Simple but Effective: CLIP Embeddings for Embodied AI

Apoorv Khandelwal[*] Luca Weihs[*] Roozbeh Mottaghi Aniruddha Kembhavi
Allen Institute for AI
{apoorvk, lucaw, roozbehm, anik}@allenai.org

Abstract

Contrastive language image pretraining (CLIP) encoders have been shown to be beneficial for a range of visual tasks from classification and detection to captioning and image manipulation. We investigate the effectiveness of CLIP visual backbones for embodied AI tasks. We build incredibly simple baselines, named EmbCLIP, with no task specific architectures, inductive biases (such as the use of semantic maps), auxiliary tasks during training, or depth maps—yet we find that our improved baselines perform very well across a range of tasks and simulators. EmbCLIP tops the RoboTHOR ObjectNav leaderboard by a huge margin of 20 pts (Success Rate). It tops the iTHOR 1-Phase Rearrangement leaderboard, beating the next best submission, which employs Active Neural Mapping, and more than doubling the % Fixed Strict metric (0.08 to 0.17). It also beats the winners of the 2021 Habitat ObjectNav Challenge, which employ auxiliary tasks, depth maps, and human demonstrations, and those of the 2019 Habitat PointNav Challenge. We evaluate the ability of CLIP's visual representations at capturing semantic information about input observations—primitives that are useful for navigation-heavy embodied tasks—and find that CLIP's representations encode these primitives more effectively than ImageNet-pretrained backbones. Finally, we extend one of our baselines, producing an agent capable of zero-shot object navigation that can navigate to objects that were not used as targets during training.

1. Introduction

The CLIP family of neural networks have produced very impressive results at a series of visual recognition tasks, including an astonishing zero-shot performance on ImageNet that matches the accuracy of a fully supervised ResNet-50 model [21]. Unsurprisingly, visual representations provided by CLIP have now also been shown to provide improvements across other computer vision tasks such as

[*]Equal contribution

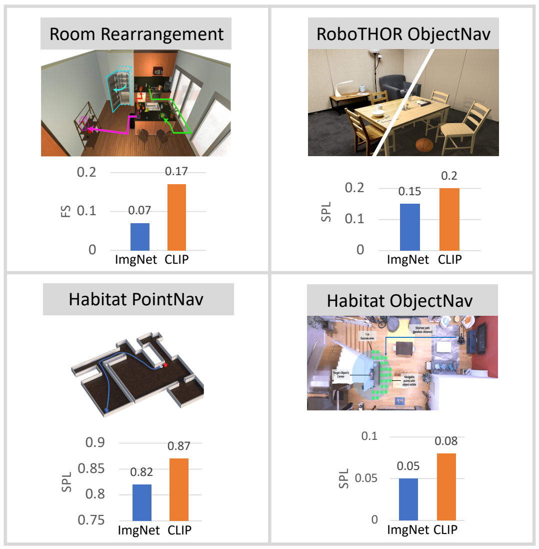

Figure 1. We show powerful visual encoders are important for Embodied AI tasks. We consider four navigation-heavy tasks and show CLIP-based encoders provide massive gains over ResNet architectures trained on ImageNet. More interestingly, using only RGB images as input, they outperform approaches employing depth images, maps, and more sophisticated architectures.

open vocabulary object detection [12], image captioning, visual question answering, and visual entailment [25]. In this work, we investigate the effectiveness of CLIP's visual representations at tasks in the domain of Embodied AI.

Shen *et al.* [25] demonstrated that CLIP features can provide gains in instruction following along a navigation graph: a task where an agent traverses the graph using linguistic instructions such as *Walk past the piano*. Building upon this outcome, we provide a thorough investigation of CLIP-powered embodied AI models at tasks that require agents to use low level instructions (such as `Move Ahead`, `Turn`, `Look Down`, and `Pick Up`) to take steps in a scene and interact with objects. Such tasks, including Object Goal Navigation [4] and Room Rearrangement [3], are inherently navigation-heavy. As a result, visual representations of observations in such tasks must not just be effective for recognizing categories, but also at encoding primitives such as walkable surfaces, free space, surface normals, and geometric structures.

We build a series of simple baselines using the CLIP ResNet-50 visual encoder. Our embodied agents input CLIP representations of observations that are frozen and not finetuned for the task at hand. These representations are combined with the goal specification and then passed into a recurrent neural network (RNN) unit to provide memory. A linear layer transforms the hidden activations of the RNN to a distribution over the agent's actions. These simple baselines have very few task specific design choices, do not use depth maps (which are commonly regarded as critical for good results), use no spatial or semantic maps, and employ no auxiliary tasks during training. And yet, without these design choices that have been empirically proven to be effective, our simple CLIP-based baselines are very effective—vaulting to the top of two leaderboards for the tasks of Object Goal Navigation and Room Rearrangement in the AI2-THOR environment, and outperforming the Habitat simulator challenge winners for Object Goal Navigation in 2021 and Point Goal Navigation in 2019.

These results are surprising and suggest that CLIP's visual representations are powerful and encode visual primitives that are useful to downstream embodied AI tasks. Indeed, we find that CLIP's representations outperform those from ImageNet pretraining by using a linear probe at four primitive tasks: object presence, object localization, reachability and free space estimation. In particular, CLIP's representations provide a +3 point absolute improvement on three of our probing studies and +6.5 point (+16 % relative) at the object localization probe.

We analyze 4 visual encoders (2 ResNet models pretrained on ImageNet and 2 ResNet models pretrained with CLIP) and use them to train 4 Object Goal Navigation agents. We then study the correlation between ImageNet Top-1 Accuracy and Success Rate for Object Goal Navigation. We find that ImageNet accuracy alone is not a good indicator of an encoder's suitability for Embodied AI tasks, and that it may be more useful to probe its representations for semantic and geometric information. However, our results do indicate that Embodied AI agents can continue to benefit from better visual encodings.

Finally, we use CLIP's visual and textual encoders in a simple architecture with few learnable parameters to train an agent for zero-shot Object Goal Navigation (*i.e.* navigating to objects that are not targets at training time). Our results are promising: the agent achieves roughly half the Success Rate for unseen objects compared to that for seen objects. They suggest that using visual representations that are strongly grounded in language can help build Embodied AI models that can generalize not only to new scenes, but also to new objects.

2. Related Work

Embodied AI. Embodied AI tasks involve agents that learn to navigate and interact with their environments. There are now many simulators and tasks in this domain. Simulators are photo-realistic, simulate physics, and enable interaction between agents and their environments [8, 9, 15, 16, 23, 24, 28, 34]. Large scene datasets accompany these simulators [5,22,34]. Many creative tasks [2–4,8,9,14,15,27,29] have been proposed, such as navigation, rearrangement, and furniture moving, and require complex combinations of skills, including perception, visual reasoning, communication, coordination, and action selection. While fast progress has been made on the task of Point Goal Navigation [33] in the presence of ideal sensors, most other tasks remain challenging.

Visual Encoders in Embodied AI. Due to inefficiency of larger models, most prior work uses lightweight CNN or ResNet models as visual encoders in Embodied AI tasks. This is particularly the case when sample or compute efficiency is of concern. Extensive experimentation has been done to compare the relative advantages of simple CNNs against ResNet-18 for Point Goal Navigation in Habitat [23] in this low sample/compute regime [32]. Although experimentation with larger ResNet visual encoders for Embodied AI has provided some evidence that such models can marginally improve downstream performance, there is also interesting evidence showing that using ImageNet-pretrained weights can harm downstream transfer to new tasks [33] (*e.g.* when using transfer learning to perform exploration with a model trained for Point Goal Navigation).

CLIP and CLIP based models. CLIP's visual representations have proven very effective at zero-shot image recognition (compared to ImageNet-pretrained ResNet representations, when a fully supervised linear classification layer is trained upon them) on benchmarks such as ImageNet, CIFAR100, Kinetics700 and UCF 101 [21]. These representations are also effective at vision-language tasks such as VQA and Visual Entailment [25]. CLIP's open vocabulary image classification ability has also been distilled into an open vocabulary object detection [12], proving to be very effective. CLIP has also proven to be effective at video tasks particularly text based video retrieval [10, 17, 20]. Finally, CLIP has been used for pixel generative tasks, including a text-based interface for StyleGAN image manipulation [19] and synthesizing novel drawings based on natural language input [11]. In the Embodied AI domain, CLIP has been employed with a Transporter [36] network to solve language-specified tabletop tasks such as packing objects and folding cloth [26]. CLIP has also been shown to help in instruction following [25] which requires the agent to move along the edges of a navigation graph. This task requires models to identify objects and locations in the scene via their names and descriptions. Buoyed by this success, we investigate

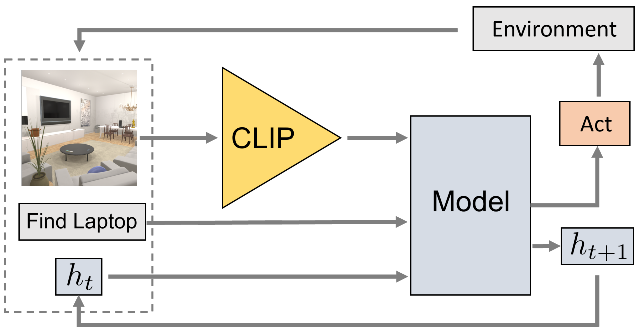

Figure 2. **Model overview.** All of our models use the same high-level architecture illustrated here, where a model (employing an RNN) receives features from the visual encoder, task definition, and previous hidden units as input, and outputs an action.

the use of CLIP towards navigation-heavy tasks that require the agent to use atomic actions (such as `Move Forward`, `Turn Left`, `Pick Up`, `Place`, etc.) to navigate within a scene. To do so successfully, visual encoders must not only identify objects, but also capture an understanding of depth, free space, surface normals, etc. Our experiments show that CLIP is remarkably effective at such tasks.

3. Using CLIP in Embodied AI

CLIP [21] is a recently released family of image and text encoders that are pretrained to contrast between corresponding and non-corresponding image–caption pairs. Although seemingly simple, this pretraining task, when employed at the scale of 400M image–caption pairs, leads to very powerful visual encoders. CLIP's visual backbones perform very well at a suite of computer vision datasets in a zero-shot setting, including matching the accuracy of the original fully supervised ResNet-50 model on the ImageNet benchmark.

CLIP's visual representations have proven to be effective for several other vision and vision–language tasks (see Sec. 2). We investigate their effectiveness at four navigation-heavy Embodied AI tasks: Object Goal Navigation (OBJECTNAV) in RoboTHOR and Habitat, Point Goal Navigation (POINTNAV) in Habitat, and Room Rearrangement (ROOMR) in iTHOR. For each of these benchmarks, we use a baseline implementation provided by its authors and substitute its visual encoder (often a shallow CNN, such as ResNet-18) with CLIP ResNet-50.

The resulting baselines have very similar core architectures. Fig. 2 shows the schematic of a general baseline model with a CLIP visual encoder. They primarily differ in their usage of the goal description (object name in OBJECTNAV, coordinates in POINTNAV, etc.) and processing of the visual features. Below, we describe this core baseline architecture in the context of OBJECTNAV in RoboTHOR. Please see the appendix for details on other tasks.

Specifically, the model for OBJECTNAV in RoboTHOR receives a $3{\times}224{\times}224$ RGB image i and integer $g \in \{0, ..., 11\}$ indicating the goal object category as input. The RGB image is encoded into a $2048{\times}7{\times}7$ tensor I by a CLIP ResNet-50 model whose weights are frozen and final attention pooling and classification layers have been removed. g is used to index a (trainable) embedding matrix and form a 32-dim goal embedding G. A two layer CNN then compresses I to form a tensor I' of shape $32{\times}7{\times}7$. The vector G is tiled to a shape of $32{\times}7{\times}7$ and concatenated with I'. This $64{\times}7{\times}7$ tensor is passed through another two-layer CNN (resulting in a $32{\times}7{\times}7$ shape) and flattened to form V: a 1568-dim goal-conditioned visual embedding. V is passed into a 1-layer GRU with 512 hidden units, along with any prior hidden state. Then, one linear layer maps the GRU output to a 6-dimensional vector of logits and another linear layer maps it to a scalar, respectively forming the actor (*i.e.* policy) and critic (*i.e.* value) heads commonly used in reinforcement learning.

In this work, we only consider CLIP ResNet-50, but our baselines can be trivially extended for other CLIP variants.

4. When is CLIP effective?

We build CLIP-based baselines (using CLIP ResNet-50) across multiple navigation-heavy tasks and across simulators. We compare these models to each task's public leaderboards and, in all cases, find that our models are competitive with SOTA methods—even surpassing the previous best entries on RoboTHOR OBJECTNAV and iTHOR ROOMR by large margins. We present these leaderboards[1] in Tables 1 to 4 and have also anonymously submitted our model to these online leaderboards (besides the 2019 Habitat POINTNAV leaderboard, which is now closed).

This is a particularly surprising result, since other methods required significant additional research and engineering efforts (*e.g.* task specific designs or massively distributed training for long durations) compared to our relatively simple modification. Importantly, **our CLIP baselines only use RGB**, whereas most models on leaderboards also use perfect depth information.

In each of the tasks, we also train a baseline agent with an ImageNet-pretrained ResNet-50. This ensures a fair comparison to the CLIP-based baseline w.r.t. parameter counts. We evaluate the checkpoint for each agent with the best validation set performance. In all four experiments, we find that pretraining with CLIP provides significant gains in success metrics compared to on ImageNet.

Hereafter, we refer to these baselines (with frozen ResNet-50 visual encoders and RGB as the only visual input) as our "CLIP agent" and "ImageNet agent" (by their method of pretraining).

[1]As of 11/16/2021

Models	SPL	SR	SPL Prox	SR Prox
ResNet-50 (CLIP)	**0.20**	**0.47**	**0.20**	**0.48**
ResNet-50 (ImageNet)	0.15	0.34	0.15	0.35
(1) EmbCLIP (Ours)	**0.20**	**0.47**	**0.20**	**0.48**
(2) Action Boost	0.12	0.28	0.12	0.30
(3) RGB+D ResNet18	0.11	0.26	0.12	0.28
. . .				

Table 1. **RoboTHOR ObjectNav Challenge (2021).** Here, we show test success metrics: Success weighted by Path Length (SPL) and Success Rate (SR). We compare our baseline agents (above) and against leaderboard entries (below).

Model	FS	SR	E	M
ResNet-50 (CLIP)	**0.17**	**0.08**	**0.89**	0.88
ResNet-50 (ImageNet)	0.07	0.03	1.06	1.05
(1) EmbCLIP (Ours)	**0.17**	**0.08**	**0.89**	0.88
(2) RN18 + ANM IL [29]	0.09	0.03	1.04	1.05
(3) RN18 + IL [29]	0.06	0.03	1.09	1.11
. . .				

Table 2. **iTHOR 1-Phase Rearrangement Challenge (2021).** We report test set metrics for both our baselines (above) and leaderboard entries (below).

4.1. Object Goal Navigation in RoboTHOR

Task. ObjectNav requires an agent to navigate through its environment and find an object of a given category. An instance of the ObjectNav task has been developed for the simulated RoboTHOR environment [8]. In RoboTHOR ObjectNav, a robotic agent[2] is placed into a near-photorealistic home environment at a random location. It is then given one of twelve goal object categories (*e.g.* apple) and must navigate to this object using `MoveAhead`, `RotateRight`, `RotateLeft`, `LookUp`, and `LookDown` actions. The agent is successful if it takes a special `Done` action and there is an instance of the goal object category that is both visible to the agent and within 1m of the agent.

Prior work. There have been two challenges for RoboTHOR ObjectNav (in 2020 and 2021), as well as several prior works that have used RoboTHOR ObjectNav as a testbed for reinforcement and imitation learning methodologies [13, 30] and for studying model robustness [7]. We present results on the 2021 RoboTHOR ObjectNav leaderboard[3]. The best performing models prior to our submission are **Action Boost** [1] which obtains a Test Set SPL of 0.12 and **RGB+D ResNet18-Imagenet** which obtains a Test Set SPL of 0.11 with code available via AllenAct [31].

Performance. We modify the AllenAct baseline model for our experiment (as described in Sec. 3) and train our ImageNet and CLIP agents for 200M environment steps. In Table 1, we display the performance of our models. Our CLIP agent dramatically outperforms both our ImageNet agent (by 1.3–1.4x in SPL and Success Rate) and the Action Boost model (by 1.7x in SPL and Success Rate).

4.2. Room Rearrangement in iTHOR

Task. A consortium of researchers in Embodied AI have recently proposed for Room Rearrangement to be the next frontier challenge [3] for existing models and methods. An instance of the RoomR task has been developed for the simulated iTHOR environment [29]. This proposed task has 1-phase and 2-phase variants, and we focus solely on the 1-phase setting in this work. In this 1-phase task, the agent is placed into a room and is given two images at every step. One image depicts the room "as it is" from the agent's egocentric viewpoint. The other image (from the same perspective) shows the room "as it should be"—namely, with several objects in different locations or in different states (*e.g.* a laptop may be on a table rather than on a sofa, or a cabinet may be open rather than closed). Using a discrete set of standard navigational actions (*e.g.* `MoveAhead`, `RotateRight`, *etc.*) and higher-level semantic actions (*e.g.* `PickUpX` for an object category `X`), the agent must rearrange the objects in the environment from their current states to their "as it should be" states.

Prior work. We report results for the 2021 iTHOR Rearrangement Challenge[4]. The best performing model prior to our submission is the **CVPR'21 ResNet18+ANM IL** model, which used a semantic mapping module based on the Active Neural SLAM (ANM) [6] architecture (proposed by Chaplot *et al.*) and was trained via imitation learning. The second best performing method (proposed by the iTHOR Rearrangement authors), which we adopt, uses an ImageNet-pretrained ResNet-18 visual encoder with frozen weights, an attention module to compare image pairs, and a 1-layer GRU to integrate observations over time.

Performance. Rather than relying on complex inductive biases in the form of semantic maps, we base our architecture on the second best performing method ("RN18 + IL" in Table 2). We train the ImageNet and CLIP agents in our experiment using imitation learning for 70 million steps. The performances of our models and the previous two best baselines are shown in Table 2. The results are dramatic: our CLIP agent achieves a 1.94x improvement over the ANM model (from 9% to 17%) in the `FixedStrict` (FS) metric used to rank RoomR models. On the other hand, our ImageNet agent (7% FS) with a ResNet-50 encoder per-

[2]Based on the LoCoBot robot (`www.locobot.org`)
[3]`leaderboard.allenai.org/robothor_objectnav`

[4]`ai2thor.allenai.org/rearrangement`

Models	SPL	SR	SoftSPL	Goal Dist
ResNet-50 (CLIP)	**0.08**	**0.18**	**0.20**	**7.92**
ResNet-50 (ImageNet)	0.05	0.13	0.17	8.69
(1) yuumi_the_magic_cat	**0.10**	0.22	0.18	9.17
(2) TreasureHunt [18]	0.09	0.21	0.17	9.20
(3) Habitat on Web (IL-HD)	0.08	**0.24**	0.16	**7.88**
(4) EmbCLIP (Ours)	0.08	0.18	**0.20**	7.92
(-) Habitat on Web^{2021}	0.07	0.21	0.15	8.26
(5) Red Rabbit^{2021} [35]	0.06	0.24	0.12	9.15
. . .				
(9) DD-PPO	0.00	0.00	0.01	10.326

Table 3. **Habitat ObjectNav Challenge (2021).** We report Test-Standard split metrics for both our baselines (above) and leaderboard entries (below). Entries with the superscript 2021 indicate challenge winners.

forms (1) only slightly better than the "RN18 + IL" baseline (6% FS), which it was based on, and (2) worse than the "RN18 + ANM IL" model (9% FS). This demonstrates that only making the visual encoder deeper is relatively ineffective. Beating the ANM-based model is somewhat remarkable, demonstrating that using CLIP representations offers a far greater improvement than even the inductive bias of a map.

4.3. Object Goal Navigation in Habitat

Task. ObjectNav in Habitat is defined similarly to the task in RoboTHOR (Sec. 4.1), including having the same action space and robotic agent design. However, Habitat uses 21 objects and does not require the agent to be looking at a target object for the episode to be successful. For the ObjectNav challenge, Habitat uses scenes from the MatterPort3D [5] dataset of real-world indoor spaces.

Prior work. ObjectNav has been a part of the 2020 and 2021 Habitat challenges. However, in the 2021 leaderboard[5], the baseline (an RGB-D agent with a ResNet-50 visual encoder trained from scratch with DD-PPO) scores 0.00 SPL on the test-standard set—this same method was shown to "solve" PointNav in Habitat [33] (and we explore this setting further in Sec. 4.4). As even the current leader (after over two years of entries) only achieves 0.10 SPL, we can see that this task remains quite challenging. As the baseline is mapless and has task-agnostic components, competitive leaderboard entries include numerous modifications to increase performance (*e.g.* augmenting scenes with 3D object scans [18], predicting semantic segmentations as additional inputs [35], and crowdsourcing human demonstrations for imitation learning).

Performance. For our experiments, we adopt the DD-PPO baseline, even though it scores very poorly on the leaderboard, and train our ImageNet and CLIP agents for 250M

[5]eval.ai/web/challenges/challenge-page/802/leaderboard/2195

Models	SPL	SR	Goal Dist
ResNet-50 (CLIP)	**0.87**	**0.97**	**0.40**
ResNet-50 (ImageNet)	0.82	0.94	0.73

Models	SPL
(1) DD-PPO	**0.89**
(2) Monocular Depth	0.85
(+) EmbCLIP (Ours)	0.84
(3) Arnold^{2019} [6]	0.70
. . .	

Table 4. **Habitat PointNav Challenge (2019).** As this leaderboard is now closed, we report metrics on the full validation set for our baselines (above) and on the minival set for RGB track leaderboard entries (below). The entry with the 2019 superscript indicates the challenge winner.

steps. In Table 3, we show that CLIP pretraining improves upon ImageNet pretraining by 1.45x SPL. Furthermore, our CLIP agent scores fourth place on the leaderboard, roughly on par with the leading entry and surpassing both winners of the 2021 challenge. This performance is particularly impressive because, unlike other entries, our agent does not (1) use depth information, (2) require additional data such as human annotations, or (3) require task-specific components. Furthermore, our agent has the second lowest distance to goal on the leaderboard, indicating that it navigates closer to the goal on average than most other models, even though it succeeds on fewer total episodes than the best models.

4.4. Point Goal Navigation in Habitat

Task. In PointNav, an agent must navigate from a random initial position to (relative) polar goal coordinates. This task has been a part of the Habitat challenge thrice: in 2019, where agents were provided ideal GPS+Compass sensors, and in 2020 and 2021, where agents were given no such sensors (to make the task more challenging and realistic). As baseline agents without GPS+Compass sensors achieve 0.00 SPL after training with DD-PPO for 100M steps, we train agents for our experiment on the 2019 challenge for ease of comparison and refer strictly to this setting below. The action space in PointNav is smaller than in ObjectNav and includes just `MoveAhead`, `RotateRight`, and `RotateLeft`. The agent should call `Done` when it reaches its goal coordinates. The Habitat challenge only permits training on the Gibson Database of 3D Spaces [34].

Prior work. Prior methods made quick progress[6] on PointNav with ideal sensors—"DD-PPO", the leading method in Table 4 (where only RGB visual input is given) achieves 0.92 SPL on the Test Std set, which is relatively

[6]eval.ai/web/challenges/challenge-page/254/leaderboard/836

Task	Pretraining	Pooling	Score
Object Presence	ImageNet	Average	0.502
	CLIP	**Average**	**0.530**
	CLIP	Attention	0.529
Object Localization	ImageNet	Average	0.387
	CLIP	**Average**	**0.452**
Reachability	ImageNet	Average	0.638
	CLIP	**Average**	**0.677**
	CLIP	Attention	0.668
Free Space	ImageNet	Average	0.287
	CLIP	**Average**	**0.315**
	CLIP	Attention	0.257

Table 5. **Do visual representations encode primitives?** We report test set scores for the experiments in Sec. 5. We compute F1 scores for Object Presence and Object Localization, and accuracy for Reachability and Free Space.

high compared to the success metrics we see in other tasks (Tables 1 to 3). And, when provided with depth and additionally trained on the Matterport3D [5] dataset, this same method achieves 0.94 SPL and 0.996 Success Rate on this set [33], so the task is regarded as "solved". This method consists of task-agnostic components and is trained for 2.5B steps, demonstrating continual improvements through this training period (although it achieves 90% of total performance after 100M steps). The second-best performing method on the leaderboard appears to predict monocular depth with an external method from RGB for use as an additional input.

Performance. Because the DD-PPO agent has a task-agnostic design and this training method is now commonly used, we adopt it as the baseline for our experiment. We train the ImageNet and CLIP agents in our experiment for 250M steps. Again, our CLIP agent outperforms our ImageNet agent (+0.05 SPL and +0.03 Success Rate, with a 1.8x reduction in distance to goal). Since we based our agent on the best performing leaderboard entry, we expect it to be competitive. However, we do not expect it to surpass that entry, because our model is trained for **10x fewer** steps—training agents at the billion-scale for benchmarking purposes is infeasibly expensive. Still, we observe that our model's performance is still increasing at the end of our training period. And, by the findings in Wijmans *et al.* [33], we hypothesize that it could surpass the best-performing entry by a few points given the same duration of training. Moreover, our CLIP agent beats the challenge winner, which introduced ANM [6], by +0.14 SPL.

5. Why is CLIP effective?

In the previous section, we showed that CLIP representations are remarkably effective across multiple embodied tasks, especially when compared to their ImageNet counterparts. We wish to bring some insight as to *why* this is the case.

To this end, we design a collection of experiments measuring how well visual representations from ImageNet and CLIP pretrained models encode the following semantic and geometric primitives: object presence, object presence at a location, object reachability, and free space. We posit that visual representations that encode these primitives effectively will lead to more capable Embodied AI agents, particularly given the nature of the 4 tasks we are investigating.

For each of these primitives, we train simple (linear) classifiers to predict the outcome of interest from image features (with ResNet-50 encoders pretrained with either ImageNet or CLIP). See Fig. 3 for examples of these tasks. Our results, summarized in Table 5, show that classifiers trained on CLIP representations outperform on each of these tasks by values ranging from 3 to 6.5 points in comparison to those trained on ImageNet representations. Object localization shows a very large improvement (+6.5 absolute and +16% relative improvement). Locating objects within an observation is useful for the two ObjectNav tasks as well as the RoomR task. These results support the improvements we see in Sec. 4: they demonstrate that CLIP features natively encode information that is relevant to navigational and similar embodied skills.

To enable the above evaluation, we have generated a small dataset of frames containing objects from scenes in iTHOR. In iTHOR there are 4 room types (kitchens, living rooms, bedrooms, and bathrooms) for each of which there are 20 train, 5 val, and 5 test scenes. As bathrooms are quite small, we exclude this type, leaving a total of 60 train, 15 val, and 15 test scenes. We sample 100 frames in each train scene and 50 frames from each val/test scene (from random locations and orientations). We additionally generate ground truth data for each frame that corresponds to each task (with details below) with the metadata available in iTHOR. For each frame in this dataset, we extract convolutional features using a ResNet-50 with CLIP or ImageNet pretraining and pool them into embeddings. At each training step, we classify them with a single linear layer and supervise using the ground truth data. We train with a batch size of 128, an Adam optimizer, and a learning rate of 0.001. We evaluate the best validation checkpoint for each model on our test set.

Pooling. We use two methods to pool the $2048 \times 7 \times 7$ feature tensor created by the ResNet-50 encoder into a vector embedding. Note: pooling for "object localization" differs slightly, as described in the corresponding section below.

Average: We average pool our conv. features from 7×7 to 1×1 and flatten, resulting in a 2048-dim embedding.

Attention: We use the attention pooling mechanism from CLIP to produce a 1024-dim embedding. This is only applicable for features from a CLIP model and this module is

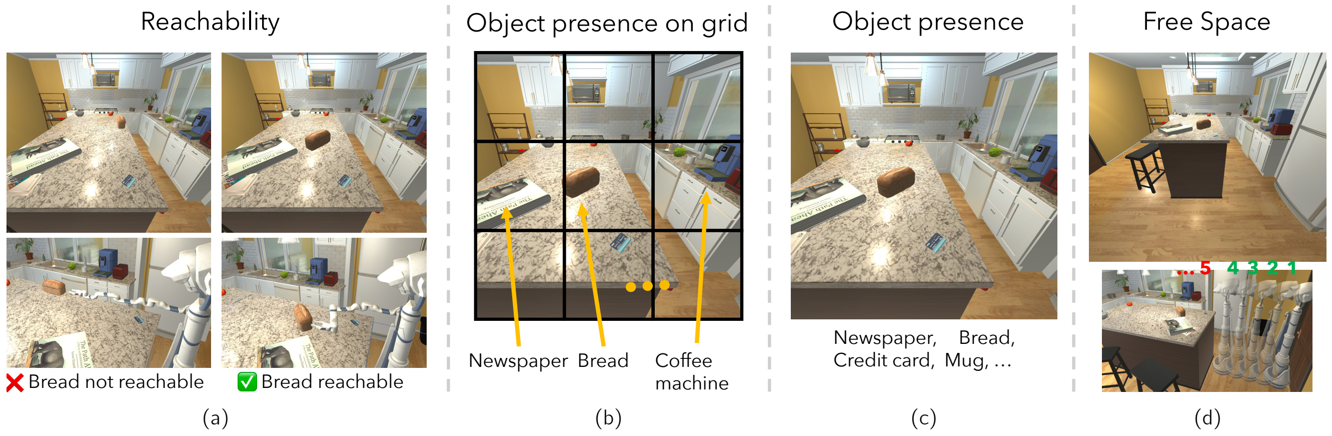

Figure 3. **Examples of visual encoder evaluations.** (a) Top: Two egocentric images, one in which the bread is reachable and one in which the bread is not reachable; bottom: two third-person images demonstrating the meaning of reachability (for visualization purposes only). (b) For each location on a 3×3 grid, the model must predict which objects are visible. (c) Predicting which objects are visible in an image in a multi-label fashion. (d) Top: image given to the model from which it must predict how many steps forward it can take before colliding with an object; bottom: third person viewpoint showing that the agent can, in this case, take 4 steps forward before colliding with the table.

not trained further in our experiments.

Object presence. In this task, we train our models to predict whether objects are present in the given image. See Fig. 3c. This task is not trivially solved by object detection, because objects in our images may be very small (*e.g.* only a few pixels wide)—like in the embodied task of object navigation, our classifier needs to indicate whether an object is likely to be present ahead, even if it is not explicitly visible. We have selected a set of 52 object categories, so the linear layer has an output dimension of 52. We apply a Sigmoid activation on the outputs of this classifier to produce independent class probabilities $o = (o_0, \ldots, o_{51})$ (where $o_i > 0.5$ indicates that an object of category i is present somewhere in the image). We supervise this with a binary cross-entropy loss.

Object localization. This task is quite similar to "object presence" (above) and uses the same set of 52 object categories. We divide the image into a 3×3 rectangular grid and the model must now predict whether objects are present in each of the grid sections (see Fig. 3b). Spatial information about objects is highly relevant for navigational tasks (*e.g.* ObjectNav), as this should help agents orient their vision and movement. We average pool our convolutional features from 7×7 to 3×3 (*i.e.* corresponding to each grid section), instead of 1×1. We also disregard attention pooling of CLIP features here. During training, we then convolve these features with a 1×1 kernel from 2048 to 52 channels (*i.e.* number of objects)—this convolution is equivalent to a single linear layer. We apply a Sigmoid activation and then have indicators for the presence of each object in each grid section. Our prediction is supervised with a binary cross-entropy loss.

Reachability. In this task, we train our models to predict whether an agent would be able to reach out and pick up some type of object from its current pose (see Fig. 3a). This skill is especially relevant for embodied tasks like ArmPointNav in ManipulaTHOR [9], which involves picking up and moving objects with a robotic arm. Our dataset has 57 object categories and we have balanced the number of positive and negative examples per category (*i.e.* whether an object is reachable when present). Our linear layer accordingly has an output dimension of 57 and we apply a Sigmoid activation on its outputs to produce independent class probabilities. During training, we iterate over each object in the set of all training images and supervise with a binary cross-entropy loss (between whether that object is indeed reachable and the model's predicted probability for that object class).

Free space. In this task, we train our models to predict how many steps an agent can move forward (in 0.25m increments) given an egocentric view (see Fig. 3d). Determining the amount of free space ahead is important for agents to plan where to move and how to avoid collisions. We design this as a classification problem for classes $c = \{0, \ldots, 9, \geq 10\}$, so the linear layer has an output dimension of 11. We apply a Softmax activation on the outputs of this classifier to produce probabilities and supervise these with a categorical cross-entropy loss.

6. Does ImageNet performance correlate with Embodied AI success?

Our analysis in Sec. 4 shows that using CLIP encoders provides large gains across a suite of benchmarks. Now, we ask: How do we choose an appropriate visual encoder when

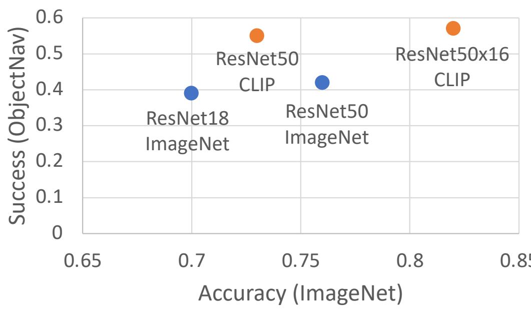

Figure 4. Comparison of OBJECTNAV Success Rates by ImageNet accuracies for ResNet models with ImageNet or CLIP pretraining.

Seen Objects	Unseen Objects				
All	All	Apple	Basketball	House Plant	Television
0.170	0.081	0.147	0.067	0.053	0.060

Table 6. **Zero-shot Success Rates for OBJECTNAV.** We report metrics for 8 seen and 4 unseen objects in validation scenes.

the next generation of networks get built? Is high accuracy on ImageNet sufficient to predict a high Success Rate on Embodied AI tasks?

To answer this question, we use 4 pretrained visual encoders and train OBJECTNAV models using each of them. We use the same architecture for each, as described in Sec. 3. The four encoders we use are: ResNet-18 and ResNet-50 pretrained on ImageNet, and ResNet-50 and ResNet-50x16 pretrained via CLIP. Fig. 4 shows a plot of OBJECTNAV Success Rate (Test) vs ImageNet Top-1 Accuracy. Within the ImageNet models, larger models lead to an improved Top-1 Accuracy and also an improved Success Rate, but the gain in Success Rate is small. This is also true within the CLIP models. However, the Success Rate gains between the ImageNet and CLIP models is larger. When we consider the same architecture (ResNet-50), higher Top-1 Accuracy does not lead to a higher Success Rate.

These results and our probing experiments (Sec. 5) suggest that Embodied AI is still limited by visual encoders and there seems to be room to improve merely by improving the visual encoding. However, one must probe visual representations for semantic and geometric information; moving forward, merely looking at ImageNet's Top-1 Accuracy alone is not a good indicator for success in Embodied AI.

7. Zero-shot OBJECTNAV

We are highly motivated by the effectiveness of CLIP's vision and language encoders in zero-shot image classification. As we've seen that CLIP produces meaningful visual representations for observations in an embodied environment (Sec. 5), we are hopeful that OBJECTNAV agents can learn using object goals encoded through the CLIP text encoder. We train an agent in RoboTHOR for 60M steps with the simple baseline model in Fig. 5. This model has very few learnable parameters and operates entirely on CLIP representations as inputs. We train on 8 of the 12 objects and show success metrics from evaluation in Table 6 for seen and unseen objects. On seen objects, this agent achieves a 0.17 Success Rate, while on unseen it achieves 0.08—roughly half. This is an interesting result, particularly when one considers the simplicity of the model used and the small number of episodes trained (just 60M) in comparison to other models in this paper and and on leaderboards. At 60M episodes, the agent achieves a 0.45 Success Rate for training scenes, so we expect this model to keep improving.

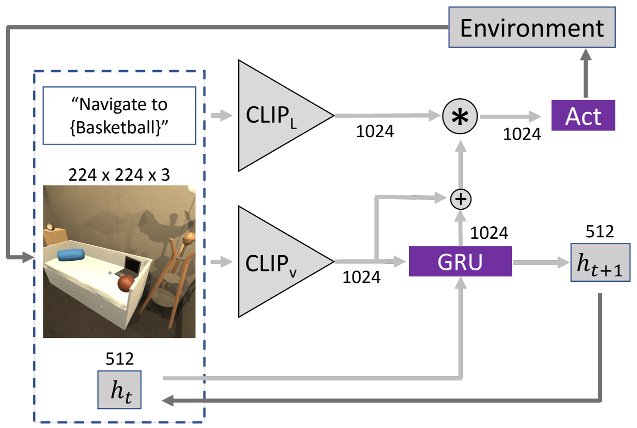

Figure 5. **Zero-shot OBJECTNAV Model Architecture.** This model has very few learnable parameters (in purple), including the GRU, the actor (1024 → 6 linear layer) and critic (1024 → 1 linear layer).

This is a good first step, and future work should build upon this result to explore architectures and training strategies that can improve zero-shot performance.

8. Discussion

Limitations. We employed frozen backbones for our experiments. Developing robust training techniques to update the backbones while completing the task is an interesting direction to explore. Another limitation of the work is that our encoders are trained with visual and textual information. Incorporating interactive actions during representation learning can potentially lead to richer representations.

Conclusion. We demonstrated the effectiveness of CLIP representations for navigation-heavy Embodied AI tasks, even beating specialized architectures at some challenges. To diagnose CLIP's effectiveness, we studied how well these representations encode relevant primitives for navigational tasks and found that CLIP models outperformed ImageNet counterparts. Finally, we also leveraged CLIP to enable an initial baseline for zero-shot OBJECTNAV.

References

[1] Robothor challenge action boost. https://github.com/964728623/robothor_challenge_objnav21. 4

[2] Abhishek Kadian*, Joanne Truong*, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? *IEEE Robotics and Automation Letters*, 2020. 2

[3] Dhruv Batra, Angel Xuan Chang, S. Chernova, Andrew J. Davison, Jun Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, Manolis Savva, and Hao Su. Rearrangement: A challenge for embodied ai. *arXiv*, 2020. 1, 2, 4

[4] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects. *arXiv*, 2020. 1, 2

[5] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *3DV*, 2017. 2, 5, 6

[6] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020. 4, 5, 6

[7] Prithvijit Chattopadhyay, Judy Hoffman, Roozbeh Mottaghi, and Ani Kembhavi. Robustnav: Towards benchmarking robustness in embodied navigation. In *ICCV*, 2021. 4

[8] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. In *CVPR*, 2020. 2, 4

[9] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ManipulaTHOR: A Framework for Visual Object Manipulation. In *CVPR*, 2021. 2, 7

[10] Han Fang, Pengfei Xiong, Luhui Xu, and Yu Chen. Clip2video: Mastering video-text retrieval via image clip. *arXiv*, 2021. 2

[11] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *arXiv*, 2021. 2

[12] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. *arXiv*, 2021. 1, 2

[13] Unnat Jain, Iou-Jen Liu, Svetlana Lazebnik, Aniruddha Kembhavi, Luca Weihs, and Alexander G. Schwing. Gridtopix: Training embodied agents with minimal supervision. In *ICCV*, 2021. 4

[14] Unnat Jain, Luca Weihs, Eric Kolve, Mohammad Rastegari, Svetlana Lazebnik, Ali Farhadi, Alexander G. Schwing, and Aniruddha Kembhavi. Two body problem: Collaborative visual task completion. In *CVPR*, 2019. 2

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

[16] Chengshu Li, Fei Xia, Roberto Mart'in-Mart'in, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *CoRL*, 2021. 2

[17] Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. Clip4clip: An empirical study of clip for end to end video clip retrieval. *arXiv*, 2021. 2

[18] Oleksandr Maksymets, Vincent Cartillier, Aaron Gokaslan, Erik Wijmans, Wojciech Galuba, Stefan Lee, and Dhruv Batra. Thda: Treasure hunt data augmentation for semantic navigation. In *ICCV*, 2021. 5

[19] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and D. Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. *arXiv*, 2021. 2

[20] Jes'us Andr'es Portillo-Quintero, José carlos Ortíz-Bayliss, and Hugo Terashima-Mar'in. A straightforward framework for video retrieval using clip. *arXiv*, 2021. 2

[21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1, 2, 3

[22] Santhosh K. Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel Xuan Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv*, 2021. 2

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

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

[25] Sheng Shen, Liunian Harold Li, Hao Tan, Mohit Bansal, Anna Rohrbach, Kai-Wei Chang, Zhewei Yao, and Kurt Keutzer. How much can clip benefit vision-and-language tasks? *arXiv*, 2021. 1, 2

[26] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *CoRL*, 2021. 2

[27] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting

Grounded Instructions for Everyday Tasks. In *CVPR*, 2020. 2

[28] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. *arXiv*, 2021. 2

[29] Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. Visual room rearrangement. In *CVPR*, 2021. 2, 4

[30] Luca Weihs, Unnat Jain, Iou-Jen Liu, Jordi Salvador, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander Schwing. Bridging the imitation gap by adaptive insubordination. In *NeurIPS*, 2021. the first two authors contributed equally. 4

[31] Luca Weihs, Jordi Salvador, Klemen Kotar, Unnat Jain, Kuo-Hao Zeng, Roozbeh Mottaghi, and Aniruddha Kembhavi. Allenact: A framework for embodied ai research. *arXiv*, 2020. 4

[32] Erik Wijmans, Irfan Essa, and Dhruv Batra. How to train pointgoal navigation agents on a (sample and compute) budget. *arXiv*, 2020. 2

[33] Erik Wijmans, Abhishek Kadian, Ari S. Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *ICLR*, 2020. 2, 5, 6

[34] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: real-world perception for embodied agents. In *CVPR*, 2018. 2, 5

[35] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectgoal navigation. In *ICCV*, 2021. 5

[36] Andy Zeng, Peter R. Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. In *CoRL*, 2020. 2

A. Baseline Architecture Details

We provide further details here for implementations of baseline architectures for tasks in Sec. 4. We have already given such a description for RoboTHOR OBJECTNAV in Sec. 3. Each of these architectures is a replica of the baseline provided by task authors, with the visual encoder substituted with a frozen CLIP ResNet-50 encoder. And, for our ImageNet baseline agents, we simply use a frozen ResNet-50 that is pretrained on ImageNet instead.

A.1. Room Rearrangement in iTHOR

The model for ROOMR in iTHOR receives two $3{\times}224{\times}224$ RGB images at every step i_1 and i_2. These RGB image are encoded into $2048{\times}7{\times}7$ tensors I_1 and I_2 by a CLIP ResNet-50 model whose weights are frozen and final attention pooling and classification layers have been removed. These feature maps are stacked to form a $6144{\times}7{\times}7$ tensor $s = [I_1, I_2, I_1 * I_2]$. An attention mask formed by applying a $1{\times}1$ convolution to s with an output dimension of 512. s is then convolved to 512 channels (resulting in a shape of $512{\times}7{\times}7$) with another $1{\times}1$ kernel. The attention mask is then used for attention pooling, resulting in a 512-dim embedding V. V is passed into a 1-layer GRU with 512 hidden units, along with any prior hidden state. An actor head (one linear layer) maps the GRU output to a 6-dimensional vector of logits and a critic head (another linear layer) maps it to a scalar.

A.2. Habitat OBJECTNAV and POINTNAV

We use the same architecture for our POINTNAV and OBJECTNAV baselines in Habitat, with the only difference being the input goal and how it is encoded (to a 32-dim encoding G). Like in RoboTHOR, the input goal for Habitat OBJECTNAV is an integer $g \in \{0, ..., 20\}$ indicating an object category. In this case, g is used to index an embedding matrix and form G. In Habitat POINTNAV, the input goal is a 2-dim polar coordinate (to the target position, expressed relatively to the agent's current position). Here, g is passed through a linear layer to form G. Weights for both the embedding matrix and linear layer are learned during training.

The model receives G, a $3{\times}224{\times}224$ RGB image i, the action from the previous step a (as an integer index in the action space). The RGB image is encoded into a $2048{\times}7{\times}7$ tensor I by a CLIP ResNet-50 model whose weights are frozen and final attention pooling and classification layers have been removed. I is average pooled spatially (from $7 times 7$ to $1 times 1$) and flattened to form V: a 2048-dim goal-conditioned visual embedding. The previous action a is used to index an embedding matrix to form a 32-dim encoding A. G, V, and A are passed into a 2-layer GRU with 512 hidden units, along with any prior hidden state. An actor head (one linear layer) maps the GRU output to a 6-dimensional vector of logits and a critic head (another linear layer) maps it to a scalar.