What do navigation agents learn about their environment?

Kshitij Dwivedi, Gemma Roig
Goethe University Frankfurt
dwivedi@em.uni-frankfurt.de, roig@cs.uni-frankfurt.de

Aniruddha Kembhavi, Roozbeh Mottaghi
PRIOR @ Allen Institute for AI
anik@allenai.org, roozbehm@allenai.org

Abstract

*Today's state of the art visual navigation agents typically consist of large deep learning models trained end to end. Such models offer little to no interpretability about the learned skills or the actions of the agent taken in response to its environment. While past works have explored interpreting deep learning models, little attention has been devoted to interpreting embodied AI systems, which often involve reasoning about the structure of the environment, target characteristics and the outcome of one's actions. In this paper, we introduce the Interpretability System for Embodied agEnts (i*SEE*) for Point Goal and Object Goal navigation agents. We use i*SEE *to probe the dynamic representations produced by these agents for the presence of information about the agent as well as the environment. We demonstrate interesting insights about navigation agents using i*SEE*, including the ability to encode reachable locations (to avoid obstacles), visibility of the target, progress from the initial spawn location as well as the dramatic effect on the behaviors of agents when we mask out critical individual neurons.*

1. Introduction

The research area of Embodied AI – teaching embodied agents to perceive, communicate, reason and act in their environment – continues to receive a lot of interest from the computer vision, natural language processing and robotics communities. A growing body of work has resulted in the emergence of several powerful and visually rich simulators including AI2-THOR [20], Habitat [25] and iGibson [37]; works that require agents to navigate [2], reason [5], collaborate [18], manipulate [12] and follow instructions [3].

While fast progress is being made across a variety of tasks and benchmarks, most solutions being employed are black box neural networks trained to either imitate a se-

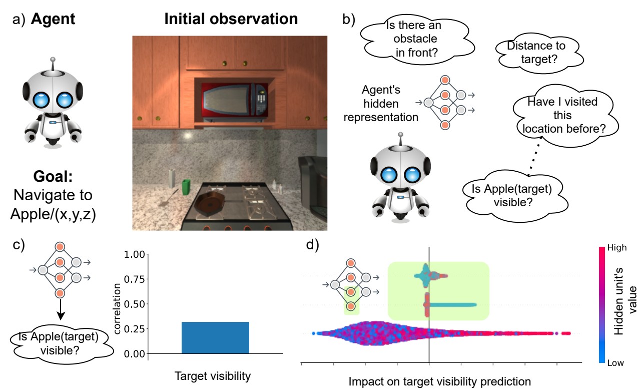

Figure 1. **The iSEE framework.** (a) An agent learns to perform the OBJECTNAV or POINTNAV tasks. (b) We wish to explore what information is encoded in the hidden representations of the agent. (c) To achieve this, we evaluate how well the agent's hidden representation can predict human interpretable concepts e.g. target visibility in ObjectNav. (d) Then we apply an explainablity method SHAP [23] to identify the top-k relevant units.

quence of human/oracle actions or trained via reinforcement learning with a careful selection of positive and negative rewards. These models offer little to no interpretability out-of-the-box about the concepts and skills learned by the model or about the actions taken by the model in response to a task or observation. Developing interpretable systems is particularly important in embodied AI since we expect these systems to eventually be deployed onto robots that will navigate the real physical world and interact with people in it.

In the image classification literature, a number of interpretability methods have been developed over the past few years [7, 14, 28, 49]. These methods rely on probing model activations via various inputs or generating synthetic inputs that lead to a spike in an activation. While such methods are useful in probing Embodied AI models, they do not take into account the rich metadata (such as perfect segmentation, depth maps, precise object localization, etc.) available in synthetic environments commonly used to train these models. Simulated worlds provide us a unique opportunity to expand interpretability research to embodied agents and develop new methods that take advantages of rich metadata.

We propose a framework to interpret the hidden representations of embodied agents trained in simulated worlds. We apply our framework to two navigation tasks (Figure 1a): Object Navigation (OBJECTNAV) [6], the task of navigating to a target object and Point Goal Navigation (POINTNAV) [2], the task of navigating to a specified relative co-ordinate, within the AI2THOR environment; but our methods are general and can be easily applied to more tasks and other environments. We train agents to perform these tasks and then probe their hidden representations to evaluate if they encode aspects of their task, progress and surroundings (Figure 1b and 1c). We then apply the model interpretation method SHAP [23] to identify which hidden units are most relevant for predicting these concepts (Figure 1d). Our framework allows us to gather evidence towards answering two fundamental questions about a trained model: (1) Has the model learned a particular concept ? (2) Which units within a recurrent layer encode this concept ? Using this framework, we were able to find several interesting insights about OBJECTNAV and POINTNAV agents.

The key contributions of this work are:

- A new interpretability framework specialized for navigation agents with no linearity assumptions between concepts and hidden units.
- New insights about what navigation agents encode and in which units:
  - sparse target representation in OBJECTNAV (50/512 units) and POINTNAV (5/512 units);
  - learning of concepts such as reachable locations and visit history by OBJECTNAV agents; encoding of progress towards target and less reliance on visual information by POINTNAV agents.
- Ablation experiments showing no impact on model performance after removal of 10% units suggesting redundancy in the representation.

2. Related Work

We explore representations stored within an agent's hidden units by predicting a human interpretable piece of information about the agent and its environment. Our work is related to two directions in interpretability research: (1) Interpretability of individual hidden units and (2) Explaining model's predictions.

Interpretability of hidden units. A common approach to investigate what a hidden unit encodes is to find the input image also referred to as "preferred image" that leads to a maximal activation of the unit of interest. The preferred image can be from within the examples in a dataset [49, 50] or obtained using gradient descent by optimizing over the input [13, 16, 28, 29, 39, 42]. One disadvantage of the methods using preferred images is that it is difficult to quantify the association of a unit with a concept. To address this issue, NetDissect [7, 51] uses overlap of a unit's spatial activation with groundtruth segmentation maps of a human interpretable concept as a measure to quantify a unit's association with a concept. The idea was further extended in Net2vec [14] to investigate whether a single unit or a group of units encode a concept. However, these approaches require groundtruth pixel-level annotation for every concept of interest and therefore for new concepts, new annotations are required. On the other hand, simulation environments [20, 25, 37] have annotations readily available as a part of the metadata. However, given the vast amount of metadata beyond simply object information, there is a need to develop new methods for these environments to interpret embodied agents. Recent embodied AI works [43, 48] have started focusing in interpretability by linear decoding of concepts from hidden units [43] and finding computational structure of the agent's recurrent units using fixed point analysis [48]. Patel et al. [31] explored interpretation of emergent communication in collaborative embodied agents. However these works do not focus on identifying which hidden units encode a given concept which is one of the main contributions of the present work.

Explaining model predictions. Saliency methods [4, 30, 33, 35, 40] use gradients to find which pixels of an image are relevant for model's prediction. Additive feature attribution methods [24, 34, 38] investigate the effect of adding an input feature in model prediction. A disadvantage of these methods is that they focus on explaining the model predictions on raw pixel level. To explain the model prediction using human-interpretable concepts, TCAV [19] and subsequent works [15, 17, 21] were proposed that use concept vectors instead of raw pixels to explain model prediction. To find concept vectors additional human annotations are required. In the embodied environments [20, 25, 37], we have the advantage of already annotated human interpretable concepts.

The above two directions of research have been considered as independent directions of interpretability research – one focusing on interpreting what the hidden units learn and the other on interpreting the decisions made by the model. In this work, we observe the potential of linking two approaches to interpret what the hidden units learn by using human interpretable concepts. Specifically, we train an interpretable model (Gradient boosted Tree) to predict human intepretable concepts from the hidden units of the model and then apply a global model explainability method SHAP [23] to explain which units are relevant for which concept prediction. In this work, we use SHAP because (a) it provides a unique solution with three desirable properties: local accuracy, missingness and consistency [24], (b) it unifies several model agnostic [34, 38] and tree based explanation methods [1], and (c) it provides explanation on both local (single example) and global (dataset) levels.

Embodied tasks. Several approaches have been proposed [8–11, 22, 26, 27, 32, 37, 41, 45–47, 52] to tackle the navigation problem, which is a core task in Embodied AI. In this paper, we analyze standard base models for two popular navigation tasks, PointNav [2] and ObjectNav [6].

3. Interpretability Framework

We introduce the **I**nterpretability **S**ystem for **E**mbodied ag**E**nts (iSEE). iSEE probes agents at their understanding of the task given to them, their progress at this task and the environment they act in. This probing is done via training simple machine learning models that input network activations and output the desired information. Simulated environments provide us with a gamut of metadata about the agent, task and surroundings, allowing us to train a series of models for probing this information. iSEE also helps identifying specific neural units that store this information. This is done via computing the SHapley Additive exPlanations (SHAP) [36] values for individual neural units. Finally we study the effect of switching off individual neural units on the downstream tasks that the agents are trained for.

We study embodied agents trained for POINTNAV [2] (navigation towards a specific coordinate in a room) and OBJECTNAV [6] (navigation towards a specific object). Our agents encode their visual observations via a convolutional neural network and encode their target/goal via an embedding layer. The outputs of the visual and goal encoders are fed into a gated recurrent unit (GRU) to add memory. The hidden units of the GRU are then linearly transformed into the policy (distribution over actions) (Figure 2a). There are more complex, customized models for each of these tasks that achieve higher performance. However, we utilize these simple, generic models that can be applied to various tasks and make the comparisons across tasks more fair. In this work, we use iSEE to probe the hidden units in the GRU and use gradient boosted trees (GBT) as the ML model to determine the presence of relevant information within these hidden units (Figure 2b). We focus here specifically on GRU units since (a) we are interested in analyzing dynamic visual representations (GRU units) as opposed to static visual representations (CNN visual encoder) and (b) some of our models use a frozen visual encoder and only optimize the parameters within the GRU.

We now describe the metadata extracted from the simulator, probing for this metadata via building GBTs and using SHAP to identify individual hidden units that store the relevant information.

3.1. Metadata

We probe agents at their understanding of the target, their position in the scene, the reachability of objects in their surroundings and their memory of visited locations as they navigate their world. This information is easily extracted by us from the metadata provided by the simulator.

Target Information: Agents trained for the OBJECTNAV and POINTNAV tasks must navigate to the location of a specified object or a point, respectively. In either case, one might expect an agent to be able to estimate its positioning with respect to the goal. Therefore, at a given timestep t, we extract metadata containing the distance (R_t) and orientation (θ_t) of the agent from the target (Figure 2c). In OBJECTNAV, an agent is successful if the object lies within 1m of the agent and is visible; thus we additionally extract target visibility ($visible_t$). Since an object may be visible in the frame but not within the specified distance to determine success, we also extract the percent of pixels covered by the target object using segmentation masks provided by AI2-THOR ($Area_t$).

Agent's information: Memory of how far and in what direction one has travelled can be relevant to avoiding revisiting locations in the scene. Therefore, we extract the agent's distance (R_a) and orientation (θ_a) with respect to its starting location (Figure 2c).

Reachability: For an agent to successfully navigate in a scene it should be able to detect obstacles and its path around them. Thus, we extract metadata to detect whether a particular location with respect to the agent's current location is reachable or not. Given an agent's location, we first extract all reachable gridpoints in the scene. Then, with the agent's location as the center we consider three concentric circles with radii=2, 4,and 6 times the grid size and locate points on these circle that are at angles from 0 to 360 in the steps of the agents rotation angle (=30 degrees). For each of these points R_r, θ_{angle}, where r is the radius and $angle$ is the orientation of the grid point with respect to agent in degrees, we check whether the closest reachable gridpoint is within $gridSize/\sqrt{2}$ or not. Figure 2d illustrates such reachable gridpoints in the scene.

Visited History: The metadata extracted above captures a global summary of the agent's movements. We also extract its local visit history. This is done by checking if a location ($\texttt{visited}_l$), rotation ($\texttt{visited}_{lr}$) and camera horizon ($\texttt{visited}_{lrh}$) has been visited by the agent or not.

3.2. Metadata extraction

As the agent traverses around in a scene, we extract the GRU activations of the agent along with the agent and scene metadata described above. This is done within the training and validation scenes. The latest model architectures and training algorithms for POINTNAV and OBJECTNAV lead to very capable agents that (a) exhibit little variability in their trajectories (b) do not collide often (c) make few mistakes such as revisiting locations. Such trajectories are less useful to probe agents, since the events of interest occur sparsely. Hence we use human trajectories (trajectories

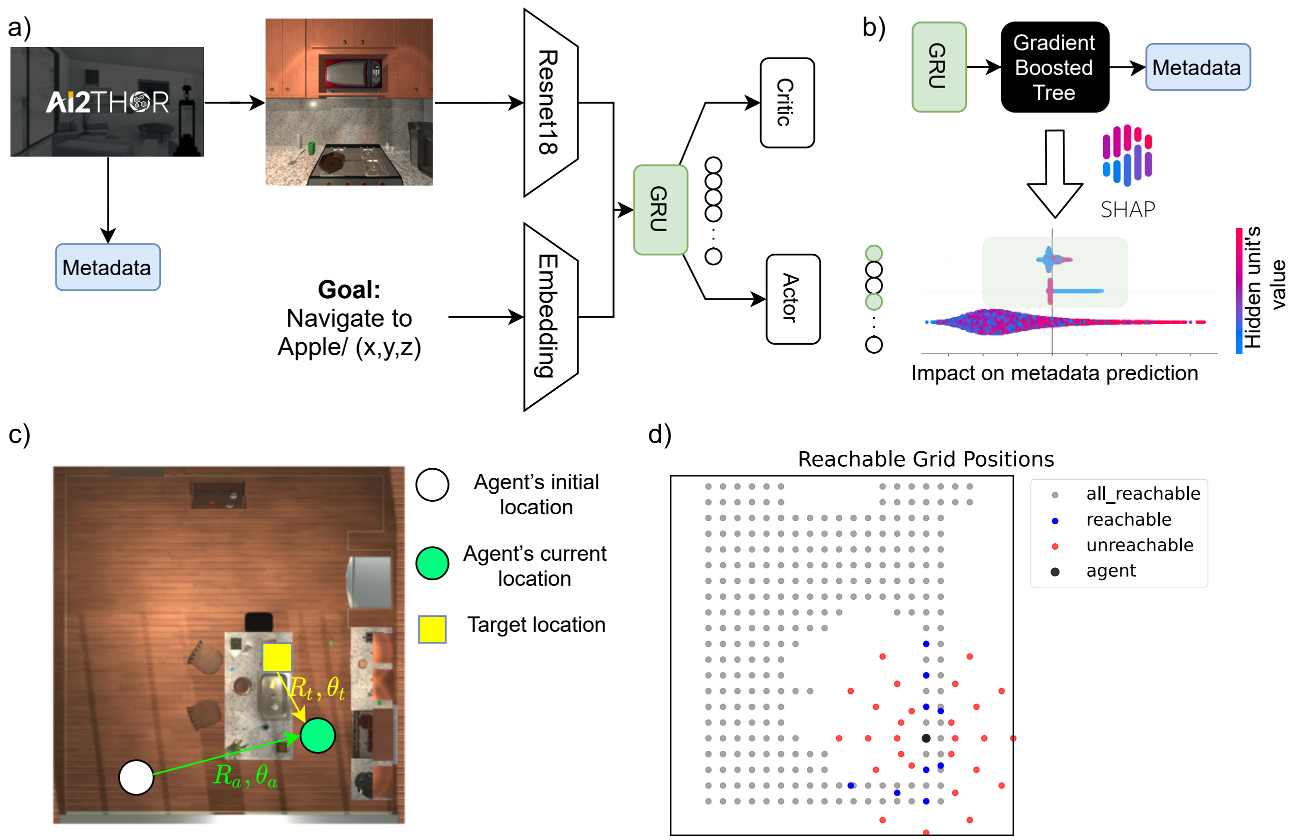

Figure 2. **iSEE:** a) At a given timestep, AI2THOR generates an observation that is fed as input to the agent along with a goal embedding. For that time step, we also extract relevant event metadata from AI2THOR which is unseen by the agent. b) After sampling rollouts from multiple training and validation episodes, we train a gradient boosted tree to predict metadata from the agent's hidden representation (GRU units). We then apply SHAP, an explainability method that identifies the top-k most relevant units for predicting a given metadata type. c) At a given timestep, we extract agent's orientation with respect to its initial spawn location (R_a, θ_a) and target location (R_t, θ_t). d) We extract reachable positions at distance 2,4,6 times the grid size and different angles with step size of 30 degrees to identify whether these locations can be reached by the agent or not.

specified by humans navigating around) that encourage exploration and have intentional collisions and mistakes. Using a pre-defined set of human trajectories also enables us to fairly compare findings across agents.

3.3. Metadata prediction

We train GBTs to predict specific metadata concepts using the GRU's hidden units as inputs. GBTs are trained using episodes within the training scenes and evaluated using correlation between the predicted metadata and groundtruth metadata on the validation episodes. For a given model, we trained one GBT of $depth = 10$ for each concept using xgboost library. For binary variables (such as target visibility) we use the logistic loss function and for continuous variables (such as distance from target/agent's initial position) we use the mean squared error loss function. Total training and evaluation time of GBT was 8 seconds on a single NVIDIA RTX 2070 GPU. We use GBTs because: (1) they are more interpretable in comparison to many other ML models when the mapping from inputs to outputs is not linear; (2) allow exact computation of SHAP values as compared to other models where SHAP values can only be approximated [23].

3.4. Identifying explainable units using SHAP

Given a set of hidden units, SHAP computes the importance of each individual unit by quantifying its contribution towards predicting a concept. SHAP values are based on a game theory concept called Shapley values [36]. We first train a GBT to predict a concept using all hidden units. We then use a subset of hidden units and mask other units to predict a concept using pretrained GBT. Then we add in a new hidden unit and compute the change in the model's prediction capability. This difference quantifies the contribution of a hidden unit with regards to the chosen subset. By averaging this contribution over all possible subsets of hidden units, we get the Shapley value of the unit of interest. For instance, we use this method to compute the contribution of a specific GRU hidden unit towards predicting the visibility of the specified target. Note that the obtained

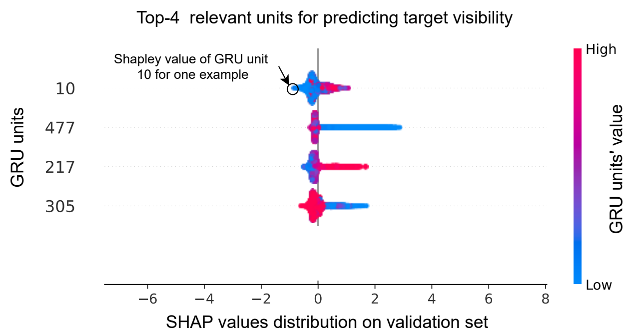

Figure 3. **Schematic to read a SHAP plot:** The plot shows the top-4 relevant GRU units to predict the target visibility. Each row shows the distribution of SHAP values of a given GRU unit for all the examples in the validation set with each dot in the row corresponding to an individual data point. The color of the dot indicates whether the GRU unit's output was low or high for that data point.

Shapley value indicates the impact of the hidden unit on the model's outcome for a single example. To quantify the global impact of hidden unit on model's outcome we calculate the mean of absolute SHAP values over all examples in the validation set (for more details please see Appendix A).

Figure 3 is a SHAP beeswarm plot to visualize the global contribution of the top-k relevant GRU units. We use this plot to explain how one can interpret SHAP plots. This plot visualizes the contribution of the top 4 relevant units to predict target visibility. Each row corresponds to a given GRU unit, and each dot in the row corresponds to the GRU unit's Shapley value for a given example. Each row displays the distribution of SHAP values on all the samples of the validation set. The location of a dot on the x axis shows whether the impact of the GRU unit on model's prediction (i.e. Shapley value) is positive or negative. The GRU unit's value for a sample is visualized using the colorbar on right. As an example, for the circled dot in Figure 3, the Shapley value of GRU unit 10 is negative and the color of the dot indicates that GRU unit 10's value is also low. For the examples on the right side of x-axis the shapley values are positive and the GRU unit's values are also higher. This means that GRU unit 10 is positively correlated with target visibility. Using a similar logic GRU unit 477 seems to be negatively correlated with target visibility. In a nutshell, the SHAP plot shows the global contribution of a GRU unit in prediction of a concept (rows sorted by contribution), displays the distribution over the validation examples (points in each row) and indicates whether a unit is positively or negatively correlated with the concept (colors of the points in accordance with the x-axis values).

4. Experimental Setup

We use the AllenAct [44] framework to train models for the tasks ObjectNav and PointNav tasks in the iTHOR rooms within AI2Thor [20]. For both tasks, we use the same split of rooms for training and validation.

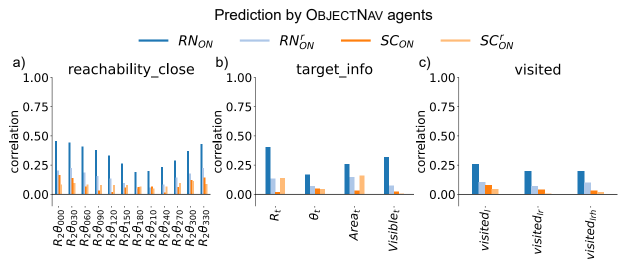

Figure 4. **Metadata prediction by ObjectNav GRU units:** a) Reachability b) Target information c) Visited history

4.1. ObjectNav Models and Baselines

We consider two models for ObjectNav. The first model uses a frozen ResNet18 as the visual encoder and is named RN_{ON}, while the second uses a 5 layer CNN (referred to as SimpleConv) as the visual encoder, denoted by SC_{ON}. In SC_{ON}, the visual encoder is optimized using the gradients of the actor critic loss. The visual representation is concatenated with the goal embedding which is then fed to a GRU. The GRU is connected to two linear layers predicting the policy and value. To ascertain if the representations learned by ObjectNav agents are due to training, we consider two randomly initialized models with the same architetcures as the baselines. For the random ResNet model, named RN^r_{ON}, we initialize ResNet with ImageNet weights and initialize the GRU randomly. For the random SimpleConv model, named SC^r_{ON}, both the visual encoder and GRU are initialized randomly. RN_{ON} and SC_{ON} are trained for 300 Million steps using the default hyperparameters from the AllenAct framework.

4.2. PointNav Models and Baselines

Similar to ObjectNav models we consider a ResNet based model (RN_{PN}) and a SimpleConv based model (SC_{PN}). The distance and orientation to target are used as a sensory input to the model for target information. The corresponding random baselines are named RN^r_{PN} and SC^r_{PN}. RN_{PN} and SC_{PN} are trained for 300 Million steps using the default hyperparameters from AllenAct.

	ObjectNav		**PointNav**	
	ResNet18	SimpleConv	ResNet18	SimpleConv
Trained	RN_{ON}	SC_{ON}	RN_{PN}	SC_{PN}
Random	RN^r_{ON}	SC^r_{ON}	RN^r_{PN}	RN^r_{PN}

4.3. Human Trajectories

After training the ObjectNav and PointNav models, we collect human sampled trajectories for the training and validation rooms. The training trajectories contain 59 episodes with average episode length of 480 while validation trajectories contain 42 episodes with average episode

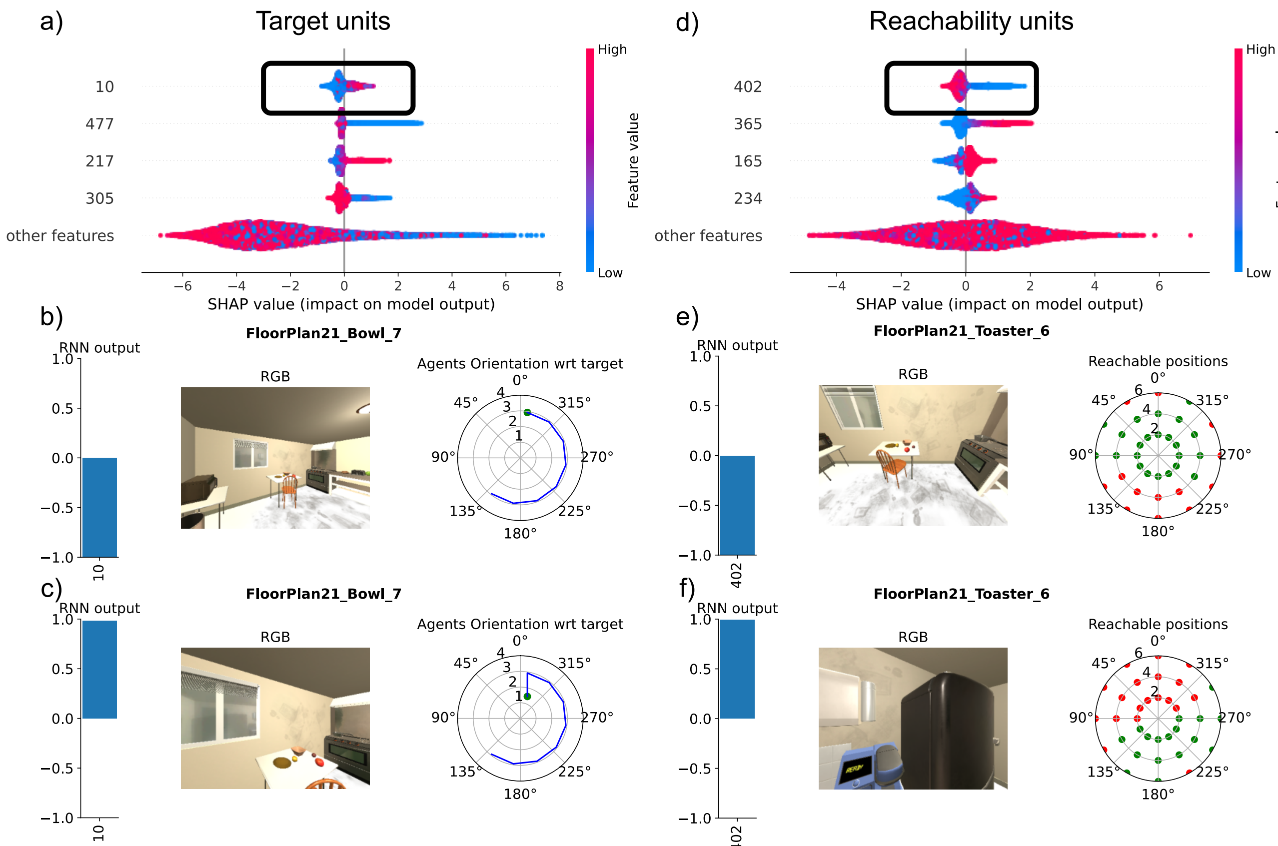

Figure 5. **Visualization of hidden units.** a) Target Visibility Unit: Top-4 most relevant units to predict target distance. b) The bar plot on left shows unit 10's (target unit) response. The center image is agent's current observation. The polar plot on right shows the distance (in meters) and orientation of the agent (in degrees) wrt target. In this case, the agent is at around 3 meters away from the target and is oriented around 0 degrees. The response of unit 10 is negative. c) In this case, the agent is now closer to target (around 1 meters) and unit 10's (target unit) response is positive. d) Reachability Unit: Top-4 most relevant hidden units to predict reachability at distance $2 \times gridSize$ and theta zero. e) The bar plot on the left shows unit 402's (reachability unit) response. The polar plot on right shows if the locations at radii of $2, 4, 6 \times gridSize$ and a given orientation in degrees are reachable or not. In this case, all the locations ahead are green i.e. reachable. The response of unit 402 is negative. f) All the locations ahead are red i.e. not reachable. The response of unit 402 is positive.

length of 470. The subject was encouraged to completely explore the rooms with intentional collisions and visits to previously visited locations with an episode length upper limit of 500. All 8 models are forced to follow these trajectories. The corresponding metadata and GRU activity was extracted resulting in 28,000 training samples and 20,000 validation samples for GBT training.

5. Results

5.1. ObjectNav

The validation performance of ObjectNav models saturates at around 50 million steps, therefore we select a checkpoint right after 50 million steps from both models. RN_{ON} (success = 0.458, SPL = 0.23) significantly outperforms SC_{ON} (success = 0.124, SPL = 0.056). Here success indicates the fraction of episodes the agent successfully reached the target and SPL refers to Success weighted by Path Length introduced in [2]. We consider concepts derived from metadata that are related to target information (R_t,θ_t,$\texttt{visible}_t$,$\texttt{Area}_t$) , reachability ($R_r\theta_{angle}$ where r is the radius and angle is the orientation of the neighboring grid point w.r.t. the agent), agent's information (R_a,θ_a) and visited history ($\texttt{visited}_l$,$\texttt{visited}_{lr}$,$\texttt{visited}_{lrh}$).

Metadata prediction: We train GBTs to predict metadata from the GRU units. We observe that RN_{ON} predicts reachability much better than the other three ObjectNav models (Figure 4a) with a correlation of 0.45 and ROC_AUC=0.75 for reachability in front ($R_2\theta_{000}$). We also observe an interesting pattern that prediction of reachability drops as one moves from 0 (front of the agent) to 180 (behind) degree then it starts increasing from 180 to 330 degrees suggesting the reachability of locations in front is more predictable than behind the agent. In Figure 4a, we show the results for reachability with radius = $2 \times gridsize$. We observe a similar pattern for radius = $4 \times gridsize$ and radius = $6 \times gridsize$ (refer to Appendix B).

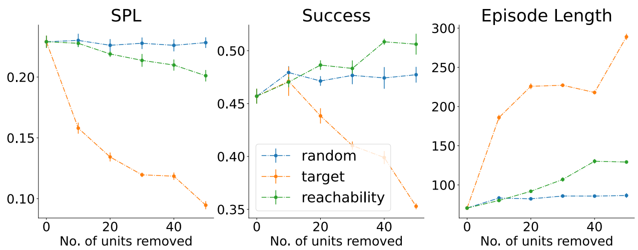

Figure 6. **Impact of removing units from RN_{ON}.**

For the target information (R_t,θ_t,$\texttt{Area}_t$,$\texttt{visible}_t$) RN_{ON} shows a higher correlation than the other three models (Figure 4b). Visited history also ($\texttt{visited}_l$,$\texttt{visited}_{lr}$,$\texttt{visited}_{lrh}$) shows a higher correlation for RN_{ON} (Figure 4c). The agent's information (R_a,θ_a) is not predicted well and RN_{ON} model shows a correlation similar to baselines suggesting this information is not learned by the agent during training (refer to Appendix B). Overall, we observe that RN_{ON} learns the reachability, target relevant information and visited history from OBJECTNAV training. This suggests that these three features are very crucial for performing this task.

While we present the results only on four concepts we also considered collision but found that it was poorly predicted for all 4 models (refer to Appendix B).

Hidden unit visualization: To identify which hidden units are relevant to the mentioned concepts we apply SHAP on the two most interesting concepts ($\texttt{visible}_t$ and $R_2\theta_{000}$). In Figure 5a we show the top-4 units that are most relevant in predicting the target visibility. On observing the SHAP plot of unit 10 (Figure 5a) we see that when the unit's value is higher it has a positive impact on target visibility and vice-versa suggesting that the unit's value is high when the target is visible (for aggregate SHAP values over units see Appendix E). The polar plots show the agent's trajectory (Figure 5 b,c), blue line represents trajectory and green dot indicates the agent's current location wrt target. Bar plot shows the RNN unit's response for current observation. Here, the target is a bowl; when the agent is away from the target its response is negative (Figure 5b) and when it is closer its response is positive (Figure 5c). These results also suggest that this unit might be positively correlated to target visibility.

In Figure 5d, we show the top-4 units most relevant in predicting $R_2\theta_{000}$ (for distribution of aggregate SHAP values over units see Appendix E). On observing the SHAP plot of unit 402 (Figure 5d) we can see that when the unit's value is higher it has negative impact on $R_2\theta_{000}$ and vice-versa suggesting that the unit value is high when the location ahead is not reachable. In Figure 5e,f, the dots are located at $radii = 2, 4, 6 \times stepsize$ from the agent and at angles from 0 to 330 in steps of 30°, where 0 is the front of the agent. Dot color indicates if the location is reachable (green) or not (red). Here, when the location in front of the agent is reachable the unit's response is negative (Figure 5e) and when there is an obstacle in front the unit's response is positive (Figure 5f). These results suggest that this unit might be detecting obstacles ahead.

Unit ablation: While SHAP provides a way to quantify the impact of hidden units on the prediction of a particular metadata concept, it does not imply causality. To identify causality we perform an ablation and measure the impact on the evaluation metrics. We remove units relevant to $\texttt{visible}_t$ and $R_2\theta_{000}$ prediction and measure the impact on the model's performance in terms of SPL, success, and episode length. We compare the ablation results to removing a random selection of units as a baseline. To remove a unit, we set the unit's activity as a constant that is equal to the mean of that unit's activity over the training episodes.

In Figure 6, we observe that removing only 10 target units leads to a huge drop in SPL as compared to removing as many as 50 random units or units encoding reachability. As we remove more target units, the success also begins to drop. This suggests that target units are crucial and removing them first deteriorates the agents ability to identify targets thus leading to longer episodes and low SPL scores and beyond a certain point, the agent ability to be successful is also affected. Removing reachability units also leads to drop in SPL but the impact is not as drastic as in the case of target units. Interestingly removing reachability units lead to increase in success rate potentially due to an increase in exploration. Removing randomly selected units do not significantly impact any of the performance measures.

5.2. POINTNAV

Similar to OBJECTNAV, we choose checkpoints after 50 million steps for our POINTNAV models. RN_{PN} (success = 0.925, SPL = 0.755) and SC_{PN} (success = 0.878, SPL = 0.712) are highly successful at this task. We consider concepts derived from metadata that are related to target information (R_t,θ_t) , reachability ($R_r\theta_{angle}$ where r is the radius and angle is the orientation of the neighboring grid point with respect to the agent), agent's information (R_a,θ_a) and visited history ($\texttt{visited}_l$,$\texttt{visited}_{lr}$,$\texttt{visited}_{lrh}$).

Metadata prediction: We train the GBTs to predict metadata from the GRU units. We first observe from Figure 7a (left) that reachability is predicted at all the angles well. Another interesting thing to note is that models that are not even trained on the POINTNAV task (RN^r_{PN} and SC^r_{PN}) can predict reachability. This result is surprising as compared to OBJECTNAV, where the only model that predicted reachability well was the one that performed well on the OBJECTNAV task (RN_{ON}). Further, RN_{ON} only predicted the reachability in the view of the agent. Our intuition for the above result is that this could be due to additional information from GPS + compass sensor that provides the distance

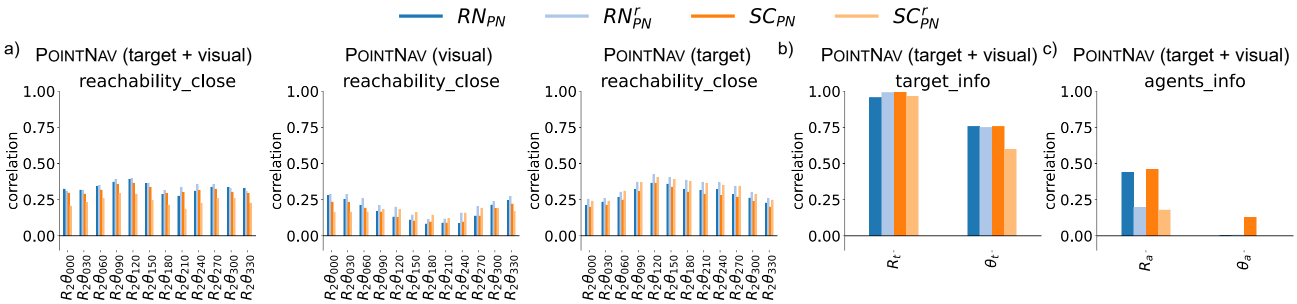

Figure 7. **Metadata prediction by PointNav GRU units:** a) Reachability b) Target information and c) Agent information

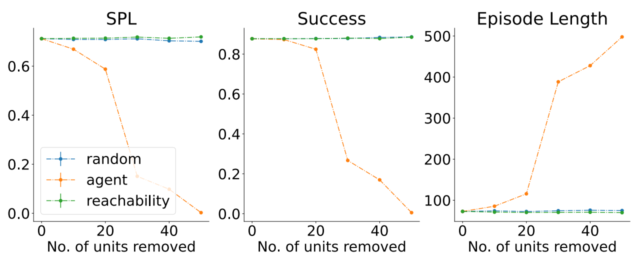

Figure 8. **Impact of removing units from** SC_{PN}**.**

and orientation of the target. To tease apart the prediction due to visual sensor and GPS sensor we perform an ablation study where in one case we replace the output of the GPS sensor by random noise (visual-only; Figure 7a center) and in the other we replace the image with all zeros (only GPS; Figure 7a right).

In the visual-only case, we now observe a pattern similar to ObjectNav where reachability in the field of view is more predictable than out of view. However, it is important to note that prediction of reachability does not improve with training RN_{PN} suggesting that ResNet with ImageNet weights is sufficient to predict reachability required to solve PointNav. SC^r_{PN} however does not seem to predict front reachability ($R_2\theta_{000}$) as effectively as SC_{PN} suggesting that a random initialization is not sufficient to predict reachability required to solve PointNav.

In the target-only case, we observe that the reachability of the backside of the agent is more predictable compared to the angles in the field of view. One possible explanation for this could be that when the distance between target and the agent changes in a given step that means the position at the back was reachable since the agent was there in the previous step. Therefore, using the change in GPS sensor values reachability at back can be predicted in some cases.

The target distance and orientation is predictable when the GPS sensor is available for all the models (Figure 7b and Appendix C). This finding is expected as we provide this information as input, and when the GPS sensor is noise it can not be predicted (refer to Appendix C). Interestingly when the GPS sensor is available (Figure 7c and Appendix C), hidden units in trained PointNav models can predict the distance of the agent (R_a) from the initial spawn location. When using the SHAP method to find the relevant units for predicting R_a, we observe that top most relevant units have a constant value (refer to Appendix D) at almost every step in the episode and show very low variance in its output. On further inspection, we found that the 2 units in top-20 most relevant units for R_a prediction were also relevant for target distance R_t prediction. To predict R_a, GBT might be using a combination of a constant unit(s) and unit that encodes the target information.

Unit ablation: Similar to ObjectNav we perform ablations by removing units and measuring the impact on the metrics. As shown in Figure 8 removing random and reachability units have almost no impact on the performance. Even after removing 50 units we observe similar performance on all three metrics. Removing the units that are relevant for predicting R_a causes a significant drop in the performance and on dropping 50 units both SPL and success rate almost reach zero. The episode length also reaches the highest possible value (500) set in the task definition i.e. the episode ends if agent takes 500 steps. On further inspection, we found that in top-50 R_a units, there are 6 units from the top-50 R_t units. This is the key reason why PointNav performance dropped as the target distance information is lost. We further performed an ablation by removing only these 6 target units, which resulted in a drastic drop.

6. Conclusion

We propose iSee to investigate if concepts about the agent, environment and task are encoded in the hidden representation of embodied agents. While we focus on visual navigation agents trained in AI2-THOR, the framework is generic and can be applied to agents trained on any task in any virtual environment with relevant metadata available. Our analysis shows the ObjectNav agent encodes target orientation, reachability and visited locations history in order to avoid obstacles and visiting the same locations repeatedly. PointNav agents encode target orientation and its progress towards the target and show less reliance on visual information.

References

[1] Treeinterpreter. https://github.com/andosa/treeinterpreter. [Accessed Nov, 2021]. 2

[2] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *arXiv*, 2018. 1, 2, 3, 6

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 1

[4] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 2015. 2

[5] Dhruv Batra, Angel X. Chang, Sonia Chernova, Andrew J. Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, Manolis Savva, and Hao Su. Rearrangement: A challenge for embodied AI. *arXiv*, 2020. 1

[6] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv*, 2020. 2, 3

[7] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, 2017. 1, 2

[8] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *NeurIPS*, 2020. 3

[9] Prithvijit Chattopadhyay, Judy Hoffman, Roozbeh Mottaghi, and Aniruddha Kembhavi. Robustnav: Towards benchmarking robustness in embodied navigation. *ArXiv*, abs/2106.04531, 2021. 3

[10] Changan Chen, Ziad Al-Halah, and Kristen Grauman. Semantic audio-visual navigation. In *CVPR*, 2021. 3

[11] Heming Du, Xin Yu, and Liang Zheng. Learning object relation graph and tentative policy for visual navigation. In *ECCV*, 2020. 3

[12] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Manipulathor: A framework for visual object manipulation. In *CVPR*, 2021. 1

[13] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009. 2

[14] Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *CVPR*, 2018. 1, 2

[15] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In *NeurIPS*, 2019. 2

[16] Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. 2

[17] Yash Goyal, Amir Feder, Uri Shalit, and Been Kim. Explaining classifiers with causal concept effect (cace). *arXiv*, 2019. 2

[18] Unnat Jain, Luca Weihs, Eric Kolve, Ali Farhadi, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander Schwing. A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In *ECCV*, 2020. 1

[19] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *ICML*, 2018. 2

[20] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. 1, 2, 5

[21] Chih kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Pradeep Ravikumar, and Tomas Pfister. On completeness-aware concept-based explanations in deep neural networks. In *NeurIPS*, 2020. 2

[22] Juncheng Li, Xin Wang, Siliang Tang, Haizhou Shi, Fei Wu, Yueting Zhuang, and William Yang Wang. Unsupervised reinforcement learning of transferable meta-skills for embodied navigation. In *CVPR*, 2020. 3

[23] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2020. 1, 2, 4, 11

[24] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NeurIPS*, 2017. 2

[25] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *ICCV*, 2019. 1, 2

[26] Bar Mayo, Tamir Hazan, and Ayellet Tal. Visual navigation with spatial attention. In *CVPR*, 2021. 3

[27] Arsalan Mousavian, Alexander Toshev, Marek Fišer, Jana Košecká, Ayzaan Wahid, and James Davidson. Visual representations for semantic target driven navigation. In *ICRA*, 2019. 3

[28] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *NeurIPS*, 2016. 1, 2

[29] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. 2

[30] Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldermariam. Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models. *arXiv*, 2019. 2

[31] Shivansh Patel, Saim Wani, Unnat Jain, Alexander Schwing, Svetlana Lazebnik, Manolis Savva, and Angel X. Chang. Interpretation of emergent communication in heterogeneous collaborative embodied agents. In *ICCV*, 2021. 2

[32] Santhosh K Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation. In *ECCV*, 2020. 3

[33] Sylvestre-Alvise Rebuffi, Ruth Fong, Xu Ji, and Andrea Vedaldi. There and back again: Revisiting backpropagation saliency methods. In *CVPR*, 2020. 2

[34] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *KDD*, 2016. 2

[35] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017. 2

[36] Lloyd S Shapley. *17. A value for n-person games*. 2016. 3, 4

[37] Bokui Shen, Fei Xia, Chengshu Li, Roberto Mart'in-Mart'in, Linxi (Jim) Fan, Guanzhi Wang, S. Buch, Claudia. Pérez D'Arpino, Sanjana Srivastava, Lyne P. Tchapmi, Micael Edmond Tchapmi, Kent Vainio, Li Fei-Fei, and Silvio Savarese. igibson, a simulation environment for interactive tasks in large realistic scenes. In *IROS*, 2021. 1, 2, 3

[38] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *ICML*, 2017. 2

[39] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv*, 2013. 2

[40] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv*, 2017. 2

[41] Saim Wani, Shivansh Patel, Unnat Jain, Angel X. Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation. In *NeurIPS*, 2020. 3

[42] Donglai Wei, Bolei Zhou, Antonio Torrabla, and William Freeman. Understanding intra-class knowledge inside cnn. *arXiv*, 2015. 2

[43] Luca Weihs, Aniruddha Kembhavi, Kiana Ehsani, Sarah Pratt, Winson Han, Alvaro Herrasti, Eric Kolve, Dustin Schwenk, Roozbeh Mottaghi, and Ali Farhadi. Learning generalizable visual representations via interactive gameplay. In *ICLR*, 2021. 2

[44] Luca Weihs, Jordi Salvador, Klemen Kotar, Unnat Jain, Kuo-Hao Zeng, Roozbeh Mottaghi, and Aniruddha Kembhavi. Allenact: A framework for embodied ai research. *arXiv*, 2020. 5

[45] Erik Wijmans, Abhishek Kadian, Ari S. Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *ICLR*, 2020. 3

[46] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *CVPR*, 2019. 3

[47] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. In *ICLR*, 2019. 3

[48] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectgoal navigation. In *ICCV*, 2021. 2

[49] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv*, 2015. 1, 2

[50] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 2

[51] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. 2015. 2

[52] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017. 3