be identified.
Human :	go to the previous injured person
Agent :	... reached near last victim!

A.2 AI2Thor for Urban Household setting

A.2.1 VLN by Imitation Learning

ALFRED [45] is a benchmarking environment built on Unity3D game engine to learn a policy from natural language instructions and egocentric vision to sequences of actions for household tasks. The dataset of visual demonstrations is collected by PDDL planning for a collection of tasks and using the underlying game simulator. The natural language description of high-level goal and low-level instructions is collected from MTurkers.

Output action space:

1. –pad– (dummy; not important)
2. –seg– (dummy; not important)
3. –stop–
4. LookDown 15 degrees
5. MoveAhead 25 cm
6. RotateLeft 90 degrees
7. PickupObject
8. SliceObject
9. LookUp 15 degrees
10. RotateRight 90 degrees
11. OpenObject
12. PutObject
13. CloseObject
14. ToggleObjectOn
15. ToggleObjectOff

A.2.2 Embeddings for Object Search

Embedding Training

We describe our technique to train our Graph Embeddings. We used standard off the shelf pre-trained Word2Vec (trained on Google News) and FastText embeddings (trained on en Wikipedia), with no modifications. RoboCSE [16] uses ANALOGY[32], a semantic matching method, to learn multi-relational embeddings processed from the AI2Thor environment data, where the nodes represent the objects and edges denote the relation between them.

To train the Graph Embeddings, we start by extracting a knowledge graph using a subset of floor plans for each Room type in the AI2Thor environment. The knowledge graph captures the co-location of objects and the relationship between them. Eg. *Book* is **located** in the living room **on** a *table*. The nodes in the graph represents objects (eg. Book, Table) and edges represents relationship (eg. on). We then used Deepwalk (number of walks: 20, walk length: 8 and window size: 3) to generate a 25 dimensional embedding. Once trained we do not modify the embeddings during navigation.

Navigation: Oscillations

Navigating to the object by choosing the object with the highest cosine similarity (with respect to *query object*) could potentially suffer from oscillations. For example,consider the agent is at position A, and is navigating to position B. On taking a few steps towards position B, the object at position B could potentially become occluded. Then the object would navigate towards the (next visible) object with highest cosine similarity. This could be at position C. Now on taking a few steps towards position C, object at position B becomes visible again. This cycle repeats. In algorithm **??**, the agent takes $p_0^{\beta T}$ steps at a time, which is some fraction of steps of the plan. While this heuristic does not alleviate the possibility of oscillations, in our experiments, we observed negligible oscillations.

Hyperparameters for Reproducibility

The code is shared at https://github.com/vidhiJain/SpatialEmbeddings

Here, we report the hyperparameters which can be used to replicate the results presented in Table 5.1.

`VISIBLE AREA THRESHOLD = 0.01`; Objects that are lower than this ratio, are discarded.

`STEP MULTIPLE = 5`; How many `‘‘Steps * 0.25 units’’` to move in each time step.

`Random Seed: 33` to initialize agent in AI2Thor and randomize the object initialization in the environment.

Appendix B

Appendix Attribution Methods

Most attribution techniques involve perturbations to simulate counterfactual reasoning. These perturbations could be in the network state or the input space.

B.1 Shapley value

Let n_i represent a player indexed at i. Consider $N = n_1, n_2, \cdots, n_i, ...n_{|N|}$ represent the set of players in a coalition. Let S be a subset of N. Let v be the function that maps the set of players to their team's reward/payoff. This describes the value of the set of players. Note that the whole can be greater than the sum of parts, i.e. it is possible that $v(\{n_i\}) + v(\{n_j\}) \neq v(\{n_i, n_j\})$ where n_i, n_j are players.

Marginal attribution to a player n_i in a coalition set $S \in N \backslash \{n_i\}$ is

$$m_i(S) = v(S \cup \{n_i\}) - v(S)$$

Marginal attribution of the player n_i is decided based on what the team's value is with and without n_i. Though this is efficient way of assigning attributions, it greatly depends on the team set S.

As this calculation depends on the order of joining, a more 'fair' solution could be to take all possible joining orders of the players to calculate average marginal attributions.

Consider $\boldsymbol{\sigma}$ as a vector of player indices denoting the order of their joining. Let $\boldsymbol{m}_\sigma$ be a vector of marginal attributions of each player considering all the before it in the team S.

Let $\Pi(N)$ be all possible permutations of $|N|$ players. Then Shapley value $\phi(N, v)$ describing the fair payoff to each player is given as:

$$\boldsymbol{\phi}(N, v) = \frac{1}{|N|!} \sum_{\sigma \in \Pi(N)} \boldsymbol{m_\sigma}$$

Expressing Shapley value for each player's attribution, we have

$$\phi_i(N, v) = \sum_{S \in N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} (v(S \cup \{i\}) - v(S))$$

Shapley values is a unique method that satisfies the following axioms to ensure fairness in the feature attributions. [36]

1. Dummy: If a feature does not contribute to the output, it should not get any attributions
2. Symmetry: Clone of a feature must receive equal attribution
3. Efficiency: Attributions must add to the total value or reward obtained by the coalition.
4. Linearity: Attribution for weighted sum of two outputs must be same as the weighted sum of attributions for each of the games.

B.2 TreeSHAP

Some of the other tools to analyze feature importance in ensembles of decision trees are:

1. Feature importance in `XGBoost` library: `gain` type shows the average gain across all splits where feature was used. The `weight` shows the number of times the feature is used to split data. This type of feature importance can favourize numerical and high cardinality features. Be careful! There are also `cover, total_gain, total_cover` types of importance.

2. Permutation based importance in `scikit learn` library permutation based importance is computationally expensive (for each feature there are several repeast of shuffling). The permutation based method can have problem with highly-correlated features. Feature importance or attribution is an approximation of how important features are in the data.

TreeSHAP [34] considers averages attributions over all orderings of the features, instead of just the decision path taken in a set of trees. This guarantees consistency for local explanations as this approach impartially assigns credit to input features regardless of their depth in the tree. TreeSHAP is shown to be better and intuitive for explanations as compared to existing approaches.

B.3 Integrated Gradients

Integrated Gradients (IG) determines the salient inputs by gradually varying the network input from a baseline x' to the original input x and aggregating the gradients.

$$IG(x, x')_i = (x_i - x'_i) \int_{\alpha=0}^{\alpha=1} \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} d\alpha$$

This integral is approximated with summation as follows:

$$IG_{approx}(x, x'; m) = (x_i - x'_i) \sum_{k=0}^{m} \frac{\partial F(x' + \frac{k}{m}(x - x'))}{\partial x_i} \times \frac{1}{m}$$

B.4 XRAI

XRAI computes the effective attributions of integrated gradients over overly segmented image. The image is segmented based on similarity such as color, which makes the segment boundaries align with the edges. The segmentation is done at multiple scales to obtain a set of overlapping image segments.

Assume that attribution mask over an image I of size $H \times W$ is A of the same size. Using graph-based segmentations over multiple scale parameters, we obtain a set of segments $\mathcal{S}$.

Let a pixel be indexed by i in the original image. For a segment s, the gain can be calculated by $g_s = \sum_{i \in s \setminus M} \frac{A_i}{area(s \setminus M)}$. The segment with maximum gain is selected as attribution to update the XRAI saliency set $\mathcal{M}$. The process is repeated with the remaining segments until the area of the mask set is equal to that of the image.

XRAI method seems to produce slightly better visual attributions over other variants of IG, which create grainy regions.

However, this method depends on the size of segmentation scales selected for computation. Further, dilation added to the final attribution masks to include edges may depict an inflated version of model's actual feature importance.

B.5 DeepSHAP

DeepSHAP [10] is a difference-from-reference approach. Both DeepLIFT and DeepSHAP overcome the issues of gradient saturation and discontinuity in gradient-based attribution approaches like IG. The most common way to calculate the attribution is by Rescale rule, which is inspired from Shapley value computation. Here the positive and negative influences are separated and calculated in all possible orderings to compute attribution for the target.

difference-from-reference $\delta z = z - z_0$ where z is the activation of a neuron. z_0 is the baseline/reference value of the activation of a neuron. Let y be a target neuron, and $x_1, x_2, \ldots, x_n$ be the input affecting the neuron $y = f(x_1, x_2, \ldots, x_n)$. Contribution of each neuron is $\sum_i C_{\delta x_i \delta y} = \delta y$ where δy can be non-zero even if $\frac{\partial y}{\partial x}$ is zero.

By separating positive and negative contributions to a neuron as $\delta y = \delta x^{+} + \delta x^{-}$, the RevealCancel rule approximates the Shapley values of $\delta x^{+}$and δx^{-} for the target neuron y.

DeepSHAP improves the RevealCancel rule by splitting by the mean than splitting by zero as it better separates x_i nodes, resulting in a better approximation of Shapley values.

Bibliography

[1] Aishwarya Agrawal et al. “Don’t Just Assume; Look and Answer: Overcoming Priors for Visual Question Answering”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 4971–4980.

[2] Maruan Al-Shedivat et al. “On the Complexity of Exploration in Goal-Driven Navigation”. In: *CoRR* abs/1811.06889 (2018). arXiv: `1811.06889`. URL: `http://arxiv.org/abs/1811.06889`.

[3] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. “Neuronlike adaptive elements that can solve difficult learning control problems”. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5 (1983), pp. 834–846. DOI: `10.1109/TSMC.1983.6313077`.

[4] Rhyse Bendell et al. “Towards Artificial Social Intelligence: Inherent Features, Individual Differences, Mental Models, and Theory of Mind”. In: *Advances in Neuroergonomics and Cognitive Engineering*. Ed. by Hasan Ayaz, Umer Asgher, and Lucas Paletta. Cham: Springer International Publishing, 2021, pp. 20–28. ISBN: 978-3-030-80285-1.

[5] Michael Bloesch et al. “CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM”. In: June 2018, pp. 2560–2568. DOI: `10.1109/CVPR.2018.00271`.

[6] Piotr Bojanowski et al. “Enriching Word Vectors with Subword Information”. In: *arXiv preprint arXiv:1607.04606* (2016).

[7] Carlos Campos et al. “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM”. In: *arXiv preprint arXiv:2007.11898* (2020).

[8] Devendra Singh Chaplot et al. “Learning To Explore Using Active Neural SLAM”. In: *International Conference on Learning Representations*. 2020. URL: `https://openreview.net/forum?id=HklXn1BKDH`.

[9] Devendra Singh Chaplot et al. “Object Goal Navigation using Goal-Oriented Semantic Exploration”. In: *In Neural Information Processing Systems*. 2020.

[10] Hugh Chen, Scott Lundberg, and Su-In Lee. “Explaining models by propagating Shapley values of local components”. In: *Explainable AI in Healthcare and Medicine.* Springer, 2021, pp. 261–270.

[11] Tao Chen, Saurabh Gupta, and Abhinav Gupta. “Learning Exploration Policies for Navigation”. In: *International Conference on Learning Representations.* 2019. URL: `https://openreview.net/forum?id=SyMWn05F7`.

[12] Valerie Chen, Abhinav Gupta, and Kenneth Marino. “Ask Your Humans: Using Human Instructions to Improve Generalization in Reinforcement Learning”. In: *International Conference on Learning Representations.* 2021. URL: `https://openreview.net/forum?id=Y87Ri-GNHYu`.

[13] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. *Minimalistic Gridworld Environment for OpenAI Gym.* `https://github.com/maximecb/gym-minigrid`. 2018.

[14] Maxime Chevalier-Boisvert et al. “BabyAI: First Steps Towards Grounded Language Learning With a Human In the Loop”. In: *International Conference on Learning Representations.* 2019. URL: `https://openreview.net/forum?id=rJeXCo0cYX`.

[15] Angela Dai et al. “BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration”. In: *ACM Transactions on Graphics 2017 (TOG)* (2017).

[16] Angel Daruna et al. *RoboCSE: Robot Common Sense Embedding.* 2019. arXiv: `1903.00412 [cs.RO]`.

[17] Abhishek Das et al. “Embodied Question Answering”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2018.

[18] Matt Deitke et al. “RoboTHOR: An Open Simulation-to-Real Embodied AI Platform”. In: (2020).

[19] Daniel Gordon et al. “IQA: Visual Question Answering in Interactive Environments”. In: *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on.* IEEE. 2018.

[20] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: `10.1109/TSSC.1968.300136`.

[21] Peter Hase and Mohit Bansal. *Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior?* 2020. arXiv: `2005.01831 [cs.CL]`.

[22] L. Huang et al. “Using humans’ theory of mind to study artificial social intelligence in minecraft search and rescue”. In: *to be submitted to the) Journal of Cognitive Science* (2021).

[23] W. Johnson. “Agents that Learn to Explain Themselves”. In: *AAAI*. 1994.

[24] A. Kapishnikov et al. “XRAI: Better Attributions Through Regions”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 4947–4956.

[25] Andrei Kapishnikov et al. “Guided Integrated Gradients: An Adaptive Path Method for Removing Noise”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 5050–5058.

[26] Been Kim et al. *Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)*. 2018. arXiv: `1711.11279 [stat.ML]`.

[27] Alexander Ku et al. “Room-Across-Room: Multilingual Vision-and-Language Navigation with Dense Spatiotemporal Grounding”. In: *Conference on Empirical Methods for Natural Language Processing (EMNLP)*. 2020.

[28] Tejas D. Kulkarni et al. “Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS’16. Barcelona, Spain: Curran Associates Inc., 2016, 3682–3690. ISBN: 9781510838819.

[29] C. Lacave and F. Díez. “A review of explanation methods for Bayesian networks”. In: *The Knowledge Engineering Review* 17 (2002), pp. 107 –127.

[30] S. Lapuschkin et al. “Unmasking Clever Hans predictors and assessing what machines really learn”. In: *Nature Communications* 10 (2019).

[31] M. Lent, William Fisher, and Michael Mancuso. “An Explainable Artificial Intelligence System for Small-unit Tactical Behavior”. In: *AAAI*. 2004.

[32] Hanxiao Liu, Yuexin Wu, and Yiming Yang. “Analogical inference for multi-relational embeddings”. In: *arXiv preprint arXiv:1705.02426* (2017).

[33] Scott M Lundberg et al. “Explainable machine-learning predictions for the prevention of hypoxaemia during surgery”. In: *Nature Biomedical Engineering* 2.10 (2018), p. 749.

[34] Scott M. Lundberg et al. “From local explanations to global understanding with explainable AI for trees”. In: *Nature Machine Intelligence* 2.1 (2020), pp. 2522–5839.

[35] Mateusz Malinowski and Mario Fritz. “A Multi-World Approach to Question Answering about Real-World Scenes based on Uncertain Input”. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 1682–1690. URL: `http://papers.nips.cc/paper/5411-a-multi-world-approach-to-question-answering-about-real-world-scenes-based-on-uncertain-input.pdf`.

[36] Luke Merrick and Ankur Taly. “The Explanation Game: Explaining Machine Learning Models with Cooperative Game Theory”. In: *CoRR* abs/1909.08128 (2019). arXiv: `1909.08128`. URL: `http://arxiv.org/abs/1909.08128`.

[37] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: `1301.3781 [cs.CL]`.

[38] Volodymyr Mnih et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *ArXiv* abs/1602.01783 (2016).

[39] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: *ArXiv* abs/1312.5602 (2013).

[40] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. “Deepwalk: Online learning of social representations”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 701–710.

[41] Wojciech Samek et al. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Jan. 2019. ISBN: 978-3-030-28953-9. DOI: `10.1007/978-3-030-28954-6`.

[42] Victor Sanh et al. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *arXiv preprint arXiv:1910.01108* (2019).

[43] J. Schulman et al. “Proximal Policy Optimization Algorithms”. In: *ArXiv* abs/1707.06347 (2017).

[44] Lloyd S. Shapley. *A Value for N-Person Games*. Santa Monica, CA: RAND Corporation, 1952. DOI: `10.7249/P0295`.

[45] Mohit Shridhar et al. “ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. URL: `https://arxiv.org/abs/1912.01734`.

[46] K. Simonyan, A. Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *CoRR* abs/1312.6034 (2014).

[47] Kunal Pratap Singh et al. “MOCA: A Modular Object-Centric Approach for Interactive Instruction Following”. In: *ArXiv* abs/2012.03208 (2020).

[48] D. Smilkov et al. “SmoothGrad: removing noise by adding noise”. In: *ArXiv* abs/1706.03825 (2017).

[49] C. Stachniss, D. Hahnel, and W. Burgard. “Exploration with active loop-closing for FastSLAM”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 2. 2004, 1505–1510 vol.2.

[50] Gregory J. Stein, Christopher Bradley, and Nicholas Roy. “Learning over Subgoals for Efficient Navigation of Structured, Unknown Environments”. In: *Proceedings of The 2nd Conference on Robot Learning*. Ed. by Aude Billard et al. Vol. 87. Proceedings of Machine Learning Research. PMLR, 2018, pp. 213–222. URL: `http://proceedings.mlr.press/v87/stein18a.html`.

[51] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. "Visualizing the Impact of Feature Attribution Baselines". In: *Distill* (2020). https://distill.pub/2020/attribution-baselines. DOI: `10.23915/distill.00022`.

[52] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 3319–3328. URL: `http://proceedings.mlr.press/v70/sundararajan17a.html`.

[53] William R Swartout and Johanna D Moore. "Explanation in second generation expert systems". In: *Second generation expert systems*. Springer, 1993, pp. 543–585.

[54] Niko Sünderhauf. *Where are the Keys? – Learning Object-Centric Navigation Policies on Semantic Maps with Graph Convolutional Networks*. 2019. arXiv: `1909.07376 [cs.LG]`.

[55] Jesse Thomason et al. "Vision-and-Dialog Navigation". In: *Conference on Robot Learning (CoRL)*. 2019.

[56] Erik Wijmans et al. "Embodied Question Answering in Photorealistic Environments with Point Cloud Perception". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[57] Shawn Xu, Subhashini Venugopalan, and Mukund Sundararajan. "Attribution in Scale and Space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[58] Brian Yamauchi. "A frontier-based approach for autonomous exploration". In: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'* (1997), pp. 146–151.

[59] Shichao Yang and Sebastian Scherer. "CubeSLAM: Monocular 3-D Object SLAM". In: *IEEE Transactions on Robotics* PP (May 2019), pp. 1–14. DOI: `10.1109/TRO.2019.2909168`.

[60] Wei Yang et al. "Visual Semantic Navigation using Scene Priors". In: *ICLR*. 2019.

[61] Yichi Zhang and J. Chai. "Hierarchical Task Learning from Language Instructions with Unified Transformers and Self-Monitoring". In: *ArXiv* abs/2106.03427 (2021).