BUILDING GENERALIZABLE AGENTS WITH A REALISTIC AND RICH 3D ENVIRONMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Towards bridging the gap between machine and human intelligence, it is of utmost importance to introduce environments that are visually realistic and rich in content. In such environments, one can evaluate and improve a crucial property of practical intelligent systems, namely *generalization*. In this work, we build *House3D*, a rich, extensible and efficient environment that contains 45,622 human-designed 3D scenes of houses, ranging from single-room studios to multi-storeyed houses, equipped with a diverse set of fully labeled 3D objects, textures and scene layouts, based on the SUNCG dataset (Song et al., 2017). With an emphasis on semantic-level generalization, we study the task of concept-driven navigation, *RoomNav*, using a subset of houses in House3D. In RoomNav, an agent navigates towards a target specified by a semantic concept. To succeed, the agent learns to comprehend the scene it lives in by developing perception, understand the concept by mapping it to the correct semantics, and navigate to the target by obeying the underlying physical rules. We train RL agents with both continuous and discrete action spaces and show their ability to generalize in new unseen environments. In particular, we observe that (1) training is substantially harder on large house sets but results in better generalization, (2) using semantic signals (*e.g.* segmentation mask) boosts the generalization performance, and (3) gated networks on semantic input signal lead to improved training performance and generalization. We hope House3D[1], including the analysis of the RoomNav task, serves as a building block towards designing practical intelligent systems and we wish it to be broadly adopted by the community.

1 INTRODUCTION

Recently, deep reinforcement learning has shown its strength on multiple games, such as Atari (Mnih et al., 2015) and Go (Silver et al., 2016), vastly overpowering human performance. Underlying these big achievements is a well-defined and efficient simulated environment for the agent to freely explore and learn. Till now, many proposed environments have encoded some aspects of human intelligence. This includes 3D understanding (DeepMind Lab (Beattie et al., 2016) and Malmo (Johnson et al., 2016)), real-time strategy decision (TorchCraft (Synnaeve et al., 2016) and ELF (Tian et al., 2017)), fast reaction (Atari (Bellemare et al., 2013)), long-term planning (Go, Chess), language and communications (ParlAI (Miller et al., 2017) and (Das et al., 2017b)).

Nonetheless, it still remains an open problem whether and how the advances of deep reinforcement learning in these simulated environments can be transferred to the real world. Towards this direction, it is of utmost importance to build environments that emulate the real world, with its rich structure, content and dynamic nature. To facilitate learning, these environments should respond in real-time and should provide a large amount of diverse and complex data. While these properties are much desired, they still don't guarantee an important goal of artificial intelligence, namely *generalization*. Generalization is the ability of an agent to successfully accomplish a task in new unseen scenarios. This is of essence for practical applications; for example an in-home robot or a self-driving car trained in a set of houses or cities, respectively, should be easily deployed in a new house or a new city, which can be visually completely different from its training scenarios.

[1] Available at AnonymousURL

While generalization is thought of as being tied to learning, it is undoubtedly bounded by the diversity of the environments an agent is trained in. To facilitate generalization, an environment needs to be able to provide a large amount of data that will allow the agent to test its ability to recognize and act under novel conditions. The unbiased and unrestricted nature of the newly generated environments is necessary in order to verify whether an agent has developed intelligent skills rather than just memorization (overfitting). Note that the connection of generalization and large-scale datasets has shaped the recent advances in image and object recognition (Russakovsky et al., 2015; He et al., 2015).

Contrary to supervised learning where generalization is formally defined and well studied (Vapnik, 2013), in reinforcement learning, generalization is interpreted in a variety of ways. In DeepMind lab (Beattie et al., 2016; Higgins et al., 2017), diversity in the environment is introduced by pixel-level color or texture changes and maze layouts. Tobin et al. (2017) explore pixel-level generalization by introducing random noise to alter object colors. Finn et al. (2017b) study the generalization skills of an agent across similar task configurations while rewards are provided only for a subset of tasks. Pathak et al. (2017) test an agent on more difficult levels of the same game.

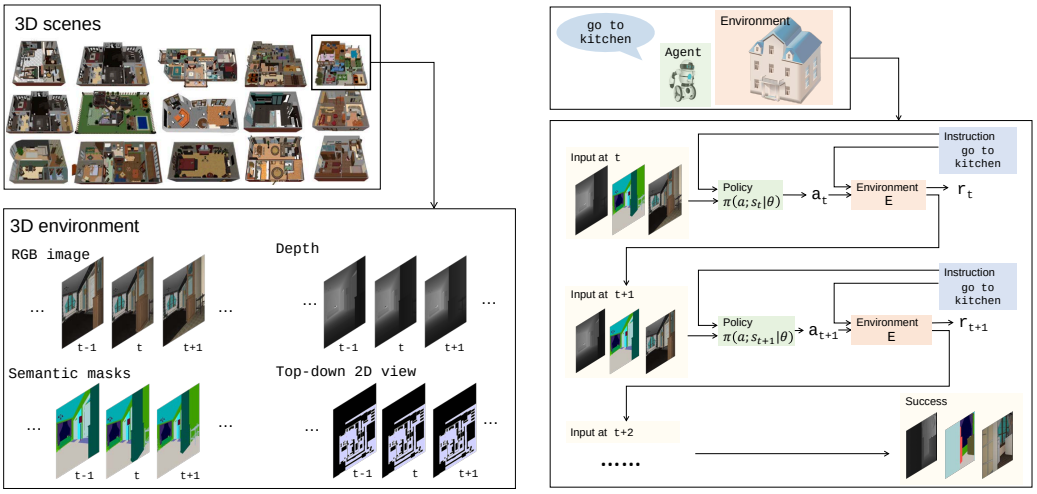
However, pixel-level variations (*e.g.* color and texture of objects) in the environment or levels of difficulty yield very similar visual observations to the agent. In the real world, a human perceives and understands complicated visual signals. For example, a human can easily recognize the kitchen room when visiting a friend's house even if the decoration and design are substantially new. For an agent to succeed in the real world, it needs to interpret novel structure layouts and diverse object appearances. The agent needs to be able to attach *semantic* meaning to a novel scene while generalizing beyond compositional visual changes. In this work, we focus on *semantic-level* generalization in which training and test environments are visually distinct, but share the same high-level conceptual properties. Specifically, we propose *House3D*, a virtual 3D environment consisting of thousands of indoor scenes equipped with a diverse set of scene types, layouts and objects. An overview of House3D is shown in Figure 1a. House3D leverages the SUNCG dataset (Song et al., 2017) which contains 45K *human-designed* real-world 3D house models, ranging from single studios to houses with gardens, in which objects are fully labeled with categories. We convert the SUNCG dataset to an *environment*, House3D, which is efficient and extensible for various tasks. In House3D, an agent can freely explore the space while perceiving a large number of objects under various visual appearances.

In our proposed House3D, we show that the agent can indeed learn high-level concepts and can generalize to unseen scenarios in a new benchmark task, *RoomNav*. In RoomNav, an agent starts at a random location in a house and is asked to navigate to a destination specified by a high-level semantic concept (*e.g.* *kitchen*), following simple physics (*e.g.* no object penetration). An example of RoomNav is shown in Figure 1b. From House3D, 270 houses are manually selected and split into two training sets (20 and 200 houses) and a held-out test set (50 houses) for evaluation. These selected houses are suitable and large enough for navigation tasks. We show that a large and diverse training set leads to improved generalization in RoomNav when using gated-CNN and gated-LSTM policies trained with standard deep reinforcement learning methods, *i.e.* A3C (Mnih et al., 2016) and DDPG (Lillicrap et al., 2015). This is contrary to the small training set regime, where overfitting is prominent. Furthermore, depth information plus semantic signal (*e.g.* segmentation) result in better generalization. The former (depth) facilitates immediate action, while the latter (segmentation) aids semantic understanding in the new environment. This empirical observation indicates the significance of building a practical vision system for real-world robots and also suggests the separation of vision and robotic learning when handling complicated real-world tasks.

The remaining of the paper is structured as follows. Section 2 summarizes relevant work. Section 3 describes our environment, House3D, in detail and section 4 describes the task, RoomNav. Section 5 introduces our gated networks and the applied algorithms. Finally, experimental results are shown in Section 6.

2 Related Work

Environment: The development of environments largely pushes the frontier of reinforcement learning. Table 1 summarizes the attributes of some of the most relevant environments and compares them to House3D. Other than this family of environments closely related to House3D, there are more sim-



(a) House3D environment

(b) RoomNav task

Figure 1: An overview of House3D environment and RoomNav task. **(a)** We build an efficient and interactive environment upon the SUNCG dataset (Song et al., 2017) that contains 45K diverse indoor scenes, ranging from studios to two-storied houses with swimming pools and fitness rooms. All 3D Objects are fully annotated into over 80 categories. Agents in the environment have access to observations of multiple modalities (e.g., RGB images, Depth, Segmentation masks (from object category), top-down 2D view, etc. **(b)** We focus on the task of semantic based navigation. Given a high-level task description, the agent explores the environment to reach the target room.

ulated environments such as OpenAI Gym (Brockman et al., 2016), ParlAI (Miller et al., 2017) for language communication as well as some strategic game environments (Synnaeve et al., 2016; Tian et al., 2017; Vinyals et al., 2017). Most of these environments are pertinent to one particular aspect of intelligence, such as dialogue or a single type of game, and make it hard to facilitate the study of more problems. On the contrary, we focus on building a platform that intersects with multiple research directions, such as object and scene understanding, 3D navigation, embodied question answering (Das et al., 2017a), while allowing users to customize the level of complexity to their needs. Two concurrent works (Brodeur et al., 2017; Savva et al., 2017) introduce a very similar platform to House3D, indicating the interest for large-scale interactive and realistic 3D environments.

We build on SUNCG (Song et al., 2017), a dataset that consists of thousands of diverse synthetic indoor scenes equipped with a variety of objects and layouts. Its visual diversity and rich content naturally allow the study of semantic generalization for reinforcement learning agents. While SUNCG is an appealing 3D scene dataset due its large size and its complex and rich scenes, it is not the only publicly available 3D dataset. Alternative options, yet smaller in size, include Al2-THOR (Zhu et al., 2017), SceneNet RGB-D (McCormac et al., 2017), Stanford 3D (Armeni et al., 2016) and Matterport 3D (Chang et al., 2017).

3D Navigation: There has been a prominent line of work on the task of navigation in real 3D scenes (Leonard & Durrant-Whyte, 1992). Classical approaches decompose the task into two subtasks by building a 3D map of the scene using SLAM and then planning in this map (Fox et al., 2005). More recently, end-to-end learning methods were introduced to predict robotic actions from raw pixel data (Levine et al., 2016). Some of the most recent works on navigation show the effectiveness of end-to-end learning. Gupta et al. (2017) learn to navigate via mapping and planning using shortest path supervision. Sadeghi & Levine (2017) teach an agent to fly using solely simulated data and deploy it in the real world. Dhiraj et al. (2017) collect a dataset of drones crashing into objects and train self-supervised agents on this data in order to avoid obstacles.

A number of recent works also use deep reinforcement learning for navigation in simulated 3D scenes. Mirowski et al. (2016); Jaderberg et al. (2016) improve an agent's navigation ability in mazes by introducing auxiliary tasks. Parisotto & Salakhutdinov (2017) propose a new architecture which stores information of the environment on a 2D map. However, these works only evaluate the

Environment	3D	Realistic	Large-scale	Fast	Customizable
Atari (Bellemare et al., 2013)				•	
OpenAI Universe (Shi et al., 2017)		•	•		•
Malmo (Johnson et al., 2016)	•		•	•	•
DeepMind Lab (Beattie et al., 2016)	•			•	•
VizDoom (Kempka et al., 2016)	•			•	•
AI2-THOR (Zhu et al., 2017)	•	•		•	
Stanford2D-3D (Armeni et al., 2016)	•	•		•	
Matterport3D (Chang et al., 2017)	•	•		•	
House3D	•	•	•	•	•

Table 1: A summary of popular environments. The attributes include **3D**: 3D nature of the rendered objects, **Realistic**: resemblance to the real-world, **Large-Scale**: a large set of environments, **Fast-speed**: fast rendering speed and **Customizable**: flexibility to be customized to other applications.

agent's generalization ability on pixel-level variations or small mazes. We argue that a much richer environment is crucial for evaluating semantic-level generalization.

Gated Modules: In our work, we focus on the task of RoomNav, where the goal is communicated to the agent as a high-level instruction selected from a set of predefined concepts. To modulate the behavior of the agent in RoomNav, we encode the instruction as an embedding vector which gates the visual signal. The idea of gated attention has been used in the past for language grounding (Chaplot et al., 2017), and transfer learning by language grounding (Narasimhan et al., 2017). Similar to those works, we use concept grounding as an attention mechanism. We believe that our gated reinforcement learning models serve as a strong baseline for the task of semantic based navigation in House3D. Furthermore, our empirical results allow us to draw conclusions on the models' efficacy when training agents in a large-scale, diverse dataset with an emphasis on generalization.

Generalization: There is a recent trend in reinforcement learning focusing on the problem of generalization, ranging from learning to plan (Tamar et al., 2016), meta-learning (Duan et al., 2016; Finn et al., 2017a) to zero-shot learning (Andreas et al., 2016; Oh et al., 2017; Higgins et al., 2017). However, these works either focus on over-simplified tasks or test on environments which are only slightly varied from the training ones. In contrast, we use a more diverse set of environments and show that the agent can work well in unseen scenes. House3D provides the agent with scenes that are both visually and structurally different thus forcing it to perceive in order to accomplish its task.

In this work, we show improved generalization performance in complex 3D scenes when using depth and segmentation masks on top of the raw visual input. This observation is similar to other works which use a diverse set of input modalities (Mirowski et al., 2016; Tai & Liu, 2016). Our result suggests that it is possible to decouple real-world robotics from vision via a vision API. The vision API trained in the desired scenes, e.g. an object detection or semantic segmentation system, can bridge the gap between simulated environment and real-world (Tobin et al., 2017; Rusu et al., 2016; Christiano et al., 2016).

3 House3D: An Extensible Environment of 45K 3D Houses

Towards building an ultimately practical AI system, there is a need for a realistic environment that is rich in content and structure, and closely resembles the real world. Such an environment can serve as the testbed for various problems which require visual understanding, language grounding, concept learning and more. For the environment to be of value, it is important that it carries complex visual signals and is of large enough scale to enable semantic learning and generalization. For this, we develop an efficient and flexible environment of thousands of indoor scenes, which we call House3D. An overview of House3D is shown in Figure 1a. Our environment allows the agent to navigate inside complex scenes which consist of a diverse set of layouts, objects and rooms and are accompanied by useful semantic labels.

3.1 DATASET

The 3D scenes in House3D are sourced from the SUNCG dataset (Song et al., 2017), which consists of 45,622 human-designed 3D scenes ranging from single-room studios to multi-floor houses. The SUNCG dataset was designed to encourage research on large-scale 3D object recognition problems and thus carries a variety of objects, scene layouts and structures. On average, there are 8.9 rooms and 1.3 floors per scene with the max being 155 rooms and 3 floors, respectively. There is a diverse set of room and object types in each scene. In total, there are over 20 different room types, such as bedroom, living room, kitchen, bathroom etc., with over 80 different object categories. On average, there are 14 different objects in each room. In total, the SUNCG dataset contains 404,508 different rooms and 5,697,217 object instances drawn from 2644 unique object meshes.

3.2 ANNOTATIONS

SUNCG includes a diverse set of labels for each scene. Based on these labels, at every time step an agent can have access to the following signals in our environment: a) the visual signal of its current first person view, consisting of RGB values, b) semantic segmentation masks for all the objects visible in its current view, and c) depth information. These signals are greatly desired since they enable thorough exploration to determine their practical value for different tasks or can serve as a predictive target when learning models. For example, for the task of navigation one can easily swap the RGB values with depth information or with semantic masks in order to quantify the importance of these different input signals.

Other than the visual input, each scene in SUNCG is fully annotated with the 3D location and size of all rooms and object instances in the form of a 3D bounding box. Rooms and objects are also marked with their type, e.g. bedroom or shoe cabinet respectively. This information allows for a detailed mapping from each 3D location (x, y, z) to an object instance (or None if the space is unoccupied), as well as the room type. Furthermore, more features can be built on top of these existing annotations, such as top-down 2D occupancy maps, connectivity analysis and shortest paths, or any arbitrary physics, all potentially helpful for a variety of applications.

3.3 BUILDING AN INTERACTIVE ENVIRONMENT

3.3.1 RENDERER

To build a realistic 3D environment, we develop a renderer for the 3D scenes from SUNCG, and define actions that obey simple physical rules for an agent to navigate in the scenes. The renderer is based on OpenGL and can run on both Linux and MacOS. The renderer provides RGB images, segmentation masks and depth maps.

As highlighted above, the environment needs to be *efficient* to be used for large-scale reinforcement learning. On a NVIDIA Tesla M40 GPU, our implementation can render 120×90-sized frames at over 600 fps, while multiple renderers can run in parallel on one or more GPUs. When rendering multiple houses simultaneously, one M40 GPU can be fully utilized to render at a total of 1800 fps. The default simple physics, currently written in Python, adds a negligible overhead to the rendering, especially when used with multi-processing. The high throughput of our implementation enables efficient learning for a variety of interactive tasks, such as on-policy reinforcement learning.

The environment along with a python API for easy use is available at `AnonymousURL`.

3.3.2 INTERACTION

In House3D, an agent can live in any location (x, y, z) within a 3D scene, as long as it does not collide with any object instance (including walls) within a small distance range, *i.e.* robot's radius. Doors, gates and arches are considered passage ways for the agent, meaning that an agent can walk through those structures freely. These default design choices add negligible runtime overhead and closely follow the behaviour of a real robot navigating inside a house: a robot can not walk through walls or other objects but can pass freely through free space, including doors and gates, in order to reach different parts of the scene. Note that more complex interaction rules can be incorporated (*e.g.* manipulating objects) within House3D using our flexible API, depending on the task at hand.

4 ROOMNAV: A BENCHMARK TASK FOR CONCEPT-DRIVEN NAVIGATION

To test whether an agent can learn semantic concepts, we consider a natural use case of a home robot in Figure 1b. A human may give a high level instruction to the robot, for example, "Go to the *kitchen*", so that one can later ask the robot to turn on the oven. The robot needs to behave appropriately conditioned on the house it is located in as well as the instruction, *e.g.* the semantic concept "kitchen". Moreover, a practical commercial home robot also needs the ability of *generalization*: the robot can be intensively trained for an arbitrarily long time by its manufacturer but it needs to be immediately and successfully deployed to the user's house, even if the new house is of a completely different design and consists of different objects than the training environments.

The key challenges for building real-world home robots are: (1) understanding of the scene from its visual sensors; (2) execution of high-level conceptual instructions; (3) safe navigation and exploration in complex indoor environments; (4) generalization to unseen scenes under the desired concepts.

To study the aforementioned abilities of an agent, we develop a benchmark task, *Concept-Driven* Navigation (RoomNav), based on our House3D environment. For our experimentation, we restrict the instruction to be of the form "go to `RoomType`", where `RoomType` denotes a pre-defined room type. In RoomNav, the room types define the semantic concepts that an agent needs to interpret across a variety of scenes of distinct visual appearances. To ensure fast experimentation cycles, we perform experiments on a subset of House3D. We manually select 270 houses suitable for a navigation task and split them into a small set (20 houses), a large set (200 houses) and a test set (50 houses), where the test set is used to evaluate the generalization of the trained agents.

4.1 TASK FORMULATION

Suppose we have a set of episodic environments $\mathcal{E} = \{E_1, \ldots, E_n\}$ and a set of semantic concepts $\mathcal{I} = \{I_1, \ldots, I_m\}$. During each episode, the agent is interacting with one environment $E \in \mathcal{E}$ and is given an instruction $I \in \mathcal{I}$. In the beginning of an episode, the agent is randomly placed somewhere in E. At each time step t, the agent receives a visual signal X_t from E via its first person view sensor. Let $s_t = \{X_1, \ldots, X_t, I\}$ denote the state of the agent at time t. The agent needs to propose action a_t to navigate and rotate its sensor given s_t. The environment will give out a reward signal r_t and terminates when it succeeds or times out.

The objective of this task is to learn an optimal policy $\pi(a_t|s_t, I)$ that leads to the targeted room according to I. During training, we train the agent in a training set of environments $\mathcal{E}_{\text{train}}$. For measuring the generalization of the learned policy, we will also evaluate the policy in another set of environments $\mathcal{E}_{\text{test}}$, such that $\mathcal{E}_{\text{test}} \cap \mathcal{E}_{\text{train}} = \emptyset$.

4.2 DETAILS

Here we provide further details for the task, RoomNav. For more specifications see the Appendix.

Environment Statistics: The selected 270 houses are manually verified to be suitable for the task of navigation; *e.g.* they are well connected, contain many targets, and are large enough for exploration (studios excluded). We split them into 3 disjoint sets, denoted by $\mathcal{E}_{small}$, $\mathcal{E}_{large}$ and $\mathcal{E}_{test}$ respectively. For the semantic concepts, we select the five most common room types, *i.e.* kitchen, living room, dining room, bedroom and bathroom. The detailed statistics are shown in Table 2. Note that we define a small set of room types as we emphasize on exploring the problem of concept-driven navigation. One could extend this set to include objects or even subareas within rooms.

Observations: We utilize three different kinds of visual input signals for X_t, including (1) raw pixel values; (2) segmentation mask of the pixel input; and (3) depth information, and experiment with different combinations of them. We encode each concept I with a one-hot vector representation.

Action Space: Similar to existing navigation works, we define a fixed set of actions, here 12 in number including different scales of rotations and movements. Due to the complexity of the indoor scenes as well as the fact that real robots may navigate towards any direction with any (bounded)

	$\lvert\mathcal{E}\rvert$	avg. #targets	kitchen%	dining room %	living room%	bedroom%	bathroom%
$\mathcal{E}_{\text{small}}$	20	3.9	0.95	0.60	0.60	0.95	0.80
$\mathcal{E}_{\text{large}}$	200	3.7	1.00	0.35	0.63	0.94	0.80
$\mathcal{E}_{\text{test}}$	50	3.7	1.00	0.48	0.58	0.94	0.70

Table 2: Statistics of the selected environment sets for RoomNav. `RoomType%` denotes the percentage of houses containing at least one target room of type `RoomType`.

velocity, we also explore a continuous action space similar to Lowe et al. (2017). See the Appendix for more details. In all cases, if the agent hits an obstacle it remains still.

Success Measure and Reward Function: To define *success* for the task, we want to ensure that the agent *identifies* the room due to its unique properties (*e.g.* presence of appropriate objects in the room such as pan and knives for kitchen and bed for bedroom) instead of merely *reaching* there by luck. An episode is considered successful if the agent achieves both of the following two criteria: (1) the agent is located inside the target room; (2) the agent consecutively "sees" a designated object category associated with that target room type for at least 2 time steps. We assume that an agent "sees" an object if there are at least 4% of pixels in X_t belonging to that object.

Regarding the reward function, ideally two signals are enough to reflect the task requirement: (1) a collision penalty when hitting obstacles; and (2) a success reward when completing the instruction. However, this basic reward function makes it too difficult for an RL agent to learn, as the positive reward is too sparse. Thus, during training we resort to an informative reward shaping in order to provide additional supervision: we compute the approximate shortest distance from the target room to each location in the house and adopt the difference of shortest distances between the agent's movement as an additional reward signal. Note that our ultimate goal is to learn a policy that could *generalize* to unseen houses. Our strong reward shaping supervises the agent at train time and is *not available* to the agent at test time. We empirically observe that stronger reward shaping leads to better performances on both training and testing.

5 Gated-Attention Networks for Multi-Target Learning

The RoomNav task can be considered as a multi-target learning problem: the policy needs to condition on both the input s_t and the target I, namely the instruction concept. For policy representations which incorporate the target I, we propose two baseline models with a gated-attention architecture, similar to Dhingra et al. (2016) and Chaplot et al. (2017): a gated-CNN network for continuous actions and a gated-LSTM network for discrete actions. We train the gated-CNN policy using the deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015), while the gated-LSTM policy is trained using the asynchronous advantage actor-critic algorithm (A3C) (Mnih et al., 2016).

5.1 DDPG with gated-CNN policy

5.1.1 Deep deterministic policy gradient

DDPG is a widely used offline-learning algorithm for continuous action space (Lillicrap et al., 2015). Suppose we have a deterministic policy $\mu(s_t|\theta)$ (actor) and the Q-function $Q(s_t, a|\theta)$ (critic) both parametrized by θ. DDPG optimizes the policy $\mu(s_t|\theta)$ by maximizing

$$L_\mu(\theta) = \mathbb{E}_{s_t}\left[Q(s_t, \mu(s_t|\theta)|\theta)\right],$$

and updates the Q-function by minimizing

$$L_Q(\theta) = \mathbb{E}_{s_t,a_t,r_t}\left[\left(Q(s_t, a_t|\theta) - \gamma Q(s_{t+1}, \mu(s_{t+1}|\theta)|\theta) - r_t\right)^2\right]$$

Here, we use a shared network for both actor and critic, which leads to the final loss function $L_{\text{DDPG}}(\theta) = -L_\mu(\theta) + \alpha_{\text{DDPG}} L_Q(\theta)$, where α_{DDPG} is a constant balancing the two objectives.

5.1.2 Gated-CNN network for continuous policy

State Encoding: Given state s_t, we first stack the most recent k frames $X = [X_t, X_{t-1}, \ldots, X_{t-k+1}]$ channel-wise and apply a convolutional neural network to derive an im-

age representation $x = f_{\text{cnn}}(X|\theta) \in \mathbb{R}^{d_X}$. We convert the target I into an embedding vector $y = f_{\text{embed}}(I|\theta) \in \mathbb{R}^{d_I}$. Subsequently, we apply a fusion module $M(x, y|\theta)$ to derive the final encoding $h_s = M(x, y|\theta)$.

Gated-Attention for Feature Fusion: For the fusion module $M(x, y|\theta)$, the straightforward version is concatenation, namely $M_{\text{cat}}(x, y|\cdot) = [x, y]$. In our case, x is always a high-dimensional feature vector (i.e., image feature) while y is a simple low-dimensional conditioning vector (e.g., instruction). Thus, simple concatenation may result in optimization difficulties. For this reason, we propose to use a gated-attention mechanism. Suppose $x \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}^{d_y}$ where $d_y < d_x$. First, we transform y to $y' \in \mathbb{R}^{d_X}$ via an MLP, namely $y' = f_{\text{mlp}}(y|\theta)$, and then perform a Hadamard product between x and $\text{sigmoid}(y')$, which leads to our final gated fusion module $M(x, y|\theta) = x \odot \text{sigmoid}(f_{\text{mlp}}(y|\theta))$. This gated fusion module could also be interpreted as an attention mechanism over the feature vector which could help better shape the feature representation.

Policy Representation: For the policy, we apply a MLP layer on the state representation h_s, followed by a softmax operator (for bounded velocity) to produce the continuous action. Moreover, in order to produce a stochastic policy for better exploration, we apply the Softmax-Gumbel trick (Jang et al., 2016), resulting in the final policy $\mu(s_t|\theta) = \text{Gumbel-Softmax}(f_{\text{mlp}}(h_s|\theta))$. Note that since we add randomness to $\mu(s_t|\theta)$, our DDPG formulation can also be interpreted as the SVG(0) algorithm (Heess et al., 2015).

Q-function: The Q-function $Q(s, a)$ conditions on both state s and action a. We again apply a gated fusion module to the feature vector x and the action vector a to derive a hidden representation $h_Q = M(x, a|\theta)$. We eventually apply another MLP to h_Q to produce the final value $Q(s, a)$.

A model demonstration is shown in the top part of Fig. 2, where each block has its own parameters.

5.2 A3C WITH GATED-LSTM POLICY

5.2.1 ASYNCHRONOUS ADVANTAGE ACTOR-CRITIC

A3C is a variant of policy gradient algorithm introduced by Mnih et al. (2016), which reduces the variance of policy gradient by jointly updating a value function as the baseline. Suppose we have a discrete policy $\pi(a; s|\theta)$ and a value function $v(s|\theta)$. A3C optimizes the policy by minimizing the

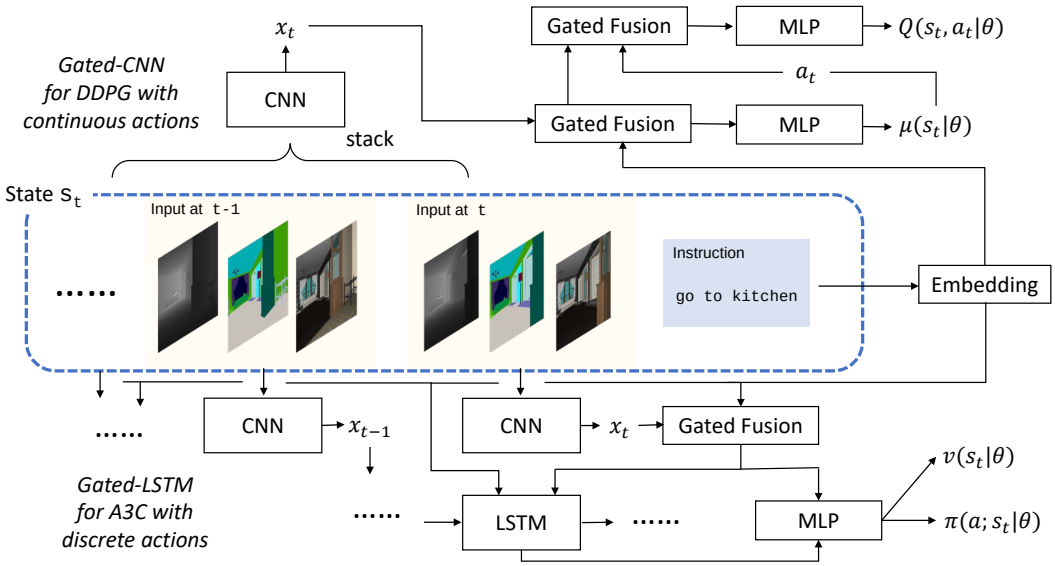


Figure 2: Overview of our proposed models. Bottom part demonstrates the gated-LSTM model for discrete action while the top part shows the gated-CNN model for continuous action. The "Gated Fusion" module denotes the gated-attention architecture.

loss function

$$L_{\text{pg}}(\theta) = -\mathbb{E}_{s_t, a_t, r_t}\left[\sum_{t=1}^{T}(R_t - v(s_t))\log\pi(a_t; s_t|\theta)\right],$$

where R_t is the discounted accumulative reward defined by $R_t = \sum_{i=0}^{T-t}\gamma^i r_{t+i} + v(s_{T+1})$.

The value function is updated by minimizing the loss

$$L_v(\theta) = \mathbb{E}_{s_t, r_t}\left[(R_t - v(s_t))^2\right].$$

Finally the overall loss function for A3C is

$$L_{\text{A3C}} = L_{\text{pg}}(\theta) + \alpha_{\text{A3C}} L_v(\theta)$$

where α_{A3C} is a constant coefficient.

5.2.2 GATED-LSTM NETWORK FOR DISCRETE POLICY

State Encoding: Given state s_t, we first apply a CNN module to extract image feature x_t for each input frame X_t. Similarly, for the target, we apply a gated fusion module to derive a state representation $h_t = M(x_t, I|\theta)$ at each time step t. Then, we concatenate h_t with the target I and the result is fed into the LSTM module (Hochreiter & Schmidhuber, 1997) to obtain a sequence of LSTM outputs $\{o_t\}_t$, so that the LSTM module has direct access to the target other than the attended visual feature.

Policy and Value Function: For each time step t, we concatenate the state vector h_t with the output of the LSTM o_t to obtain a joint hidden vector $h_{\text{joint}} = [h_t, o_t]$. Then we apply two MLPs to h_{joint} to obtain the policy distribution $\pi(a; s_t|\theta)$ as well as the value function $v(s_t|\theta)$.

A visualization of the model is shown in the bottom part of Fig. 2. Note that the parameters of CNN module are shared across time.

6 EXPERIMENTAL RESULTS

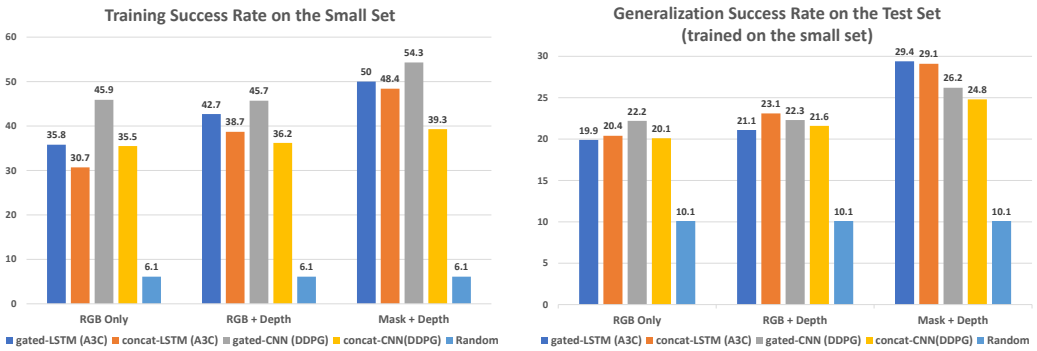
We report experimental results for our models on the task of RoomNav under different sized training sets. We compare models with discrete and continuous action spaces and empirically show the effect of using different input modalities.

6.1 DETAILS

We train our two baseline models on both the small set ($\mathcal{E}_{\text{small}}$, 20 houses) and the large set ($\mathcal{E}_{\text{large}}$, 200 houses), and examine both the training performance (success rate) on the training set as well as the generalization performance on the test set ($\mathcal{E}_{\text{test}}$, 50 houses). All the success rate evaluations use a fixed random seed for a fair comparison. In all the cases, if the agent cannot accomplish the goal within 100 steps[2], we terminate the episode and declare failure. We use gated-CNN and gated-LSTM to denote the networks with gated-attention, and concat-CNN and concat-LSTM for models with simple concatenation. We also experiment with different visual signals to the agents, including RGB image (RGB Only), RGB image with depth information (RGB+Depth) and semantics mask with depth information (Mask+Depth). To preserve the richness of the input visual signal, the resolution of the input image is 120×90.

During each simulated episode, we randomly select a house from the environment set and randomly pick an available target from the house to instruct the agent. Each episode will be forced to terminate after 100 time steps. During training, we add an entropy bonus term for both models in addition to the original loss function. For evaluation, we keep the final model for DDPG due to its stable learning curve, while for A3C, we take the model with the highest training success rate. Optimization is performed via Adam (Kingma & Ba, 2014) while the implementation is done in PyTorch (Paszke et al., 2017). More model and experiment details can be found in appendix.

[2]This is enough for success evaluation. The average number of steps for success runs is much smaller than 100. Refer to appendix B.4 for details.



(a) Training performance

(b) Generalization performance on the test set

Figure 3: Performance of various models trained on $\mathcal{E}_{\text{small}}$ (20 houses) with different input signals: RGB Only, RGB+Depth and Mask+Depth. In each group, the bars from left to right correspond to gated-LSTM, concat-LSTM, gated-CNN, concat-CNN and random policy respectively.

6.2 Training on the small set with 20 houses

Training Performance: For each of the models, we run 2000 evaluation episodes on $\mathcal{E}_{\text{small}}$ and measure overall the success rate. The results are summarized in Fig. 3a. Our empirical results show that the gated models achieve higher success rate than the models without gated-attention architecture. Especially for the DDPG case, the gated-CNN model outperforms the concat-CNN model with a large margin. We believe this is due to the fact that there are two gated-fusion modules utilized in gated-CNN model. Notably, the simpler CNN model with DDPG has stable learning curve and even achieves higher success rate than LSTM with all different input signals, which suggests that simpler CNN models can be an appropriate candidate for fitting a small set of environments.

Regarding the different combinations of input signals, adding a depth channel to the RGB channel generally improves the training performance. Furthermore, changing the RGB signal to semantic signal significantly boosts the performance for all the models.

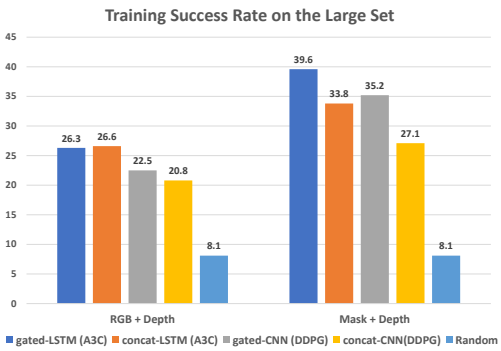
Generalization Performance on $\mathcal{E}_{\text{test}}$: Here, the models trained on $\mathcal{E}_{\text{small}}$ are evaluated on the test set $\mathcal{E}_{\text{test}}$ to measure their generalization ability. We run 2000 episodes and measure the success rate of each model. The results are shown in Fig. 3b.

Regarding different types of visual signals, we draw the same conclusions as at training time: depth improves the performance and semantic information contributes to generalization ability the most. More importantly, we observe that the generalization performance is drastically worse than the training performance, especially for the cases with RGB input: the gated-LSTM models achieve even lower success rate than concat-LSTM models at test time despite the fact that they have much better training performance. This indicates that the neural model tends to *overfit* to the environments with a small training set, while having a semantic input signal could slightly alleviate the issue of overfitting.

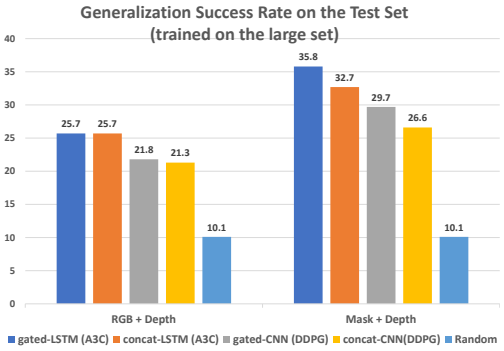
6.3 Training on the large set with 200 houses

Here we train our models on the large set $\mathcal{E}_{\text{large}}$ containing 200 different houses. For visual signals, we focus on "RGB + Depth" and "Mask + Depth". For evaluation, we measure the training success rate over 5000 evaluation episodes and the test success rate over 2000 episodes. Both train and test results are summarized in Fig. 4.

Training Performance: Compared to the performance on the small training set, the training performance on the large set drops significantly, especially for the models using RGB signal. We also notice that in the case of RGB input, gated models perform similar to concat-models. This suggests fundamental difficulties for reinforcement learning agents to learn within a *large* set of diverse and visually rich environments. Whereas, for models with semantic signals, we observe a huge gain on training performance as well as the benefits by having a gated-attention architecture. This implies



(a) Training performance



(b) Generalization performance on the test set

Figure 4: Performance of various models trained on $\mathcal{E}_{\text{large}}$ (200 houses) with input signals of RGB+Depth and Mask+Depth. In each group, the bars from left to right correspond to gated-LSTM, concat-LSTM, gated-CNN, concat-CNN and random policy respectively.

that a semantic vision system can be potentially an effective component for building a real-world robotic system.

In addition, we observe that on the large set, even with a relatively unstable algorithm, such as A3C, the models with larger capacity, i.e. LSTMs, considerably outperform the simpler reactive models, i.e., CNNs. We believe this is due to the larger scale and the high complexity of the training set, which makes it almost impossible for an agent to "remember" the optimal actions for every scenario. Instead, an agent needs to develop high-level abstract strategies, such as exploration, and memory. This also suggest a potential future direction of persuading the agent to learn generalizable abstractions by introducing more inductive bias into the model.

Generalization Performance: Regarding generalization, as we can see from Fig. 4b, after training on a large number of environments, every model now has a much smaller gap between its training performance and test performance. For the models with semantic input signals, their test performance are largely improved compared to those trained on the small set. This again emphasizes the importance of having a large set of diverse environments for training generalizable agents.

Lastly, we also analyze the detailed success rate with respect to each target instruction and provide the results in the appendix.

7 Conclusion

In this paper, we developed a new environment, House3D, which contains 45K of houses with a rich set of objects as well as natural layouts resembling the real-world. House3D is an efficient and extensible environment that could be used for a variety of applications.

Based on our House3D environment, we further introduce a concept-driven navigation task, RoomNav, which tests an agent's intelligence including understanding the given semantic concept, interpreting the comprehensive visual signal, navigation and, most importantly, generalization. In addition to our work and to this date House3D has been used for other tasks, such as question answering (Das et al., 2017a).

We develop two baseline models for RoomNav, gated-CNN and gated-LSTM, with the gated-attention architecture, which show promising results at train and test time. In our experiments, we observe that using the semantic signal as the input considerably enhances the agent's generalization ability. Increasing the size of the training environments is important but at the same time introduces fundamental bottlenecks when training agents to accomplish the RoomNav task due to the higher complexity of the underlying task.

We believe our environment will benefit the community and facilitate the efforts towards building better AI agents. We also hope that our initial attempts towards addressing semantic generalization

ability in reinforcement learning will serve as an important step towards building real-world robotic systems.

REFERENCES

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. *arXiv preprint arXiv:1611.01796*, 2016.

Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. *CVPR*, 2016.

Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, 47:253–279, 2013.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub Strub, Jean Rouat, Hugo Larochelle, and Aaron Courville Courville. HoME: a household multimodal environment. *arXiv 1711.11017*, 2017.

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. *International Conference on 3D Vision (3DV)*, 2017.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. *arXiv preprint arXiv:1706.07230*, 2017.

Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied Question Answering. *arXiv preprint arXiv:1711.11543*, 2017a.

Abhishek Das, Satwik Kottur, Stefan Lee, Jos M.F. Moura, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. *ICCV*, 2017b.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*, 2016.

Gandhi Dhiraj, Pinto Lerrel, and Gupta Abhinav. Learning to fly by crashing. *IROS*, 2017.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017a.

Chelsea Finn, Tianhe Yu, Justin Fu, Pieter Abbeel, and Sergey Levine. Generalizing skills with semi-supervised reinforcement learning. *ICLR*, 2017b.

Dieter Fox, Sebastian Thrun, and Wolfram Burgard. *Probabilistic Robotics*. MIT press, 2005.

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *CVPR*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

Irina Higgins, Arka Pal, Andrei A Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. *arXiv preprint arXiv:1707.08475*, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The Malmo platform for artificial intelligence experimentation. In *IJCAI*, pp. 4246–4247, 2016.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based AI research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pp. 1–8. IEEE, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

John J. Leonard and Hugh F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, Norwell, MA, USA, 1992. ISBN 0792392426.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. SceneNet RGB-D: Can 5m synthetic images beat generic ImageNet pre-training on indoor segmentation? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2678–2687, 2017.

Alexander H Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*, 2017.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy 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.

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.

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 *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. Deep transfer in reinforcement learning by language grounding. *arXiv preprint arXiv:1708.00133*, 2017.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. *arXiv preprint arXiv:1706.05064*, 2017.

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360*, 2017.

Adam Paszke, Sam Gross, and Soumith Chintala. Pytorch, 2017. URL http://pytorch.org/.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *ICML*, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Andrei A Rusu, Matej Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real single-image flight without a single real image. *RSS*, 2017.

Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of Bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pp. 3135–3144, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *CVPR*, 2017.

Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, and Nicolas Usunier. Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint arXiv:1611.00625*, 2016.

Lei Tai and Ming Liu. Towards cognitive exploration through deep reinforcement learning for mobile robots. *arXiv preprint arXiv:1610.01733*, 2016.

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pp. 2154–2162, 2016.

Yuandong Tian, Qucheng Gong, Wenling Shang, Yuxin Wu, and Larry Zitnick. ELF: An extensive, lightweight and flexible research platform for real-time strategy games. *arXiv preprint arXiv:1707.01067*, 2017.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

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 *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3357–3364. IEEE, 2017.

A ROOMNAV TASK DETAILS

The location information of an agent can be represented by 4 real numbers: the 3D location (x, y, z) and the rotation degree ρ of its first person view sensor, which indicates the front direction of the agent. Note that in RoomNav, the agent is not allowed to change its height z, hence the overall degree of freedom is 3.

An action can be in the form of a triple $a = (\delta_x, \delta_y, \delta_\rho)$. After taking the action a, the agent will move to a new 3D location $(x + \delta_x, y + \delta_y, z)$ with a new rotation $\rho + \delta_\rho$. The physics in House3D will detect collisions with objects under action a and in RoomNav, the agent will remain still in case of a collision. We also restrict the velocity of the agent such that $|\delta_x|, |\delta_y| \leq 0.5$ and $|\delta_\rho| \leq 30$ to ensure a smooth movement.

Continuous Action: A continuous action a consists of two parts $a = [m, r]$ where $m = (m_1, \ldots, m_4)$ is for movement and $r = (r_1, r_2)$ is for rotation. Since the velocity of the agent should be bounded, we require m, r to be a valid probability distribution. Suppose the original location of robot is (x, y, z) and the angle of camera is ρ, then after executing a, the new 3D location will be $(x + (m_1 - m_2) * 0.5, y + (m_3 - m_4) * 0.5, z)$ and the new angle is $\rho + (r_1 - r_2) * 30$.

Discrete Action: We define 12 different action triples in the form of $a_i = (\delta_x, \delta_y, \delta_\rho)$ satisfying the velocity constraints. There are 8 actions for movement: left, forward, right with two scales and two diagonal directions; and 4 actions for rotation: clockwise and counter-clockwise with two scales. In the discrete action setting, we do not allow the agent to move and rotate simultaneously.

Reward Details: In addition to the reward shaping of difference of shortest distances, we have the following rewards. When hitting an obstacle, the agent receives a penalty of 0.3. In the case of success, the winning reward is +10. In order to encourage exploration (or to prevent eternal rotation), we add a time penalty of 0.1 to the agent for each time step outside the target room. Note that since we restrict the velocity of the agent, the difference of shortest path after an action will be no more than $0.5 \times \sqrt{2} \approx 0.7$.

B EXPERIMENT DETAILS

B.1 NETWORK ARCHITECTURES

We apply a batch normalization layer after each layer in the CNN module. The activation function used is ReLU. The embedding dimension of concept instruction is 25.

Gated-CNN: In the CNN part, we have 4 convolution layers of 64, 64, 128, 128 channels perspective and with kernel size 5 and stride 2, as well as a fully-connected layer of 512 units. We use a linear layer to transform the concept embedding to a 512-dimension vector for gated fusion. The MLP for policy has two hidden layers of 128 and 64 units, and the MLP for Q-function has a single hidden layer of 64 units.

Gated-LSTM: In the CNN module, we have 4 convolution layers of 64, 64, 128, 128 channels each and with kernel size 5 and stride 2, as well as a fully-connected layer of 256 units. We use a linear layer to convert the concept embedding to a 256-dimension vector. The LSTM module has 256 hidden dimensions. The MLP module for policy contains two layers of 128 and 64 hidden units, and the MLP for value function has two hidden layers of 64 and 32 units.

B.2 TRAINING PARAMETERS

We normalize each channel of the input frame to $[0, 1]$ before feeding it into the neural network. Each of the training procedures includes a weight decay of 10^{-5} and a discounted factor $\gamma = 0.95$.

DDPG: We stack $k = 5$ recent frames and use learning rate 10^4 with batch size 128. We choose $\alpha_{\text{DDPG}} = 100$ for all the settings except for the case with input signal of "RGB+Depth" on $\mathcal{E}_{\text{large}}$, where we choose $\alpha_{\text{DDPG}} = 10$. We use an entropy bonus term with coefficient 0.001 on $\mathcal{E}_{\text{small}}$ and 0.01 on $\mathcal{E}_{\text{large}}$. We use exponential average to update the target network with rate 0.001. A training update is performed every 10 time steps. The replay buffer size is 7×10^5. We run training for 80000 episodes in all. We use a linear exploration strategy in the first 30000 episodes.

A3C: We clip the reward to the range $[-1, 1]$ and use a learning rate $1e-3$ with batch size 64. We launch 120 processes on $\mathcal{E}_{\text{small}}$ and 200 on $\mathcal{E}_{\text{large}}$. During training we estimate the discounted accumulative rewards and back-propagate through time for every 30 time steps unrolled. We perform a gradient clipping of 1.0 and decay the learning rate by a factor of 1.5 when the difference of KL-divergence becomes larger than 0.01. For training on $\mathcal{E}_{\text{small}}$, we use a entropy bonus term with coefficient 0.1; while on $\mathcal{E}_{\text{large}}$, the coefficient is 0.05. α_{A3C} is 1.0. We perform 10^5 training updates and keep the best model with the highest training success rate.

B.3 GENERALIZATION OVER DIFFERENT CONCEPTS

	test succ.	kitchen%	dining room %	living room%	bedroom%	bathroom%
gated-LSTM	35.8	37.9	50.4	48.0	33.5	21.2
gated-CNN	29.7	31.6	42.5	54.3	27.6	17.4

Table 3: Detailed test success rates for gated-CNN model and gated-LSTM model with "Mask+Depth" as input signal across different instruction concepts.

We illustrate in Table 3 the detailed test success rates of our models trained on $\mathcal{E}_{\text{train}}$ with respect to each of the 5 concepts. Note that both models have similar behaviour across concepts. In particular, "dining room" and "living room" are the easiest while "bathroom" is the hardest. We suspect that this is because dining room and living room are often with large room space and have the best connectivity to other places. By contrast, bathroom is often very small and harder to find in big houses.

Lastly, we also experiment with adding auxiliary tasks of predicting the current room type during training. We found this does not help the training performance nor the test performance. We believe it is because our reward shaping has already provided strong supervision signals.

B.4 AVERAGE STEPS TOWARDS SUCCESS

We also measure the number of steps required for an agent in RoomNav. For all the successful episodes, we evaluate the averaged number of steps towards the final target. The numbers are shown in Table 4. A random agent can only succeed when it's initially spawned very close to the target, and therefore have very small number of steps towards target. Our trained agents, on the other hand, can explore in the environment and reach the target after resonable number of steps. Generally, our DDPG models takes fewer steps than our A3C models thanks to their continuous action space. But in all the settings, the number of steps required for a success is still far less than 100, namely the horizon length.

	random	concat-LSTM	gated-LSTM	concat-CNN	gated-CNN
Avg. #steps towards targets on $\mathcal{E}_{\text{small}}$ with different input signals					
RGB+D (train)	14.2	35.9	41.0	31.7	33.8
RGB+D (test)	13.3	27.1	29.8	26.1	25.3
Mask+D (train)	14.2	38.4	40.9	34.9	36.6
Mask+D (test)	13.3	31.9	34.3	26.2	30.4
Avg. #steps towards targets on $\mathcal{E}_{\text{large}}$ with different input signals					
RGB+D (train)	16.0	36.4	35.6	31.0	32.4
RGB+D (test)	13.3	34.0	33.8	24.4	25.7
Mask+D (train)	16.0	40.1	38.8	34.6	36.2
Mask+D (test)	13.3	34.8	34.3	30.6	30.9

Table 4: Averaged number of steps towards the target in all success trials for all the evaluated models with various input signals and different environments.