Language-Based Augmentation to Address Shortcut Learning in Object-Goal Navigation

Dennis Hoftijzer*†, Gertjan Burghouts* , Luuk Spreeuwers†
*TNO, Intelligent Imaging, Den Haag
†University of Twente, EEMCS, DMB, Enschede

***Abstract*—Deep Reinforcement Learning (DRL) has shown great potential in enabling robots to find certain objects (e.g., 'find a bed') in environments like homes or schools. This task is known as *Object-Goal Navigation* (ObjectNav). Although DRL has shown impressive results, the simulators are key and may be biased or limited. This creates a profound risk of *shortcut learning* i.e., learning a policy tailored to specific visual details of training environments. Therefore, in this work, we aim to deepen our understanding of shortcut learning in ObjectNav, its implications and propose a solution. We design an experiment for inserting a shortcut bias in the appearance of training environments. As an example, we associate room types to specific wall colors (e.g., bedrooms have green walls), and observe poor generalization of a SOTA ObjectNav method to environments where this is not the case (e.g., bedrooms now have blue walls). Further analysis shows that shortcut learning is the root cause: the agent learns to navigate to target objects, by simply searching for the associated wall color of the target object's room. To solve this, we propose *Language-Based Augmentation* (L-B). Our key insight is that we can leverage the multimodal feature space of a Vision-Language (V-L) model to augment visual representations directly at the feature-level, requiring no changes to the simulator, and only an addition of one layer to the model. Where the SOTA ObjectNav method's success rate drops 69%, our proposal has only a drop of 23%.**

***Index Terms*—Reinforcement learning, Visual navigation, Vision-Language models, Shortcut learning, Augmentation**

I. Introduction

Autonomously inspecting an industrial site or rapidly retrieving a victim from a disaster area would require a robot to skillfully move through complex three-dimensional (3D) environments. Therefore, the topic of visual navigation in the computer vision community aims at enabling robotic agents to navigate in unseen environments i.e., no ground truth information such as a global geometric map is given, using only visual input from cameras.

Inspired by the success of Deep Reinforcement Learning (DRL) in a variety of Atari games [1], [2], researchers have made great strides in visual navigation by learning embodied agents (or 'virtual robot') to navigate in unseen environments using DRL [3]–[5]. Learning navigation policies requires the use of simulated environments as DRL relies on gathering experience over millions (or billions) of iterations, making it impossible to learn in real-world environments. Contrary, Embodied AI (E-AI) simulators, such as AI2-THOR [6], can leverage parallelization to speed up training, and allow for reproducing experimental setups at low cost. These simulators are accompanied with (near) photo-realistic 3D scene datasets e.g., Gibson [7] and Matterport-3D [8].

Object-goal navigation (ObjectNav) is a visual navigation task where agents are initialized at a random starting pose in an unseen environment and are tasked to navigate to an instance of an object category e.g., 'find a fridge' [9], [10]. ObjectNav not only requires navigation skills such as obstacle avoidance but also semantic reasoning about the environment e.g., 'where is a fridge most likely located?'. Despite good progress over the last few years, state-of-the-art (SOTA) methods continue to generalize poorly to unseen environments [5], [11]–[14]. This poor generalization ability is due, in large part, to limited training data and inflexible E-AI simulators [5], [13].

E-AI simulators tend to be limited by (1) the number of scenes they contain, especially when compared to recent advances in other computer vision tasks [15]–[17], and (2) due to limitations in data collection or rendering. The latter might cause dataset biases: a certain room- or object category might only occur in a single (or few) specific appearance(s). For instance, all kitchens in training scenes might have a certain floor material (e.g., a wooden floor). Consequently, training in E-AI simulators creates a profound risk of *shortcut learning* [18]: learning a simple, non-essential policy, tailored to specific visual details of the simulated environment, rather than learning any semantic reasoning or task-related skills. Efficient object-goal navigation involves learning useful semantic priors such as object-room (e.g., a fridge is in the kitchen) and object-object relations (e.g., chairs are near tables), however can easily lead to unintended shortcuts (e.g., fridge is located near wooden floor) and poor generalization to visually different scenes, where the shortcuts are no longer valid.

In this work, we introduce an *out-of-distribution* (o.o.d.) generalization test for ObjectNav agents to deepen our understanding of shortcut learning in DRL and its implications. In particular, we design a procedure for inserting a dataset bias in the visual appearance of training scenes, which offers the agent a shortcut pathway for finding a given target object. As an example of such a shortcut bias, we associate every room type to a unique wall color i.e., all kitchens have red walls, bedrooms have green walls and so forth.

How well does a SOTA ObjectNav method [19] generalize to scenes with different wall colors and to what extend do shortcuts affect the o.o.d. generalization ability? Using our procedure, we are able to evaluate generalization to scenes with a range of specific wall color changes, where the agents can no longer rely on the inserted shortcut bias. We posit the

target room i.e., the room in which the target object is located, is key. Therefore, we propose to alter the walls of the target room first and incrementally change more rooms. To shed light on the implications of shortcut learning, we differentiate wall color changes of non-target rooms (rooms other than the target room) into two types: *deceptive* vs *nondeceptive*. By moving the wall color associated with the target room to a non-target room, deceptive changes mislead a navigation model to search for the target object in the wrong room. Instead, nondeceptive changes only change the wall color of non-target rooms to a different color than the one associated with the target room. As a result, we find that (1) changing wall colors in testing scenes decreases performance significantly, and (2) shortcut learning plays an important role in the o.o.d. generalization ability of the SOTA ObjectNav method [19]. The agent learns to navigate towards target objects by simply searching for the wall color associated with the target room.

Why does the agent learn such a shortcut strategy? By further analyzing the agent's visual representations, we find that wall color serves as an unintended shortcut predictor for room type. We perform an experiment by training a simple neural classifier on visual representations from sampled frames (RGB observations), outside of the Reinforcement Learning (RL) framework, using supervised learning. We observe the classifier's predictions on held-out combinations of room type–wall color and find it is heavily biased towards wall color.

Beyond observing these shortcuts, we subsequently aim to increase domain generalization without modifying training data or the E-AI simulator, as these can be inflexible and difficult to modify. Augmentation methods e.g., randomizing textures, colors and shapes of objects or environments are commonly used to transfer policies in DRL [13], [20], [21]. However, these methods specifically require modifications to the simulator. While more sophisticated methods for partially editing individual frames during training exist (e.g., text-to-image models [22], [23]), they are slow, computationally expensive and error-prone. Instead, we take a different approach and propose *Language-Based (L-B) augmentation* (see Fig. 1). We augment directly at feature-level, without editing individual frames or requiring any changes to the simulator.

We build upon promising results from [19], where visual representation within the agent's architecture are based on a Vision-Language (V-L) model. They use the image encoder of the Contrastive Language Image Pretraining (CLIP) [15] network to encode RGB observations. CLIP jointly trains an image and text encoder, such that both produce similar representations for visual concepts in images or their names in natural language. Hence, by describing variations of the dataset bias, in natural language, we are able to augment visual representations at feature-level. If the learning algorithm is provided with enough variation during training, the model cannot rely on shortcuts anymore. Similar analysis of visual representations shows increased robustness to changing wall colors. The simple neural classifier is able to classify room types better when trained on our augmented visual representations.

Does the improved room classification, and thus more robust visual representations, also lead to better generalization in ObjectNav? By an elegant modification to a SOTA architecture [19], with only one additional layer, we generalize better to scenes with different wall colors in ObjectNav.

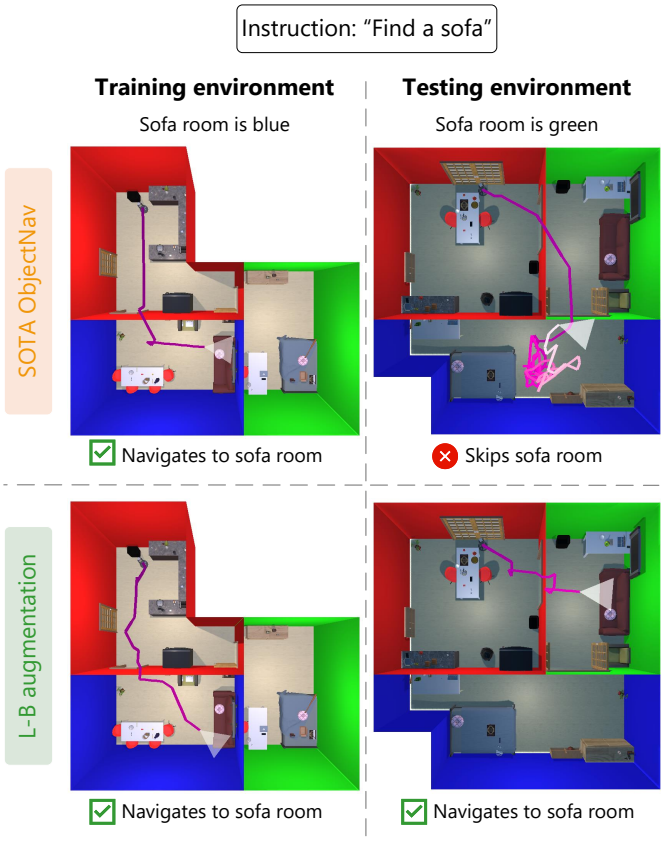


Fig. 1: We propose **Language-Based (L-B) augmentation** to generalize better to scenes with different wall colors. In this example, we interchange the wall color of the bedroom and living room, causing the SOTA objectNav method [19] to look for the sofa in the blue bedroom (wrong). With our augmentations, this is solved.

In summary, our contributions are:

1) An o.o.d. generalization test for ObjectNav to evaluate (1) how well a SOTA ObjectNav method is able to generalize to scenes with different wall colors and (2) the influence of shortcut learning on the generalization ability.
2) An o.o.d. analysis of the agent's visual representations. By training a simple neural classifier on CLIP visual representations, we show wall color serves as a shortcut predictor for room type.
3) L-B augmentation: using our o.o.d. generalization test, we show we are able to mitigate shortcut bias to a large degree without any changes to the E-AI simulator. When only changing wall colors of the target room, the SOTA ObjectNav method's [19] success rate drops 69%. Instead, when integrating L-B augmentations, we observe only 23% relative drop.

II. Related work

A. DRL for visual navigation

There has been a significant amount of research on several visual navigation tasks in recent years, which can be divided according to their target type. For instance, Point-goal Navigation (PointNav), where agents must navigate to a given target coordinate [3], [19], Room-goal Navigation (RoomNav) [24], [25] or more complicated tasks such as Vision-and-Language Navigation (VLN), where an agents must follow a given instruction such as 'turn right and exit the bathroom into the bedroom' [26], [27]. In this work, we focus on the ObjectNav task using DRL [5], [11]–[14], [19], [28]–[31], as this shows the most potential towards enabling agents to navigate to target objects.

E-AI agent architectures consisting of general-purpose neural components such as convolutional neural networks (CNN), recurrent neural networks (RNN) and fully connected layers are a common approach for a variety of visual navigation tasks [3]–[5], [12], [13], [19], [32], [33]. A visual representation is obtained by encoding RGB egocentric observations i.e., first-person camera views of the agent, using a CNN. These representations are then fed into a RNN to provide episodic memory. A linear layer maps the hidden state of the RNN to a probability distribution over a discrete set of actions. We will refer to these as *generic architectures* as they are easily applicable across a range of visual navigation tasks and are easy to modify. In [19], authors explore the effectiveness of CLIP's [15] visual encoder in such a generic architecture. They build simple baselines, named EmbCLIP [19], and show CLIP's visual representations encode useful navigation primitives such as reachability and object localization. They set new SOTA results on several visual navigation tasks, including ObjectNav. We adopt the generic architecture of EmbCLIP as a baseline, given its strong performance on a variety of settings. Our focus is on the challenge of learning a model that can better generalize to changing environments.

B. Embodied AI simulators and scene datasets

Many simulators have been developed for Embodied AI [6], [34]–[39], along with several photo-realistic 3D indoor scene datasets [7], [8], [13], [40], [41]. Scenes can be either reconstructed from 3D scans of real-world houses e.g., Habitat [34] and iGibson [38], or synthetically composed from artist created 3D assets e.g., AI2-THOR [6] (The House Of inteRactions) and variants (RoboTHOR [35], ManipulaTHOR [36]). Both methods have a crucial disadvantage. Namely, they are extremely costly to collect. Reconstructing scenes from 3D scans involves stitching images from specialized cameras whilst manually composing synthetic scenes involves carefully configuring lighting, object placement or textures. ProcTHOR recognizes this fact and instead obtains thousands of training scenes by a procedural generation process [13]. Scenes are interactive, diverse, realistic and as they are procedurally generated, the dataset can be scaled up to an arbitrary amount of scenes. Using this huge dataset, they set new SOTA results on six visual navigation tasks, including ObjectNav, in AI2-THOR, Habitat and RoboTHOR, by pre-training on a generated set of 10,000 houses (dubbed ProcTHOR-10k). In this work, we take advantage of the ProcTHOR-10k scene dataset. The appearance of these scenes can be fully customized by altering the appearance of individual objects and room surfaces such as walls, floors and ceilings. For instance, a red sofa can be replaced with a black one. This allows for (1) inserting a shortcut dataset bias in the appearance of training scenes and (2) creating a range of specific appearance differences between training and testing scenes to evaluate o.o.d. generalization. To ensure we only test for generalization to different wall colors, we deliberately set the appearances of training scenes identical to each other and alter only the color of specific walls in testing scenes. In this way, we can guarantee only wall color differences are influencing the performance of the agent. ProcTHOR-10k is currently the only dataset which allows for such interventions.

C. Shortcut learning

Shortcut learning is emerging as a key impediment in the generalization ability of deep neural networks (DNNs) [18]. Shortcuts are decision rules often learned by DNNs which help to perform well on a particular dataset but do not match with the human-intended ones. Accordingly, they typically fail when tested in only slightly different conditions. Prior work in shortcut learning is predominantly concerned with supervised learning [42]–[44]. In [43], authors show that ImageNet-trained CNNs exhibit a strong bias towards texture based recognition, unlike humans, which mostly rely on object shape. DNNs even exploit implicit shortcuts, which are visually not always observable, such as textures consisting of specific frequencies [44]. Similar to our work, [42] designs a procedure to observe whether DNNs prefer to adopt e.g., color, shape or size shortcuts under fair conditions, and find DNNs naturally prefer certain shortcuts. In contrast, we study the shortcut learning phenomenon in the context of DRL.

In DRL, agents might learn shortcuts due to shortcut biases in the visual appearance of training environments. A common implication is observed in transferring policies from simulation to real-world environments [18], [20], [45]. Most policies trained in simulated environments do not generalize well to the real-world due to the so-called 'reality gap'. Agents adapt to specific visual details of the simulated environment. Prior works cope with the reality gap through randomizing appearances within training environments [20], [21]. Similar data augmentation methods are applied to ObjectNav. ProcTHOR scenes offer a large visual diversity by augmenting e.g., textures and colors of walls, floors, ceilings, and objects [13]. While ProcTHOR shows incredibly powerful results, such augmentations might not be available for all simulators, and more often than not, simulators are more limited than ProcTHOR. Thus, motivating our investigation into augmentations which can be applied post-hoc and not within the training data itself. Moreover, we propose augmentations beyond the already proposed ones, where we use targeted randomization of specific visual biases in environments, in our case, wall color. V-L models e.g., CLIP [15] allows us to augment at feature-level based on prior knowledge of the environment.

III. Preliminaries

A. ObjectNav task definition

The goal for the agent is to navigate through the unseen environment and find the target object within a certain time budget. At the start of an episode, the agent is initialized at a random navigable location within the scene and is given a target object label (e.g. 'Bed') from a predefined set of possible objects. At each time step t, the agent receives an image from the RGB forward-facing camera and can take one of 6 possible actions. We do not utilize any depth sensor readings and a full description of the discrete action space is shown in Table II (Appendix). An episode terminates after 500 timesteps ($T = 500$) or when the agent issues the special DONE action.

B. ObjectNav evaluation metrics

Following standard procedure for ObjectNav [9], [10], an episode is considered successful if (1) the agent executes the DONE action; (2) The target object is within a certain distance threshold, set to 1 meter; and (3) The target object is considered visible i.e., within the camera's field of view and not fully obstructed

We report two primary performance metrics: Success and Success weighted by (normalized inverse) Path Length (SPL). Success is the average success rate over all N evaluation episodes and SPL is a measure for path efficiency [9], [10]:

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

S_i is a binary indicator denoting success of episode i; l_i is the shortest path length from starting position to the target object and p_i is the length of the path the agent travelled. SPL is bounded by $[0:1]$, where 1 is optimal performance. The shortest path length l_i is defined directly until the target object, including the distance threshold. So if the agent acts optimally and stops for example $0.5m$ before the target object, then $l_i > p_i$. To ensure SPL< 1, the denominator computes $max(p_i, l_i)$. It should be noted that SPL is a stringent measure. Achieving an SPL of 1 is infeasible (even for humans) without knowing the target object location a priori.

Additionally, we report two more evaluation measures which give more insight into agent behaviour: Distance To Target (DTT) and Episode Length. DTT is the remaining path length (in meters) of the shortest path to visibly see the target object. This indicates whether the agent close to succeeding in failed episodes and is 0 for successful episodes.

C. ProcTHOR

ProcTHOR enables E-AI to scale by procedurally generating simulated environments. Given a room specification (e.g. a house with 1 bedroom and 1 bathroom), ProcTHOR can produce a large variety of floor plans, populates each floor plan by sampling from a library of 3D assets, and supports randomization of lighting, colors and textures. In this work, we leverage ProcTHOR-10k, as these scenes can be fully customized. This customization and our proposed interventions, allow for inserting a shortcut bias in the appearance of training scenes and test for o.o.d. generalization.

IV. Method

A. Interventions on ProcTHOR-10k

In order to evaluate o.o.d. generalization, we need to (1) insert a shortcut bias and (2) create a range of specific wall color differences between training and testing scenes. By selecting a more uniform subset of scenes and setting identical appearances, we guarantee to only evaluate generalization to different wall colors, without other aspects (object appearances, room types, number of rooms, etc.) influencing the performance. Our procedure is detailed in the next subsections.

1) Scene and target object selection: We start by selecting a more uniform subset of scenes from ProcTHOR-10k. First, we select only houses with 3 rooms, which all consists of 3 room types: kitchen, bedroom and living room. For each room type, we select 3 target object categories which are semantically related (e.g. fridge in kitchen). Table III (Appendix) shows an overview of the selected target object categories. Although the selected target object categories are semantically related to a room type, some might occur in multiple room types (e.g. sofa occurs in living room and bedroom). Also, some selected target object categories might have multiple instances in each scene (e.g. 2 televisions in the living room and 1 in the bedroom). We restrict ourselves to scenes which contain exactly one instance of each target object category in the associated target room e.g., every house contains 1 television and is it positioned in the living room. We ensure this restriction by (1) selecting scenes which contain at least one instance of each target object category in the associated room type and (2) manually removing any double (or more) instances of target objects. Fig. 2 shows an example scene where we removed double instances of target objects.

2) Setting identical appearances: After the above selection, object types still might occur in different 3D assets and room surfaces (walls, floors and ceilings) still appear in many different colors and textures. For our o.o.d. generalization test, we need a set of visually identical houses. To this end, we 1) set the same appearance for each object category e.g., all fridges appear exactly alike, and 2) set equal appearances of room surfaces for each room type e.g., all kitchens have identically colored surfaces. We set the object appearances identical by assigning one 3D asset from the ProcTHOR library to each object type. We set all room surfaces identical by setting the same materials from the ProcTHOR library. Lastly, we set identical appearances of doors, and remove windows and wall decoration.

3) o.o.d. generalization test: In Fig. 3 we show an illustration of how we propose to evaluate generalization to scenes with different wall colors. The train set consists of visually identical houses, where each object type appears exactly alike, and where living rooms have blue walls, kitchen have red walls and bedrooms have green walls. For our test sets, we use houses with a different layout such that agents cannot simply memorize object locations, and permute wall colors. The 0-room test set serves as a reference. To solely evaluate generalization to different wall colors, we use the same layouts in each test set and compare performance to the reference 0-room test set.

Fig. 2: **Example 3-room house.** Here, we show a 3-room house selected from the ProcTHOR 10k train split, where we select 3 target object categories per room type e.g., a sofa and television in the living room, a bed and dresser in the bed room, and a kettle and fridge in the kitchen.

First, we change the wall color of the target object's room as this is the most simple bias (1 wall color change). Next, we change another room's wall color (2 wall color changes). Finally, we change the wall colors of all three rooms (3 wall color changes). For instance, if the target object is a fridge, we start by altering the wall color of the kitchen e.g., from red to green, whereas if the target object is a sofa, we start with changing the wall color of the living room e.g., from blue to red. Fig. 3 shows an example change from red to blue wall colors when changing the target room for 2 different layouts. We consider changing to all possible permutations (e.g. to blue and green wall color in test set 1-room) with repetition i.e., multiple room types can have the same wall color. See Section VI-D (Appendix) for an overview of all possible permutations.

We expect that moving the wall color of the target room to another room i.e., a non-target room, will have a high impact, because the agent may look in that latter, wrong room. It might look in this room because it has the learned color i.e., the color associated with the target room. We refer to this wall color change as 'deceptive'. For instance, a fridge is now erroneously searched in the living room with a red wall, instead of the kitchen which had a red wall during training but is now blue. We expect that this wall color change degrades the performance significantly. Examples of deceptive changes are shown in the bottom row (test set 2- and 3-room). Here, the learned wall color (red) is moved to a non-target room, which we expect will mislead the agent. Instead, when none of the rooms has the learned color, we expect less performance degradation, because it is not misled. We refer to such a wall color change as 'nondeceptive'.

B. Language-based augmentation

We intent to increase domain generalization by augmenting agent's visual representations, such that these are more invariant to changing environments. We integrate our L-B augmentation method in EmbCLIP [19], where agent's visual representation are based on a Vision-Language (V-L) model (Fig. 4 shows an overview). At each time step t, a visual representation or image embedding $\boldsymbol{I_t}$ is obtained by encoding a RGB observation using CLIP's [15] visual encoder (CLIP$_v$). CLIP learns to associate text strings e.g., 'blue wall' or 'red wall', with their visual concepts in images. Our key insight is that we can represent domain specific knowledge, regarding the changes in environment appearances, using natural language. By encoding text descriptions of variations of the dataset bias (e.g. 'a blue wall'), using CLIP's text encoder (CLIP$_T$), we are able to vary visual representations without actually having seen images containing these variations (e.g. an image of a blue wall). This allows us to augment directly at feature-level.

For encoding the text descriptions we use the default text prompt template recommended by [15]: 'a photo of a {label}'. We insert descriptions of variations of the dataset bias (e.g. 'red wall'). We obtain our augmented embeddings $\boldsymbol{I_t^{LB}}$ by computing differences between n encoded text descriptions of variations of the dataset bias $\boldsymbol{T_{1,\ldots,n}}$, and adding to visual representation $\boldsymbol{I_t}$:

$$\boldsymbol{I_t^{LB}} = \boldsymbol{I_t} + \alpha \cdot \Delta(\boldsymbol{T}), \tag{2}$$

$$\Delta(\boldsymbol{T}) = \begin{bmatrix} \boldsymbol{\Delta_1} \\ \boldsymbol{\Delta_2} \\ \vdots \\ \boldsymbol{\Delta_{n(n-1)}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{T_1 - T_2} \\ \boldsymbol{T_1 - T_3} \\ \vdots \\ \boldsymbol{T_n - T_1} \end{bmatrix} \tag{3}$$

where, α controls the degree of augmentation and Δ computes differences of all permutations of length 2 of text descriptions $\boldsymbol{T}$. By randomly sampling an augmented embedding from $\boldsymbol{I_t^{LB}}$ at each time step, we aim to provide the RL model (RNN) with an embedding which resembles the same room type (e.g., a living room in Fig. 4), but with a different wall color (e.g., red and blue instead of green walls in Fig. 4).

We empirically find $\alpha = 50$ to work well by tuning for our specific dataset and shortcut bias (see Appendix, Section VI-C). We standardize features before feeding into the RNN to ensure stability during training (as some features might dominate the loss function due to large norms). In our case, we insert three ($n = 3$) text descriptions of variations of the dataset bias: 'blue wall', 'red wall' and 'green wall'. This results in 6 augmented embeddings $\boldsymbol{I_t} + \boldsymbol{\Delta_{n(n-1)}}$ per visual representation $\boldsymbol{I_t}$.

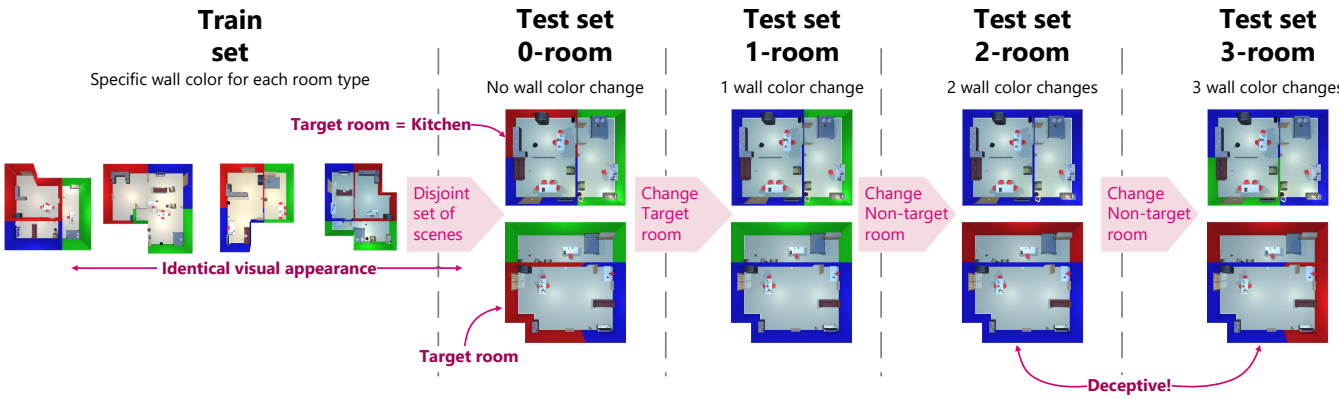


Fig. 3: **Procedure for our o.o.d. generalization test.** In this example, the target room is the kitchen (red walls in test set 0-room). We apply wall color changes to the target room first (test set 1-room) and incrementally alter more rooms (test set 2/3-room). The bottom row shows two examples of deceptive changes, where the wall color associated with the target room (red wall color) is moved to a different room type. The top row only shows nondeceptive wall color changes.

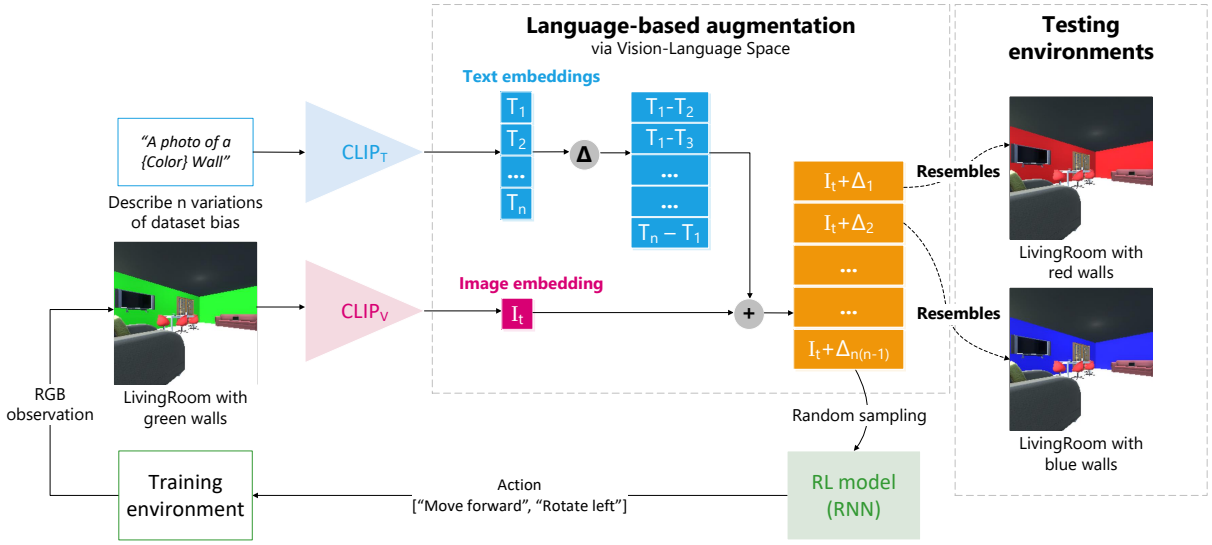


Fig. 4: **Illustration of Language-Based (L-B) augmentation via a vision-language (V-L) space.** Our key insight is that we can augment visual representations (I_t), using differences (Δ) between encoded text descriptions of variations of the dataset bias ($T_{1,\ldots,n}$). The idea is to augment the CLIP embedding of e.g. 'A living room with green walls' in such a way that the embedding resembles a 'living room with red or blue walls'. By providing the RL model with enough variation during training, using random sampling, we ensure the RL model is not able to use a shortcut strategy (Living rooms always have green walls).

V. Experiments

We perform four experiments, where we aim to: (1) evaluate generalization of EmbCLIP [19] to scenes with specific wall color changes and study to what extend shortcuts influence this generalization ability; (2) bring more insight into why the agent learns shortcuts by analysing visual representations within EmbCLIP; (3) show, using similar o.o.d. analysis, that our L-B augmented representations mitigate shortcut bias and finally; (4) validate L-B augmentation can increase domain generalization in ObjectNav by integrating within the architecture of EmbCLIP.

A. Experimental setup

1) ObjectNav dataset details: We train ObjectNav agent on 20 visually identical but biased scenes, generated using our proposed procedure (Section IV-A). See Fig. 13 and 14 (Appendix) for an overview. During training, we randomly sample 1 of 9 target objects. We analyze o.o.d. performance on a set of 5 disjoint scene layouts. We evenly distribute evaluation episodes over the 5 layouts, wall color permutations and target objects. See Section VI-E (Appendix) for more details.

2) Agent architecture and configuration: For our experiments, we use the CLIP-based method EmbCLIP [19]. There are two different variations of the ObjectNav EmbCLIP architecture: a closed-world architecture, which assumes known target objects, and a zero-shot variant. See Section VI-F (Appendix) for a description of both agents. Both are generic architectures, which obtain a visual representation by encoding RGB egocentric views using a frozen CLIP image encoder,

employing a ResNet-50 backbone. However, the closed-world variant first obtains a goal-conditioned embedding before feeding into the RNN, which involves removing the final pooling and fully connected layers from CLIP, whereas the zero-shot variant feeds the CLIP 1024-dimensional visual embedding directly. For our o.o.d. generalization test, we adopt EmbCLIP's closed-world architecture as this variant has better performance in a traditional ObjectNav setting [9]. We integrate our L-B augmentation method in the zero-shot variant, as this architecture feeds the 1024-dimensional CLIP visual embedding directly into the RNN, which allows for substituting this visual embedding with a L-B augmented embedding using random sampling.

Following typical setup for ObjectNav [13], [30], [46], the embodied agent approximately matches a LoCoBot[1]. This robot has a height of $0.88m$, radius of $0.18m$ and its RGB sensor is placed $0.88m$ from the ground. We set the horizontal camera field of view to $90°$. The robot uses a step size of $0.25m$ and a turning angle of $30°$.

3) Reward setting: At each time step t, the reward r_t is:

$$r_t = max(0, min\Delta_{0:t-1} - \Delta_t) + r_{slack} + r_{succ} \quad (4)$$

where:

- $min\Delta_{0:t-1}$ is the minimal path length from the agent to the target object that the agent has previously observed during the episode, so during time steps $\{0, ..., t-1\}$.
- Δ_t is the current path length from the agent to the target.
- r_{slack} is the slack penalty. It is set to -0.01.
- r_{succ} is a large reward for successful episodes. If the agent issues the DONE action and the episode is considered successful, $r_{succ} = 10$, otherwise it is 0.

The reward is shaped to optimize path efficiency and, therefore, SPL. The agent is encouraged to reach target objects using as few actions as possible, due to the slack penalty. Note that, moving towards the target object is encouraged by a reward of maximally the step size and moving away from the target object is not discouraged. The latter is to ensure exploration is not discouraged.

4) Implementation details: We train ObjectNav agents using the Allenact framework [47] and render frames at 224×224 resolution. To parallelize training, we use DD-PPO [3] with 2 workers on 2 Nvidia GeForce GTX 3080 Ti graphic cards (1 worker per GPU). Each worker collects in total 3840 frames of experience from 20 agents in 20 environment instances, all running in parallel. After each rollout (3840 frames), the model is updated using 4 epochs of PPO [48] in a single global batch size of 7680. We perform validation every 200,000 frames and report results of the checkpoint with the highest SPL. See Appendix (SectionVI-G) for additional training details.

B. ObjectNav o.o.d. generalization test

How well does a SOTA ObjectNav method generalize to scenes with different wall colors and to what extend do shortcuts affect the o.o.d. generalization ability? We seek answers by training EmbCLIP [19] for 20M frames (60 GPU-hours) on the 20 visually identical but biased scenes. We hypothesise the performance drop coheres with the number of wall color changes. Moreover, we posit a deceptive change will lead to more performance degradation than a nondeceptive change as the agent will be misled to search for the target object in the wrong room. In Fig. 5 we report quantitative results of our o.o.d. generalization test. Results for each target object are shown in the Appendix (Fig. 12). Next, we describe our findings.

First of all, we observe that changing the wall colors of only the target room already leads to a large decrease in performance. On average (blue mean bar), we observe a 67% relative drop in SPL ($0.39 \rightarrow 0.13$) and 56% in success rate ($68\% \rightarrow 30\%$) going from 0 wall color changes to 1 wall color change, with even lower mean performance for more wall color changes. Indeed, we find that EmbCLIP generalizes poorly to scenes with different wall colors when using limited training data.

Secondly, the Success, SPL and DTT metrics indicate deceptive changes cause lower performance than nondepective changes for both 2 and 3 wall color changes. Interestingly, however, we observe shorter episode lengths for deceptive changes. We conjecture that due to deceptive changes, the agent directly navigates towards the learned color of the target room, which is now placed in a non-target room, without exploring any other rooms. The agent will erroneously search this non-target room, but can not find the target object, and terminates the episode. In contrast, an agent will explore the entire scene when wall colors have only been changed nondeceptively, leading to higher success rate but longer episodes. We show a qualitative example of this behaviour in Fig. 6. In this example, the target object is a bed, though the agent is deceived by the living room, which now has a green wall color, and terminates the episode when it sees the sofa. This leads to a much shorter episode length than in the nondeceptive example, where the agent explores large parts of the scene.

Thirdly, we observe that the drop in mean performance coheres with the number of wall color changes. However, somewhat surprisingly, we also find that altering the wall colors of non-target rooms (2/3 wall color changes) only has an impact on performance depending on the type of color change (deceptive/nondeceptive). Notice how performance remains constant when making only nondeceptive changes, irrelevant of the number of changes (SPL and success of nondeceptive changes is approximately equal to 1 wall color change). Instead, deceptive changes cause lower performance than only altering the target room's walls (1 wall color change). The underlying reason for the decrease in mean performance is a growing share of deceptive changes within all evaluation episodes ($50\% \rightarrow 75\%$ going from 2 to 3 changes). As such, increasing the number of rooms affected by wall color changes does not necessarily decrease performance, but making a deceptive change does.

Evidently, the agent has learned a shortcut strategy i.e., simply navigating towards the wall color of the target room to find objects. Hence, we find shortcut learning plays a crucial role in our ObjectNav o.o.d. generalization test.

[1] http://www.locobot.org/

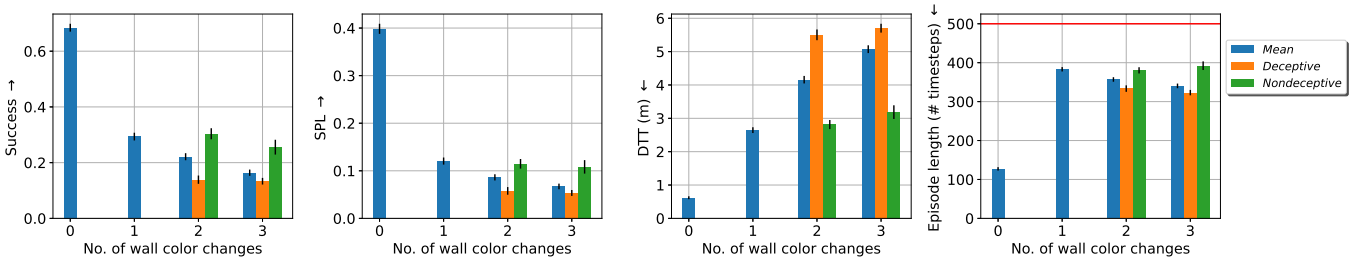


Fig. 5: **Results out-of-distribution (o.o.d.) generalization test.** Here, we report the performance of EmbCLIP [19] to scenes with different wall colors. We report the mean over all episodes, the mean over episodes with deceptive wall color changes and the mean of episodes with nondeceptive changes. We also show standard error of the mean for each. When only changing the wall color of the target object's room (1 wall color change), we already observe a large decrease in performance in all metrics.

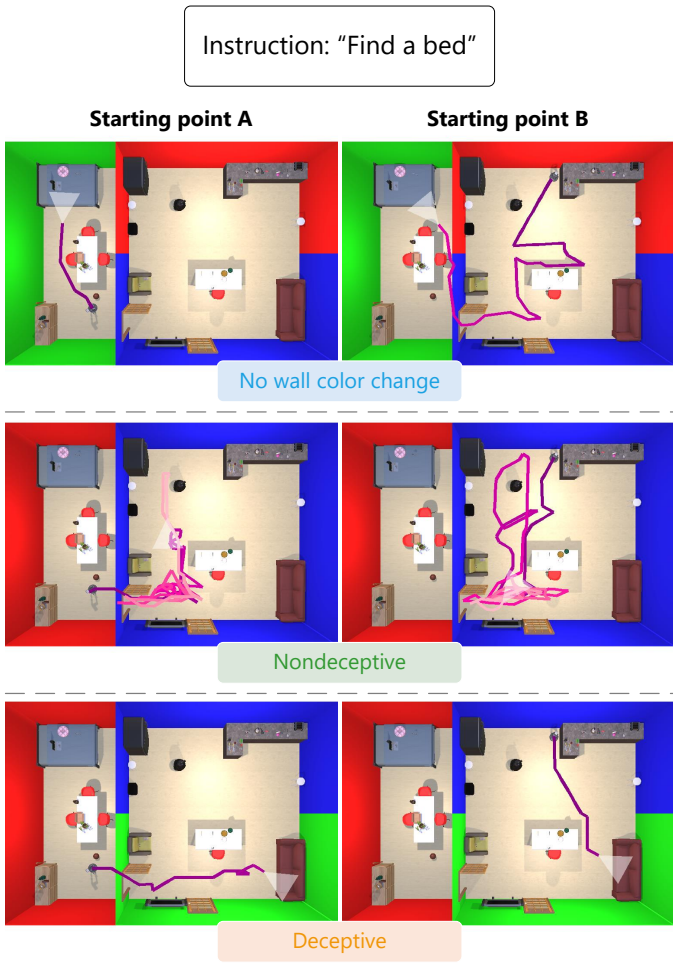


Fig. 6: **Errors and shortcuts by the SOTA ObjectNav method.** We show example trajectories from 2 different starting position (left vs right column). The top row shows 2 successful episodes in scenes without any wall color changes. Longer episodes are indicated by a brighter pink color in the path line segments. Notice how nondeceptive episodes (middle) are much longer than deceptive episodes (bottom), whilst both are unsuccessful. Also note the absolute lack of search in the bedroom when changing wall colors deceptively.

C. Analysis of agent's visual representations

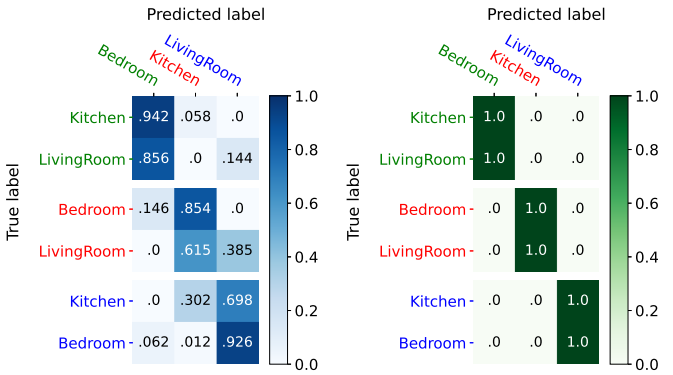
Though our o.o.d. generalization test shows shortcut learning is the root cause of poor generalization of EmbCLIP [19] to scenes with different wall colors, it does not explain why the agent learns a shortcut strategy. In this second experiment, we wish to bring more insight into why the agent learns to simply navigate towards the wall color associated with the target room. We aim to understand if we can trace this issue to the visual encoder (CLIP$_v$). Specifically, we posit wall color serves as an unintended shortcut predictor for room type in CLIP visual embeddings. To this end, we perform an experiment outside of the RL framework, using supervised learning. We train a simple neural classifier to predict room type from CLIP visual embeddings, extracted from sampled frames. The training set consists of frames where the bedroom has green walls, the kitchen has red walls and the living room has blue walls. We evaluate if the classifier adopts wall color as a shortcut predictor by observing its predictions on held-out room type-wall color combinations.

To enable the above evaluation, we generate a small dataset of sampled frames (RGB observations). Specifically, we select 1 house layout from our test split, permute wall colors, and sample 200 frames for every room type–wall color combination. Ground truth data for room type is determined by the position where the frame is sampled. Furthermore, we split our dataset into two types: 'context' vs 'contextless'. We refer to 'context' frames as frames which contain objects, which provide semantic information (or 'context') pertaining to room type (e.g. a sofa is in the living room). Instead, 'contextless' frames do not contain any useful information (i.e. walls, floors and ceilings). We create this split as we expect the classifier's prediction to be worse on contextless frames. If a frame only shows a wall, the classifier can only use the biased wall color to classify room type, leading to erroneous classification. See Appendix (Section VI-H) for more details on the dataset generation.

For each frame in our dataset, we extract the 1024-dimensional CLIP embeddings using the CLIP visual encoder employing a ResNet-50 backbone. We train a multi-layer perceptron (MLP) to predict room type from these embeddings. We use a MLP with 1 hidden layer (100 neurons wide), ReLu

activation, a batch size of 200, a learning rate of 0.0001, an adam optimizer and supervise using the generated ground truth data. Lastly, we standardize individual features using mean removal and scaling to unit variance before training the MLP.

Fig. 7 shows confusion matrices for (a) context and (b) contextless frames. The color of the labels indicates the wall color. Predicted label colors indicate the room type–wall color combinations which we trained on. In Fig. 7a we see that e.g., 94.2% of frames showing a kitchen with green walls, are classified as a bedroom since the classifier has only seen bedrooms with green walls during training. The majority of context frames are being classified directly according to their wall color, leading to classification accuracy of 16.8%, worse than random. In the contextless set (Fig. 7b), we see all rooms being classified according to their wall color leading to an overall classification accuracy of 0%. Clearly, we observe a large confusion of room types, which is caused by the biased wall color.



(a) Context (b) Contextless

Fig. 7: **Room type confusion using CLIP embeddings.** Confusion matrices reporting classifier accuracy for a classifier trained on CLIP embeddings of sampled frames. The colors of the labels indicate the wall color of the rooms, where the predicted label colors indicate the training data.

D. Analysis of L-B augmented visual representations

In the previous experiment we showed that CLIP visual representations encode shortcuts where wall color serves as an unintended shortcut predictor for room type. We find the classifier to predict room type mostly based on wall color of the biased training data, leading to poor classification accuracy on held-out room type–wall color combinations. We posit that improved room type classification will lead to more capable ObjectNav agents in our o.o.d. generalization test. Therefore, in the following experiment, we study the impact of augmenting CLIP visual representations, using the L-B augmentation from Section IV-B, on the room type classification accuracy.

Though the agent can not base its recognition of room type on wall color, it can base its recognition on object-room relations. Therefore, using L-B augmentation, we aim to improve classification accuracy on context frames i.e., frames which contain objects. In contrast, when no context is provided, the agent should not recognize the room type. Hence, we aim to achieve unbiased confusion on contextless frames i.e., a confusion matrix $\boldsymbol{C}$ where each element $C_{i,j} = 0.33$ (for $n_{classes} = 3$). To measure unbiased confusion, we define the Absolute Difference With Random (ADWR):

$$\text{ADWR} = \frac{1}{N} \sum_{i,j} |C_{i,j} - 0.33| \tag{5}$$

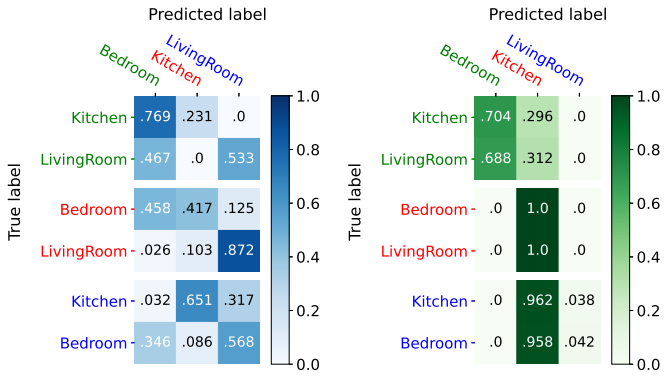
where, N is the number of elements in the confusion matrix $\boldsymbol{C}$. Ideally, ADWR $= 0$, where the classifier assigns equal probability (0.33) to each room type for all test sets. Contrary, when prediction is completely biased i.e., predicting all test samples as a single room type, ADWR $= 0.44$. Note that ADWR $\in [0, 0.44]$.

We use an identical experiment setup as in the previous experiment, where we analysed CLIP visual representations (Section V-C). Specifically, we use the same frames dataset and an identical MLP with the same hyperparameters. However, now we first use L-B augmentation to augment the 600 original CLIP embeddings from our train set. As detailed in Section IV-B, we insert $n = 3$ variations of the dataset bias (e.g. 'red wall') in the default prompt template: 'a photo of a {label}', encode using the CLIP text encoder and augment according to Equation 2 and 3. This results in 3600 augmented embeddings (6 for each original embedding). For each original embedding, we randomly sample 1 augmented embedding to obtain 600 augmented embeddings for our new train set. We test on the original CLIP embeddings of our test set, where we split in context and contextless frames.

In Fig. 8 we report confusion matrices. Using our L-B augmentations, we observe improved classification of room type in frames containing context (Fig. 8a). Although the bias is still significantly influencing the room type prediction, we see, for instance, 23.1% of green kitchen frames now being classified as their true room type, opposed to only 5.8% in Fig. 7a. This improvement results from varying CLIP features which encode wall color. By varying features which encode wall color, due to our L-B augmentations, the classifier cannot base its prediction solely on wall color.

Surprisingly, instead of unbiased confusion, we see a shift towards predicting all contextless frames as a kitchen type in Fig. 8b. We posit this is due to optimizing classification accuracy during training using log-loss, which indicates how far predictions are from ground truth. There is no incentive to classify rooms randomly, with a low probability, over classifying all test samples as a single class. Both result in low classification accuracy. Hence, there seems to be a mismatch between what we optimize for and what we aim to achieve.

In Table I, we compare results between a classifier trained on original CLIP embeddings vs after L-B augmentions. Using our method, we observe significantly increased classification accuracy on the context frames (17% $\rightarrow$ 50%), as desired. Furthermore, we see decreased ADWR on contextless frames (0.44 $\rightarrow$ 0.37), as desired. Overall, using L-B augmentation we are able to suppress the use of wall color serving as a shortcut predictor for room type.



(a) Context (b) Contextless

Fig. 8: **Room type confusion after L-B augmentations.** Confusion matrices reporting classifier accuracy for a classifier trained on L-B augmented embeddings of frames. The colors of the labels indicate the wall color of the rooms, where the predicted label colors indicate the wall color of the rooms the classifier was trained on.

E. Language-Based (L-B) augmentation in ObjectNav

Our L-B augmentations offer visual representations which are more domain invariant in the case of biased wall color and room type. Now we are curious if this also offers improved generalization in ObjectNav, when an agent needs to deal with such biases. We study if the increased room classification indeed also produces more capable ObjectNav agents.

To this end, we integrate our L-B augmentation method within the EmbCLIP architecture as detailed in Section IV-B. As our method operates in CLIP embedding space, we use the zero-shot architecture variant of EmbCLIP as this variant directly feeds the 1024-dimensional CLIP visual representation into the RNN. This allows for easily substituting this CLIP embedding with an augmented embedding, without further interventions on the EmbCLIP architecture. Notably, this architecture variant has lower base performance as it operates completely on CLIP representations. This includes encoding the target object label using CLIP's text encoder. Therefore, we compare performance against EmbCLIP's zero-shot variant. In Allenact [47], frames of each worker are processed in a batch. Each time step, we randomly sample a possible augmentation $\mathbf{\Delta_1}, ..., \mathbf{\Delta_{n(n-1)}}$ for each batch. Specifically, as we have 2 workers, this means that we choose a new augmentation each time step for both workers. We only change the model architecture in the experimental setup, all other parameters remain the same. We train both the zero-shot EmbCLIP variant (88 GPU-hours) and a L-B augmented variant (90 GPU-hours) for 30M frames on the set of visually identical training scenes.

Fig. 9 shows a comparison of the zero-shot variant of EmbCLIP and when we integrate our L-B augmentations. We observe similar agent behaviour for the zero-shot EmbCLIP variant as for the closed-world variant in Fig. 5. Performance already degrades significantly after changing the wall colors of the target room (1 wall color change). We observe 69% relative drop in success rate ($45\% \rightarrow 14\%$) and 82% drop in SPL ($0.22 \rightarrow 0.04$). In contrast, our method shows improved domain generalization. When changing the wall colors of the target room (1 wall color change), our method incurs only 23% relative drop in success rate ($39\% \rightarrow 30\%$) and 29% drop in SPL ($0.17 \rightarrow 0.12$). We observe less performance degradation with increasing number of wall color changes than EmbCLIP. These results demonstrate L-B augmentation is able to mitigate dataset bias to a large degree in our o.o.d. generalization test.

TABLE I: **Original CLIP visual representation vs after Language-Based augmentations.** Table shows a comparison between a classifier trained on original CLIP embeddings vs after L-B augmentations. We report accuracy on the context frames and ADWR (see text) on the contextless frames.

	Accuracy (↑) (Context)	ADWR (↓) (Contextless)
Original CLIP	0.17	0.44
Language-based augmentation (ours)	**0.50**	**0.37**

VI. Conclusion and Limitations

DRL and the use of E-AI simulators shows promising results for ObjectNav. However, bias in the appearance of simulated training environments might cause agents to learn shortcuts, which hamper o.o.d. generalization. Therefore, in this work, we evaluated how well a SOTA ObjectNav method generalizes to scenes with different wall colors, and studied to what extend shortcut learning influences this o.o.d. generalization. We found that, when deliberately limiting training data, only changing wall colors in testing scenes decreases performance significantly. Moreover, we showed that shortcut learning is the root cause by making deceptive wall color changes, which demonstrate the agent learned to simply search for the wall color associated with the target room.

Subsequently, we introduced Language-Based (L-B) augmentation to address this issue. By encoding text descriptions of variations of the dataset bias, and leveraging the multimodal embedding space of CLIP, we were able to augment agent's visual representation directly at feature-level. Analysis of agent's visual representations showed improvement when probed for room type classification. Finally, experiments showed that our L-B augmentation method is able to improve domain generalization to scenes with different wall colors in ObjectNav. When changing the target object's room, our method incurs 23% relative drop in success rate whilst the SOTA ObjectNav method's success rate drops 69%.

To demonstrate the usefulness of our approach, we consider an extreme case of dataset bias, where every instance of each room type is associated to a specific wall color. However, biases in training data might occur with some variation e.g., 10% of room types might occur with an unbiased wall color. In future work, it would be interesting to see the effect of introducing some variations on shortcut learning.

Instead of visual biases, agents may still learn shortcuts pertaining to semantic biases e.g., fridge is in the kitchen. Efficient ObjectNav requires agents to leverage useful semantic priors about the environment, such as object-object or object-room relations. This allows agents to decide which regions

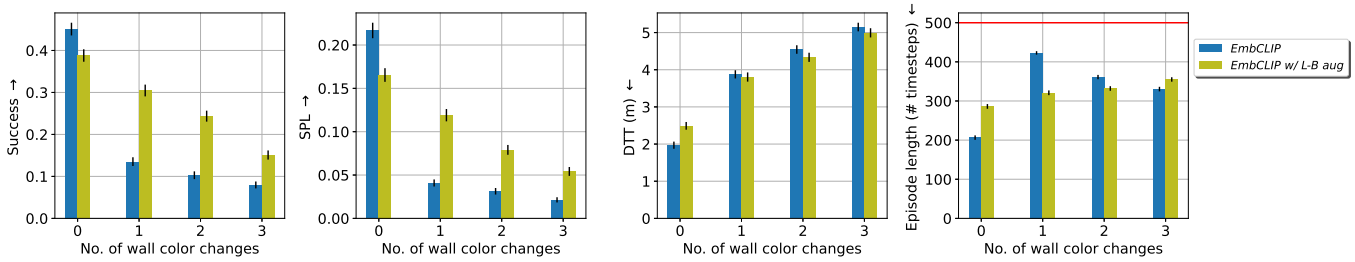


Fig. 9: **EmbCLIP [19] vs EmbCLIP with L-B augmentations.** We report results of our o.o.d. generalization test for the zero-shot variant of EmbCLIP and when integrating our L-B augmentations. Using our L-B augmentations we generalize better to scenes with different wall colors. Note, for instance, our method's success rate drops 23% ($0.39 \rightarrow 0.30$) when changing the target room (1 wall color change), whilst we observe 69% relative drop for EmbCLIP ($0.45 \rightarrow 0.14$).

to explore next and to decide where the target objects is most likely located. However, by relying on these priors, agents may struggle in environments which offer more unusual situations. For instance, when the fridge is placed in the living room. Future work might explore how to use natural language to guide the exploration of agents in such situations.

Acknowledgment

I would like to thank one person in particular, my supervisor at TNO: Gertjan Burghouts. He has been incredibly helpful and understanding, even when I was struggling at times. Looking back, this provided a unique learning experience to not only develop my academic skills, but also to better myself on a more personal level. I would also like to thank Luuk Spreeuwers. He has provided excellent supervision not only during my Master's thesis, but throughout my entire time as a student at the University of Twente. I am truly grateful for the support of both of you.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[2] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "Decentralized distributed PPO: solving pointgoal navigation," *CoRR*, vol. abs/1911.00357, 2019. [Online]. Available: http://arxiv.org/abs/1911.00357

[4] P. Marza, L. Matignon, O. Simonin, and C. Wolf, "Teaching agents how to map: Spatial reasoning for multi-object navigation," *CoRR*, vol. abs/2107.06011, 2021. [Online]. Available: https://arxiv.org/abs/2107.06011

[5] O. Maksymets, V. Cartillier, A. Gokaslan, E. Wijmans, W. Galuba, S. Lee, and D. Batra, "Thda: Treasure hunt data augmentation for semantic navigation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 15 354–15 363.

[6] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "AI2-THOR: an interactive 3d environment for visual AI," *CoRR*, vol. abs/1712.05474, 2017. [Online]. Available: http://arxiv.org/abs/1712.05474

[7] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," *CoRR*, vol. abs/1808.10654, 2018. [Online]. Available: http://arxiv.org/abs/1808.10654

[8] A. X. Chang, A. Dai, T. A. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from RGB-D data in indoor environments," *CoRR*, vol. abs/1709.06158, 2017. [Online]. Available: http://arxiv.org/abs/1709.06158

[9] P. Anderson *et al.*, "On evaluation of embodied navigation agents," *CoRR*, vol. abs/1807.06757, 2018. [Online]. Available: http://arxiv.org/abs/1807.06757

[10] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, "Objectnav revisited: On evaluation of embodied agents navigating to objects," *CoRR*, vol. abs/2006.13171, 2020. [Online]. Available: https://arxiv.org/abs/2006.13171

[11] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," *CoRR*, vol. abs/2007.00643, 2020. [Online]. Available: https://arxiv.org/abs/2007.00643

[12] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable objectnav," *CoRR*, vol. abs/2104.04112, 2021. [Online]. Available: https://arxiv.org/abs/2104.04112

[13] M. Deitke *et al.*, "Procthor: Large-scale embodied ai using procedural generation," 2022. [Online]. Available: https://arxiv.org/abs/2206.06994

[14] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman, "PONI: potential functions for objectgoal navigation with interaction-free learning," *CoRR*, vol. abs/2201.10029, 2022. [Online]. Available: https://arxiv.org/abs/2201.10029

[15] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," *CoRR*, vol. abs/2103.00020, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[16] C. Jia, Y. Yang, Y. Xia, Y. Chen, Z. Parekh, H. Pham, Q. V. Le, Y. Sung, Z. Li, and T. Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision," *CoRR*, vol. abs/2102.05918, 2021. [Online]. Available: https://arxiv.org/abs/2102.05918

[17] H. Pham, Z. Dai, G. Ghiasi, H. Liu, A. W. Yu, M. Luong, M. Tan, and Q. V. Le, "Combined scaling for zero-shot transfer learning," *CoRR*, vol. abs/2111.10050, 2021. [Online]. Available: https://arxiv.org/abs/2111.10050

[18] R. Geirhos, J. Jacobsen, C. Michaelis, R. S. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *CoRR*, vol. abs/2004.07780, 2020. [Online]. Available: https://arxiv.org/abs/2004.07780

[19] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi, "Simple but effective: CLIP embeddings for embodied AI," *CoRR*, vol. abs/2111.09888, 2021. [Online]. Available: https://arxiv.org/abs/2111.09888

[20] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," 2017. [Online]. Available: https://arxiv.org/abs/1703.06907

[21] F. Sadeghi and S. Levine, "(cad)^{2}rl: Real single-image flight without a single real image," *CoRR*, vol. abs/1611.04201, 2016. [Online]. Available: http://arxiv.org/abs/1611.04201

[22] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022. [Online]. Available: https://arxiv.org/abs/2204.06125

[23] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," *CoRR*, vol. abs/2102.12092, 2021. [Online]. Available: https://arxiv.org/abs/2102.12092

[24] Y. Wu, Y. Wu, A. Tamar, S. Russell, G. Gkioxari, and Y. Tian, "Learning

and planning with a semantic model," *CoRR*, vol. abs/1809.10842, 2018. [Online]. Available: http://arxiv.org/abs/1809.10842

[25] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building generalizable agents with a realistic and rich 3d environment," *CoRR*, vol. abs/1801.02209, 2018. [Online]. Available: http://arxiv.org/abs/1801.02209

[26] G. Georgios, S. Karl, W. Karan, D. Soham, M. Eleni, R. Dan, and D. Kostas, "Cross-modal map learning for vision and language navigation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[27] D. Shah, B. Osinski, B. Ichter, and S. Levine, "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," 2022. [Online]. Available: https://arxiv.org/abs/2207.04429

[28] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," 2018. [Online]. Available: https://arxiv.org/abs/1810.06543

[29] T. Campari, P. Eccher, L. Serafini, and L. Ballan, "Exploiting scene-specific features for object goal navigation," *CoRR*, vol. abs/2008.09403, 2020. [Online]. Available: https://arxiv.org/abs/2008.09403

[30] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra, "Zson: Zero-shot object-goal navigation using multimodal goal embeddings," 2022. [Online]. Available: https://arxiv.org/abs/2206.12403

[31] Q. Zhao, L. Zhang, B. He, H. Qiao, and Z. Liu, "Zero-shot object goal visual navigation," 2022. [Online]. Available: https://arxiv.org/abs/2206.07423

[32] R. Ramrakhya, E. Undersander, D. Batra, and A. Das, "Habitat-web: Learning embodied object-search strategies from human demonstrations at scale," 2022. [Online]. Available: https://arxiv.org/abs/2204.03514

[33] S. Wani, S. Patel, U. Jain, A. X. Chang, and M. Savva, "Multion: Benchmarking semantic map memory using multi-object navigation," *CoRR*, vol. abs/2012.03912, 2020. [Online]. Available: https://arxiv.org/abs/2012.03912

[34] M. Savva *et al.*, "Habitat: A platform for embodied AI research," *CoRR*, vol. abs/1904.01201, 2019. [Online]. Available: http://arxiv.org/abs/1904.01201

[35] M. Deitke *et al.*, "Robothor: An open simulation-to-real embodied AI platform," *CoRR*, vol. abs/2004.06799, 2020. [Online]. Available: https://arxiv.org/abs/2004.06799

[36] K. Ehsani, W. Han, A. Herrasti, E. VanderBilt, L. Weihs, E. Kolve, A. Kembhavi, and R. Mottaghi, "Manipulathor: A framework for visual object manipulation," *CoRR*, vol. abs/2104.11213, 2021. [Online]. Available: https://arxiv.org/abs/2104.11213

[37] A. Szot *et al.*, "Habitat 2.0: Training home assistants to rearrange their habitat," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 251–266. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/021bbc7ee20b71134d53e20206bd6feb-Paper.pdf

[38] B. Shen *et al.*, "igibson, a simulation environment for interactive tasks in large realistic scenes," *CoRR*, vol. abs/2012.02924, 2020. [Online]. Available: https://arxiv.org/abs/2012.02924

[39] C. Li *et al.*, "igibson 2.0: Object-centric simulation for robot learning of everyday household tasks," *CoRR*, vol. abs/2108.03272, 2021. [Online]. Available: https://arxiv.org/abs/2108.03272

[40] S. K. Ramakrishnan *et al.*, "Habitat-matterport 3d dataset (HM3D): 1000 large-scale 3d environments for embodied AI," *CoRR*, vol. abs/2109.08238, 2021. [Online]. Available: https://arxiv.org/abs/2109.08238

[41] Z. Li *et al.*, "Openrooms: An open framework for photorealistic indoor scene datasets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7190–7199.

[42] L. Scimeca, S. J. Oh, S. Chun, M. Poli, and S. Yun, "Which shortcut cues will dnns choose? A study from the parameter-space perspective," *CoRR*, vol. abs/2110.03095, 2021. [Online]. Available: https://arxiv.org/abs/2110.03095

[43] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *CoRR*, vol. abs/1811.12231, 2018. [Online]. Available: http://arxiv.org/abs/1811.12231

[44] S. Wang, R. Veldhuis, C. Brune, and N. Strisciuglio, "Frequency shortcut learning in neural networks," in *NeurIPS 2022 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2022. [Online]. Available: https://openreview.net/forum?id=zAfUHtSGWw

[45] D. Gordon, A. Kadian, D. Parikh, J. Hoffman, and D. Batra, "Splitnet: Sim2sim and task2task transfer for embodied visual navigation," *CoRR*, vol. abs/1905.07512, 2019. [Online]. Available: http://arxiv.org/abs/1905.07512

[46] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, "Clip on wheels: Zero-shot object navigation as object localization and exploration," 2022. [Online]. Available: https://arxiv.org/abs/2203.10421

[47] L. Weihs, J. Salvador, K. Kotar, U. Jain, K. Zeng, R. Mottaghi, and A. Kembhavi, "Allenact: A framework for embodied AI research," *CoRR*, vol. abs/2008.12760, 2020. [Online]. Available: https://arxiv.org/abs/2008.12760

[48] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[50] J.-B. Alayrac *et al.*, "Flamingo: a visual language model for few-shot learning," 2022. [Online]. Available: https://arxiv.org/abs/2204.14198

[51] D. Batra *et al.*, "Rearrangement: A challenge for embodied AI," *CoRR*, vol. abs/2011.01975, 2020. [Online]. Available: https://arxiv.org/abs/2011.01975

[52] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot, "Navigating to objects in the real world," 2022. [Online]. Available: https://arxiv.org/abs/2212.00922

[53] J. Kulhánek, E. Derner, and R. Babuška, "Visual navigation in real-world indoor environments using end-to-end deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4345–4352, 2021.

[54] J. Clark and D. Amodei, "Faulty reward functions in the wild," 2016. [Online]. Available: openai.com/blog/faulty-reward-functions/

[55] T. VII, "The first level of super mario bros. is easy with lexicographic orderings and time travel ...after that it gets a little tricky," 01 2013.

[56] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *CoRR*, vol. abs/1606.06565, 2016. [Online]. Available: http://arxiv.org/abs/1606.06565

[57] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013. [Online]. Available: https://arxiv.org/abs/1312.6199

[58] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," 2017. [Online]. Available: https://arxiv.org/abs/1707.07328

APPENDIX

A. Action space description

Table II shows a description of the 6 actions the agent can take. Also, we simulate actuation noise to better resemble actuation in the real-world.

B. Scene and target selection

We select 3 target objects per room type. The selection is based on the object's overall frequency of occurrence in ProcTHOR-10k and if they have a clear semantic relation with one of the room types. We select different sized objects for each room type. See table III for an overview of all the target objects selected.

C. Tuning degree (α)

The degree of augmentation α controls the amount of variation to add to the visual representations. Essentially, α scales the norm of the differences of the encoded CLIP text embeddings $\Delta(\boldsymbol{T})$, which we add to the CLIP visual embedding $\boldsymbol{I_t}$ (see Equation 2). We standardize the resulting embeddings to ensure more stability during training by removing the mean of each individual feature and scaling to unit variance.

We empirically tune α by iteratively training classifiers on our L-B augmented embeddings for varying α. As detailed in Section IV-B, we use three ($n = 3$) text descriptions of variations of the dataset bias: 'blue wall', 'red wall' and 'green wall'. We insert each description in the default prompt template [15]: 'a photo of a {label}' and encode using CLIP's text encoder. We augment according to Equation 2 and 3. We use the generated dataset of sampled frames (Section VI-H Appendix) to select an optimal α. Specifically, we augment CLIP embeddings of 600 frames which show a bedroom with green walls (200), a kitchen with red walls (200) and a living room with blue walls (200). Our train set is formed by the resulting 3600 augmented embeddings (6 for each original embedding). Note that we do not implement random sampling here. We test on the original CLIP embeddings extracted from the context and contextless frames. We tune α based on accuracy on context frames and ADWR (Equation 5) on the contextless split. Finally, the classifier architecture is an identical MLP as used in our analysis of agent's visual representations, Section V-C and V-D.

Fig. 10 shows the result of this tuning. Note that accuracy (context) for $\alpha = 50$ is better than in Table I as we have 6 times as many training samples due to not implementing random sampling. We observe an optimal accuracy on context frames at $\alpha \approx 50$ and a steep drop-off at $\alpha \approx 80$. We posit the drop-off is caused by the optimizer converging to a different minimum in the loss function. As detailed in Section V-D, it unfortunately seems impossible to vary CLIP visual representation in such a way that we improve classification on context frames, whilst making classification on contextless frames close to random. Therefore, we choose $\alpha = 50$.

TABLE II: **Action space description.** We use a 6-action discrete action space.

Action	Description
MOVEAHEAD	Moves the agent forward (if possible) by sampling from $\mathcal{N}(\mu = 0.25m, \sigma = 0.005m)$. If it is not possible to move the agent due to a collision, the action fails and the position remains the same.
ROTATELEFT ROTATERIGHT	Rotates the agent left or right by sampling from $\mathcal{N}(\mu = 30°, \sigma = 0.5°)$
LOOKUP LOOKDOWN	Tilt the camera of the agent upward or downward by 30°
DONE	Special action of the agent to terminate the episode.

TABLE III: **Target objects selected for each room type.**

Room type	Target object category
Kitchen	Fridge Kettle Apple
Living room	Sofa Television Newspaper
Bedroom	Bed Dresser Alarm clock

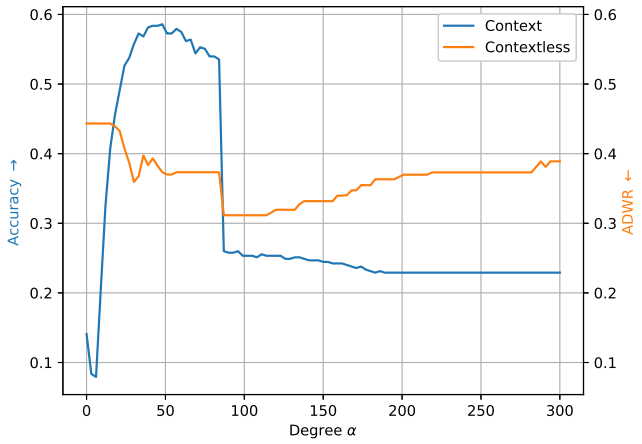


Fig. 10: **Tuning alpha.** The figure reports classifier accuracy on context frames (blue) and ADWR (Equation 5) on contextless frames (orange) for varying degree α.

D. Wall color permutations

For our o.o.d. generalization test, we permute wall colors in testing scenes. We only consider changes to red, green and blue walls and do not consider different colors. Basically, we consider all possible permutations with repetition i.e. multiple room types can have the same wall color. Table IV shows an overview of all possible wall colors changes.

TABLE IV: **Possible wall color changes.** 0 represents blue wall color, 1 represents red wall color and 2 a green wall color. As we start with altering the target room first, the possible wall color changes differ per target room.

Set	Target room	Living room	Kitchen	Bedroom
Train-set	All	0	1	2
Test set 0-room		0	1	2
Test set 1-room	Kitchen	0	0	2
		0	2	2
	Bedroom	0	1	0
		0	1	1
	Living room	1	1	2
		2	1	2
Test set 2-room	Kitchen/Bedroom	0	0	0
		0	0	1
		0	2	0
		0	2	1
	Living room/Kitchen	1	0	2
		1	2	2
		2	0	2
		2	2	2
	Living room/Bedroom	1	1	0
		1	1	1
		2	1	0
		2	1	1
Test set 3-room	ALL	1	0	0
		1	0	1
		1	2	0
		1	2	1
		2	0	0
		2	0	1
		2	2	0
		2	2	1

E. Evaluation episodes split

This section details how we distribute evaluation episodes in our o.o.d. generalization test. Basically, we evenly distribute episodes over the 9 target objects (see Table III), 5 scene layouts and possible wall color permutations (see Table IV). As there are more possible permutations for increasing number of wall color changes, the number of episodes per unique scene (combination of layout and wall color permutation) decreases. For example, only 5 unique scenes are possible for the 0-room test set as only 1 wall color permutation is possible (no wall color changes w.r.t. training set), while 10 unique scenes per target room are possible for the 1-room test set as we can alter the target room to 2 different wall colors. We run in total 1080 evaluation episodes per test set. Table V shows an overview.

F. Agent descriptions

1) Closed-world EmbCLIP architecture [19]: This variant assumes known target objects i.e., target object are drawn from a closed predetermined set of object categories. At each time step the agent receives a $3 \times 224 \times 224$ egocentric RGB observation. This image is processed into a $2048 \times 7 \times 7$ visual representation $\boldsymbol{V}_t$, using a frozen CLIP visual encoder with a ResNet-50 backbone whose final pooling and fully connected layers have been removed. A 2-layer CNN compresses the representation to obtain a $32 \times 7 \times 7$ tensor $\boldsymbol{V}'_t$. An integer $g \in \{0, ..., 9\}$ is used to indicate 1 of the 9 possible target object labels. g is used to index a trainable embedding matrix to obtain a 32-dimensional goal vector. This vector is resized and tiled to $32 \times 7 \times 7$ (copied 7×7 times) to obtain our target object embedding $\boldsymbol{G}_t$. We concatenate $\boldsymbol{V}'_t$ and $\boldsymbol{G}_t$ to a $64 \times 7 \times 7$ shape, compress using a 2-layer CNN to $32 \times 7 \times 7$ and flatten to form a 1568-dimensional goal-conditioned visual embedding $\boldsymbol{Z}_t$. Next, $\boldsymbol{Z}_t$ is concatenated with a 6-dimensional embedding vector $\boldsymbol{a'_{t-1}}$ representing the previous action. This 1574-dimensional vector is passed into a 1-layer GRU, along with the previous hidden belief state. The GRU has 512 hidden units and its output feeds into two linear layers forming the actor and critic heads. The actor head maps the hidden state to a 6-dimensional vector which, after a softmax function, produces the probability distribution over the 6 discrete actions i.e. the agent's policy π. Each time step t, the agent executes the action a_t with the highest probability from π. Lastly, the critic head maps the hidden state to a scalar to estimate the value of the current state i.e. the expected total accumulated reward for following policy π in the current state.

2) Zero-shot EmbCLIP architecture [19]: This variant does not assume known target object categories and, therefore, operates entirely on CLIP representations. Basically, this variant does not remove the pooling and fully connected layers from the CLIP visual encoder. Instead, it feeds the 1024-dimensional CLIP embedding $\boldsymbol{I}_t$ directly into a 1-layer GRU with 1024 hidden units. Also, the target object label is inserted into a text prompt: 'Navigate to label' and encoded using CLIP's text encoder to obtain a text embeddings $\boldsymbol{T_t}$. The output of the GRU $\boldsymbol{o_t}$ is added to $\boldsymbol{I}_t$ and element-wise multiplied with the text embedding of the target object label i.e., $(\boldsymbol{o_t} + \boldsymbol{I}_t) * \boldsymbol{T_t}$. Finally, this vector is passed into actor and critic heads, which

TABLE V: **Distribution of episodes overview.** We run 1080 evaluation episodes per test set and evenly distribute episodes over the possible unique scenes (combination of layout and wall color permutation) and 9 target objects in each set.

	Test set 0-room	Test set 1-room	Test set 2-room	Test set 3-room
No. of wall color permutations per target room (Table IV)	1	2	8	8
No. of unique scenes per target room	5	10	40	40
Episodes per unique scene	216	108	27	27
Episodes per target object per unique scene	24	12	3	3
Total	**1080**	**1080**	**1080**	**1080**

formulate the agent's policy π and estimate the value of the current state respectively.

G. Additional training details

We use the Allenact framework [47] to train ObjectNav agents. Table VI details the hyperparameters we set for all of our training runs. We use DD-PPO [3] to train agents. Also, we employ Generalized Advantage Estimation (GAE) [48], parameterized by $\lambda = 0.95$.

H. Dataset of sampled frames

1) Sampling frames: For our o.o.d. analysis of agent's visual representations, we generate a small dataset of sampled frames (RGB observations). For this dataset, we sample 200 frames for every room type-wall color combination (3 room types, 3 wall colors) by initializing agents with a random position in a certain room, and sampling its RGB frame. Ground truth data for room type is determined by the agent's position. For instance, if the agent is positioned in e.g., the kitchen, the frame encodes a kitchen room type. To ensure the agent is not initialized in e.g., the kitchen but looking towards the bedroom, we limit the possible orientations the agent is initialized at. We limit the orientations such that the agent is not looking into other rooms than the one it is positioned in. This results in 9 sets of frames (200 each), each belonging to a certain room type and wall color combination (e.g. kitchen with red walls). The training set consists of frames showing a bedroom with green walls, a kitchen with red walls and a living room with blue walls. We test on the held-out combinations of room type and wall color, which we split into 'context' vs 'contextless'.

2) Splitting into context/contextless: This section describes how we split the test set of held-out frames into 'context' vs 'contextless'. First, we cherry pick 6 sampled frames, which we define as ground truth context or contextless. We pick one for each wall color. The frames are shown in Fig. 11. Next we extract the 1024-dimensional CLIP embeddings for all frames in our dataset. To form our split, we select the 150 most similar frames to each of the 3 contextless ground truth samples, and 200 most similar frames to each of the 3 context ground truth samples. The similarity is based on cosine score of their encoded CLIP representations. We use the CLIP visual encoder, employing a ResNet-50 backbone. Some of the frames are similar to multiple of the cherry picked frames and are, thereby, duplicate in either the contextless or context split. We remove any duplicates. This results in 195 contextless frames and 576 frames with context. Finally, our test set is formed by the intersection of frames not seen during training (held-out combinations of room type and wall color) and which are either in the context or contextless split.

TABLE VI: **Training hyperparameters.** We set these parameters for all ObjectNav experiments.

Hyperparameter	**Value**
No. of GPUs	2
No. environments per GPU	20
Rollout length	192
No. mini-batches per rollout	1
PPO epochs	4
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
Value loss coefficient	0.5
Entropy loss coefficient	0.01
PPO clip parameter (ϵ)	0.1
Gradient clip norm	0.5
Optimizer	Adam
Learning rate	3e-4

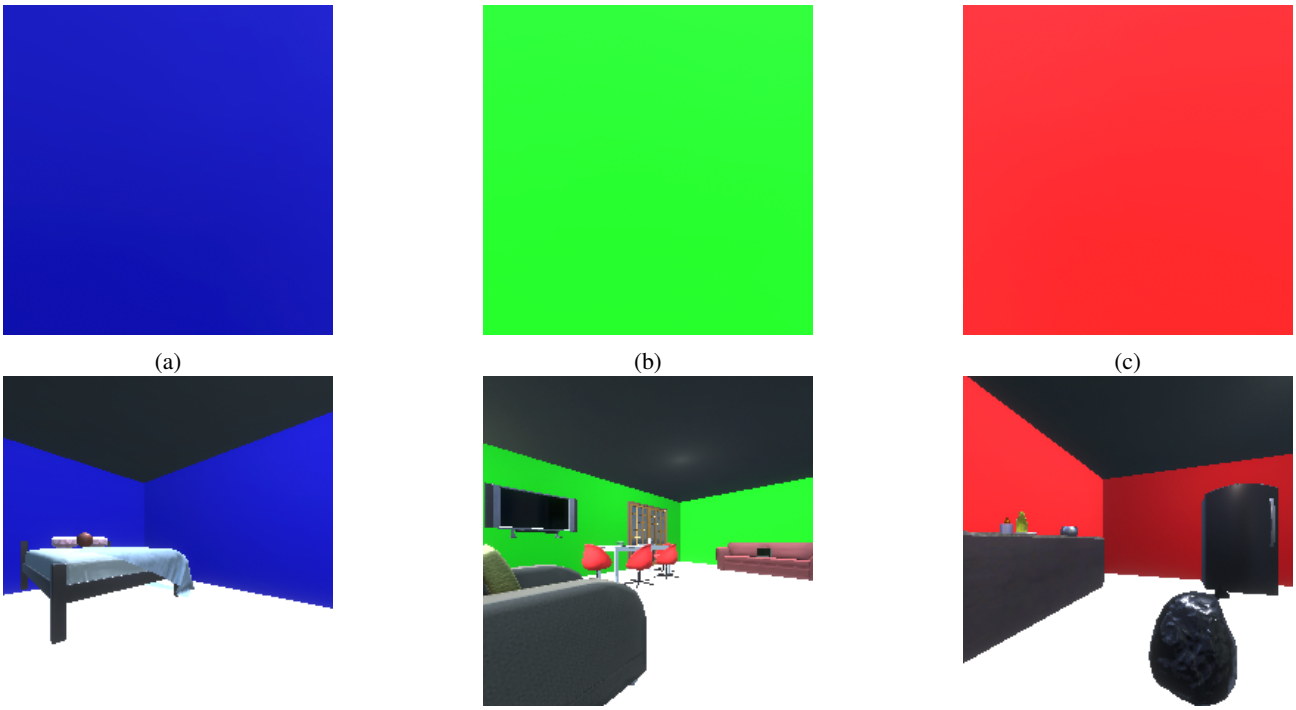


Fig. 11: **Ground truth for contextless (top) vs contextl (bottom) frames.** The figure shows the cherry picked samples for splitting our dataset into contextless (a-c) vs context (d-f). The split is based on cosine similarity of the CLIP representations of these ground truth frames.

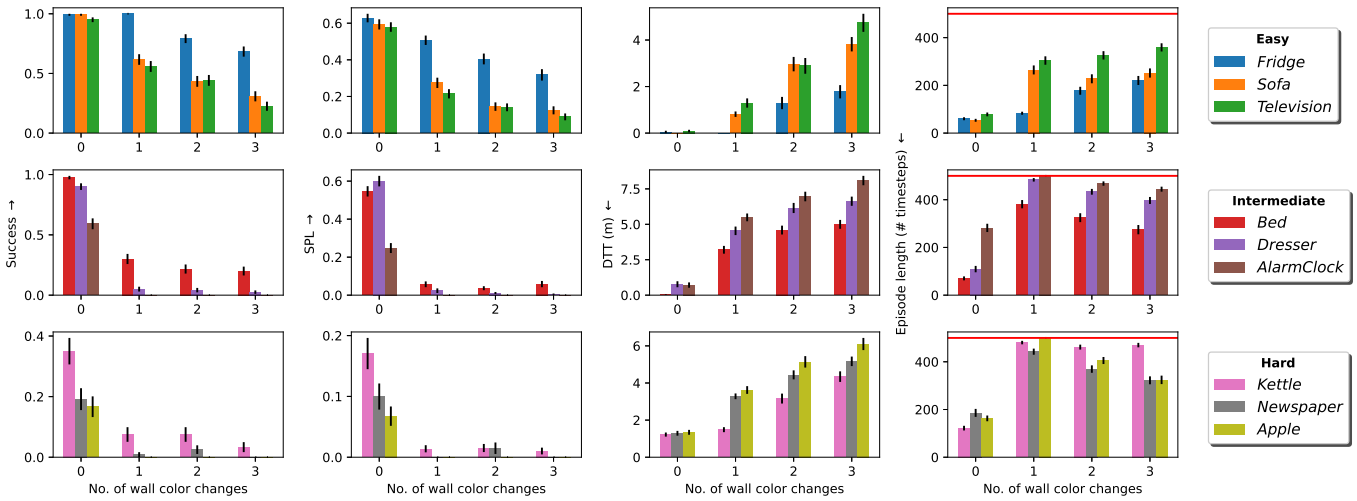


Fig. 12: **Additional results o.o.d. generalization test (Section V-B).** Here, we provide additional results. We show performance for each target object, separated by difficulty. We use the closed-world variant of EmbCLIP [19] for these results.

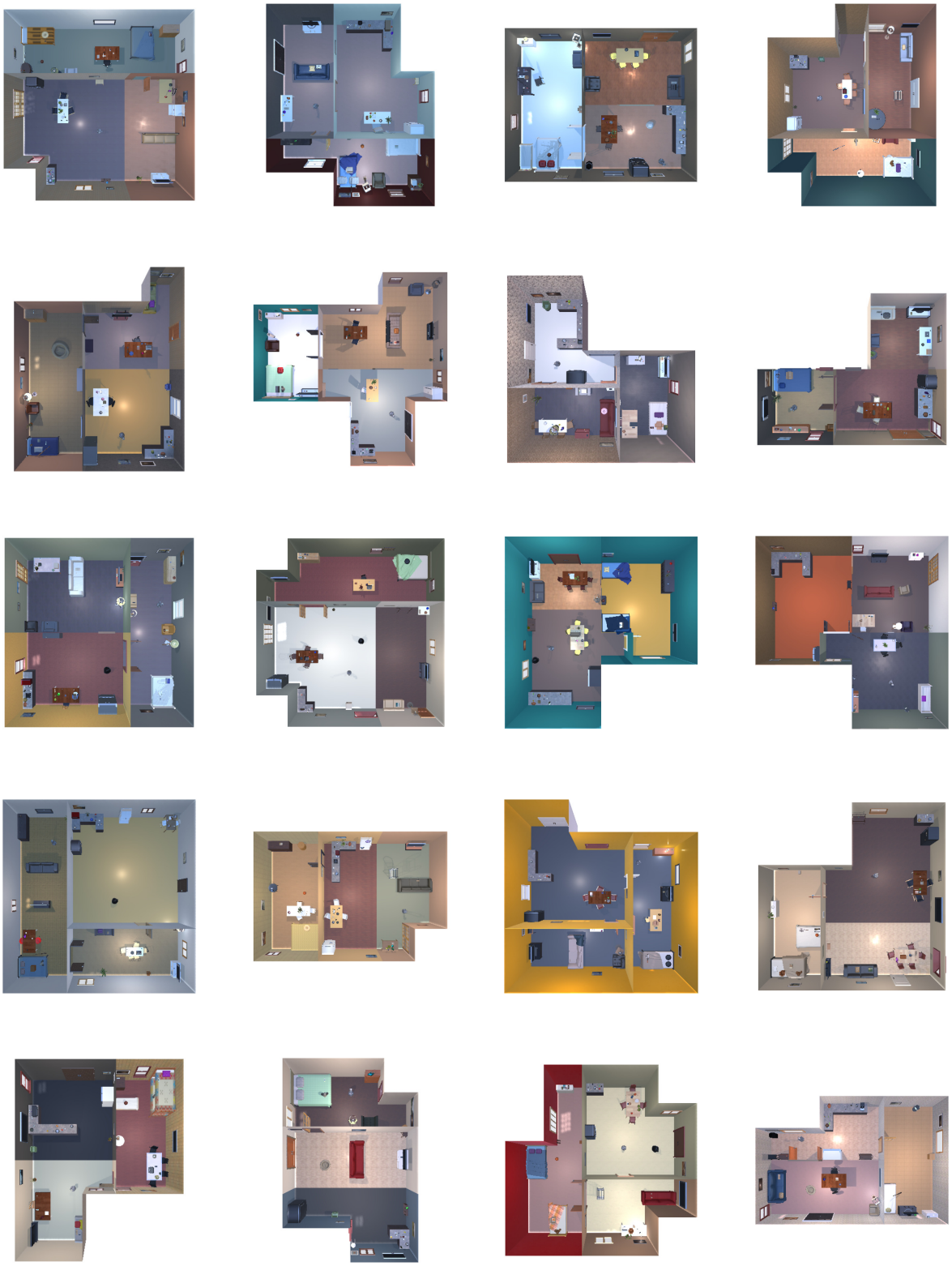


Fig. 13: **Original scenes from ProcTHOR-10k.** Top-down views of the 25 original scenes from the ProcTHOR-10k dataset we use. See Fig. 14 for top-down views of the houses after our proposed interventions, as described in Section IV-A.

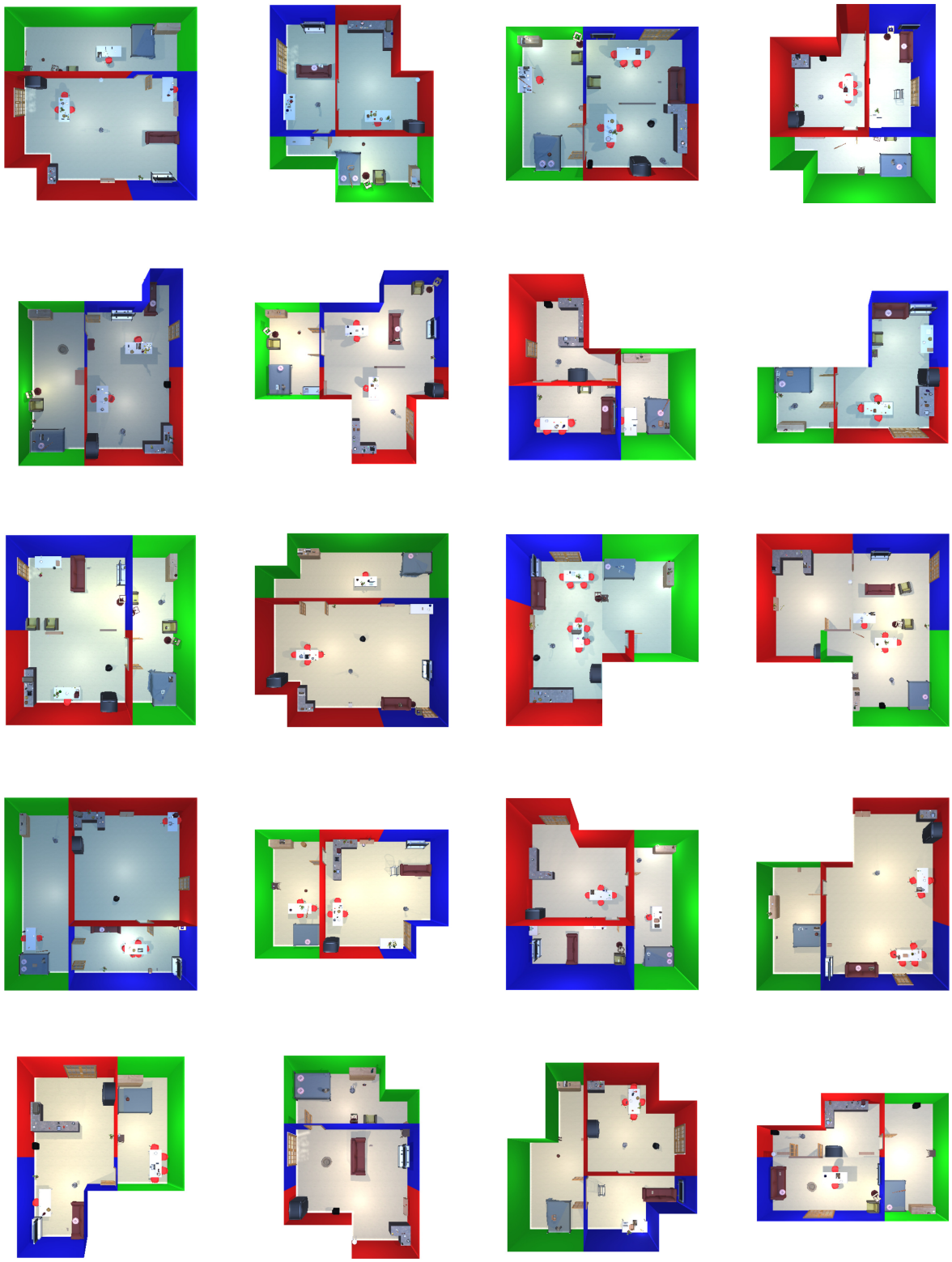


Fig. 14: **3D scene dataset for ObjectNav experiments.** Top-down images of the 25 scenes used for our ObjectNav experiments. Notice how all the houses are visually identical i.e., each room category has the same unique wall color and each object category appears in the same 3D asset from the ProcTHOR asset library (e.g. the sofa is always the same red sofa asset). We split in 20 training scenes and 5 testing scenes. The figure only shows houses where bedrooms have green walls, kitchens have red walls and living rooms have blue walls. For our o.o.d. generalization test, we permute wall colors of testing scenes.