Online Learning of Reusable Abstract Models for Object Goal Navigation

Tommaso Campari[1,2]   Leonardo Lamanna[2,3]   Paolo Traverso[2]   Luciano Serafini[2]   Lamberto Ballan[1]

[1] University of Padova, Italy   [2] Fondazione Bruno Kessler (FBK), Trento, Italy
[3] University of Brescia, Italy

{tcampari,llamanna,traverso,serafini}@fbk.eu   lamberto.ballan@unipd.it

Abstract

In this paper, we present a novel approach to incrementally learn an Abstract Model of an unknown environment, and show how an agent can reuse the learned model for tackling the Object Goal Navigation task. The Abstract Model is a finite state machine in which each state is an abstraction of a state of the environment, as perceived by the agent in a certain position and orientation. The perceptions are high-dimensional sensory data (e.g., RGB-D images), and the abstraction is reached by exploiting image segmentation and the Taskonomy model bank. The learning of the Abstract Model is accomplished by executing actions, observing the reached state, and updating the Abstract Model with the acquired information. The learned models are memorized by the agent, and they are reused whenever it recognizes to be in an environment that corresponds to the stored model. We investigate the effectiveness of the proposed approach for the Object Goal Navigation task, relying on public benchmarks. Our results show that the reuse of learned Abstract Models can boost performance on Object Goal Navigation.
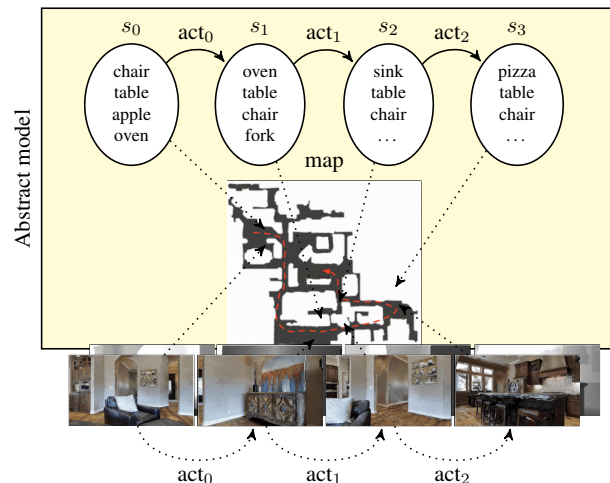
Figure 1. During its navigation in a complex 3D scenario, an agent incrementally acquires knowledge about the environment by storing rich semantic information in an Abstract Model. For instance, when the robot is in s_0, chair and table are visible; by performing act_0, other objects become visible, thus the Abstract Model is updated to s_1. Our work shows how knowledge can be incrementally learned and effectively reused over time.

1. Introduction

In Embodied AI, agent's intelligence emerges from the interaction with the environment as the result of sensorimotor activities [39]. While acting in a real environment, an agent should acquire and effectively represent some knowledge of its surrounding, obtained through sensors (such as RGB or depth cameras). However, this knowledge acquisition process is a key challenge. To this end, two major directions can be followed. On the one hand, knowledge can be codified in a sub-symbolic model (e.g., a neural network), which is learned, for instance, by designing supervised or reinforcement learning techniques that can be directly applied to the sensory data [17, 42]. On the other hand, one can adopt a symbolic/semantic representation of the environment (e.g., by exploiting a semantically rich relational

structure) which captures the high-level critical aspects of the environment, abstracting away useless details [24, 37].

In our work, we follow this second approach, in the attempt of obtaining a more abstract and general knowledge representation that can be, eventually, reused across time. To this end, an agent, such as a robot navigating in a complex scenario, will represent the acquired knowledge of the environment in an Abstract Model that encodes the following key features: (i) some semantic insights about objects, scene elements, and their relations; e.g., it represents a specific state such as "the agent is close to a fridge and a table is visible from that position" (see Fig. 1); (ii) the elements of the Abstract Model are "grounded" to the perceptions; for instance, the agent stores in the Abstract Model some in-

formation of each encountered object, such as its position, the corresponding visual features, etc.; (iii) the Abstract Model is dynamically updated to incorporate the additional information the agent acquires during its operations; i.e., if a new object is discovered, it should be added to the model; (iv) the models learned in the past should be reusable by the agent whenever it recognizes to be in an environment that corresponds to the stored model. This last property is essential because it is here that we can observe the maximum utility of the learned Abstract Model.

In this work we specifically focus on the Object Goal Navigation task [7], in which an agent is asked to go close to an instance of a given object class. Recent approaches often tackle this problem by constructing environment's semantic maps [10, 13] and exploiting SLAM [14, 40]. Instead, we propose to acquire and store the environment knowledge in an abstract, and semantically rich, model. Concretely, such a model is represented by a finite automata whose set of states explicitly describe (at a semantic level) what's an agent views, given a pose. Thus, a state corresponds to an agent pose, a set of object classes (those visible from that pose), and an estimation of the position of each object. We incrementally learn (online) the Abstract Model by navigating the environment, similarly to [12, 13]. The information associated to each abstract state is obtained from the low-level perceptions, acquired from RGB and RGB-D images processed through segmentation models [21] and the Taskonomy model bank [46]. The learned Abstract Model is then stored for future reuse. Therefore, once an agent recognizes that the current environment is similar to one that it has visited before, the proper Abstract Model representing the information previously acquired within the visited environment can be reloaded by the agent, and then updated with the new observations. To implement this feature, we design a "relocation" mechanism that allows the agent to match states of different Abstract Models. We evaluate our approach on the popular Habitat simulator [35], with 3D real environments from the MatterPort3D dataset [11]. In our experiments we focus on the Object Goal Navigation task, and we show that the Abstract Model is helpful to improve the success rate (e.g., avoiding some false positive detections) and the optimality of the planned path.

Summing up, the contributions of this paper are three-fold: (i) the proposed framework allows an agent to incrementally enhance and reuse previously acquired knowledge, relevant to the current environment; (ii) we integrate sub-symbolic techniques such as image processing, path planning, global policy learning, with symbolic reasoning on Abstract Models; (iii) our experimental analysis shows that learning and reusing Abstract Models is an effective way to exploit previously acquired knowledge, obtained from noisy observations (e.g., from inaccurate semantic segmentations), for the Object Goal Navigation Task.

2. Related Works

Embodied AI. In recent years, several large scale datasets for Embodied AI tasks have been presented, such as Matterport3D [11] and Gibson [43]. These datasets contain 3D reconstruction of environments, which enabled the creation of various photorealistic simulators, such as Habitat [35] or GibsonEnv [43]. Thanks to the new experimental setup offered by these environments, nowadays there are numerous exciting tasks, as described in [2]. Some examples are Point Goal [42] (approach a specific point), Object Goal [7, 10] (approach a specific object), and Vision and Language Navigation [3, 32] (follow instructions in natural language).

In this context, the most common approaches are based on Reinforcement Learning models that exploit policies based on RNNs [9, 17, 25, 26, 30, 34]. For example, [26] solves the point goal task by learning a RL policy, using the A3C algorithm [27] and exploiting auxiliary depth prediction and loop closure classification. [30] proposes a Deep RL framework which uses an LSTM-based policy for Object Goal Navigation. [17] proposes Scene Memory Transformer, an attention-based policy [41] that can exploit the least recent steps performed by the agent. In this case, the policy training is performed using the Deep Q-Learning algorithm [28]. Starting from the work done in [46], [36] shows that Mid-Level Vision produces policies that learn faster and generalize better w.r.t. learning from scratch, especially for the point goal navigation task. Previously described approaches require task specific end-to-end training with RL. In contrast our approach is more general, since we train a RL policy to maximize the environment exploration, and can be used to solve different tasks.

Recent works [10, 12, 13] exploit explicit maps constructed from images. Notably, [12] proposes Active Neural SLAM (ANS) which constructs an obstacle map from depth observations. An RL algorithm is then applied on such a map, with the objective of learning a global policy that selects a point, reached via path planning, to maximize the environment exploration. An extension is then proposed in [13], where the occupancy map is enriched with semantic information about objects in the scene. The Global Policy, trained specifically for the Object Goal Navigation task, exploits the semantic information available in the map. Using semantic maps to store information about the environment for future reuse, which is our main objective, requires relocation algorithms, which are not considered in previous approaches. In our approach, instead of relying on relocation w.r.t. semantic maps, we store information of previously visited environments in an Abstract Model, where each state is associated with some visual features sufficient for relocation. Finally, [10] proposes a new method to construct semantic maps that exploit an encoder-decoder model with a Spatial Memory Transformer. The generated semantic maps are then tested for the Object Goal Navigation task.

Furthermore, they propose also an experiment in which pre-computed maps are reused in the Object Goal Navigation task. However, this approach assumes perfect relocation (i.e. the absolute position of the robot is provided).

Learning Abstract Models. Abstract Models learning in planning has the objective of inducing an Abstract Model that describes actions, starting from sequences of observations about their execution. [15, 19] propose to learn action models in a structured language starting from complete observations. [29] learn action models from noisy and incomplete observations. [48] learn an action model on a target domain by transfer learning from a set of source domains and by partial observations. [1] propose a method for learning action models from either complete or incomplete observations. The work by [8], provides a framework for learning first-order symbolic representations from complete information about action execution. In all these approaches, learning is performed from symbolic observations, and sensory perceptions in a continuous environment are not considered. In this work, we also tackle the problem of abstracting the sensory perceptions into a finite set of states.

Alternative approaches learn a discrete Abstract Model from continuous observations. Causal InfoGAN [23] learns discrete or continuous models from high dimensional sequential observations. This approach fixes a priori the size of the discrete domain model. LatPlan [4, 5] takes in input pairs of high dimensional raw data (e.g., images) corresponding to transitions and learn an action model. LatPlan is an offline approach, while our approach instead learns online and without fixing the dimension of the Abstract Model. The work by [24] proposes an online method to learn Abstract Models by mapping continuous perceptions to deterministic state transition systems. With respect to our approach, they requires an input draft Abstract Model, and do not deal with complex perceptions like RGB-D images.

Our approach shares some similarities with the work on planning by reinforcement learning [18, 22, 31, 33, 44], since we learn by acting in the environment. However, these works focus on learning policies and assume the set of states and the correspondence between continuous observations from sensors and states are fixed.

3. Preliminaries

Object Goal Navigation. In the Object Goal Navigation (OGN) task [34], an agent is required to go close to an object of a specific class (such as *fridge* or *bed*) – referred as to *object goal* – starting from a random position within an unknown and static environment, in less than 500 actions. A particular instance ("run") of this task is called *episode*. To reach the object goal, the agent is allowed to execute a set of actions (also called *steps*): namely, `move_forward` (by $25cm$), `turn_left`, `turn_right` (by 30°), `stop`. At

every step, the agent can observe the environment via a set of sensors providing an RGB-D image and the agent pose $\langle x, y, \theta \rangle$, relative to the initial one (which is $\langle 0, 0, 0 \rangle$). The agent ends an episode by executing the `stop` action; if its distance from the closest object goal is less than a threshold (set to $1m$), the episode is considered a success, otherwise a failure. The solution of the OGN task involves multiple challenges. Firstly, the agent must explore the environment in an effective way by exploiting SLAM techniques to learn a map of the environment. Then, it has to recognize new objects in the environment whenever they are into its current view, through object detection. Finally, it must be able to approach the goal objects, by using path planning algorithm to decide which actions it has to execute.

In the standard OGN task each episode is indipendent from the other and no information is transferred across episodes. We call it *memory-less* setting. We also introduce a new setting called *with-memory*, where the agent can exploit the knowledge acquired in previous episodes. In particular, if the agent realizes that is visiting an already visited environment, it can retrieve and exploit the knowledge previously acquired. We believe that the *with-memory* setting is much closer to real scenarios, where an agent should accumulate and reuse previously accumulated knowledge. Notice that the *with-memory* setup introduces new challenges, concerning how and which previously acquired knowledge can be reused in the current situation (relocation and aggregation). Furthermore, in the *with-memory* setup, dealing with previously acquired noisy knowledge is even more challenging, due to the errors accumulation.

Abstract Model. The knowledge of the agent about an environment is represented by a finite state machine in which each states is associated to the "corresponding" visual features. Formally, a finite state machine is a triple $\mathcal{D} = \langle S, A, \delta \rangle$ where S is a finite set of states, A is the set of actions that the agent can execute, and δ is a deterministic transition function among states, i.e, a function $\delta : S \times A \rightarrow S$. Each state $s \in S$ is associated to a triple $\langle \mathcal{F}_s, \mathcal{C}_s, \{\mathcal{F}_{s,c}\}_{c \in \mathcal{C}_s} \rangle$ where $\mathcal{F}_s$ is a set of numeric features associated to the state (i.e. a feature vector extracted from the RGB image); $\mathcal{C}_s$ is a finite set of identifiers for the objects visible by the agent in the state s; for all $c \in \mathcal{C}_s$, $\mathcal{F}_{s,c}$ is a set of real features associated to the view of the object c in the state s (it includes for instance the estimated relative position, the bounding box and a set of visual features).

Since the agent is aware of different environments, it keeps multiple Abstract Models $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(n)}$. We don't assume a one-to-one correspondence between models and environments, since the agent might associate different models to the same environment. For example, the agent could erroneously build two models for the same environment because the second time it enters in the environment,

it does not realize that this has been already visited.

4. Approach

An overview of the main cycle executed at every step by the agent to reach an object goal G is shown in Figure 2. This cycle is composed of three main phases: *1) Knowledge extraction*, *2) Knowledge Modeling*, and *3) Reasoning*. The approach extends [12,13] by allowing the agent to learn Abstract Models and reuse them.

Knowledge Extraction. At every iteration the agent perception is composed of the RGB-D image, corresponding to the egocentric view of the agent, and the agent current relative pose. The Segmentator module [21] extracts object segmentations from the RGB-D image. The Map Builder module creates an egocentric map with a classical SLAM approach [20] from the current depth image and pose. Finally, the State Creator module generates an abstract state $s = \langle \mathcal{F}_s, \mathcal{C}_s, \{\mathcal{F}_{s,c}\}_{c \in \mathcal{C}_s} \rangle$ where: $\mathcal{F}_s$ is the set of state features extracted from the RGB image by the auto-encoder of the Taskonomy model bank [46]; $\mathcal{C}_s$ are the object classes extracted by the Segmentator module; for all $c \in \mathcal{C}_s$, $\mathcal{F}_{s,c}$ contains: the position on the map, the bounding box, and the distance from the agent of every visible object of type c. The object position is estimated by adding the depth value of the bounding box centroid to the agent pose.

Knowledge Modeling. In the Knowledge Modeling phase the environment map and the current Abstract Model are updated with the knowledge extracted in the previous phase. Namely: the current map is extended with the additional information present in the egocentric map, and the state s extracted by the State Creator is added to the current Abstract Model if not present. Finally, the transition function is extended with (s_{prev}, a, s), where s_{prev} is the previous state, and a is the last executed action.

In the *with-memory* setting, if s matches with a state in a previously learned Abstract Model, this model is reloaded and merged with the current one, using the procedure described in Section 4.1.

Reasoning. In the Reasoning phase, given the object goal class G, the agent looks if the current Abstract Model contains a state with an object of type G (i.e., $\exists s \in S$ s.t. $G \in \mathcal{C}_s$). In such a case, the agent selects one object of type G and sets the position of the object as goal point on the map. If the Abstract Model contains multiple states with objects of type G, then the agent ranks these objects according to the number of states from which they are visible, and selects the closest one among the top five. We prefer mostly seen objects in order to mitigate the errors of the Segmentator. Indeed, the more points of view from which an object is detected, the less probable that it is a false positive of the Segmentator. Alternatively, if the Abstract Model

does not contain any state with G (e.g., in exploring a new environment, the agent might not have seen any object of type G), a goal position is computed by our Global Policy, based on [12,13]. Namely, given the current map, the policy seeks for a position on the map that maximizes the environment exploration. Once the target position is set, either by the reasoner or by the Global Policy, the agent computes a plan with a path planner, based on the Fast Marching algorithm [38], to reach the target position, and executes the first action. To compute a plan, all unexplored areas of the map are considered navigable; this enables the agent to discover new scene elements and objects, thus enriching both the map and the Abstract Model.

4.1. Abstract Model Reuse

In the *with-memory* setting the Abstract Model learned at each episode is stored by the agent for future re-use. Therefore, the knowledge of the agent is constituted by n Abstract Models $\{\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(n)}\}$. When the agent starts a new episode, it initializes a new Abstract Model $\mathcal{D}^{(n+1)}$. At every step of this episode, the agent looks if its current state $s = \langle \mathcal{F}_s, \mathcal{C}_s, \{\mathcal{F}_{s,c}\}_{c \in \mathcal{C}_s} \rangle$ matches a state in $\{\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(n)}\}$. Matches between states of different Abstract Models are computed by the cosine distance among the state features $\mathcal{F}_s$. Therefore the best match is computed as follows:

$$s^* = \operatorname*{argmin}_{\substack{s^{(i)} \in S^{(i)} \\ i \in 1, \ldots, n}} cos\_dist(\mathcal{F}_s, \mathcal{F}_{s^{(i)}})$$

where $S^{(i)}$ is the set of states of the i-th model $\mathcal{D}^{(i)}$. If $cos\_dist(s, s^*)$ is lower than a given threshold (an hyperparameter of our model), and s^* is a state of the i-th Abstract Model, then $\mathcal{D}^{(n+1)}$ is merged into $\mathcal{D}^{(i)}$ and the resulting one $\mathcal{D}^{(i)}$ is considered as the current Abstract Model. The resulting model contains all states of the two merged models, and the knowledge is incrementally enhanced through episodes. After such a merging the agent does not look for further matching in the current episode.

Notice that the match could not be perfect since the poses of the robot in the matched states s and s^* might be slightly different. This matching difference can propagate to the object positions recorded in the Abstract Model, thus the agent can rely on wrong information. To prevent these potential errors, we propose two different strategies: namely *hard* and *soft*. In the *hard* strategy, we assume that the matching is always perfect and the agent blindly believes in the matched Abstract Model, i.e., it goes to the goal object position returned by the Model reasoner without looking for other goal objects on its path. In the *soft* strategy, the agent tries to mitigate the effects of non-perfect matches. To do this, the agent looks for the goal object in the area around the goal object position given by $\mathcal{D}$. The dimension of the area around the goal object is proportional to the distance
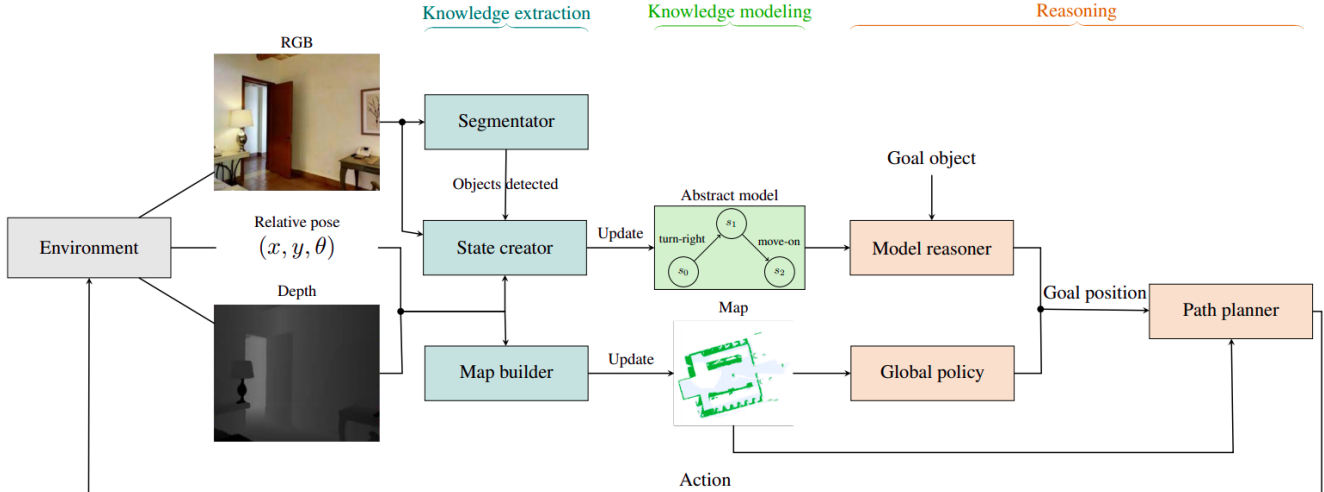
Figure 2. Overview of the proposed architecture for Object Goal Navigation.

among the agent position in the matching state and the goal object one. Moreover, during its path, it continuously looks for a goal object, possibly terminating the episode before reaching the area around the goal object position suggested by the Abstract Model.

4.2. Implementation Details

We used the Habitat Simulator [35] with the Matterport3D dataset [11], which contains 90 different scenes with a total of 194K RGB-D images. Habitat allows to simulate the navigation in these 90 different scenes.

The Global Policy, which select the exploration goal, is trained for 10M steps on the 56 training scenes of Matterport3D (50 as training and 6 as validation) using the Proximal Policy Optimization (PPO) RL algorithm. The Global Policy consists of 5 Conv Layers with the ReLU activations and MaxPooling2D as in [12, 13]. For the Semantic Segmentation, we used the RedNet model [21] pretrained on the 40 classes of Matterport3D (check [45] for details about the performance in the OGN task). The features extractor, which computes $\mathcal{F}_s$ from the RGB data, is the encoder of the Taskonomy model bank [46], and $\mathcal{F}_s$ is a vector with dimension 2048. The cosine distance threshold for states matching is set to 0.3. The SLAM algorithm, which computes the egocentric map, is based on [20].

5. Experiments

In our experiments, we evaluate our approach on the OGN task. Our aim is to show that the *reuse* of previously acquired knowledge, in the form of Abstract Models, can enhance the navigation in existing approaches. Furthermore, we empirically demonstrate our claims with a limitation and failure analysis and a qualitative comparison about reusing *vs* not reusing previously acquired knowledge.

5.1. Evaluation Metrics

The OGN task is evaluated with four standard metrics: the *Success Rate*, the *Success weighted by Path Length (SPL)*, the *SoftSPL*, and the *Distance To Success (DTS)*.

The *Success Rate* is defined as the ratio between the successful and the total number of episodes.

The *SPL* [2] estimates the efficiency of the agent in reaching the goals, and it is defined as:

$$\text{SPL} = \frac{1}{N} \sum_{i=1}^{N} S_i \frac{l_i}{\max(p_i, l_i)}$$

where N is the number of episodes, l_i is the shortest-path distance from the agent's starting position to the closest goal point in an episode i, p_i is the length of the path followed by the agent in the episode i, and S_i is a boolean success indicator of the i-th episode.

The *SoftSPL* [10] is similar to the *SPL*, but measures the path optimality in all the episodes, without penalizing the unsuccessful ones with a zero score; it is defined as:

$$\text{SoftSPL} = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{d_{T_i}}{d_{init_i}} \right) \frac{l_i}{\max(p_i, l_i)}$$

where d_{init_i} is the geodesic distance between the initial position of the agent and the target point, and d_{T_i} is the geodesic distance between the final position of the agent and the goal point. Both refers to the i-th episode.

Finally, the *DTS* measures the mean distance from the closest goal point, mathematically:

$$\text{DTS} = \frac{1}{N} \sum_{i=0}^{N} max(||x_i - g_i||_2 - d, 0)$$

where $||x_i - g_i||_2$ computes the L2 distance for the i-th episode and d is the success threshold $(1m)$.

5.2. Reusing Abstract Models

Here, we report our investigation on different way of reusing knowledge and their advantages. To this aim we have developed the following four models:

Active Neural SLAM (ANS*): it is our implementation of the ANS model [12][1]. This constitutes our baseline, since it does not exploit any previously acquired knowledge.

Hard Pre-explored (ANS*+HP): it is the simplest extension of **ANS***, based on our approach (Section 4). Firstly, the agent is provided with an Abstract Model for each environment. Then these Abstract Models are initialized by performing 10000 exploration steps; for every episode the agent can reuse one of the pre-acquired Abstract Models, using the Hard strategy.

Soft Pre-explored (ANS*+SP): similar to **ANS*+HP**, but the agent reuses the provided Abstract Models by applying the Soft strategy (see Section 4.1).

Soft Incremental (ANS*+SI): this is our "full model". In the first episode, the agent is provided with no Abstract Models; then, in the consecutive steps, the agent can reuse and incrementally extend all the Abstract Models learned in previous episodes, using the Soft strategy.

The fundamental difference between **ANS*/ANS*+SI** and **ANS*+HP/ANS*+SP** is that the formers do not take pre-acquired knowledge, while the latters require such a knowledge. Moreover, **ANS*+SI** is the only model in which the agent extends the Abstract Models with the additional knowledge acquired through episodes. Finally, all the versions but **ANS*** use the *with-memory* setup described in 3.

Table 1 shows the results of our variants on the validation set of Matterport3D, composed of 2195 episodes in 11 environments. This is a standard benchmark for the OGN task [10, 13]. **ANS*+HP** achieves higher results than **ANS***, as expected, since **ANS*+HP** is provided with additional input knowledge. Furthermore, **ANS*+SP** obtains better results than **ANS*+HP**, due to the fact that the Soft strategy mitigates the errors introduced by the matching of Abstract Models in different episodes (Section 4.1). Remarkably, **ANS*+SI** outperforms all the other versions, providing a relative improvement of +8.13% in *success* and +11.9% in *SPL* w.r.t. **ANS*+SP**. From the results of Table 1, we can deduce that the incremental learning of Abstract Models is more effective than providing the agent with the pre-acquired input Abstract Models.

---

[1]We checked the coherence of our implementation by running the same experiments as in [12]; the performance are comparable: namely **ANS [12]** reports 7.056, 0.321 and 0.119 of DTS, Success, and SPL, respectively; our implementation obtained 6.721, 0.313 and 0.127 on the same metrics.

Method	DTS↓	Success↑	SPL↑	SoftSPL↑
ANS*	6.417	0.240	0.102	0.191
ANS*+HP	6.352	0.251	0.105	0.206
ANS*+SP	6.294	0.258	0.117	0.214
ANS*+SI	**6.155**	**0.279**	**0.131**	**0.233**

Table 1. Results achieved by our variants on the Matterport3D validation set.

Method	DTS↓	Success↑	SPL↑
ANS*	6.721	0.313	0.127
ANS*+SI	**6.347**	**0.354**	**0.150**

Table 2. Results obtained on a subset of the validation set of Matterport3D, containing object classes which are in both MS-COCO and MatterPort3D datasets (658 episodes across 11 environments).

Furthermore, the fact that the agent starts in each episode from a different location allows the **ANS*+SI** variant to cover spaces inside the environment that are hardly reachable with a single long pre-exploration. **ANS*+SI** is able to relocate the agent in 69.7% of the episodes, confirming that the method exploited the matching system.

5.3. Effects of Knowledge Accumulation

This section aims to investigate how accumulating knowledge in the Abstract Models affects the agent performance. To do this experiment, we need to limit the quantity of noise recorded in the Abstract Model (e.g., the false positives given from the Semantic Segmentator). Hence, we selected a subset of the Matterport3D validation set on which the Semantic Segmentator is more stable. This subset was built as in [13] and contains episodes with a goal object in one of the following classes: *chair, sofa, plant, bed, toilet, tv, table,* and *sink*.

Table 2 reports the results obtained with **ANS*** and **ANS*+SI**. Our results show that reusing Abstract Models (**ANS*+SI**) allows the agent to take better paths (15% SPL) and brings the agent closer to the goal object (6.34m DTS) compared to the version without knowledge reusage. The analysis on the evolution through episodes of the success rate is visible in Figure 3. Here, we plotted in dashed line the average success rate for each episode across the 11 environments (e.g., point 0 represents the average success rate across the 11 environments in their first episode). The solid thick line represents the moving average of the success rate with a window size of 5. From the figure, it is clear that accumulating knowledge over episodes consistently enhances the success rate.

5.4. Semantic Maps and Abstract Models

Another method to represent knowledge about the environments are Semantic Maps [10]. Semantic maps are obstacle map enriched with information about object classes.
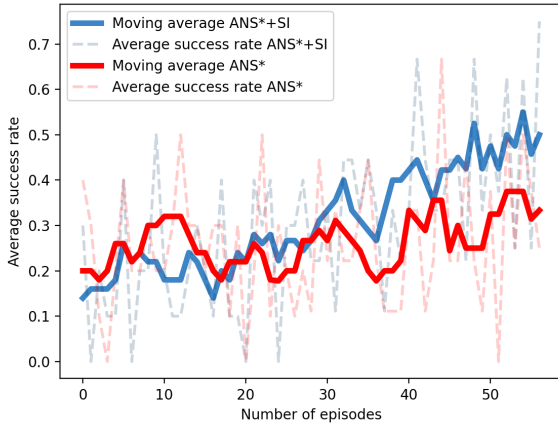
Figure 3. A plot of the moving average success rate in the ANS*+SI model. The window size was set to 5.

Method	DTS↓	Success↑	SPL↑	SoftSPL↑
SMNet [10]	7.316	0.096	0.057	0.087
SMNet (GT)	**5.658**	0.312	**0.207**	**0.282**
ANS*+SI	6.155	0.279	0.131	0.233
SemExp*+SI	5.785	**0.347**	0.151	0.274

Table 3. Results obtained on the validation set of the Matterport3D dataset (2195 episodes across 11 environments). Note that **SMNet (GT)**, as explained in [10], exploits ground truth free space maps extracted directly from the Habitat API.

Here, we compare systems that use Semantic Maps with our method, and investigate how semantic maps and Abstract Models can be suitably combined.

A method for OGN that exploits a precomputed semantic map is **SMNet** [10]. In **SMNet**, the plan to reach an object goal G is obtained by computing the shortest path to an object of type G. The method makes the simplifying assumption that the absolute position of the agent is known, therefore re-location is not needed. [10] also considers a version, called **SMNet(GT)** that assumes ground truth free space maps. Another system that exploits semantic maps is **SemExp** [13]. **SemExp** is an evolution of **ANS** where the base architecture is the same, but the Global Policy takes semantic maps in input. Such a policy seeks to directly find the object goal, instead of maximizing the environment exploration. To understand how semantic maps and Abstract Models can be combined, we integrate our **SI** approach on top of **SemExp**. This version is called **SemExp*+SI**. Table 3 compares all this different versions of reusing knowledge. We used the same split used in [10], which is the validation set of the Matterport3D dataset (2195 episodes across 11 environment). Remarkably, **ANS*+SI** and **SemExp*+SI** outperform **SMNet** by large margin. Furthermore, the Global

Policy exploited in **SemExp*+SI** increases all the metrics w.r.t. **ANS*+SI** counterparts. The higher *success* rate of **SemExp*+SI** w.r.t. **ANS*+SI** ($+6.8\%$) suggests that the way in which the environments are explored plays a crucial role in how the Abstract Models are learned. It's interesting to notice that **SemExp*+SI** has similar performances to **SMNet (GT)**, that exploits pre-computed maps with ground truth free space.

5.5. Limitations and Failure analysis

One of the major limitations in our model comes from the abstraction of input sensory data. The output of the Semantic Segmentator, as well as the visual features stored in the memory, could be affected by errors. This could lead to semantic drift and put a bound on the quality of the knowledge representation. Furthermore, in the *without-memory* setup, the Abstract Model does not provide a significant added value with respect to simpler representations (such as semantic maps). However, this is not the case in the *with-memory* setting, in which the abstraction encoded in the Abstract Models is a cornerstone for the reuse previously acquired knowledge.

Hereafter, we report a failure analysis aiming at understanding why the agent fails in the **SI** setting. In particular, we try to understand quantitatively how the errors introduced by the Semantic Segmentator affect the reliability of the Abstract Models. To this end, we randomly sampled 200 failed episodes in which the Abstract Model was reloaded from the experiment in Table 2. Then we annotated the failures w.r.t. five classes: (i) **Last Mile (Navigation Failure):** the agent correctly navigated to an instance of the goal object but was not able to reach it ($DTS < 2$m); (ii) **Hallucination (Abstract Model Failure):** the agent approached the goal point extracted from the Abstract Model, but there was no goal object occurrence nearby the suggested location; (iii) **Detection (Sensors Failure):** the agent, during its path to the goal point suggested by the Abstract Model, found a wrong instance of the goal object and approached it; (iv) **Exploration (Abstract Model Incompleteness):** the reloaded Abstract Model has no information about possible goal object locations, and the agent could not find any instance in 500 steps; (v) **Misc:** the agent reloaded the Abstract Model but has a generic failure (e.g., agent is trapped by reconstruction debris at spawn).

Notably, we have two possible failure cases directly linked to the reloaded Abstract Model: Exploration, where the agent has not enough information about the environment, and Hallucination, where the agent relies on wrong information. In Figure 4 are reported the statistics about the failed episodes. We can observe how the Abstract Model generated by **SemExp*+SI** gave fewer failures wrt **ANS*+SI** in the Exploration and Hallucination failures, highlighting how the different Global Policy affects the cre-
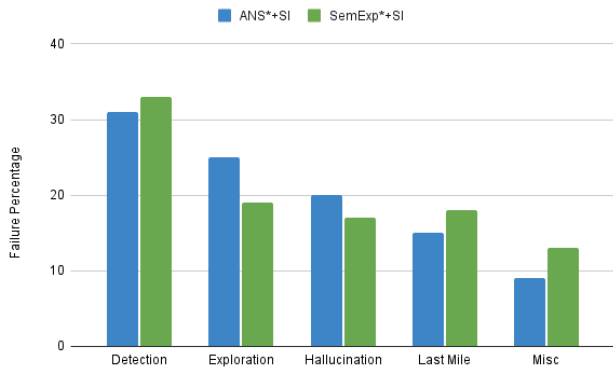
Figure 4. Barplot of failure cases for **ANS*+SI** and **SemExp*+SI** models on the validation set of Matterport3D.

ation of the Abstract Models. Furthermore, the majority of the failures are in Detection for both models. This also suggests that a better reconstruction of the Semantic Segmentations can improve a large margin of performance.

5.6. Qualitative examples

In Figure 5 and 6, we report a qualitative comparison among **ANS*** and **ANS*+SI** in the same episode of the scene *2azQ1b91cZZ* provided in the MatterPort3D dataset.

ANS* starts by exploring the environment, but it never encounters an instance of the Sofa object class. Furthermore, the exploration leads the agent very far from the nearest sofa, more than 10 meters. This is because the environment of the scene *2azQ1b91cZZ* is very large w.r.t. the average dimension of other environments, and the agent is likely to take paths towards areas far from the goal object ones. Therefore, with a limited number of 500 steps, the agent cannot easily find an object when navigating into the environment for the first time. Figure 6, shows the same episode with the exploitation of the incrementally learned Abstract Model. Particularly, at step 15, the agent's state matches a state of the previously learned Abstract Model that, therefore, is reused. The agent's coordinates system is rescaled to match the reused domain one, and the goal point is chosen. Subsequently, starting from the selected goal object position, is created an exploration area that the agent must explore to find the goal object. At step 82, the agent has reached the selected area and consequently approaches the sofa it was looking for.

6. Conclusion and Future Works

This paper presents a novel approach that allows an agent (i) to incrementally acquire and store knowledge about a set of unknown environments, and (ii) to reuse the acquired knowledge, represented as an Abstract Model, when the agent returns to an already visited state. We evaluate
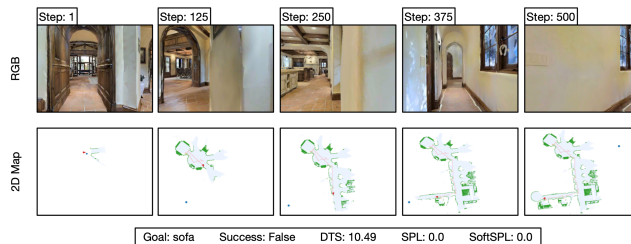

Figure 5. A failed episode with the **ANS*** variant. The agent explored the environment for 500 steps without finding any Sofa occurrence. Green pixels on the map are the obstacles, light blue pixels are explored areas, and the blue point is the goal position.
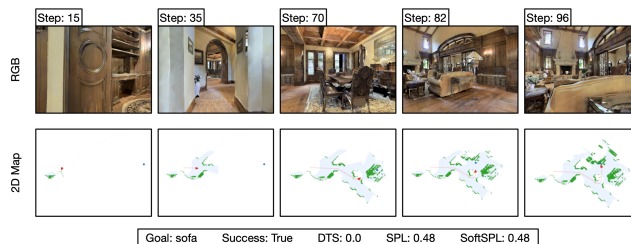

Figure 6. A success with **ANS*+SI** variant. The agent correctly matched its current state with one in the domain and used the information stored to navigate towards a Sofa successfully.

the proposed method on the Object Goal Navigation task. Our experiments show that reusing Abstract Models is effective. An ablation study on different strategies to reuse such knowledge confirmed that incremental learning works better than reusing Abstract Models learned offline. The failure analysis highlights that reusing Abstract Model does not constitute the major reason of failure, which is mostly ascribable to the Semantic Segmentation module. The qualitative analysis highlights how the effect of reusing Abstract Models affect the agent's behaviour. Future works will focus on integrating more semantic information about the environment (e.g., correlation between room types and object types which are present in the room) and tackling other Embodied AI tasks that require reasoning at a symbolic level (e.g., Image Goal Navigation [47], Embodied QA [16], and Rearrangement [6]).

References

[1] Diego Aineto, Sergio Jiménez Celorrio, and Eva Onaindia. Learning action models with minimal observability. *Artificial Intelligence*, 275:104–137, 2019. 3

[2] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 2, 5

[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 *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[4] Masataro Asai. Unsupervised grounding of plannable first-order logic representation from images. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2019. 3

[5] Masataro Asai and Alex Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In *Proc. of the AAAI Conference on Artificial Intelligence*, 2018. 3

[6] Dhruv Batra, Angel X Chang, Sonia Chernova, Andrew J Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, et al. Rearrangement: A challenge for embodied AI. *arXiv preprint arXiv:2011.01975*, 2020. 8

[7] 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 preprint arXiv:2006.13171*, 2020. 2

[8] Blai Bonet and Hector Geffner. Learning first-order symbolic representations for planning from the structure of the state space. In *Proc. of the European Conference on Artificial Intelligence (ECAI)*, 2019. 3

[9] Tommaso Campari, Paolo Eccher, Luciano Serafini, and Lamberto Ballan. Exploiting scene-specific features for object goal navigation. In *Proc. of the European Conference on Computer Vision Workshops*, 2020. 2

[10] Vincent Cartillier, Zhile Ren, Neha Jain, Stefan Lee, Irfan Essa, and Dhruv Batra. Semantic mapnet: Building allocentric semanticmaps and representations from egocentric views. In *Proc. of the AAAI Conference on Artificial Intelligence*, 2021. 2, 5, 6, 7

[11] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *Proc. of the International Conference on 3D Vision (3DV)*, 2017. 2, 5

[12] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural SLAM. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. 2, 4, 5, 6

[13] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2, 4, 5, 6, 7

[14] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[15] Stephen Cresswell, Thomas Leo McCluskey, and Margaret Mary West. Acquiring planning domain models using LOCM. *Knowledge Eng. Review*, 28(2):195–213, 2013. 3

[16] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 8

[17] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[18] Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. Towards deep symbolic reinforcement learning. In *Proc. of Advances in Neural Information Processing Systems Workshops*, 2016. 3

[19] Peter Gregory and Stephen Cresswell. Domain model acquisition in the presence of static relations in the LOP system. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2016. 3

[20] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4, 5

[21] Jindong Jiang, Lunan Zheng, Fei Luo, and Zhijun Zhang. Rednet: Residual encoder-decoder network for indoor rgb-d semantic segmentation. *arXiv preprint arXiv:1806.01054*, 2018. 2, 4, 5

[22] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. 3

[23] Hanard Kurutach, Aviv Tamar, Ge Yang, Stuart Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 3

[24] Leonardo Lamanna, Alfonso Gerevini, Alessandro Saetti, Luciano Serafini, and Paolo Traverso. On-line learning of planning domains from sensor data in pal: Scaling up to large state spaces. In *Proc. of the AAAI Conference on Artificial Intelligence*, 2021. 1, 3

[25] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *Proc. of the AAAI Conference on Artificial Intelligence*, 2017. 2

[26] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016. 2

[27] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proc. of the International Conference on Machine Learning (ICML)*, 2016. 2

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 2

[29] Kira Mourão, Luke S. Zettlemoyer, Ronald P. A. Petrick, and Mark Steedman. Learning STRIPS operators from noisy and incomplete observations. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2012. 3

[30] Arsalan Mousavian, Alexander Toshev, Marek Fišer, Jana Košecká, Ayzaan Wahid, and James Davidson. Visual representations for semantic target driven navigation. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019. 2

[31] Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 1997. 3

[32] Zhiwei Deng Karthik Narasimhan Olga Russakovsky. Evolving graphical planner: Contextual global planning for vision-and-language navigation. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[33] Malcolm R. K. Ryan. Using abstract models of behaviours to automatically generate reinforcement learning hierarchies. In *Proc. of the International Conference on Machine Learning (ICML)*, 2002. 3

[34] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931*, 2017. 2, 3

[35] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2, 5

[36] Alexander Sax, Jeffrey O Zhang, Bradley Emi, Amir R. Zamir, Silvio Savarese, Leonidas Guibas, and Jitendra Malik. Learning to navigate using mid-level visual priors. In *Proc. of the International Conference on Robot Learning (CoRL)*, 2020. 2

[37] Luciano Serafini and Paolo Traverso. Learning abstract planning domains and mappings to real world perceptions. In *Proc. of the International Conference of the Italian Association for Artificial Intelligence (AI*IA)*, 2019. 1

[38] James A Sethian. Fast-marching level-set methods for three-dimensional photolithography development. In *Optical Microlithography IX*, volume 2726, pages 262–272. International Society for Optics and Photonics, 1996. 4

[39] Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial Life*, 11(1–2):13–30, Jan. 2005. 1

[40] Cyrill Stachniss, John J. Leonard, and Sebastian Thrun. Simultaneous localization and mapping. In *Springer Handbook of Robotics*, pages 1153–1176. Springer, 2016. 2

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2

[42] Erik Wijmans, Abhishek Kadian, Ari 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 *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[43] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[44] Fangkai Yang, Daoming Lyu, Bo Liu, and Steven Gustafson. PEORL: integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2018. 3

[45] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectnav. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 5

[46] Amir R. Zamir, Alexander Sax, William Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 4, 5

[47] 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 *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 8

[48] Hankz Hankui Zhuo and Qiang Yang. Action-model acquisition for planning via transfer learning. *Artificial Intelligence*, 212:80–103, 2014. 3