OVRL-v2: Semantic Navigation without Semantic Mapping or Detection

Karmesh Yadav[1] Arjun Majumdar[2] Ram Ramrakhya[2] Alexei Baevski[1]
Oleksandr Maksymets[1] Dhruv Batra[1,2]
[1]Meta AI [2]Georgia Institute of Technology

Abstract

Can an autonomous agent navigate to semantic goals (e.g. *'find chair' or 'goto location in <this picture>'*) *in new environments without any semantic detection or mapping modules? Intuition would suggest 'no', but what does the empirical evidence say? We present a single neural network architecture composed of task-agnostic components (ViTs, convolutions, and LSTMs) that achieves state-of-art results on both* IMAGENAV *and* OBJECTNAV *without any object detection, segmentation, or mapping modules.*

First, we present a negative result – naively replacing ResNets with ViTs does not work for navigation tasks. Second, we identify the key bottleneck – both `[CLS]` *token embedding and global-average pooling remove spatial structure that is important for navigation. We propose using a compression layer operating over ViT patch representations to preserve spatial information, and find that it leads to ViTs outperforming ResNets. Third, we find that ViT models with compression layers are so good at learning, that they expose and 'hack' a flaw in the reward proposed by prior work [1], which leads them to never end the episode, and move in and out of the target location collecting a reward. Fortunately, we show that there is a simple fix. Finally, we demonstrate* positive scaling laws for the first time – *increasing model size from ViT-*SMALL *to ViT-*BASE *improves success rates.*

Putting it all together, we present a simple ViT+LSTM architecture that advances state-of-art on IMAGENAV *from 54.2% to 82.0% success (+27.8, 51.3% relative) and on* OBJECTNAV *from 60.0% to 64.0% (+4.0, 6.7% relative). Our agents use only RGB and GPS+Compass; no egocentric depth, no egocentric semantic segmentation, no object detection, no semantic, geometric, or topological mapping.*

1. Introduction

Imagine a home assistant robot that can find things in the house. For instance, we might ask it in natural or templated language to *'find a sweatshirt'*. Or we may show the agent a picture of our favorite sweatshirt and ask it to find it. Designing systems for *autonomous navigation to semantic goals* is a challenge of broad scientific and societal interest.

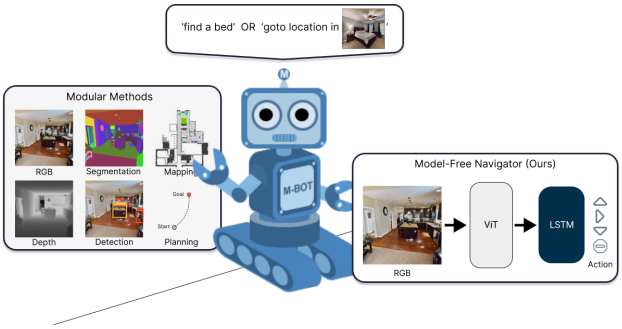

Figure 1. **OVRL-v2** is a model-free navigator with a ViT+LSTM architecture that achieves SOTA results on IMAGENAV and OBJECTNAV without mapping, detectors, or segmentors of any kind.

In the embodied AI research community, two concrete goal specifications have emerged for semantic navigation. In OBJECTNAV [4], an agent is spawned in an unseen environment and asked to find any instance of an object category given its name *'find <name>'*. In IMAGENAV [45], it is asked to find a location 'described' by a goal image.

Both problems test the agent's semantic understanding and episodic memory – what objects or parts of a scene are visible in the goal image (is it part of a kitchen or bathroom)? Where are these objects (seen in the goal image in IMAGENAV or mentioned by name in OBJECTNAV) typically found in a house? Where does the agent find itself at initialization? And how should it strategically search the environment to find the object or scene that it is looking for without looping over the same area multiple times?

These questions seem intuitive, and this intuition leads us to expect that an agent capable of performing well at these tasks must consist of semantic detection and mapping modules – *e.g.*, detecting objects (or extracting semantic features) in 2D images [28] or in 3D point clouds, accumulating detections (or features) into a 2D (top-down) or 3D map [7, 8, 25, 32, 44], using detections to build relational graphs [41], or create topological maps [15].

In this work, we challenge this intuition and present a surprising result – a single neural network architecture composed of task-agnostic components (ViTs, convolutions, and LSTMs) shown in Fig. 1 achieves state-of-the-art re-

sults on both IMAGENAV and OBJECTNAV– without any detection, segmentation, or mapping modules of any kind.

Our key technical contributions and insights are as follows:

1. **Negative Result: ViTs under-perform ResNets.** In order to achieve strong results without any task-specific modules, we require a high-capacity state-of-the-art general-purpose visual 'backbone' such as vision transformers (ViTs). However, our first finding is a negative result on naively following this modern trend of replacing ResNets with ViTs [6, 10]. Specifically, we find that ViT-based agents trained from scratch perform quite poorly on IMAGENAV compared to ResNets (achieving only 36.1% success *vs*. 59.9% for ResNets). This is despite a substantially higher model capacity (ViT-SMALL has ∼4 times more parameters than a half-width ResNet50).

2. **Primary Finding: Compression layers rescue ViTs.** We find that a key issue with using ViTs for navigation problems is that both [CLS] token embedding and global-average pooling remove a spatial structure that is important for the task. We propose using a compression layer (consisting of a 2D convolution plus flattening) operating over ViT patch representations to preserve spatial information, and find that it leads to ViTs outperforming ResNets (67.4% *vs*. 59.9% success on IMAGENAV).

3. **Secondary Finding: ViT agents learn to hack rewards.** In fact, ViT models with compression layers are so good at learning, they expose a flaw in the reward proposed by prior work [1]. Specifically, [1] proposed a reward-shaping term to encourage the agent to look at the goal upon reaching it. Our ViT-based agent discovers that this reward can be 'hacked' by never-ending the episode, moving towards the target, turning to look at the goal, moving away from the target, turning back and repeating. We show that the reason for this behavior is that the reward term from [1] is not a difference of potential functions as recommended by the theory [22]. Interestingly, we only noticed this theoretical mistake after our experiments 'started working'. Fortunately, there is a simple fix to the reward, which we propose.

4. **Key Result: visual pretraining unlocks positive scaling laws.** We demonstrate for the first time, *positive scaling laws* with ViT-based agents on IMAGENAV. Specifically, we find that visual representation learning (using masked autoencoding (MAE) [16]) not only improves performance, but also enables model scaling. With this pretraining, we are able to increase a model size by switching from ViT-SMALL to ViT-BASE and observe gains in success rate from 80.5% to 82.0% and SPL (success weighted by path efficiency) from 55.2% to 58.7%. These encouraging trends stand in contrast to negative results in prior work (*e.g*., [35]) that did not observe any benefits from scaling.

5. **Single architecture achieves SOTA on IMAGENAV and OBJECTNAV.** Putting it all together (ViTs, compression layers, reward fixing, pretraining, and scaling), we present a simple ViT+LSTM architecture that pushes state-of-the-art success rate on IMAGENAV from 54.2% (by [37]) to 82.0% (+27.8, 51.3% relative improvement) and on OBJECTNAV from 60.0% (by [7]) to 64.0% (+4.0, 6.7% relative improvement). Our agents use only RGB and GPS+Compass; no egocentric depth (as used by [28]), no egocentric semantic segmentation (as used by [28]), no object detection (as used by [41]), no semantic or geometric mapping (as used by [7, 8, 25, 32, 44]). We believe our work unlocks a new line of work in embodied AI.

2. Related Work

Semantic Navigation. Semantic Navigation approaches can be divided into three categories: a) *Classical methods* [40] - b) Modular learning methods [7, 8, 15, 26, 28, 32] and c) End-to-end learning approaches [1, 33, 37, 43]. While many modular learning methods build explicit semantic maps [8, 15], others simply use object detectors or segmentors without mapping [20, 28]. In comparison, we show that it is possible to achieve state-of-the-art performance on semantic navigation tasks without using semantic mapping, object detection, or segmentation of any kind.

Self-Supervised Learning (SSL) for Computer Vision. Recently, there has been increasing interest in self-supervised visual representation learning [3, 5, 9, 10, 16]. SSL algorithms differ in the kind of objectives they use: contrastive [9, 10] *vs*. non-contrastive, where non-contrastive objectives could be distillation-based [3, 5] or reconstruction-based [16]. [6] showed how these models trained without supervision manage to capture semantic information in their representations. SSL methods have also been coupled with state-of-the-art architectures like the vision transformer (ViT) [12] to build even more powerful models [10, 16]. In this work, we use [16] to pretrain ViT-based visual encoders used for semantic navigation.

Self Supervised Learning (SSL) for Embodied AI. Recent methods have explored using self-supervised visual representations to learn effective control policies [24, 30]. [24] demonstrated the efficacy of frozen MAE representations for motor control tasks, while [30] proposed incorporating a reconstruction-based self-supervised objective alongside online model-based RL training.

Closely related to this work, EmbCLIP [17] and OVRL [38] demonstrate the effectiveness of visual pretraining for visual navigation. EmbCLIP [17] uses off-the-shelf CLIP [23] encoders, which are frozen during policy learning. In our work, we show the effectiveness of finetuning, not just for our method but also for ViT-based CLIP models. Both OVRL [38] and EmbCLIP [17] use ResNet-based visual encoders. By contrast, in this work we focus on adapting more recent ViT-based backbones for visual navigation.

3. Background: Tasks and Visual Pretraining

This work studies two semantic navigation tasks: image-goal navigation (IMAGENAV) [45] and object-goal navigation (OBJECTNAV) [4]. To address these tasks, we design an embodied agent leveraging a vision transformer (ViT) [12]. This section provides an overview of each task, then describes an approach that we use for pretraining ViTs.

3.1. Semantic Navigation

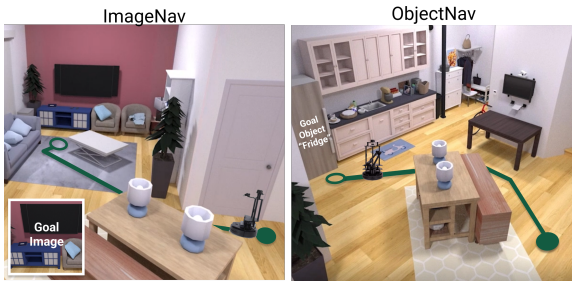

Figure 2. **Semantic Navigation Tasks.** In IMAGENAV [45] the goal is 'described' by an image and in OBJECTNAV [4] the goal is described in words (e.g., *'fridge'*). We demonstrate the effectiveness of our 'model-free navigator' (*i.e.*, agent) on both tasks.

Fig. 2 illustrates the IMAGENAV [45] and OBJECTNAV [4] tasks. In both, an agent starts at a random position and orientation in a 3D environment. The agent must explore the environment to find a goal location. In IMAGENAV, the goal is an image (*e.g.*, a picture of a sofa) that is taken from the goal position. In OBJECTNAV, the agent is given the name of an object (*e.g.*, *'sofa'*) that it needs to find.

In these tasks, agents perceive the environment using an egocentric RGB camera. Agents navigate using a discrete action space. In IMAGENAV, the standard set of actions includes: MOVE_FORWARD ($0.25m$), TURN_LEFT ($30°$), TURN_RIGHT ($30°$) and STOP to indicate that the agent thinks it has reached the goal. In OBJECTNAV, agents can also LOOK_UP ($30°$) and LOOK_DOWN ($30°$).

Agents are evaluated in new environments that are not used during training, which allows measuring how well navigation behaviors generalize. Two standard metrics are used to assess agent's navigation performance: the success rate (SR) and success weighted by path length (SPL) [2]. SPL rewards agents that take shorter paths to the goal, thus measuring how efficiently the agent explores new environments.

3.2. Masked Autoencoders (MAEs)

Semantic navigation tasks require understanding visual cues to navigate in new environments. Thus, agents require strong visual representations. We use masked autoencoding (MAE) [16] – an efficient self-supervised visual representation learning algorithm designed for pretraining vision transformers [12] (ViTs) – to improve the performance of our ViT-based agent. As illustrated in Fig. 3, MAE derives its efficiency from an asymmetric encoder-decoder design. Specifically, an input image is first divided into non-overlapping patches, a high fraction (75%) of which are randomly masked during pretraining. The encoder only processes the remaining unmasked patches, which reduces the computational burden during pretraining. A small decoder is tasked with reconstructing the full input image. Both the encoder and decoder are ViTs, which naturally handle processing the variable number of patches. The high masking percentage is achievable due to the natural redundancy across patches in real-world images, which makes the full image predictable from only a small subset of the constituent parts. After pretraining, the decoder is discarded, and only the encoder is used for downstream tasks.

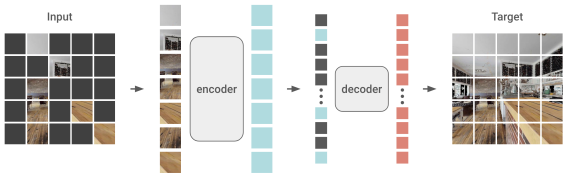

Figure 3. **MAE [16] architecture** used to pretrain ViTs on images from HM3D [27] and Gibson [34]. Figure adapted from [16].

4. Approach

We use a general-purpose agent architecture for *both* semantic navigation tasks (IMAGENAV and OBJECTNAV). As shown in Fig. 4, both agents primarily consist of a from-scratch or pretrained ViT-based visual encoder, a goal encoder, and a recurrent policy network. This section describes several key components of our approach.

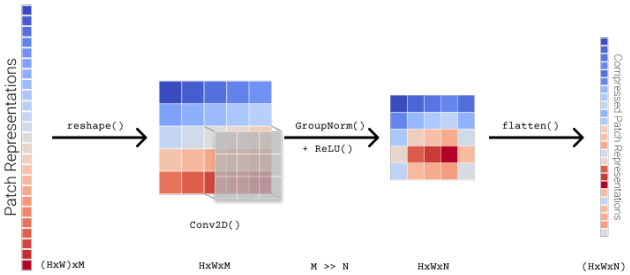

Figure 5. **Compression Layer.** We propose using a compression layer to encode the output patches from a ViT encoder. Unlike [CLS] token representations and global average pooling operations, a compression layer maintains spatially distinct features.

Compression layers for ViTs. As shown in Fig. 4, our semantic navigation agents process the RGB observation O_t with a ViT-based visual encoder $f_{\theta_{obs}}$. Specifically, the input images are converted into non-overlapping 16×16 patches, concatenated with a [CLS] token, and then processed with a ViT, which outputs a representation for each patch and the [CLS] token. In tasks such as image classification, it is common to represent the image using either

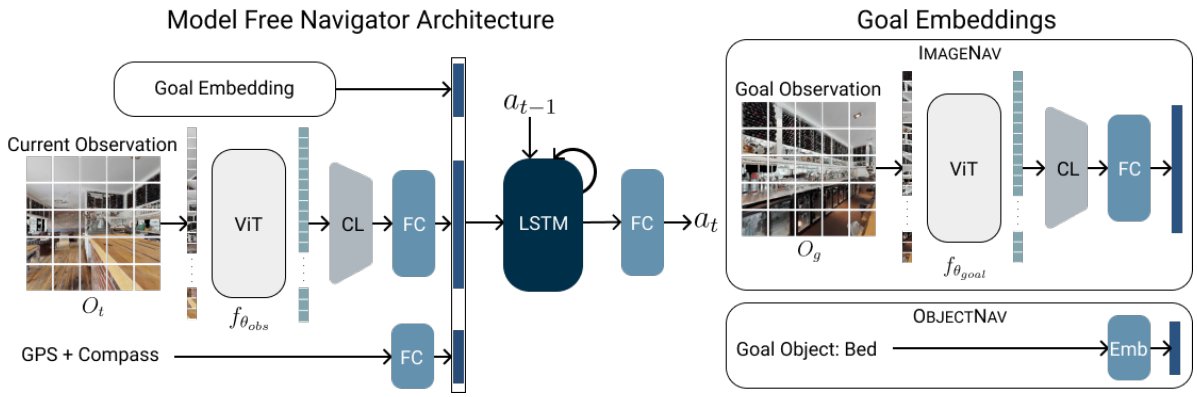

Figure 4. **OVRL-v2 architecture.** In our model-free navigator, observations O_t are encoded using a from-scratch or pretrained ViT then fed to a compression layer (CL) and fully-connected layer (FC). The output representation is concatenated with a goal embedding and (optionally) a GPS+Compass encoding. Finally, an LSTM-based policy outputs actions a_t. In IMAGENAV, the visual encoder pipeline is replicated and used to encode goal images O_g. In OBJECTNAV, the embedding is used to encode categorical object goals (*e.g.*, *'bed'*)

(a) the [CLS] token output or (b) the average pooling of the patch representations (*i.e.*, global average pooling).

We hypothesize that both solutions remove the spatial structure that exists in the patch representations, which may be helpful for visual navigation. Thus, we opt for an alternative approach. As illustrated in Fig. 5, we use a compression layer that first reshapes the patch representations (generated by a ViT) back into a grid. Next, we process them with a convolutional layer that reduces the dimensionality (*i.e.* compresses) of each patch.[1] Finally, we flatten the output to maintain spatially distinct features. A similar technique has been used previously (*e.g.*, [33, 37]) to process grid features produced by a ResNet, which we adapt here for ViTs.

Semantic Navigation with ViTs As illustrated in Fig. 4, the output from the visual encoder f_{obs} is concatenated with a goal representation and an embedding of a GPS+Compass sensor (only used for OBJECTNAV) that provides pose information. The concatenated output is processed by a recurrent LSTM-based policy network, which predicts actions.

The key difference between agents for each task is the method used to encode the goal. In IMAGENAV, the image-goal O_g is encoded with a visual encoder f_{goal} identical to the encoder used for visual observations (f_{obs}). For OBJECTNAV, the object category (*e.g.*, *'sofa'*) is processed by a randomly initialized, learned embedding.

We train our IMAGENAV agent with reinforcement learning (RL) using DD-PPO [33] and the reward function described in the next section. For OBJECTNAV, we train our agent using human demonstrations with a distributed version of behavior cloning [28]. Further training and evaluation details are provided in the appendix.

IMAGENAV Rewards. The reward used for semantic navigation is typically composed of three components: (a) a sparse reward for successfully completing the task, (b) a per-timestep penalty to incentivize efficiently solving the task, and (c) one or more reward-shaping terms to simplify the optimization problem. Commonly, the per-timestep change in the geodesic distance to the goal is used for reward-shaping [29]. Formally, this reward can be written:

$$r_t = c_s \times ([d_t < r_g] \;\&\; [a_t = \texttt{STOP}]) + (d_{t-1} - d_t) - \gamma \tag{1}$$

where c_s is a success weighting, d_t is the geodesic distance to the goal, r_g is the goal radius, a_t is the agent's action, and γ is the per-timestep 'slack' penalty (typically 0.01).

A key limitation of the reward function in Eq. (1) is that it does not require looking at the goal object or image to obtain the full reward. To resolve this issue, [1] proposed an additional angle-to-goal reward based on success and a reward-shaping term that incentivize looking towards the goal when the agent is within the goal radius r_g. Specifically, the ZER [1] reward is defined as:

$$\begin{aligned} r_t = {} & c_s \times ([d_t < r_g] \;\&\; [a_t = \texttt{STOP}]) + \\ & c_a \times ([\theta_t < \theta_g] \;\&\; [a_t = \texttt{STOP}]) \\ & + ([d_t < r_g] \times (\theta_{t-1} - \theta_t)) \\ & + (d_{t-1} - d_t) - \gamma \end{aligned} \tag{2}$$

where c_a is angle success weighting, θ_t is the angle to the goal at timestep t, and θ_g is an angle success threshold (set to 25° in our experiments).

While the reward function in Eq. (2) can improve IMAGENAV performance (as demonstrated in [1]), it has a subtle flaw: the reward is hackable. Specifically, agents can enter and exit the goal radius multiple times to accumulate the angle-to-goal reward term and maximize the reward without ever succeeding at the task. In our experiments, we find that agents are able to learn this undesirable behavior, leading to circuitous paths and poor navigation performance.

[1] The convolution layer is a `3×3 Conv + GroupNorm + ReLU`.

In this work, we propose a principled fix to reward function in Eq. (2). Our key insight is that we can transform the angle-to-goal shaping term ($[d_t < r_g] \times (\theta_{t-1} - \theta_t)$) into a difference of potential functions, which are provably optimal for reward-shaping [22]. Specifically, we define a new angle-to-goal function $\hat{\theta}_t$ that is equal to π outside of the goal radius and equal to the angle-to-goal otherwise:

$$\hat{\theta}_t = \begin{cases} \theta_t & d_t < r_g \\ \pi & d_t \geqslant r_g \end{cases} \tag{3}$$

With this definition, agents can be appropriately rewarded (or penalized) for entering (or exiting) the goal radius using the difference in the per timestep measure $\Delta_{\hat{\theta}_t} = \hat{\theta}_{t-1} - \hat{\theta}_t$. This term will be 0 outside the goal radius, $\geqslant 0$ when entering, and $\leqslant 0$ when exiting. Furthermore, $\Delta_{\hat{\theta}_t}$ has the property that any positive reward accumulated inside the goal radius will entirely be lost if the agent exits, resulting in a zero-sum path. We take advantage of these properties, and propose the full reward structure defined as follows:

$$\begin{aligned} r_t = c_s \times ([d_t < r_g] \;\&\; [a_t = \texttt{STOP}]) + \\ c_a \times ([\theta_t < \theta_g] \;\&\; [a_t = \texttt{STOP}]) \\ + (\hat{\theta}_{t-1} - \hat{\theta}_t) + (d_{t-1} - d_t) - \gamma \end{aligned} \tag{4}$$

In Sec. 5, we find that the reward defined in Eq. (4) substantially improves IMAGENAV performance (particularly in terms of path efficiency as measured by SPL).

Visual Encoder Pretraining. Our proposed approach of using a ViT-based visual encoder within a model-free navigation agent (Fig. 4) can be trained end-to-end from scratch (*e.g.*, using the RL rewards described in the previous section). In addition, we investigate pretraining the ViT-based visual encoder using the masked autoencoding (MAE) algorithm described in Sec. 3.2. For pretraining, we collect an in-domain image dataset from HM3D [27] and Gibson [34] scenes. This follows the observation in prior work (*e.g.*, [35, 37]), which demonstrates that pretraining on in-domain data (as opposed to datasets like ImageNet) improves downstream performance. Further details about the pretraining dataset and hyperprameters are provided in Appendix A.

5. Experimental Findings

In this section, we first establish an IMAGENAV baseline that is competitive with existing SoTA methods. We then use this strong baseline to systematically address the following research questions:

1. **Do ViTs work out-of-the-box for IMAGENAV?** No. We discover that despite higher model capacity, ViT-based agents trained from-scratch underperform smaller ResNet agents by considerable margins.

2. **How does adding a compression layer affect performance?** We find that using a compression layer to maintain spatial structure in image representations significantly improves navigation performance on IMAGENAV.

3. **Does performance scale with larger ViTs?** When trained from scratch we observe mixed results. However, self-supervised visual pretraining reverses this trend, resulting in across-the-board improvements.

4. **Can strong semantic navigation agents 'hack' the reward function in Eq. (4)?** No. Agents that *can* 'hack' the ZER reward [1] are no longer able to 'hack' the new reward function, which includes our proposed corrections.

5. **How does OVRL-V2 performance compare with the IMAGENAV SoTA?** OVRL-V2 significantly improves over prior work, including approaches that use additional cameras that provide panoramic views of the environment.

6. **Do the architectural improvements transfer to OBJECTNAV?** Yes. OVRL-V2 outperforms the OBJECTNAV SoTA in terms of SR without using a depth sensor or segmentation module, as commonly used for OBJECTNAV.

5.1. Establishing a Strong IMAGENAV Baseline

#	Visual Encoder	Reward	AVGPOOL	SPL (↑)	SR (↑)
1	ResNet50	ZER [1]	Yes	18.8	27.7
2	ResNet50	ZER [1]	No	27.9	**60.6**
3	ResNet50	Eq. (4) (ours)	No	**33.5**	59.9

Table 1. **Our IMAGENAV baseline** uses a principled reward function (Eq. (4)) and does not downsample (AVGPOOL) input images.

As a starting point, we use a baseline agent from [37] with a similar architecture to the model-free navigator described in Sec. 4. Instead of the ViT-based visual encoder used in our approach, this baseline uses a half-width ResNet50 with GroupNorm [33] that is trained from scratch. We train this agent with ZER rewards [1] (Eq. (2)) and report results in Tab. 1 row 1.

Next, we make two improvements to strengthen this baseline. First, we discover that removing an AVGPOOL operation that is used in numerous prior works (*e.g.*, [33, 37]) to downsample images before agent processing, significantly improves performance. Specifically, removing this AVGPOOL operation improves IMAGENAV SR by +32.9% absolute and SPL by +9.1% (Tab. 1 row 1 *vs.* 2). We do not use this AVGPOOL in the remaining experiments.

Finally, to avoid the potential for reward hacking (discussed in Sec. 4), we switch to the reward function from Eq. (4). In Tab. 1 row 3, we observe that this leads to a small (-0.7%) drop in SR, but a large improvement in SPL of +5.6% absolute (a +20.1% relative improvement). Unless otherwise specified, we use the corrected reward function from Eq. (4) in the remaining experiments. We use this strengthened baseline from Tab. 1 row 3 to study the effects of switching to a ViT-based visual encoder, next.

5.2. Using ViTs in a Semantic Navigation Agent

Negative result. In Tab. 2 (rows 2 and 3), we switch the visual encoder in the baseline agent from a ResNet50 to a

#	Visual Encoder	SPL (↑)	SR (↑)
1	ResNet50	33.5	59.9
2	ViT-SMALL [CLS]	23.7	36.1
3	ViT-SMALL Global Average Pool	21.1	35.0
4	ViT-SMALL Compression Layer	**37.1**	**67.4**

Table 2. **Compression layers** make ViTs work for IMAGENAV.

ViT-SMALL. In row 2 we use the [CLS] token representation produced by the ViT to represent images. In row 3, we use global average pooling over the patch representations generated by the ViT. As compared with the ResNet50 baseline (row 1), we find that both solutions lead to substantially reduced performance despite the increase in model capacity as measured by number of parameters (50.9M for ViT-SMALL and 21.5M for ResNet50 agent). Specifically, the better of the two options ([CLS] in row 2), results in a drop of -23.8% in SR and -9.8% in SPL. These results indicate that, unfortunately, ViTs are not drop-in replacements for the ResNets commonly used for semantic navigation.

Compression layers rescue ViTs. In Tab. 2 row 4, we discover that using the compression layer described in Sec. 4 reverses the negative results in rows 2 and 3. Specifically, the compression layer (row 4) improves IMAGENAV SR by +7.5% and SPL by +3.6% over the ResNet50 baseline in row 1. This suggests that preserving spatial structure (using a compression layer) is critical for semantic navigation. Thus, we use compression layers in the remaining experiments.

5.3. Scaling with and without Visual Pretraining

#	Visual Encoder	Pretrained	SPL (↑)	SR (↑)
1	ViT-SMALL	No	37.1	67.4
2	ViT-BASE	No	37.7	65.0
3	ViT-SMALL	Yes	55.2	80.5
4	ViT-BASE	Yes	**58.7**	**82.0**

Table 3. **Visual pretraining** using MAE [16] enables positive scaling of the ViT-BASE architectures on IMAGENAV.

The encouraging performance of our ViT-SMALL agent (from Tab. 2 row 4, replicated in Tab. 3 row 1) suggests that further model scaling may lead to additional improvements. Unfortunately, we find that this is not the case. In Tab. 3, we observe that simply switching from the 50.9M parameter ViT-SMALL (row 1) to the 179.2M parameter ViT-BASE (row 2) produces another negative result: SR drops -2.4% while SPL minimally increases by +0.7%.

By contrast, we discover that visual pretraining (rows 3 and 4) resolves this issue. Specifically, we pretrain ViTs using MAE (as described in Sec. 3) on the HGSP dataset (details in Appendix A). First, with pretraining we observe a large boost in navigation performance for the ViT-SMALL agent. Specifically, SR improves by +13.1% and SPL by +18.1% (row 1 *vs*. 3). Next, we find the negative scaling in rows 1 *vs*. 2 is reversed. With pretraining, switching from ViT-SMALL to ViT-BASE results in a +1.5% improvement in SR and +3.5% gain in SPL (rows 3 *vs*. 4) – *i.e*., we finally see *positive* results from model scaling. Such positive scaling was not observed in prior works such as [35].

We refer to this IMAGENAV agent – a ViT-BASE with compression layers, MAE pretraining, and finetuned using our principled rewards – as OVRL-v2. The accumulated improvements from our initial baseline are +54.3% in SR and +39.9% in SPL (Tab. 1 row 1 *vs*. Tab. 3 row 4).

5.4. Comparing Reward Functions

#	Visual Encoder	Pretrained	Reward	SPL (↑)	SR (↑)
1	ViT-BASE	Yes	ZER [1]	44.1	76.5
2	ViT-BASE	Yes	Eq. (4) (ours)	**58.7**	**82.0**

Table 4. **Our corrected reward function** mitigates reward hacking, which leads to improved IMAGENAV performance.

In Tab. 4, we revisit the reward function used to train our semantic navigation agents to confirm that using our corrected reward function (Eq. (4)) is indeed necessary. In row 1, we find that switching back to the ZER reward (Eq. (2)) results in a substantial performance drop of -5.5% in SR and a -14.6% in SPL. In Fig. 6, we observe that this drop is due in-part to reward hacking. Specifically, after 300M step of training the agent using ZER rewards [1] (orange) learns to hack the reward. This corresponds to a dramatic increase in the training reward (Fig. 6 left), yet precipitous drops in SPL (middle) and SR (right). Using our fix (blue), OVRL-v2 does not hack the reward and we observe steady improvements in performance over the course of training.

5.5. Comparisons with the IMAGENAV SoTA

#	Method	Cameras	SPL (↑)	SR (↑)
1	ZER [1]	1 Cam	21.6	29.2
2	ZSON [19]	1 Cam	28.0	36.9
3	CRL [46]	1 Cam	10.2	20.4
4	OVRL-V1 [37]	1 Cam	27.0	54.2
5	Mem-Aug Nav [21]	4 Cam	56.0	69.0
6	CLIP ViT-BASE (baseline)	1 Cam	37.4	51.7
7	ViT-SMALL (baseline)	1 Cam	37.1	67.4
8	OVRL-V2 (ours)	1 Cam	**58.7**	**82.0**

Table 5. **IMAGENAV performance** of our method (row 8) compared with prior SoTA (rows 1 - 5) in the single camera and 4 camera setups. OVRL-V2 uses a single camera and attains better performance than methods from both the setups.

In Tab. 5, we compare OVRL-V2 (row 8) with prior work on IMAGENAV (rows 1 - 5)[2] and two baselines: a

[2]Details about each method is provided in the appendix (Appendix B).

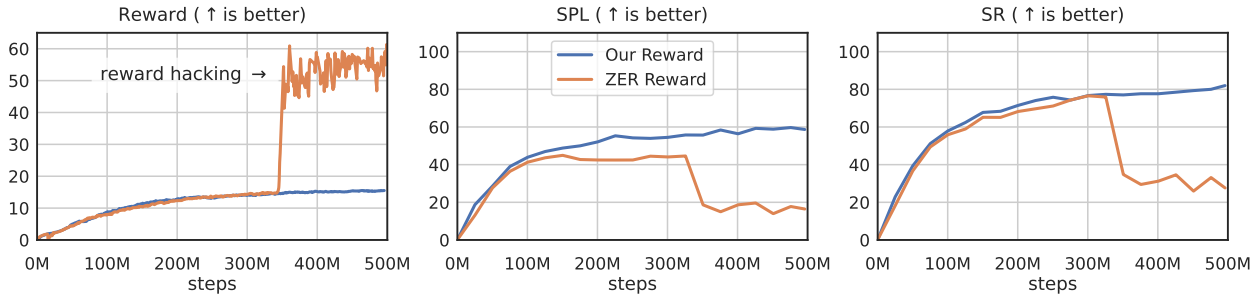

Figure 6. **Reward hacking.** With the ZER reward [1] (orange curve) agents learn to hack the reward leading to large increases in training reward (left), yet substantial drops in validation path efficiency or SPL (middle) and success rate or SR (right). This undesirable behavior is resolved with the reward function introduced in Eq. (4) (blue curve), and performance steadily increases during training.

			VAL		TEST-STANDARD	
#	Method	Camera	SPL (↑)	SR (↑)	SPL (↑)	SR (↑)
1	DD-PPO [36]	RGBD	14.2	27.9	12.0	26.0
2	Habitat-Web [28]	RGBD-S	23.8	57.6	22.0	55.0
3	Habitat-Web*	RGB	16.0	41.5	-	-
4	ProcTHOR [11]	RGB	-	-	32.0	54.0
5	Stretch [7]	RGBD-S	-	-	34.0	60.0
6	BadSeed	-	-	-	33.0	65.0
7	ByteBOT	-	-	-	37.0	68.0
8	OVRL-v1 *	RGB	26.8	62.0	27.0	60.0
9	OVRL-v2 (ours)	RGB	28.1	**64.7**	29.0	64.0

Table 6. **ObjectNav results** on the HM3DSem Val and Test-Standard splits. We compare to prior work on the HM3DSem dataset and attain highest success rate on both the splits. Unpublished works that were submitted to the Test-Standard ObjectNav leaderboard are in gray. (* denotes our implementation).

version of our full approach that uses the CLIP visual encoder (similar to EmbCLIP [17]) instead of MAE (row 6) and the ViT-Small baseline first presented in Tab. 2 row 4, which uses our full method with the ViT-Small architecture and without MAE pretraining (row 7). We observe that OVRL-v2 outperforms all of these methods, exceeding the next best single camera (1 Cam) method, OVRL-v1 (row 4) by +27.8% in SR and +31.7% in SPL. OVRL-v2 also outperforms Mem-Aug Nav [21], a method that uses 4 cameras providing panoramic views of the environment and goal, by +13.0% in SR and 2.7% in SPL.

Additionally, we observe that the ViT-Small baseline (row 7) is competitive with state-of-the-art methods, achieving the second highest SR and third highest SPL. Finally, the CLIP ViT-Base baseline (row 6) significantly underperforms OVRL-v2 with a drop of -30.3% in SR and -21.3% in SPL. This indicates that self-supervised pretraining – opposed to CLIP's large-scale vision-and-language pretraining – is a key ingredient in the success of OVRL-v2.

5.6. Transferring to ObjectNav

This section presents a comparison of OVRL-v2 with prior works on ObjectNav. In Tab. 6, we find that OVRL-v2 achieves 64.0% SR and 29.0% SPL on ObjectNav Test-Standard split (row 9). This improves over the the prior state-of-the-art (Stretch [7]) in terms of SR by +4.0% (64.0% row 9 *vs*. 60.0% row 5). Stretch [7] uses a depth camera and an explicit semantic prediction and mapping module, and outperforms OVRL-v2 in terms of SPL by +5% (34.0% *vs*. 29.0% in rows 5 *vs*. 9). These results indicate that additional sensors (depth) and modules (semantic prediction and mapping) are not required for SoTA SR in ObjectNav. However, they can improve path efficiency.

Similarly, OVRL-v2 outperforms Habitat-Web [28] (row 2), which uses an explicit semantic predictor, by +9.0% on SR and +7.0% on SPL (rows 2 *vs*. 9). We also compare to ProcTHOR [11] (row 4), which uses procedural scene generation to pretrain a policy in $10k$ environments, then finetunes on HM3D ObjectNav. ProcTHOR achieves 54.0% SR and 32.0% SPL which is -10% worse on SR and +3.0% better on SPL compared to OVRL-v2. Our approach improves over OVRL-v1 by +4.0% in SR and +2.0% SPL.

Additionally, we compare OVRL-v2 with methods that do not use pretrained visual encoders. First, we compare with a Habitat-Web RGB only baseline (row 3) and find OVRL-v2 is +27.7% better on SR and +14.1% better on SPL, demonstrating the value in using a pretrained visual encoder for imitation learning (IL). We also compare with a DD-PPO [36] RGBD RL baseline (row 1), and find OVRL-v2 is +36.8% better on SR and +13.9% better on SPL.

6. Analysis and Ablations

This section presents ablations and a failure analysis.

SSL Algorithm In Tab. 7a, we use different pretraining algorithms to initialize the visual encoder in our ImageNav agent. We find that Data2Vec [3] (row 3) attains similar performance to the MAE [16] (row 4) initialization we use in Sec. 5. Both methods substantially outperform a variation of OVRL-v2 initialized with CLIP [23] weights and then finetune (row 2) or frozen (row 1) as done in EmbCLIP [17].

Using the visual representations. In Tab. 7b, we study the

SSL Algorithm	SPL (↑)	SR (↑)
CLIP (Frozen) [17]	51.7	37.4
CLIP (Finetuned)	65.5	47.7
Data2Vec	**59.3**	81.0
MAE	58.7	**82.0**

(a) **SSL algorithms** MAE and Data2Vec perform similarly and surpass (frozen or finetuned) CLIP.

Representation Type	SPL (↑)	SR (↑)
[CLS]	55.0	73.8
Global Average Pool	**59.3**	76.9
Compression Layer	58.7	**82.0**

(b) **Compression layers** work better than using the [CLS] token or global average pooling.

Encoder LR	SPL (↑)	SR (↑)
2.5×10^{-4} (Default)	37.8	51.0
1.5×10^{-6} (Tuned)	**58.7**	**82.0**
0.0 (Frozen)	43.2	61.8

(c) **Learning rate tuning** substantially improves IMAGENAV performance.

Augmentations	SPL (↑)	SR (↑)
False	43.2	57.8
True	**58.7**	**82.0**

(d) **Without augmentations** agents overfit in training.

Pretraining Dataset	SPL (↑)	SR (↑)
OSD	57.5	81.1
HGSP	**58.7**	**82.0**

(e) **Pretraining datasets** minimally impact performance.

Table 7. **Ablations** of OVRL-V2 on IMAGENAV.

impact of using a compression layer with a pretrained visual encoder (supplementing from-scratch analysis in Sec. 5.2). First, we find that global average pooling (row 2) outperforms using the [CLS] token representation (row 1). Next, we observe that while global average pooling and compression layer perform similarly in terms of SPL (58.7% *vs*. 59.3%), compression layers lead to a substantially higher success rate (82.0% *vs*. 76.9%). We hypothesize that preserving spatial structure with compression layers is particularly useful for recognizing the goal location (and stopping in a correct place), thus leading to a higher SR.

Visual Encoder Learning Rate. In initial experiments, we found finetuning representations pretrained with MAE or Data2Vec led to overfitting and poor generalization. Thus, we experimented with tuning the learning rate (LR) used specifically for weights of the visual encoder. In Tab. 7c, we observe that tuning the LR leads to massive improvements in SR of +31.0% and SPL of +20.9% (row 1 *vs*. 2). In fact, we find fintuning with a bad LR (row 1) is worse than simply freezing the representations (row 3).

Pretraining Dataset In Tab. 7e we compare the effect of pretraining with a dataset (OSD [13]) that was used in prior work [37]. We observe similar performance with both datasets, indicating that both choices are similarly 'in-domain' with respect to the downstream IMAGENAV task.

Image Augmentations. In Tab. 7d, we ablate the use of image augmentations during policy learning. We discover that augmentations play a vital role in preventing overfitting of the pretrained ViT agent. Without augmentations, OVRL-V2's SR drops by -24.2% and SPL drops by -15.5%.

ImageNav Failures. We present qualitative analysis of the failure cases of OVRL-V2 in Fig. 7. We perform this analysis by manually labeling each case with a cause (*e.g.*, 'exploration failure'). We find that in approximately one-third of the failures the goal-image is not semantically meaningful ('bad goals') such as blank walls. These failures account

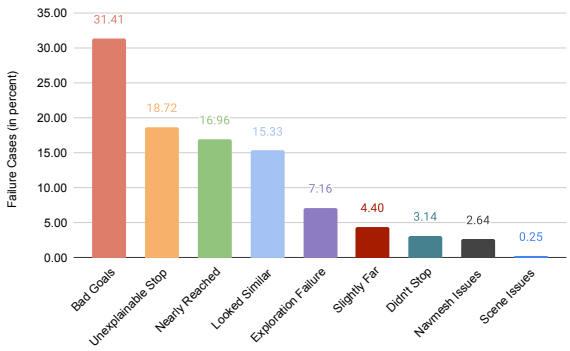

Figure 7. **Breakdown of the OVRL-V2 IMAGENAV Failures.**

for nearly 6% of the dataset, suggesting the upper bound on IMAGENAV success is 94%. Other key issues include 'unexplained stop' where the reason for stopping is unclear, 'nearly reached' when the agent stops short, and 'looked similar' finding a location very similar to the goal-image (but incorrect). The appendix includes additional details.

Several of these failures may be reduced with simple techniques. For example, the 'nearly reached' cases may be reduced by using a stronger success criteria during training. Similarly, increasing image resolution may reduce 'looked similar' failures as the agent may recognize additional visual cues to distinguish the correct goal. For other cases, such as 'unexplained stop' further investigation is required.

7. Conclusion

In this paper, we demonstrate that a model-free navigation agent (OVRL-V2) composed of task-agnostic components (ViTs, convolutions, and LSTMs) can achieve state-of-the-art results on both IMAGENAV and OBJECTNAV. We show that a compression layer can make ViTs work so well that they learn to 'hack' rewards. We present a simple, yet principled reward fix to boost performance. Finally, we discover that visual pretraining with MAE enables positive scaling trends with larger ViT architectures.

References

[1] Ziad Al-Halah, Santhosh K Ramakrishnan, and Kristen Grauman. Zero experience required: Plug & play modular transfer learning for semantic visual navigation. *arXiv preprint arXiv:2202.02440*, 2022. 1, 2, 4, 5, 6, 7, 12, 13

[2] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 3

[3] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Thirunavukkarasu Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language. *arXiv preprint arXiv.2202.03555*, 02 2022. 2, 7

[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 preprint arXiv:2006.13171*, 2020. 1, 3, 11

[5] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:9912–9924, 2020. 2

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660, 2021. 2, 12

[7] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020. 1, 2, 7

[8] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12875–12884, 2020. 1, 2

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning (ICML)*, 2020. 2

[10] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021. 2

[11] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, et al. Procthor: Large-scale embodied ai using procedural generation. *arXiv preprint arXiv:2206.06994*, 2022. 7

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 3

[13] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10786–10796, 2021. 8, 11, 12

[14] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in neural information processing systems*, 31, 2018. 11

[15] Meera Hahn, Devendra Singh Chaplot, Shubham Tulsiani, Mustafa Mukadam, James Matthew Rehg, and Abhinav Gupta. No rl, no simulation: Learning to navigate without navigating. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2

[16] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022. 2, 3, 6, 7, 11

[17] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. *CVPR*, 2022. 2, 7, 8

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 11

[19] Arjun Majumdar, Gunjan Aggarwal, Bhavika Devnani, Judy Hoffman, and Dhruv Batra. Zson: Zero-shot object-goal navigation using multimodal goal embeddings. *arXiv preprint arXiv:2206.12403*, 2022. 6, 12

[20] Oleksandr Maksymets, Vincent Cartillier, Aaron Gokaslan, Erik Wijmans, Wojciech Galuba, Stefan Lee, and Dhruv Batra. Thda: Treasure hunt data augmentation for semantic navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15374–15383, October 2021. 2

[21] Lina Mezghani, Sainbayar Sukhbaatar, Thibaut Lavril, Oleksandr Maksymets, Dhruv Batra, Piotr Bojanowski, and Karteek Alahari. Memory-augmented reinforcement learning for image-goal navigation. *arXiv preprint arXiv:2101.05181*, 2021. 6, 7, 11, 12

[22] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 99, pages 278–287, 1999. 2, 5

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 2, 7, 12

[24] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real world robot learning with masked visual pre-training. In *6th Annual Conference on Robot Learning*, 2022. 2

[25] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. Poni: Potential functions for objectgoal navigation with interaction-free learning. In *CVPR*, pages 18890–18900, June 2022. 1, 2

[26] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. Poni: Potential functions for objectgoal navigation with interaction-free learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18890–18900, 2022. 2

[27] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 3, 5, 11, 12

[28] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search from human demonstrations at scale. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 4, 7, 11

[29] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9339–9347, 2019. 4, 11

[30] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked World Models for Visual Control. *arXiv e-prints*, page arXiv:2206.14244, June 2022. 2

[31] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 11

[32] Justin Wasserman, Karmesh Yadav, Girish Chowdhary, Abhinav Gupta, and Unnat Jain. Last-mile embodied visual navigation. In *6th Annual Conference on Robot Learning*, 2022. 1, 2

[33] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *International Conference on Learning Representations (ICLR)*, 2020. 2, 4, 5

[34] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9068–9079, 2018. 3, 5, 11, 12

[35] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendrfa Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022. 2, 5, 6

[36] Karmesh Yadav, Santhosh Kumar Ramakrishnan, John Turner, Aaron Gokaslan, Oleksandr Maksymets, Rishabh Jain, Ram Ramrakhya, Angel X Chang, Alexander Clegg, Manolis Savva, et al. Habitat challenge 2022, 2022. 7, 11

[37] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline Visual Representation Learning for Embodied Navigation. *arXiv e-prints*, page arXiv:2204.13226, Apr. 2022. 2, 4, 5, 6, 8, 11, 12

[38] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation. *arXiv preprint arXiv:2204.13226*, 2022. 2

[39] Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Theo Gervet, John Turner, Aaron Gokaslan, Noah Maestre, Angel Xuan Chang, Dhruv Batra, Manolis Savva, et al. Habitat-matterport 3d semantics dataset. *arXiv preprint arXiv:2210.05633*, 2022. 11

[40] Brian Yamauchi. A frontier-based approach for autonomous exploration. *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, 1997. 2

[41] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *ICLR*, 2019. 1, 2

[42] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021. 11

[43] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectgoal navigation. In *CoRL*, pages 16117–16126, 2021. 2

[44] Minzhao Zhu, Binglei Zhao, and Tao Kong. Navigating to objects in unseen environments by distance prediction. *arXiv preprint arXiv:2202.03735*, 2022. 1, 2

[45] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Kumar Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. *ICRA*, 2017. 1, 3

[46] Yilun Du, Chuang Gan, and Phillip Isola. Curious representation learning for embodied intelligence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10408–10417, 2021. 6, 12

A. Experimental Details

Pretraining Dataset. For pretraining, we create a dataset using the 800 HM3D [27] and 72 Gibson [34] training scenes. We collect a total of 1.45M RGB images using a camera with a resolution of 512×512 and a 90° FoV, attached to an oracle agent that navigates in a scene from a random start position to a random goal position using the shortest path finding algorithm. We refer to this dataset as the HM3D-Gibson Shortest Path (HGSP) dataset. We determined the HGSP dataset size (of 1.45M) based on the observation in [37] that pretraining on 10% (1.45M) of the Omnidata Starter Dataset (OSD) [13] leads to the same downstream performance as training on 100% of the dataset totalling 14.5M images.

Pretraining Details. We pretrain visual encoders using MAE [16] using the same hyperparameters as [16]. We pretrain ViT-SMALL and ViT-BASE encoders, initialized from scratch, for 800 epochs on the HGSP dataset. We use these pretrained encoders to initialization the visual encoder for downstream task. We use the same pretrained encoder for both the IMAGENAV and OBJECTNAV experiments.

Data Augmentation For downstream tasks (IMAGENAV and OBJECTNAV), we use image augmentations during training and evaluation (*i.e.* at test-time). Specifically, we apply random color-jitter followed by random shifts [42] (*i.e.* random translations). We apply the same image augmentation across time and to all of the samples on each GPU in our distributed training setup. For IMAGENAV, we use a color-jitter of 0.3 for brightness, contrast, saturation and hue levels followed by random shifts with a padding of 4 pixels. For OBJECTNAV, we use a value of 0.4 for color jitter and a padding of 16 for random shifts. These data augmentation settings follow the IMAGENAV experiments in [37].

IMAGENAV Benchmark. We perform IMAGENAV experiments using a standard dataset released by [21]. This benchmark uses the Habitat simulator [29,31] and situates the task in the Gibson [34] environments, which include 72 training and 14 validation scenes. The validation set includes 300 episode for each scene (4,200 episodes total). In this benchmark, agents are simulated as a cylinder with a height of 1.5m, radius of 0.1m, and sensors placed 1.25m above the center of the base. The RGB camera has a resolution of 128×128 and a 90° field-of-view (FoV). Agents can take up to 1000 steps in the environment, and an agent is successful if it calls STOP within 1m of the goal position.

IMAGENAV Training Details. We train agents in the Gibson environments for 500M timesteps (25k updates) using a total of 320 environment running in parallel. Every environment collects (up to) 64 frames of experience which is followed by 2 PPO epochs with 2 mini-batches. Unless specified explicitly, we use a learning rate of 2.5×10^{-4} for training the agent and update the parameters using the AdamW optimizer [18] with a weight decay of 10^{-6}. We train agents with the reward functions in Eq. (2) and Eq. (4) from the main paper, using the following settings: success weighting $c_s = 5.0$, angle success weighting $c_a = 5.0$, goal radius $r_g = 1.0$, angle threshold $\theta_g = 25°$, and slack penalty $\gamma = 0.01$. We evaluate performance every 25M steps of training. We report metrics based on the highest success rate (SR) achieved on the validation set.

OBJECTNAV Benchmark. We conduct OBJECTNAV experiments using the HM3DSEM dataset [39], which uses the Habitat simulator [29, 31] and HM3D [27] environments. The dataset consists of 80 training, 20 validation, and 20 testing scenes. We report results on the v0.1 HM3DSEM VAL and TEST-STD splits that were used in the 2022 Habitat Challenge [36] OBJECTNAV benchmark. In this benchmark, the agent models a LocoBot [14] with a height of 0.88m, radius of 0.18m, and sensors placed at the top of the agent's head. The RGB camera has a 640×480 resolution and a 79° horizontal FoV. In each episode, agents must find an object drawn from one of 6 categories: *'chair'*, *'bed'*, *'plant'*, *'toilet'*, *'tv/monitor'*, and *'sofa'*. Agents are allowed 500 steps in the environment, and episodes are considered successful if the agent stops within 0.1m of a viewpoint that is (a) within 1m of any instance of an object from the goal category and (b) from which that instance is visible, following the evaluation protocol laid out in [4].

OBJECTNAV Human Demonstrations. For training our imitation learning agent, we use the dataset of OBJECTNAV human demonstrations collected by Habitat-Web [28, 39] for HM3DSEM dataset using Amazon Mechanical Turk. The dataset consists of $77k$ human demonstrations for 80 HM3DSEM scenes that were released by [36]. For each scene, we have ∼158 episodes for each unique goal object category with a randomly set start location amounting to ∼950 demonstrations per scene. The dataset amounts to a total of ∼12.1M steps in experience, with each episode averaging ∼159 steps.

OBJECTNAV Training Details. We train all OBJECTNAV agents in the HM3D environment for ∼400M steps (25K updates) using 512 parallel environments. Similar to IMAGENAV, we use a weight decay of 10^{-6}, a learning rate of 10^{-4} for the visual encoder and 10^{-3} LR for everything else with the AdamW optimizer. We evaluate checkpoints after every 1000 policy updates and report metrics for the checkpoints with the highest validation SPL.

B. IMAGENAV Baselines

This section describes the state-of-the-art IMAGENAV methods from prior work listed in Tab. 5 (rows $1-5$).

1. Zero Experience Required (**ZER**) [1] proposes a novel reward function (detailed in Eq. (2)) for IMAGENAV, which is used to train a general purpose agent composed of an ResNet-9 visual encoder (for observations and goals) and a GRU-based policy network that are trained from scratch. Our initial from-scratch ResNet-50 IMAGENAV baseline presented in Tab. 1 row 1 uses a similar architecture and the same reward function as [1].

2. Zero-Shot OBJECTNAV (**ZSON**) [19] uses the reward function from [1] to train an IMAGENAV agent consisting of a pretrained ResNet-50 encoder from [37] for visual observations and a CLIP [23] visual encoder for processing goal-images. The agents in [19] are trained in HM3D [27] and then evaluated in Gibson [34].

3. Curious Representation Learning (**CRL**) [46] uses curiosity-based exploration to collect data for visual representation learning. The visual encoder is then used within a general purpose agent that is finetuned for IMAGENAV. We report results reproduced by [37].

4. Offline Visual Representation Learning (**OVRL-v1**) [37] pretrains a ResNet-50 encoder using the DINO [6] self-supervised representation learning algorithm on images from the Omnidata Starter Dataset (OSD) [13]. The pretrained ResNet-50 is used within a general purpose agent to process visual observations and goal images while finetuing for IMAGENAV with the reward function from [1].

5. Memory-Augmented RL for IMAGENAV (**Mem-Aug Nav**) [21] enhances a general purpose agent with an attention-based memory module. In [21], agents use four RGB sensors for observations and goals, which provide a panoramic view of the environment. By contrast, our agents and the agents in Tab. 5 (rows $1-4$) use a single RGB camera. Thus, we highlight results from [21] in gray.

C. Compression Layer Implementation

PyTorch-style pseudocode for creating the compression layer described in Sec. 4 is presented in Algorithm 1. The layer operates on patch representations generated by a ViT-based visual encoder. It reshapes the patches into a grid and then uses a convolutional layer to compress them to a lower dimension. The dimension is determined such that when the compressed patches are subsequently concatenated (*i.e.*, flattened) into a vector the size is approximately 2,048.

D. IMAGENAV Failure Analysis Details

Here we describe each of the categories used for the IMAGENAV failure analysis within the main paper (which shows the distribution of these failures) in greater detail:

Bad Goals. We found that a large number goals in the validation dataset face the wall or capture a noisy parts of the scene. These goals are not semantically meaningful – *i.e.*, they either do not 'describe' a unique location in the environment or do not provide any indication of where the goal might be found.

Looked Similar. The goal image and agent observation at the stopping time have some visual similarities that may have caused the agent to believe it has reached the goal.

Unexplainable Stop. The reason why the agent called stop is unclear to the annotator.

Nearly Reached. The agent's distance to the goal is between 1.0m to 1.5m and the agent is looking in the direction of the goal image.

Slightly Far. Distance to goal is farther than 1.5m but the

Algorithm 1 PyTorch-style Compression Layer

```
def create_compression_layer(
    patch_dim: int,
    num_patches: int,
    approx_output_size: int = 2048,
):
    # num channels per patch
    num_channels = int(
        round(approx_output_size / num_patches)
    )

    # create layer
    layer = nn.Sequential(
        PatchReshape(),
        nn.Conv2d(
            in_channels=patch_dim,
            out_channels=num_channels,
            kernel_size=3,
            padding=1,
            bias=False,
        ),
        nn.GroupNorm(
            num_groups=1,
            num_channels=num_channels,
        ),
        nn.ReLU(inplace=True),
        nn.Flatten(),
    )

    # actual output size
    output_size = num_channels * num_patches

    return layer, output_size


class PatchReshape(nn.Module):
    def forward(self, x):
        # batch size X num patches X patch dim
        N, L, D = x.shape
        # assume square image
        H = W = int(L**0.5)
        # reshape to square grid
        x = x.reshape(N, H, W, D)
        # put channels first
        x = x.permute(0, 3, 1, 2)
        return x
```

agent is looking towards the goal image.

Exploration Failure. The exploration strategy of the agent fails. For example, when the agent keeps looping in one small region of the environment or stop after reaching an area that is far from the goal.

Didn't Stop. The agent saw the goal but did not stop.

Navmesh Issues. The agent needs to pass through a narrow passage that is nearly the same size as the agent.

Scene Issues. There is a hole in the scene (*i.e.*, a missing part of the underlying 3D geometry of the environment), which confuses the agent.

E. Additional Ablations

In this section, we present the results of additional ablations conducting on the IMAGENAV task with randomly initialized and pretrained model-free navigators.

#	Pretrained	Augmentations	SPL (↑)	SR (↑)
1	False	False	31.6	50.0
2	False	True	**37.1**	**67.4**
3	True	False	43.2	57.8
4	True	True	**58.7**	**82.0**

Table 8. **Image augmentations** improve the IMAGENAV performance of from-scratch and pretrained agents.

Image Augmentations. In Tab. 8, we ablate the use of image augmentations for a randomly initialized and pretrained policy. We find that in both cases agents benefit from using augmentation during policy learning. Without pretraining, augmentations improve SR by +17.4% and SPL by +5.5% (rows 1 *vs.* 2). With pretraining, augmentations lead to gains in SR of +24.2% and gains in SPL of +15.5% (rows 3 *vs.* 4). Additionally, we find that using augmentations without pretraining (row 2) leads to a higher SR of 67.4% than using pretraining without augmentations (row 3), which results in a SR of 57.8%.

#	Model Size	# Params	SPL (↑)	SR (↑)
1	Half-width RESNET-50	21.6M	33.5	59.9
2	Full-width RESNET-50	59.1M	**38.0**	60.0
3	VIT-SMALL	50.9M	37.1	**67.4**

Table 9. **Increasing ResNet-50 model size** improves SPL, but does not improve SR. Switching to VIT-SMALL substantially improves SR with a minimal drop in SPL despite fewer parameters.

Increasing Model Size. The default version of our randomly initialized navigation agent uses a VIT-SMALL as the vision backbone. In Table 9, we demonstrate that when trained from scratch, VIT-SMALL agents are more successful than RESNET counterparts, including a full-width RESNET-50 model that uses 8.2M more parameters than the VIT-SMALL variant. Specifically, we find that using a full-width RESNET-50 leads to +4.5% improvement in SPL compared to the half-width RESNET-50. However, even with fewer parameters, VIT-SMALL attains 7.4% higher SR, while only being 0.9% worse in terms of SPL.

F. Reward Hacking Visualization

In Fig. 8, we visualize the reward hacking behaviour of an IMAGENAV agent trained with the ZER [1] reward, as discussed in Sec. 4. Towards the end of the trajectory near the goal (left), the agent repeatedly enters and exits the goal radius to accumulate the angle-to-goal reward term in Eq. (2) rather than successfully complete the episode. With the submission, we also include a set of videos for 3 episodes, contrasting the performance of an OVRL-v2 agent trained with the ZER reward [1] (`reward_hacking_agent.mp4`) against an agent trained with our reward (`our_agent.mp4`).

Figure 8. **Reward hacking** behavior of agents trained with ZER [1] rewards. The episode's start and goal position are denoted by the green and peach square respectively. The orange dashed line indicates the agent's trajectory, which repeatedly enters and exits the goal radius (not shown) to accumulate (*i.e.*, hack) the angle-to-goal term in the ZER [1] reward.

G. Code

Source code and model weights will be publicly released upon acceptance.