PoliFormer: Scaling On-Policy RL with Transformers Results in Masterful Navigators

Kuo-Hao Zeng **Zichen Zhang** **Kiana Ehsani** **Rose Hendrix** **Jordi Salvador**

Alvaro Herrasti **Ross Girshick** **Aniruddha Kembhavi** **Luca Weihs**

PRIOR @ Allen Institute for AI
poliformer.allen.ai

Abstract: We present POLIFORMER (**Poli**cy Trans**former**), an RGB-only indoor navigation agent trained end-to-end with reinforcement learning ***at scale*** that generalizes to the real-world without adaptation despite being trained purely in simulation. POLIFORMER uses a foundational vision transformer encoder with a causal transformer decoder enabling long-term memory and reasoning. It is trained for hundreds of millions of interactions across diverse environments, leveraging parallelized, multi-machine rollouts for efficient training with high throughput. POLIFORMER is a masterful navigator, producing state-of-the-art results across two distinct embodiments, the LoCoBot and Stretch RE-1 robots, and four navigation benchmarks. It breaks through the plateaus of previous work, achieving an unprecedented 85.5% success rate in object goal navigation on the CHORES-$\mathbb{S}$ benchmark, a 28.5% absolute improvement. POLIFORMER can also be trivially extended to a variety of downstream applications such as object tracking, multi-object navigation, and open-vocabulary navigation *with no finetuning*.

Keywords: Embodied Navigation, On-Policy RL, Transformer Policy

1 Introduction

Reinforcement Learning (RL) has been used extensively to train embodied robotic agents to complete a variety of indoor navigation tasks. Large-scale, on-policy, end-to-end RL training with DD-PPO [1] enables near-perfect *PointNav*[1] performance when using a shallow GRU-based [2] architecture. However, this approach fails to result in the same breakthroughs for harder navigation problems like Object Goal Navigation *(ObjectNav* [3]) where an agent must explore its environment to locate and navigate to an object of the requested type. RL approaches for ObjectNav have generally not advanced beyond shallow GRU architectures due to challenges presented by training instability and unreasonably long training times with wider and deeper models, such as scaled-up transformers [4].

In a departure from on-policy RL, which is sample inefficient and often uses complex reward shaping and auxiliary losses [5], Imitation Learning (IL) has recently shown promise for ObjectNav. Ehsani *et al.* (2023) [6] demonstrated that the transformer-based SPOC agent, when trained to imitate heuristic shortest-path planners, can be trained stably, is sample efficient, and is significantly more effective than prior RL approaches on their benchmark. SPOC, however, ultimately falls short of mastery, plateauing at a success rate of $\sim$57%. Critically, as we show in our experiments, the performance of SPOC does not seem to improve significantly when further scaling up data and model depth; we suspect this is a consequence of insufficient state-space exploration as expert trajectory datasets frequently contain few examples of error recovery, which can lead to sub-optimal performance due to compounding errors [7] or non-trivial domain shifts during inference.

[1] Point Goal Navigation is the task of navigating to a goal location using privileged GPS coordinates.

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

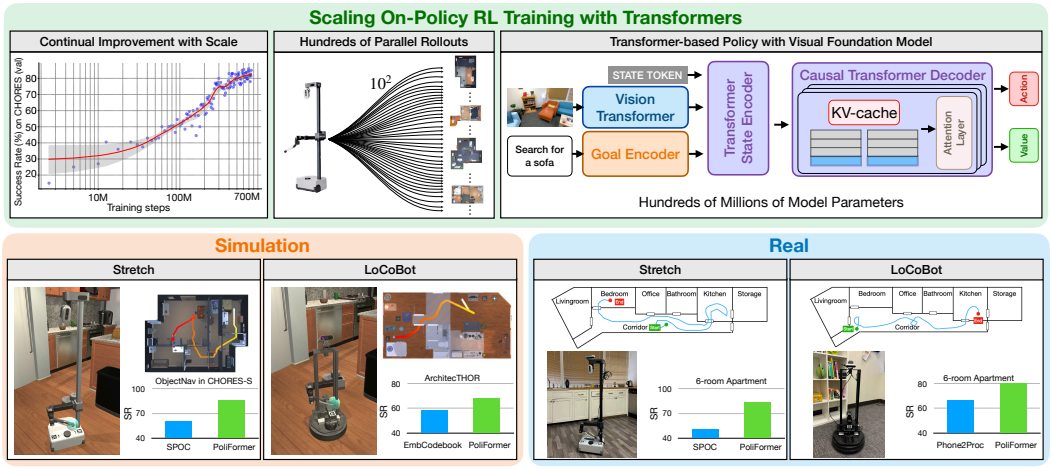

Figure 1: POLIFORMER, a transformer-based policy trained using RL at scale in simulation, achieves significant performance improvements in simulation (*bottom-left*) and the real world (*bottom-right*), across two embodiments. SR denotes Success Rate. We scale on-policy RL training across multiple dimensions: (*top-left*) we observe continual performance improvement with scaling RL training; (*top-middle*) we leverage hundreds of parallel rollouts for higher throughput; (*top-right*) we develop a transformer-based policy scaling model parameters to hundreds of millions.

In contrast to IL, RL requires that agents learn via interactive trial-and-error, allowing for deep exploration of the state space. This exploration has the potential to produce agents that can learn behaviors that are superior to those in the expert demonstrations. This raises the question: can we bring together the modeling insights from SPOC-like architectures and the exploratory power of RL to train a masterful navigation agent? Unfortunately, this cannot be done naively due to RL's sample inefficiency and complexities in using deep (transformer) architectures with RL algorithms. In this work, we develop an effective method for training large transformer-based architectures with RL, breaking through the plateaus of past work, and achieving SoTA results across four benchmarks.

Our method, POLIFORMER, is a transformer-based model trained with on-policy RL in the AI2-THOR [8] simulator at scale, that can effectively navigate in the real world without any adaptation. We highlight three primary design decisions that make this result possible. *(i)* Scale in Architecture: Building on the SPOC architecture, we develop a fully transformer-based policy model that uses a powerful visual foundation model (DINOv2 [9], a vision transformer), incorporates a transformer state encoder for improved state summarization, and employs a transformer decoder for explicit temporal memory modeling (Fig. 1, *top-right*). Importantly, our transformer decoder is causal and uses a KV-cache [10], which allows us to avoid huge computational costs during rollout collection and makes RL training affordable. *(ii)* Scale in Rollouts: We leverage hundreds of parallel rollouts and large batch sizes, which leads to high training throughput and allows us to train using a huge number of environment interactions (Fig. 1, *top-middle*). *(iii)* Scale in Diverse Environment Interactions: Training POLIFORMER with RL at scale in 150k procedurally generated PROCTHOR houses [11] using optimized Objaverse [12, 13, 14] assets results in steady validation set gains (Fig. 1, *top-left*).

POLIFORMER achieves excellent results across multiple navigation benchmarks in simulation. On CHORES-$\mathbb{S}$, it achieves an impressive 85.5% Success Rate, a higher than +28.5% absolute improvement over the previous SoTA model [6]. Similarly, it also obtains SoTA Success Rates on ProcTHOR (+8.7%), ArchitecTHOR (+10.0%) and AI2-iTHOR (+6.9%). These results hold across two embodiments, LoCoBot [15] and Stretch RE-1 [16], with distinct action spaces (Fig. 1, *bottom-left*). In the real world (Fig. 1, *bottom-right*), it outperforms ObjectNav baselines in the sim-to-real zero-shot transfer setting using LoCoBot (+13.3%) and Stretch RE-1 (+33.3%).[2]

We further train POLIFORMER-BOXNAV that accepts a bounding box (*e.g.*, from an off-the-shelf open-vocab object detector [18, 19] or VLMs [20, 21]) as its goal specification in place of a given

[2]POLIFORMER even outperforms Phone2Proc [17], a baseline finetuned in 3D-reconstructed test scenes.

category. This abstraction makes POLIFORMER-BOXNAV a general purpose navigator that can be "prompted" by an external model akin to the design of Segment Anything [22]. It is extremely effective at exploring its environment and, once it observes a bounding box, beelines towards it. POLIFORMER-BOXNAV is a leap towards training a foundation model for navigation; with no further training, this general navigator can be used in the real world for multiple downstream tasks such as open vocabulary ObjectNav, multi-target ObjectNav, human following, and object tracking.

In summary, our contributions include: (i) POLIFORMER, a transformer-based policy trained by RL at scale in simulation that achieves SoTA results over four benchmarks in simulation and in the real world across two different embodiments. (ii) A training recipe that enables effective end-to-end policy learning with large-scale neural models via on-policy RL. (iii) A general purpose navigator, POLIFORMER-BOXNAV, that can be used zero-shot for multiple downstream navigation tasks.

2 Related Work

IL and RL on Embodied Navigation. Recent advancements include Point Goal Navigation [1], Object Goal Navigation [3, 23, 24, 25], Exploration [26, 27, 28], and Social Navigation [29], as well as successful sim-to-real transfer [6] and high performance in various downstream applications [16, 30, 31, 32, 33, 34, 35]. The prevalent approaches to building capable navigation agents include both end-to-end training [1, 6] and modular methods that leverage mapping [36, 37, 38] or off-the-shelf foundation models [39, 40, 41, 42, 43]. In these frameworks, the policy models are typically optimized via Imitation Learning (IL) using expert trajectories [44, 45, 46, 47, 48, 49, 50] or Reinforcement Learning (RL) within interactive environments [29, 51, 52, 53, 54, 55, 56, 57, 58, 59] and with carefully tuned reward shaping and auxiliary losses [5, 60, 61, 62, 63].

Transformer-based Policies for Embodied Agents. Recent works use transformer-based architectures for embodied agents. Decision Transformer (DT [64]) learns a policy offline, conditioning on previous states, actions, and future returns, providing a path for using transformers as sequential decision-making models. ODT [65] builds on DT and proposes to blend offline pretraining and online finetuning, showing competitive results in D4RL [66]. More recently, MADT [67] shows few-shot online RL finetuning on the offline trained DT in multi-agent game environments. The Skill Transformer [68] learns a policy via IL for mobile manipulation tasks. While the focus of these works is control towards specific tasks (relying on IL pre-training), we train *from scratch* using pure RL targeting a navigation policy that can be used zero-shot for many downstream tasks.

There has been a huge effort on improving the scale and training stability of transformer policies. GTrXL [69] proposes to augment causal transformer policies with gating layers towards stable RL training in [70]. PDiT [71] proposes to interleave perception and decision transformer-based blocks showing its effectiveness in fully-observed environments. GATO [72] learns a large-scale transformer-based policy on a diverse set of tasks, including locomotion and manipulation. Performer-MPC [73] proposes learnable Model Predictive Control policies with low-rank attention transformers (Performer [74]) as learnable cost functions. SLAP [75] learns a policy for mobile manipulation based on a hybrid design. Radosavovic *et al.* [76] learn a causal transformer using RL and IL for real-world humanoid locomotion. However, their input is proprioceptive state, without visual observations, and their learned policy is intended to be stationary (walking steadily in the real-world). NavFormer [77] learns image-goal conditioned transformer policies using offline-RL for target navigation. In contrast to existing approaches, we achieve efficient large-scale transformer-based on-policy RL training for a partially-observed, long-horizon, navigation task using RGB sensor inputs and show that our policy can be seamlessly deployed in the real world. We accomplish this without complex inductive biases or multi-task dataset aggregation due to the sheer amount of procedural scene layouts and visual variety of objects in simulation.

Toward Navigation Foundation Models. Many recent works attempt to produce general-purpose "foundational" navigation models. The causal transformer for humanoid outdoor locomotion [76], relying on proprioceptive state, shows emergent behaviors like arm swinging and in-context adaptation. GNM [78] trains image goal-conditioned policies (*i.e.*, conditioning on an image at the desired goal location) by IL across various embodiments, which allows them to scale up the size of their

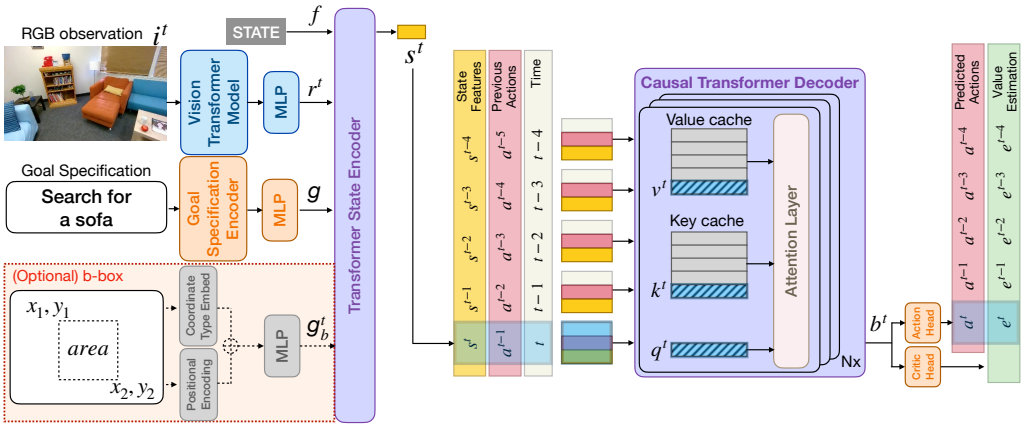

Figure 2: POLIFORMER is a fully transformer-based policy model. At each timestep t, it takes an ego-centric RGB observation i^t, extracts visual representations r^t using a vision transformer model, further encodes state features s^t using the visual representations and goal features g (and optional detected bounding box goal features g_b^t), models state belief b^t over time, employing a causal transformer decoder, and, finally, predicts action logits a^t and a value estimation e^t via linear actor and critic heads, respectively. For rollout collection and inference, we leverage the KV-cache [10] as our temporal cache strategy to prevent recomputing the forward pass for all prior timesteps at each new timestep, saving memory and speeding up both training and inference.

training data and manages to generalize to controlling unseen embodiments. NOMAD [79], which extends ViNT [80] uses a diffusion policy and also uses image goal conditioning. Unlike these works, we navigate from rich RGB image inputs and specify goals with natural-language text (or with bounding boxes), as such goals are far more easily available in real-world settings.

3 Method

RL, while effective and intuitive for training policies, has not yet been scaled in embodied AI to the same degree as in other domains. RL has primarily been applied to short-horizon tasks [81, 82], smaller environments [83, 84], or tasks utilizing privileged knowledge [1]. We scale learning in three directions; (i) Network Capacity, via POLIFORMER's deep, high-capacity, model architecture (Sec. 3.1); (ii) Parallelism and Batch size via our highly-parallelized training methodology that leverages large batch sizes for efficiency (Sec. 3.2), and (iii) Training Environments via our optimization of the simulation environment to support high-speed training (Sec. 3.3).

3.1 The POLIFORMER Architecture

We now detail POLIFORMER's transformer architecture (see Fig. 2), which is inspired by SPOC [6]. While we make several subtle, but important, changes to the SPOC architecture (detailed below), our largest architectural difference is in the transformer decoder: we replace the standard transformer decoder block [4, 85] with the implementation used in the Llama 2 LLM [86]. This change has dramatic implications for training and inference speed.

At each timestep t, POLIFORMER takes an ego-centric RGB observation i^t as input and employs a frozen vision transformer model to extract a visual representation r^t. This representation, along with an embedding g of the goal, are summarized by a transformer state encoder to produce the state representation s^t. The causal transformer decoder then encodes the state feature s^t (along with $s^0, \ldots, s^{t-1}$) into a belief b^t. Finally, the linear actor and critic heads project b^t to predict action logits a^t and a value estimate e^t, respectively.

Vision Transformer Model. Inspired by prior work [6, 87], we choose DINOv2 [9] as our visual foundation backbone, because of its remarkable dense prediction and sim-to-real transfer abilities. The visual backbone takes an ego-centric RGB observation $i \in \mathbb{R}^{H \times W \times 3}$ as input and produces a patch-wise representation $r \in \mathbb{R}^{\frac{H}{14} \times \frac{W}{14} \times h}$, where H and W are the observation height and width, and h is the hidden dimension of the visual representation. We reshape the visual representation into

a $\ell \times h$ matrix, $\ell = H \cdot W/196$, and project the representation to produce $v \in \mathbb{R}^{\ell \times d}$, where d is the input dimension to the transformer state encoder. For effective sim-to-real transfer, it is important to keep the vision transformer model frozen when training in simulation.

Goal Encoder. For fair comparison, when training agents on ObjectNav benchmarks using the LoCoBot, we follow EmbCLIP [39] and use a one-hot embedding layer to encode the target object category. On benchmarks using the Stretch RE-1 robot, we follow SPOC [6] and use a FLAN-T5 small [88, 89] model to encode the given natural language goal and use the last hidden state from the T5 model as the goal embedding. Before passing the goal embedding to the transformer state encoder, we always project the embedding to the desired dimension d, resulting in $g \in \mathbb{R}^{1 \times d}$.

In select experiments (detailed in Sec. 4), we specify the goal object via a bounding box (b-box), either in addition to or as an alternative to text. In this case, the goal encoder processes the b-box using both the sinusoidal positional encoding and the coordinate-type embeddings to embed the top-left, bottom-right b-box coordinates and its area (5 values in total), followed by an MLP to project the hidden feature to the desired dimension d, resulting in $g_b \in \mathbb{R}^{5 \times d}$.

Transformer State Encoder. This module summarizes the state at each timestep as a vector $s \in \mathbb{R}^d$. The input to this encoder includes the visual representation v, the goal feature g (and/or g_b), and an embedding f of a `STATE` token. We concatenate these features together and feed them to a non-causal transformer encoder. This encoder then returns the output corresponding to the `STATE` token as the state feature vector. Since the transformer state encoder digests both visual and goal features, the produced state feature vector can also be seen as a goal-conditioned visual state representation.

Causal Transformer Decoder. We use a causal transformer decoder to perform explicit memory modeling over time. This can enable both long-horizon (*e.g.*, exhaustive exploration with backtracking) and short-horizon (*e.g.*, navigating around an object) planning. Concretely, the causal transformer decoder constructs its state belief b^t using the sequence of state features $\mathbf{s} = \{s^j|_{j=0}^{j=t}\}$ within the same trajectories.

Unlike RNN-based causal decoders [90] which, during rollout collection, require only constant time to compute the representation at each timestep, standard implementations of causal transformers require t^2 time to compute the representation at timestep t. This substantial computational cost makes large-scale on-policy RL with causal transformer models slow. To overcome this challenge, we leverage the KV-cache technique [10] to keep past feed-forward results in two cache matrices, one for **K**eys and one for **V**alues. With a KV-cache, our causal transformer decoder only performs feedforward computations with the most current state feature which results in computation time growing only linearly in t rather than quadratically. While this is still mathematically slower than the constant time required for RNN-based decoders, empirically, we found only a small difference between KV-cache-equipped causal transformers and RNNs in overall training FPS for our setting. Compared to other temporal cache strategies, we found that KV-Cache offers the most significant speed improvements. Please see App. B.3 and Fig. 5 for a discussion of the impact on training speed resulting from different cache strategies.

3.2 Scalable RL Training Recipe

While KV-cache accelerates the causal transformer, this alone is not sufficient to enable efficient and fast RL training. In this section, we describe the methodology we use to achieve faster training. Although each of these individual findings may have been discussed in other works, we emphasize the critical importance of these hyperparameters for efficient training and, consequently, stellar results.

We parallelize the training process using multi-node training, increasing the number of parallel rollouts by a factor of 4 compared to previous approaches [30, 59, 87] (we use 192 parallel rollouts for Stretch RE-1 agents and 384 for LoCoBot agents). We employ the DD-PPO [1] learning algorithm across 4 nodes, utilizing a total of 32 A6000 GPUs and 512 CPU cores. This scaling accelerates the training speed by approximately $3.4\times$, a near-linear gain compared to single-node training; we train for $\sim$4.5 days with 4 nodes, while this would have required 15.3 days with a single node.

Our batch size during training, in number of frames, is equal to the number of parallel rollouts (R) multiplied by the length of these rollouts (T) and thus, by increasing R as above, we multiplicatively increase the total batch size. We follow SPOC [6] and use a small *constant* learning rate of $2 \cdot 10^{-4}$ throughout the experiments. Instead of annealing the learning rate during training, we instead follow PROCTHOR [11] and *increase the batch size* by changing the rollout length from T=32, to T=64, and finally to T=128 (resulting in a final batch size of 49,152 frames for LoCoBot agents). We make these increases every 10M steps until we reach T=128.

3.3 Scaling Environment Interactions

For the experiments on LoCoBot, we use the original PROCTHOR-10k houses for training for a fair comparison to baselines. With the KV-cache technique, our training steps per second is $\sim$2.3k for training POLIFORMER, using 384 rollouts on 32 GPUs. For the experiments on Stretch, following [6], we use the PROCTHOR-150k houses with $\sim$40k annotated Objaverse 3D assets. As on-policy RL requires the agent to interact with the environment on-the-fly, we found the continuous physics simulation used by the Stretch RE-1 agent in AI2-THOR to be too slow to efficiently train at scale. To overcome this, we discretely approximate the agent's continuous movement via a teleportation-with-collision-checks approach. In particular, to move the agent, we perform a physics cast using a capsule collider representing the agent along the desired movement direction. If the cast does not hit any object (suggesting no potential collisions), we teleport the agent to the target location, otherwise we let AI2-THOR simulate continuous movement as usual. For rotations, we first teleport the agent to the target rotation pose. If the agent's capsule collides with any object, we teleport the agent back and let the AI2-THOR simulate the continuous rotation. For our tasks, these approximations increase simulation speed by $\sim$40%. We also found that AI2-THOR scene loading and resetting was a significant bottleneck when using Objaverse assets. We streamline AI2-THOR's Objaverse 3D asset loading pipeline by saving objects in the `Msgpack` format (natively usable with Unity's C# backend) as opposed to Python-only `Pickle` files. This change dramatically decreases the loading time of a new scene from $\sim$25s to $\sim$8s. With all these changes, our training steps per second increases from $\sim$550 to $\sim$950 using 192 rollouts on 32 GPUs for training Stretch RE-1 agent, reducing the training time by $\sim$42%.

4 Results

We now present our experimental results. In Sec. 4.1 we demonstrate that scaling RL with POLIFORMER produces SoTA results on four simulation benchmarks across two embodiments (LoCoBot and Stretch RE-1). Then, in Sec. 4.2, we show that POLIFORMER, despite being trained purely in simulation, transfers very effectively to the real world, outperforming previous work; we again show these results on the above two embodiments. In Sec. 4.3, we provide ablations for various design choices. Finally, we qualitatively show that POLIFORMER-BOXNAV can be extended to a variety of downstream applications in a zero-shot manner in Sec. 4.4.

Baselines. For our baselines, we chose a set of prior works in both imitation learning and reinforcement learning. SPOC [6] is a supervised imitation learning baseline trained on shortest path expert trajectories in AI2THOR. SPOC* is similar to SPOC, but is trained on more expert trajectories (2.3M *vs.* 100k). We build this baseline to verify if SPOC easily scales with more data. Emb-SigLIP [6] is an RL baseline also used by [6], but trained with comparable GPU hours with SPOC. The baselines [11, 63, 87] for LoCoBot-embodiment are all trained by RL using the same training steps used by POLIFORMER. We have three experiment configurations for our studies, specifying the goal as (a) natural language instruction text, (b) a b-box for the target object, and (c) b-box & text. Following [6], b-boxes in simulation are the *ground-truth* b-boxes, and so such models are not comparable fairly with RGB-only models. In the real world we replace GT b-boxes with outputs from the open-vocabulary object detector Detic [18], which uses only RGB images as input, and so all comparisons are fair.

Implementation Details. For LoCoBot benchmarks, we follow [87] to train our model and baselines on 10k training scenes in ProcTHOR houses for 435M training steps. The evaluation sets consist of 800 tasks in 20 scenes for AI2-iTHOR, 1200 tasks in 5 scenes for ARCHITECTHOR,

Inputs	Model	Loss	CHORES-S ObjectNav Success (SEL)
RGB+text	SPOC [6]	IL	57.0 (46.2)
	SPOC*	IL	60.0 (30.5)
	EmbSigLIP [6]	RL	36.5 (24.5)
	POLIFORMER	RL	**85.5** (**61.2**)
RGB +text+b-box	SPOC	IL	85.0 (61.4)
	POLIFORMER	RL	**95.5** (**71.4**)
RGB+b-box	POLIFORMER	RL	92.0 (73.9)

(a) Stretch RE-1 on CHORES-S

Inputs	Model	PROCTHOR-10k Success (SPL)	ARCHITECTHOR Success (SPL)	AI2-iTHOR Success (SPL)
RGB+text	PROCTHOR [11][3]	67.7 (49.0)	55.8 (38.3)	70.0 (57.1)
	SGC [63]	70.8 (48.6)	53.8 (34.8)	71.4 (59.3)
	EmbCodebook [87]	73.7 (48.4)	58.3 (35.6)	78.4 (23.7)
	POLIFORMER	**82.4** (**58.5**)	**68.3** (**45.1**)	**85.3** (**72.7**)
RGB +text+b-box	POLIFORMER	90.4 (66.6)	81.9 (55.6)	94.9 (83.5)
RGB+b-box	POLIFORMER	87.4 (56.2)	85.7 (47.6)	92.1 (78.6)

(b) LoCoBot on ProcTHOR-10k (val), ArchitecTHOR and AI2-iTHOR (test)

Table 1: Across four ObjectNav benchmarks, POLIFORMER obtains SoTA performance. (a) Results on the CHORES-S ObjectNav benchmark, which uses the Stretch RE-1 embodiment, POLIFORMER dramatically outperforms the previous SoTA, IL-trained SPOC. (b) On three LoCoBot-embodiment test suites, POLIFORMER outperforms all prior work (all trained using RL).

Vision Backbone	Encoder	Decoder	PROCTHOR-10k (val) Success	ARCHITECTHOR Success
CLIP (ResNet50)	1x CNN	1x GRU	67.7	55.8
DINOv2 (ViTs)	1x CNN	1x GRU	73.1	60.8
DINOv2 (ViTs)	3x Tx	3x GRU	73.6	59.8
DINOv2 (ViTs)	3x Tx	1x Tx	77.2	59.1
DINOv2 (ViTs)	3x Tx	3x Tx	80.4	63.9
DINOv2 (ViTb)	3x Tx	3x Tx	**82.4**	**68.3**

(a) Ablations on design choices for scaling model capacity

Model	Stretch RE-1	LoCoBot
ProcTHOR [11]	-	26.7
Phone2Proc [17]	-	66.7
SPOC [6]	50.0	-
POLIFORMER (ours)	**83.3**	**80.0**
SPOC+Detic [6]	83.3	-
POLIFORMER +Detic (ours)	**88.9**	-

(b) Real-world results - Success

Table 2: We present (a) ablation studies on design choices for scaling up model capacity; and (b) the real-world results, on two different embodiements.

and 1500 tasks in 150 scenes for PROCTHOR-10k. For Stretch-RE1, we train our model and baselines on 150k ProcTHOR houses populated with Objaverse assets processed by ObjaTHOR. Then, we evaluate on 200 tasks in 200 scenes as in [6]. Training takes 4.5 days on 32 GPUs. Please see App. B for more details about training, inference, and environment setups.

4.1 POLIFORMER Achieves SoTA on four Benchmarks

Table 1a shows that, on the CHORES-S benchmark, POLIFORMER achieves 28.5% absolute gain in success rate over the previous best model. EmbSigLIP baseline results are taken from [6]; we suspect that its poor performance is, in part, due to its training budget being limited to match the training budget of SPOC.

POLIFORMER works across embodiments. Table 1b shows that the remarkable performance of POLIFORMER is not limited to one embodiment. The LoCoBot and Stretch-RE1 robots have different body sizes, action spaces, and camera configurations. Nevertheless, POLIFORMER is able to achieve 8.7%, 10.0%, and 6.9% absolute gain over the best baseline on PROCTHOR-10k (val), ARCHITECTHOR, and AI2-iTHOR.

4.2 POLIFORMER Generalizes to the Real World

We evaluated POLIFORMER on two real-world benchmarks for two different embodiments to show its sim-to-real transfer capabilities. We used the same evaluation set of 15 tasks for LoCoBot (3 different starting poses with 5 different targets), and 18 tasks for Stretch-RE1 (3 different starting poses with 6 different goal specifications), similar to [17] and [6], respectively. We do not use any real-world finetuning, and the testing houses were not seen during training. In the text-only setting, POLIFORMER achieves a remarkable improvement of 33.3% and 13.3% over the baselines for Stretch-RE1 and LoCoBot respectively (Tab. 2b). POLIFORMER even outperforms Phone2Proc [17], a baseline that has access to privileged information about the layout of the environment for finetuning.

4.3 Ablation Studies

The recipe for building the effective RL navigation model is the end product of several design decisions guided by experimentation, the combination resulting in our SoTA performance. Table 2a, shows the process of scaling POLIFORMER. Row 1 *vs.* Row 2: Moving from the CLIP ResNet-50 visual encoder to the ViT-small DINOv2 improves the success rate by an average of 5.2%.

[3]We report ProcTHOR results from [87] as these are the most up-to-date for recent AI2-THOR versions.

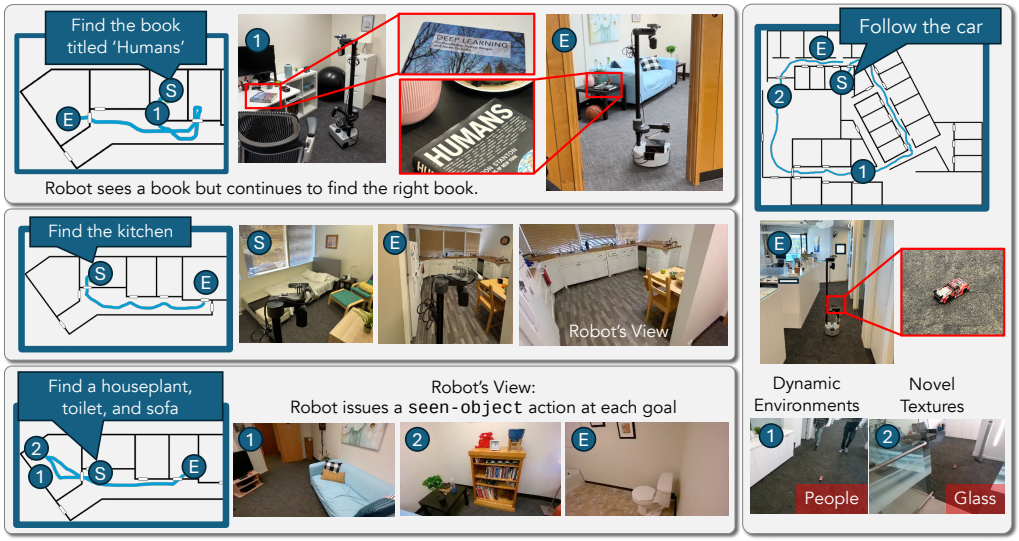

Figure 3: We use POLIFORMER-BOXNAV zero-shot to find a book with a particular title, navigate to a kitchen, navigate to multiple objects sequentially, and follow a toy car around an office building.

Row 3 *vs.* Row 5: Using a 3-layer transformer decoder instead of 3-layer GRU increases the performance by 5.5%. Row 4 *vs.* Row 5: Increasing the number of decoder layers from 1 to 3 increases the performance by 4%. Row 5 *vs.* Row 6: Using a larger capacity visual encoder (ViT-b instead of ViT-s), results in a 3.2% gain. In all, these changes result in a 13.6% absolute avg. improvement.

Has POLIFORMER performance saturated? Figure 1 (*top-left*) shows CHORES-$\mathbb{S}$ ObjectNav validation success rate as we train POLIFORMER for $\sim$700M environment steps. As that plot shows, POLIFORMER's performance does not appear to have converged, and we expect that performance will continue to improve with more compute. This suggests that achieving near-perfect ObjectNav performance may only require further scaling our training approach.

4.4 Scaling POLIFORMER to Everyday Tasks

By specifying POLIFORMER's goal purely using b-boxes, we produce POLIFORMER-BOXNAV. While POLIFORMER-BOXNAV lacks the ability to leverage some helpful priors about where objects of certain types should reside (comparing rows 6 & 7 in Table 1a we see POLIFORMER-BOXNAV performs slightly worse than a model having both b-box and text goal specifications), this design decision makes POLIFORMER-BOXNAV a **fully general-purpose, promptable, navigation system** that can navigate to *any goal specifiable using a b-box*. Figure 3 shows four qualitative examples where, by using an open-vocabulary object detector (Detic [18]) and a VLM (GPT-4o [91]), POLIFORMER-BOXNAV is able to, zero-shot, navigate to (1) a book with a particular title, (2) a given room type, (3) multiple objects sequentially, and (4) a toy car as the car is driven around an office building. Simply by specifying goals as b-boxes and leveraging off-the-shelf systems, we obtain complex navigation behaviors that would otherwise need to be trained for individually: a painful process requiring new reward functions or data collection. Finally, we encourage readers to visit our website to view more qualitative results presented in video format.

5 Discussion

Limitations: Training RL agents for long-horizon tasks with a large search space requires extensive compute and demands careful reward shaping. While we believe POLIFORMER is capable of scaling to other tasks, it requires crafting new reward models for novel tasks such as manipulation. More discussion on limitations in App. E. **Conclusion:** In this paper we provide a recipe for scaling RL for long-horizon navigation tasks. Our model, POLIFORMER, achieves SoTA results on four simulation benchmarks and two real-world benchmarks across two different embodiments. We also show that POLIFORMER has remarkable potential for use in downstream everyday tasks.

References

[1] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1gX8C4YPr.

[2] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014. doi:10.3115/V1/D14-1179. URL https://doi.org/10.3115/v1/d14-1179.

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

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

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

[6] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti, et al. Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. In *CVPR*, 2024.

[7] N. Rajaraman, Y. Han, L. F. Yang, K. Ramchandran, and J. Jiao. Provably breaking the quadratic error compounding barrier in imitation learning, optimally. *ArXiv*, abs/2102.12948, 2021. URL https://api.semanticscholar.org/CorpusID:232046264.

[8] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, A. Kembhavi, A. K. Gupta, and A. Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *ArXiv*, abs/1712.05474, 2017. URL https://api.semanticscholar.org/CorpusID:28328610.

[9] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning robust visual features without supervision, 2023.

[10] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, J. Heek, K. Xiao, S. Agrawal, and J. Dean. Efficiently scaling transformer inference. In *MLSys*, 2023.

[11] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, K. Ehsani, J. Salvador, W. Han, E. Kolve, A. Kembhavi, and R. Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/27c546ab1e4f1d7d638e6a8dfbad9a07-Abstract-Conference.html.

[12] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A Universe of Annotated 3D Objects. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2022. URL https://api.semanticscholar.org/CorpusID:254685588.

[13] Y. Yang, F. Sun, L. Weihs, E. VanderBilt, A. Herrasti, W. Han, J. Wu, N. Haber, R. Krishna, L. Liu, C. Callison-Burch, M. Yatskar, A. Kembhavi, and C. Clark. Holodeck: Language guided generation of 3d embodied AI environments. *CoRR*, abs/2312.09067, 2023. doi:10.48550/ARXIV.2312.09067. URL https://doi.org/10.48550/arXiv.2312.09067.

[14] A. I. for AI. ObjaTHOR: Python package for importing and loading external assets into ai2thor. https://github.com/allenai/objathor, 2024.

[15] A. Gupta, A. Murali, D. P. Gandhi, and L. Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. In *NeurIPS*, 2018.

[16] C. C. Kemp, A. Edsinger, H. M. Clever, and B. Matulevich. The Design of Stretch: A Compact, Lightweight Mobile Manipulator for Indoor Human Environments. In *2022 International Conference on Robotics and Automation, ICRA 2022, Philadelphia, PA, USA, May 23-27, 2022*, pages 3150–3157. IEEE, 2022. doi:10.1109/ICRA46639.2022.9811922. URL https://doi.org/10.1109/ICRA46639.2022.9811922.

[17] M. Deitke, R. Hendrix, L. Weihs, A. Farhadi, K. Ehsani, and A. Kembhavi. Phone2Proc: Bringing robust robots into our chaotic world, 2022.

[18] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra. Detecting twenty-thousand classes using image-level supervision. In S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part IX*, volume 13669 of *Lecture Notes in Computer Science*, pages 350–368. Springer, 2022. doi:10.1007/978-3-031-20077-9_21. URL https://doi.org/10.1007/978-3-031-20077-9_21.

[19] M. Minderer, A. Gritsenko, and N. Houlsby. Scaling open-vocabulary object detection. In *NeurIPS*, 2023.

[20] J. Bai, S. Bai, S. Yang, S. Wang, S. Tan, P. Wang, J. Lin, C. Zhou, and J. Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023.

[21] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[22] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W. Lo, P. Dollár, and R. B. Girshick. Segment anything. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 3992–4003. IEEE, 2023. doi:10.1109/ICCV51070.2023.00371. URL https://doi.org/10.1109/ICCV51070.2023.00371.

[23] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra. ZSON: zero-shot object-goal navigation using multimodal goal embeddings. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/d0b8f0c8f79d3a621af945cafb669f4b-Abstract-Conference.html.

[24] S. Wani, S. Patel, U. Jain, A. X. Chang, and M. Savva. MultiON: Benchmarking Semantic Map Memory using Multi-Object Navigation. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6e01383fd96a17ae51cc3e15447e7533-Abstract.html.

[25] R. Ramrakhya, E. Undersander, D. Batra, and A. Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5163–5173. IEEE, 2022. doi:10.1109/CVPR52688.2022.00511. URL https://doi.org/10.1109/CVPR52688.2022.00511.

[26] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural slam. *ICLR*, 2020.

[27] B. Yamauchi. A frontier-based approach for autonomous exploration. In *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. IEEE, 1997.

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

[29] X. Puig, E. Undersander, A. Szot, M. D. Cote, T. Yang, R. Partsey, R. Desai, A. W. Clegg, M. Hlavac, S. Y. Min, V. Vondrus, T. Gervet, V. Berges, J. M. Turner, O. Maksymets, Z. Kira, M. Kalakrishnan, J. Malik, D. S. Chaplot, U. Jain, D. Batra, A. Rai, and R. Mottaghi. Habitat 3.0: A Co-Habitat for Humans, Avatars and Robots. *CoRR*, abs/2310.13724, 2023. doi:10.48550/ARXIV.2310.13724. URL https://doi.org/10.48550/arXiv.2310.13724.

[30] S. Yenamandra, A. Ramachandran, K. Yadav, A. Wang, M. Khanna, T. Gervet, T. Yang, V. Jain, A. W. Clegg, J. M. Turner, Z. Kira, M. Savva, A. X. Chang, D. S. Chaplot, D. Batra, R. Mottaghi, Y. Bisk, and C. Paxton. Homerobot: Open-vocabulary mobile manipulation. *CoRR*, abs/2306.11565, 2023. doi:10.48550/ARXIV.2306.11565. URL https://doi.org/10.48550/arXiv.2306.11565.

[31] K. Ehsani, A. Farhadi, A. Kembhavi, and R. Mottaghi. Object manipulation via visual target localization. In S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXIX*, volume 13699 of *Lecture Notes in Computer Science*, pages 321–337. Springer, 2022. doi:10.1007/978-3-031-19842-7_19. URL https://doi.org/10.1007/978-3-031-19842-7_19.

[32] N. Yokoyama, A. Clegg, E. Undersander, S. Ha, D. Batra, and A. Rai. Adaptive skill coordination for robotic mobile manipulation. *ArXiv*, abs/2304.00410, 2023. URL https://api.semanticscholar.org/CorpusID:261886226.

[33] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y. Chao, and D. Fox. RVT: Robotic View Transformer for 3D Object Manipulation. *CoRR*, abs/2306.14896, 2023. doi:10.48550/ARXIV.2306.14896. URL https://doi.org/10.48550/arXiv.2306.14896.

[34] J. Gu, D. S. Chaplot, H. Su, and J. Malik. Multi-skill Mobile Manipulation for Object Rearrangement. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=Z3IClM_bzvP.

[35] J. Yang, C. Glossop, A. Bhorkar, D. Shah, Q. Vuong, C. Finn, D. Sadigh, and S. Levine. Pushing the limits of cross-embodiment learning for manipulation and navigation. In *arXiv preprint arXiv:2402.19432*, 2024.

[36] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[37] M. Chang, T. Gervet, M. Khanna, S. Yenamandra, D. Shah, S. Y. Min, K. Shah, C. Paxton, S. Gupta, D. Batra, et al. GOAT: Go to any thing. *arXiv preprint arXiv:2311.06430*, 2023.

[38] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot. Navigating to objects in the real world. *Science Robotics*, 2023.

[39] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi. Simple but effective: CLIP embeddings for embodied AI. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 14809–14818. IEEE, 2022. doi:10.1109/CVPR52688.2022.01441. URL https://doi.org/10.1109/CVPR52688.2022.01441.

[40] M. Shridhar, L. Manuelli, and D. Fox. CLIPort: What and where pathways for robotic manipulation. In *CoRL*, 2022.

[41] A. Majumdar, K. Yadav, S. Arnaud, Y. J. Ma, C. Chen, S. Silwal, A. Jain, V. Berges, P. Abbeel, J. Malik, D. Batra, Y. Lin, O. Maksymets, A. Rajeswaran, and F. Meier. Where are we in the search for an artificial visual cortex for embodied intelligence? *CoRR*, abs/2303.18240, 2023. doi:10.48550/ARXIV.2303.18240. URL https://doi.org/10.48550/arXiv.2303.18240.

[42] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K. Lee, Y. Kuang, S. Jesmonth, N. J. Joshi, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and C. K. Fu. Do as I can, not as I say: Grounding language in robotic affordances. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR, 2022. URL https://proceedings.mlr.press/v205/ichter23a.html.

[43] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, et al. PIVOT: Iterative visual prompting elicits actionable knowledge for vlms. In *arXiv preprint arXiv:2402.07872*, 2024.

[44] S. Srivastava, C. Li, M. Lingelbach, R. Mart'in-Mart'in, F. Xia, K. Vainio, Z. Lian, C. Gokmen, S. Buch, C. K. Liu, S. Savarese, H. Gweon, J. Wu, and L. Fei-Fei. BEHAVIOR: Benchmark for Everyday Household Activities in Virtual, Interactive, and Ecological Environments. In *Conference on Robot Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:236957374.

[45] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun, M. Anvari, M. Hwang, M. Sharma, A. Aydin, D. Bansal, S. Hunter, K. Kim, A. Lou, C. R. Matthews, I. Villa-Renteria, J. H. Tang, C. Tang, F. Xia, S. Savarese, H. Gweon, K. Liu, J. Wu, and L. Fei-Fei. BEHAVIOR-1K: A Benchmark for Embodied AI with 1, 000 Everyday Activities and Realistic Simulation. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 80–93. PMLR, 2022. URL https://proceedings.mlr.press/v205/li23a.html.

[46] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. S. Ryoo, G. Salazar, P. R. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. T. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. RT-1: robotics transformer for real-world control at scale. In K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi:10.15607/RSS.2023.XIX.025. URL https://doi.org/10.15607/RSS.2023.XIX.025.

[47] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. S. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. T. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. RT-2: vision-language-action models transfer web knowledge to robotic control. *CoRR*, abs/2307.15818, 2023. doi:10.48550/ARXIV.2307.15818. URL https://doi.org/10.48550/arXiv.2307.15818.

[48] O. X.-E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. In *ICRA*, 2024.

[49] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *arXiv preprint arXiv:2403.12945*, 2024.

[50] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *RSS*, 2024.

[51] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, X. Yuan, P. Xie, Z. Huang, R. Chen, and H. Su. ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills. In *The Eleventh International Conference on Learning Rep-*

resentations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023. URL https://openreview.net/pdf?id=b_CQDy9vrD1.

[52] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. E. Vainio, C. Gokmen, G. Dharan, T. Jain, A. Kurenkov, C. K. Liu, H. Gweon, J. Wu, L. Fei-Fei, and S. Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In A. Faust, D. Hsu, and G. Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 455–465. PMLR, 2021. URL https://proceedings.mlr.press/v164/li22b.html.

[53] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D'Arpino, S. Buch, S. Srivastava, L. Tchapmi, M. Tchapmi, K. Vainio, J. Wong, L. Fei-Fei, and S. Savarese. iGibson 1.0: A Simulation Environment for Interactive Tasks in Large Realistic Scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*, pages 7520–7527. IEEE, 2021. doi:10.1109/IROS51168.2021.9636667. URL https://doi.org/10.1109/IROS51168.2021.9636667.

[54] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. M. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. X. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra. Habitat 2.0: Training Home Assistants to Rearrange their Habitat. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 251–266, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/021bbc7ee20b71134d53e20206bd6feb-Abstract.html.

[55] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su. SAPIEN: A SimulAted Part-based Interactive ENvironment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[56] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba. VirtualHome: Simulating Household Activities via Programs. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8494–8502. Computer Vision Foundation / IEEE Computer Society, 2018. doi:10.1109/CVPR.2018.00886. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Puig_VirtualHome_Simulating_Household_CVPR_2018_paper.html.

[57] C. Gan, J. Schwartz, S. Alter, D. Mrowca, M. Schrimpf, J. Traer, J. D. Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, M. Sano, K. Kim, E. Wang, M. Lingelbach, A. Curtis, K. T. Feigelis, D. Bear, D. Gutfreund, D. D. Cox, A. Torralba, J. J. DiCarlo, J. Tenenbaum, J. H. McDermott, and D. Yamins. ThreeDWorld: A Platform for Interactive Multi-Modal Physical Simulation. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/735b90b4568125ed6c3f678819b6e058-Abstract-round1.html.

[58] Z. Zhang and L. Weihs. When learning is out of reach, reset: Generalization in autonomous visuomotor reinforcement learning. *arXiv preprint arXiv: Arxiv-2303.17600*, 2023.

[59] M. Khanna, Y. Mao, H. Jiang, S. Haresh, B. Schacklett, D. Batra, A. Clegg, E. Undersander, A. X. Chang, and M. Savva. Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation. In *CVPR*, 2024.

[60] Z. D. Guo, M. G. Azar, B. Piot, B. A. Pires, and R. Munos. Neural predictive belief representations. *ICLR*, 2019.

[61] Z. D. Guo, B. A. Pires, B. Piot, J.-B. Grill, F. Altché, R. Munos, and M. G. Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*, 2020.

[62] K.-H. Zeng, L. Weihs, A. Farhadi, and R. Mottaghi. Pushing it out of the way: Interactive visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[63] K. P. Singh, J. Salvador, L. Weihs, and A. Kembhavi. Scene Graph Contrastive Learning for Embodied Navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10884–10894, October 2023.

[64] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 15084–15097, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html.

[65] Q. Zheng, A. Zhang, and A. Grover. Online decision transformer. In *ICML*, 2022.

[66] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4RL: Datasets for deep data-driven reinforcement learning. *ArXiv*, abs/2004.07219, 2020. URL https://api.semanticscholar.org/CorpusID:215827910.

[67] L. Meng, M. Wen, C. Le, X. Li, D. Xing, W. Zhang, Y. Wen, H. Zhang, J. Wang, Y. Yang, et al. Offline pre-trained multi-agent decision transformer. *Machine Intelligence Research*, 2023.

[68] X. Huang, D. Batra, A. Rai, and A. Szot. Skill transformer: A monolithic policy for mobile manipulation. In *CVPR*, 2023.

[69] E. Parisotto, F. Song, J. Rae, R. Pascanu, C. Gulcehre, S. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury, et al. Stabilizing transformers for reinforcement learning. In *ICML*, 2020.

[70] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, J. Schrittwieser, K. Anderson, S. York, M. Cant, A. Cain, A. Bolton, S. Gaffney, H. King, D. Hassabis, S. Legg, and S. Petersen. Deepmind lab. *ArXiv*, abs/1612.03801, 2016. URL https://api.semanticscholar.org/CorpusID:3221395.

[71] H. Mao, R. Zhao, Z. Li, Z. Xu, H. Chen, Y. Chen, B. Zhang, Z. Xiao, J. Zhang, and J. Yin. PDiT: Interleaving perception and decision-making transformers for deep reinforcement learning. In *AAMAS*, 2024.

[72] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. In *TMLR*, 2022.

[73] X. Xiao, T. Zhang, K. Choromanski, E. Lee, A. Francis, J. Varley, S. Tu, S. Singh, P. Xu, F. Xia, et al. Learning model predictive controllers with real-time attention for real-world navigation. In *CoRL*, 2023.

[74] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, et al. Rethinking attention with performers. In *ICLR*, 2021.

[75] P. Parashar, V. Jain, X. Zhang, J. Vakil, S. Powers, Y. Bisk, and C. Paxton. Spatial-language attention policies for efficient robot learning. In *CoRL*, 2023.

[76] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 2024.

[77] H. Wang, A. H. Tan, and G. Nejat. Navformer: A transformer architecture for robot target-driven navigation in unknown and dynamic environments. In *IEEE Robotics and Automation Letters*, 2024.

[78] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. GNM: A general navigation model to drive any robot. In *ICRA*, 2023.

[79] A. Sridhar, D. Shah, C. Glossop, and S. Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *ICRA*, 2023.

[80] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. Vint: A foundation model for visual navigation. In *CoRL*, 2023.

[81] T. Haarnoja, A. Zhou, S. Ha, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. *ArXiv*, abs/1812.11103, 2018. URL https://api.semanticscholar.org/CorpusID:57189150.

[82] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17:39:1–39:40, 2015. URL https://api.semanticscholar.org/CorpusID:7242892.

[83] W. Yang, X. Wang, A. Farhadi, A. K. Gupta, and R. Mottaghi. Visual semantic navigation using scene priors. *ArXiv*, abs/1810.06543, 2018. URL https://api.semanticscholar.org/CorpusID:53116049.

[84] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013. URL https://api.semanticscholar.org/CorpusID:15238391.

[85] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[86] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. Canton-Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. doi:10.48550/ARXIV.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

[87] A. Eftekhar, K.-H. Zeng, J. Duan, A. Farhadi, A. Kembhavi, and R. Krishna. Selective visual representations improve convergence and generalization for embodied ai. *arXiv preprint arXiv:2311.04193*, 2023.

[88] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *JMLR*, 2020.

[89] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin,

A. Roberts, D. Zhou, Q. V. Le, and J. Wei. Scaling instruction-finetuned language models, 2022. URL https://arxiv.org/abs/2210.11416.

[90] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *arXiv preprint arXiv:1412.3555*, 2014.

[91] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi:10.48550/ARXIV.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

[92] L. Weihs, J. Salvador, K. Kotar, U. Jain, K.-H. Zeng, R. Mottaghi, and A. Kembhavi. AllenAct: A framework for embodied ai research. *arXiv preprint arXiv:2008.12760*, 2020.

[93] J. Singla, A. Agarwal, and D. Pathak. Sapg: Split and aggregate policy gradients. In *arXiv preprint arXiv:2407.20230*, 2024.

Appendices for *PoliFormer: Scaling On-Policy RL with Transformers Results in Masterful Navigators*

These appendices contain additional information about our:

- Zero-shot real-world applications (App. A),
- Training procedure (App. B),
- Environment, benchmarks, and quantitative real-world experiments (App. C),
- Simulation evaluations (App. D), and
- Limitations (App. E).

Please find our project website (see the poliformer.allen.ai) that contains

- Six real-world qualitative videos where POLIFORMER performs the everyday tasks of Section 4.4 (recall also Figure 3), and
- Four qualitative videos in simulation showing our POLIFORMER's behavior in the four benchmark environments (CHORES, PROCTHOR, AI2-iTHOR, and ARCHITECTHOR).

A Details about Zero-shot Real-world Downstream Applications using an Open-Vocab Object Detector and VLM

By specifying POLIFORMER's goal purely using b-boxes, we produce POLIFORMER-BOXNAV. POLIFORMER-BOXNAV is extremely effective at exploring its environment and, once it observes a bounding box, takes a direct and efficient path towards it. We now describe how we utilize this behavior to apply POLIFORMER-BOXNAV zero-shot to a variety of downstream applications by leveraging an open vocabulary object detector (Detic [18]) and a VLM (GPT-4o [91]).

Open Vocabulary ObjectNav. To perform open vocabulary object navigation (*i.e.*, where one must navigate to any given object type), we simply prompt the Detic object detector with the novel object type, for example, `Bicycle`. As POLIFORMER-BOXNAV relies on the b-box as its goal specification, it finds a bicycle in the scene smoothly.

Multi-target ObjectNav. To enable multi-target object goal navigation, we make a few simple modifications to the inputs and output of the Detic detector. On the input side, we query with multiple prompts simultaneously (one for each object type); for instance, `HousePlant`, `Toilet`, and `Sofa`, as shown in Fig. 3 (bottom-left). We then, on the output side, only return the b-box with the highest confidence score. Since the returned b-box also contains the predicted object type, we know what the target object the agent finds is when issuing a `Done` action. Therefore, we remove the found target from the list of target types, and reset the POLIFORMER's KV-cache. If the agent issues a `Done` action without a detected b-box, we terminate the episode and consider it a failure. As a result, the agent is required to find all the targets from the list of target types to succeed in an episode.

Human Following. We change the Detic prompt to `Person`. Once a b-box is detected, POLIFORMER drives the agent to approach it. Our experiment participant continues to walk away, so the agent keeps approaching them to minimize the distance.

Object Tracking. In this example, we control a remote control car that moves in the environment, and prompt the agent to find the car. Similar to **Human Following**, we change the prompt to `Toy Truck` in this example. As a result, the agent keeps trying to move closer to the detected b-box of the RC car, while avoiding collisions with objects in the dynamic scene.

Room Navigation. In this example, shown in Fig. 3 (middle-left), we provide no detections to the agent. As the agent sees no detections, it continuously explores the scene. As the agent explores, we

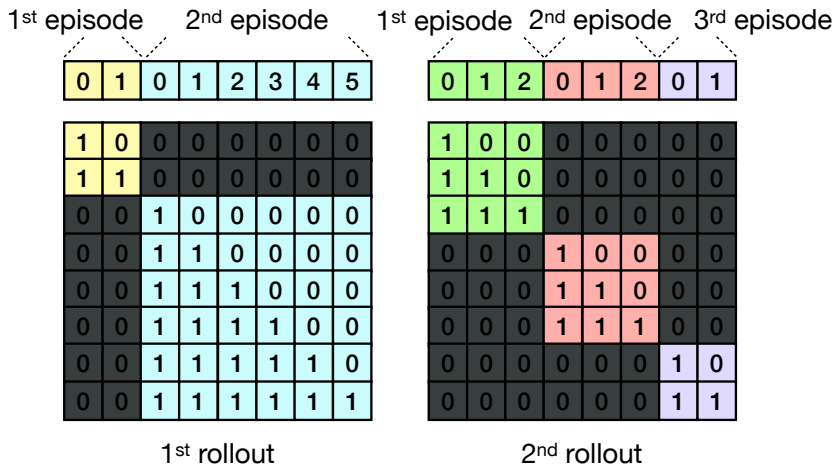

Figure 4: Attention Masks for training with block lower triangular structure.

query GPT4-o every 5 timesteps with the prompt `Am I in a Kitchen? Please return Yes or No.` with the most recent visual observation. Once GPT-4o returns `Yes`, the agent issues a `Done` action to end the episode.

Instance Description Navigation. In this example, shown in Fig. 3 (upper-left), the agent is prompted to find a specific book titled "Humans". Detic can generate open-vocabulary bounding boxes using instance-level descriptions but we found that doing this alone leads to high false-positive rates. To reduce these errors, we use GPT4-o to filter positive detections from Detic. In particular, a sample filtering prompt is "Is there a book titled "Humans" in this image? Please return Yes or No.". We find this combination works well in practice. The agent, not GPT-4o, remains responsible for deciding when it has successfully completed its task, and in the Fig. 3 example sees many books in its search but perseveres and eventually finds the correct one.

B Additional Training Details

B.1 Reward Shaping

For reward shaping, we follow EmbCodebook [87] and PROCTHOR [11] and use the implementation in AllenAct [92]: $\mathcal{R}_{penalty} + \mathcal{R}_{success} + \mathcal{R}_{distance}$, where $\mathcal{R}_{penalty} = -0.01$ encourages an efficient navigation, $\mathcal{R}_{success} = 10$ when the agent successfully completes the task ($= 0$ otherwise), and $\mathcal{R}_{distance}$ is the change of L2 distances from target between two consecutive steps. Note that we only provide a nonzero $\mathcal{R}_{distance}$ if the new distance is less than previously seen in the episode. We do not enforce a negative reward for increasing distance. This formulation encourages exploration.

B.2 Episodic Attention Mask

During training, to ensure that the causal transformer decoder cannot access observations or states across different episodes, we construct the episodic attention mask to only allow the past experiences within the same episode to be attended. In Fig. 4, we show a couple of possible rollouts collected during training. With the episodic attention mask, observations and states in an episode can only attend to previous ones within the same episode, in contrast with a naive causal mask where they could also potentially attend to observations and states in previous episodes.

B.3 Different Temporal Cache Strategies

Besides the KV-Cache, in this subsection, we ablate four different temporal cache strategies (shown in Fig. 5 top):

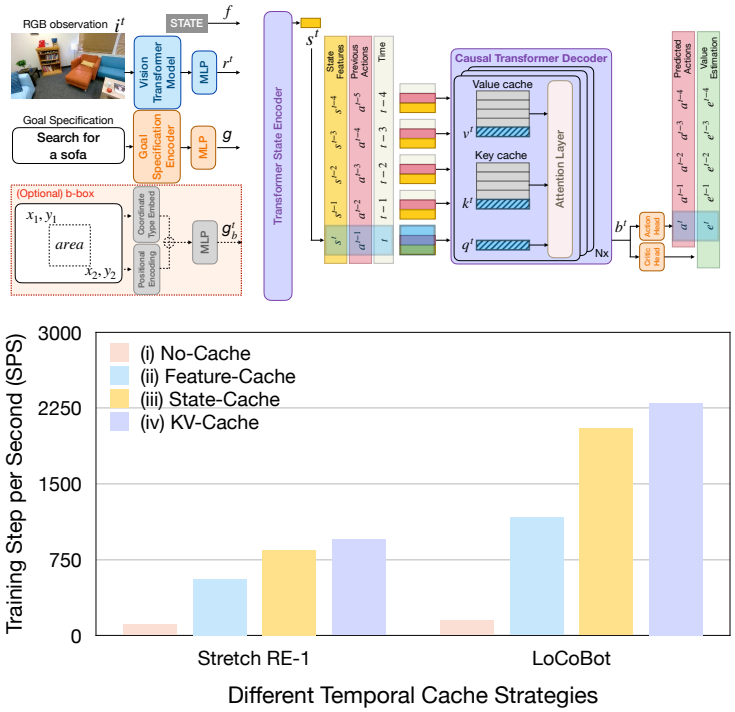

Figure 5: Different temporal cache strategies and their impact on the training speed. We ablate four different cache strategies, including (i) No-Cache, (ii) Feature-Cache, (iii) State-Cache, and (iv) KV-Cache, shown at top. The bottom chart shows the training Step per Second (SPS) achieved by different strategies, on both LoCoBot and Stretch RE-1 agents.

(i) No-Cache, we cache the raw frames (visual observation i^t) to provide past experiences for the causal transformer decoder. Therefore, POLIFORMER has to rerun the feedforward across all modules with the cached frames and the latest visual observation at each timestep.

(ii) Feature-Cache, we cache the features, including visual representation v^t and goal embedding g^t (as well as g_b^t). In this case, POLIFORMER needs to recompute the transformer state encoder and causal transformer decoder with the cached features and the latest feature at each timestep.

(iii) State-Cache, we cache the state feature vector s^t. Since it is right before the causal transformer decoder, POLIFORMER only requires to pass the cache state features and the latest state feature vector to the decoder, without recomputing the visual transformer model, goal encoder, and transformer state encoder.

(iv) KV-Cache, as described in Sec. 3.1, we cache the **K**eys and **V**alues inside the causal transformer decoder, further reducing the required computation time for the transformer decoder from t^2 to t theoretically. In addition, since this strategy operates at very end, all the recomputations required by modules preceding causal transformer decoder can be saved.

The speed profile results are shown in Fig. 5 bottom. It clearly shows that placing the cache closer to the causal transformer decoder improves the training efficiency significantly.

B.4 Hyperparameters for Training

Tab. 3 lists the hyperparameters used in our training and model architecture design. Please find more details such as scene texture randomization, visual observation augmentations, and goal specification randomization when using text instruction in our codebase.

Training and Model Details	
Parameter	**Value**
Allowed Steps	600 (Stretch RE-1), 500 (LoCoBot)
Total Rollouts	192 (Stretch RE-1), 384 (LoCoBot)
Learing Rate	0.002
Mini Batch per Update	1
Update Repeats	4
Max Gradient Norm	0.5
Discount Value Factor γ	0.99
GAE λ	0.95
PPO Surrogate Objective Clipping	0.1
Value Loss Weight	0.5
Entropy Loss Weight	0.01
Training Stages	3
Steps for PPO Update Stage 1	32
Steps for PPO Update Stage 2	64
Steps for PPO Update Stage 3	128
Transformer State Encoder Layers	3
Transformer State Encoder Hidden Dims	512
Transformer State Encoder Heads	8
Causal Transformer Deocder Layers	3
Causal Transformer Deocder Hidden Dims	512
Causal Transformer Deocder Heads	8

Table 3: Hyperparameters for training and model architecture.

C Additional Details about Environment, Benchmarks, and Real-World Experiments

Action Space. Following prior work using AI2-THOR, we discretize the action space for both LoCoBot and Stretch RE-1. For LoCoBot, we discretize the action space into 6 actions, including {`MoveAhead`, `RotateRight`, `RotateLeft`, `LookUp`, `LookDown`, `Done`}, where `MoveAhead` moves the agent forward by 0.2 meters, `RotateRight` rotates the agent clockwise by 30° around the yaw-axis, `RotateLeft` rotates the agent counter-clockwise by 30° around the yaw-axis, `LookUp` rotates agent's camera clockwise by 30° around the roll-axis, `LookDown` rotates agent's camera counter-clockwise by 30° around the roll-axis, and `Done` indicates that the agent found the target and ends an episode. We follow previous works [11, 17, 87] to use the same action space for LoCoBot for a fair comparison. For Stretch RE-1, we remove the `LookUp` and `LookDown` camera actions, and add `MoveBack`, `RotateRightSmall`, and `RotateLeftSmall` to the action space, where `MoveBack` moves the agent backward by 0.2 meters, `RotateRightSmall` rotates the agent clockwise by 6° around the yaw-axis, and `RotateLeftSmall` rotates the agent counter-clockwise by 6° around the yaw-axis. Again, this action space is identical to the one used in prior work [6] for fair comparison.

Success Criteria. We follow the definition of Object Goal Navigation defined in [3], where an agent must explore its environment to locate and navigate to an object of interest within an allowed number of steps n. The agent has to issue the `Done` action to indicate it found the target. The environment will then judge if the agent is within a distance d from the target and if the target can be seen in the agent's view. An episode is also classified as failed if the agent runs more than n steps without issuing any `Done` action. Across different benchmarks, n and d vary depending on the scenes size and complexity and agent's capabilities. We follow ProcTHOR [11] to use $n = 500$ and $d = 1$ meter for LoCoBot, and follow CHORES-$\mathbb{S}$ [6] to use $n = 600$ and $d = 2$ meters for Stretch RE-1.

SPL and SEL. Success Weighted by Path Length (SPL) and Success Weighted by Episode Legnth (SEL) are two popular evaluation metrics to evaluate how efficient an agent is to find the target. SPL is defined as $\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(l_i, p_i)}$, where N is the total number of episodes, S_i is a binary

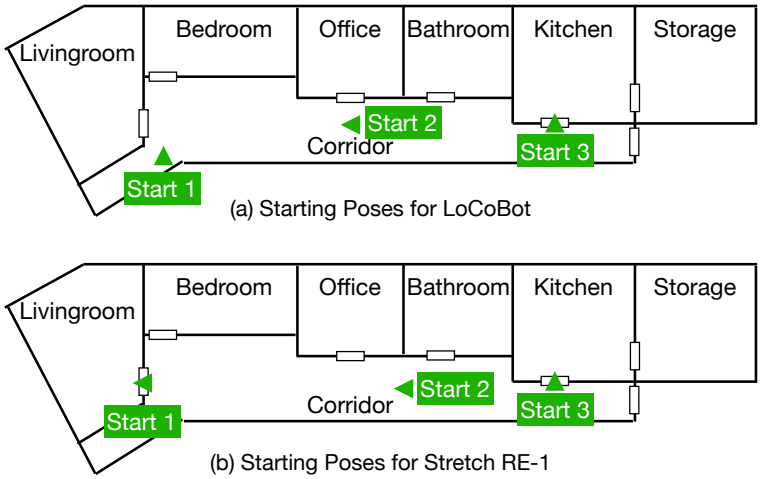

Figure 6: Starting Poses of (a) LoCoBot and (b) Stretch RE-1 used in the real world experiments. The arrow direction indicates where the agent faces with.

indicator of success for episode i, l_i is the shortest travel distance to the target, and p_i is the actual travel distance. SEL is defined similarly: $\frac{1}{N}\sum_{i=1}^{N} S_i \frac{w_i}{max(w_i,e_i)}$, where w_i is the shortest number of steps to find the target, and e_i is the actual number of steps used by the agent. By definition, SPL focuses on how far the agent has traveled, while SEL focuses on how many steps the agent has used (which also penalizes excessive in-place rotation). SPL can be derived by computing the geodesic distance between the agent's starting location and the target's location, while SEL needs a planner with privileged environment information to calculate the number of steps of expert trajectories. Therefore, we follow ProcTHOR [11] to report SPL to evaluate the LoCoBot agent, since those benchmarks do not provide planner, while we follow CHORES-$\mathbb{S}$ [6] to report SEL, since expert trajectories are available.

Real-world Experiment Setup. For the experiments using LoCoBot, we follow Phone2Proc [17] to use the same five target object categories, including `Apple`, `Bed`, `Sofa`, `Television`, and `Vase`, and the three starting poses, shown in Fig. 6 (a). Among those target categories, `Apple` can be found in the Living room and Kitchen, `Bed` can only be found in the Bedroom, `Sofa` and `Television` can only be found in the Living room, and `Vase` can be found in the Livingroom, Corridor, Office, and Kitchen. For the experiments using Stretch RE-1, we follow SPOC [6] to use the same six target object categories, including `Apple`, `Bed`, `Chair`, `HousePlant`, `Sofa`, and `Vase`, and the three starting poses, shown in Fig. 6 (b). Among the categories not mentioned above, `Chair` can be found in the Living room, Office, and Kitchen, and `HousePlant` can be found in the Living room, Office, Bathroom, and Kitchen. Note that we use the hardware design intorduced in SPOC [6] for Stretch RE-1. Instead of using the off-the-shelf camera equipped on the Stretch RE-1 (due to its narrow field of view), we use an Intel RealSense 455 fixed camera, which has a vertical field of view of 59° and a resolution of 1280×720. The camera is mounted facing forward but pointing downward, with the horizon at a nominal angle of 30°. Please find more details in App. C.1. in SPOC [6].

Details about Different Benchmarks

First, during the training and testing stages, the possible target object types the agent is tasked with searching for include `AlarmClock`, `Apple`, `BasketBall`, `Bed`, `Bowl`, `Chair`, `GarbageCan`, `HousePlant`, `Laptop`, `Mug`, `Sofa`, `SprayBottle`, `Television`, `Toilet`, and `Vase`. We then outline the differences between the environments.

(i) AI2-iTHOR: This environment consists of 60 training scenes, 20 validation scenes, and 20 test scenes, and our training and evaluation never touch the training scenes. All the scenes

are created by human designers, including the structure, layout, and object placements, producing 750 episodes for validation and 800 episodes for testing. Moreover, iTHOR has 4 different styles of scenes, including LivingRoom, Kitchen, Bathroom, and Bedroom, where each style contains the objects which are semantically associated with it. Since each iTHOR scene only contains a single room, it doesn't require much exploration but rather focuses on recognition capability.

(ii) ArchitecTHOR: This environment consists of 10 high-quality large interactive houses designed by human designers, including 5 for validation and 5 for testing. Both validation and testing scenes have 1,200 episodes each. It includes multiple rooms, larger navigable spaces, and more objects to explore. Since they are much larger than iTHOR scenes, they serve as a better benchmark to test the agent's exploration ability. Since ArchitecTHOR is designed by human designers, it also conforms to human priors on room layout and object placement.

(iii) ProcTHOR-val: This environment contains 1,550 episodes across 150 procedurally generated houses. The way these houses are generated follows the same pipeline used in ProcTHOR. Therefore, the distribution of layout styles, number of rooms, and object placements respects the ProcTHOR-10k training houses' distribution, where our LoCoBot agent is trained.

(iv) CHORES-S: This environment contains 200 episodes across 200 procedurally generated houses, using the same approach as ProcTHOR. However, it includes the Objaverse 3D assets, presented by SPOC, which introduced 41,133 assets into the ProcTHOR-150k houses, creating a more diverse scene distribution. Our Stretch Agent is trained on 80,000 houses out of the ProcTHOR-150k training houses; the 200 CHORES-S test houses and episodes are not seen during training.

D More Simulation Evaluations

Performance Variance. On CHORES-$\mathbb{S}$, since we follow SPOC [6] to apply test-time data augmentation and non-deterministic action sampling, we found that performance varies even using the same checkpoint, especially given that we are only evaluating on 200 episodes. As a result, we re-evaluate our POLIFORMER and SPOC*[4] 16 times and report mean success rate (mSR) and standard deviation (std). POLIFORMER achieves 82.5% mSR with 1.897 std, while SPOC* achieves 56.7% mSR with 2.697 std. This result indicates that POLIFORMER not only achieves a higher mSR than SPOC*, but also exhibits more reliably consistent behavior, *i.e.* a lower std, when run on the same episodes multiple times.

Inputs	Model	Loss	**EasyObjectNav**	**RegularObjectNav**	**HardObjectNav**
			Success (SEL)	Success (SEL)	Success (SEL)
RGB+text	SPOC [6]	IL	62.9 (40.5)	48.2 (38.9)	34.1 (27.4)
	SPOC*	IL	69.7 (43.3)	53.5 (34.3)	31.0 (19.6)
	POLIFORMER	RL	**89.0** (**62.1**)	**82.6** (**71.8**)	**72.3** (**62.8**)
RGB +text+b-box	SPOC	IL	90.3 (67.7)	78.7 (62.6)	70.6 (52.5)
	POLIFORMER	RL	**98.1** (**86.5**)	**90.4** (**79.6**)	**86.0** (**75.0**)
RGB+b-box	POLIFORMER	RL	97.1 (83.2)	91.9 (79.8)	87.6 (75.0)

Table 4: Large-scale evaluation results with different difficulty tiers. We evaluate performance on 2,000 episodes per tier.

Larger Scale Simulation Benchmark using Stretch RE-1. To further analyze POLIFORMER's performance through different difficulty settings, we construct 3 different levels of Object Goal Navigation benchmarks, `EasyObjectNav`, `RegularObjectNav`, and `HardObjectNav`, where each

[4] SPOC* is similar to SPOC but is trained on more expert trajectories (2.3M *vs.* 100k).

level contains 2k episodes, using Stretch RE-1. We construct these differentiated tasks by ensuring the oracle expert path length between the agent and target is 1 to 3 meters long for `EasyObjectNav`, greater than 3 meters for `RegularObjectNav`, and larger than 10 meters for `HardObjectNav`. The results are shown in Tab. 4. We observe that every model performs better as the agent is closer to the target at the episode start. In addition, on `EasyObjectNav` the agent barely needs exploration to find the target. Thereby, we find that POLIFORMER lagging behind POLIFORMER-BOXNAV by $\sim$9% could result from a *Recognition Issue*. Moreover, the gap on `HardObjectNav` is widened to $\sim$13.7%, and it could result from an additional *Exploration Issue*. The performance gap between `HardObjectNav` and `EasyObjectNav` could also support that an *Exploration Issue* exists, but not just the *Recognition Issue*.

Average Collision Rate (%)	**PoliFormer w. Text goal**	**PoliFormer w. Box goal**	**PoliFormer w. Box + Text goal**
ProcTHOR-val	2.2	2.9	2.8
ArchitecTHOR	3.0	3.6	3.7
AI2-iTHOR	2.9	3.5	3.2

Table 5: Average Collision Rates for Different Benchmarks.

Average Collision Rate. As shown in Tab. 5. We measure the average collision rate achieved by POLIFORMER using the LoCoBot agent in ProcTHOR-val, ArchitecTHOR, and AI2-iTHOR. We compute the collision rate by $\frac{\#collision}{\#steps}$ within an episode and we average the rate across all evaluation episodes to obtain an average collision rate.

ProcTHOR-10k val	**PoliFormer w. Text goal**	**PoliFormer w. Box goal**	**PoliFormer w. Box + Text goal**
x=0	82.4 (58.5)	87.4 (56.2)	90.4 (66.6)
x=1	82.2 (56.7)	85.1 (55.0)	89.8 (65.3)
x=2	80.1 (52.9)	82.7 (51.1)	86.1 (59.7)
x=3	74.5 (41.9)	75.4 (42.3)	83.4 (50.1)
ArchitecTHOR	**PoliFormer w. Text goal**	**PoliFormer w. Box goal**	**PoliFormer w. Box + Text goal**
x=0	68.3 (45.1)	85.7 (47.6)	81.9 (55.6)
x=1	67.0 (43.5)	85.1 (47.2)	81.9 (53.3)
x=2	63.3 (39.5)	78.9 (43.6)	78.2 (50.0)
x=3	60.1 (31.8)	69.6 (34.5)	71.5 (39.9)
AI2-iTHOR	**PoliFormer w. Text goal**	**PoliFormer w. Box goal**	**PoliFormer w. Box + Text goal**
x=0	85.3 (72.7)	92.1 (78.6)	94.9 (83.5)
x=1	85.3 (71.0)	91.6 (78.3)	93.4 (81.4)
x=2	85.1 (67.0)	90.0 (74.1)	92.6 (78.1)
x=3	80.8 (59.1)	86.0 (64.3)	90.6 (69.6)

Table 6: Success Rate and SPL achieved by POLIFORMER with different level of noise across different benchmarks. We inject the gaussian noise $G(0, x * \sigma)$ into the action, where x is a variable to control the scale of the noise, and $\sigma = \frac{\text{step size}}{3}$ for movement and $\sigma = \frac{\text{rotation degree}}{3}$ for rotation. The LoCoBot agent has a step size of 0.25m and rotation degree of 30 degrees by default.

Robustness of POLIFORMER. To evaluate the robustness of PoliFormer, we inject the gaussian noise into the action only at the evaluation stage. The results using LoCoBot across ProcTHOR-val, ArchitecTHOR, and AI2-iTHOR are shown in Tab. 6. POLIFORMER is robust to action noise even when experiencing substantial perturbations ($3 \times \sigma$ is a very large and unrealistic amount of noise).

E Additional Discussion on Limitations

Depth Sensor. It is important to note that POLIFORMER is not equipped with a depth sensor (which has been proven to be effective for manipulation). While the lack of the depth sensor does not affect our agent's performance on navigation, we acknowledge that integrating the depth sensor into our visual representation is an interesting direction for future work, especially when considering mobile-manipulation extensions.

Discretized Action Space. To have a fair comparison with baselines, we use the same discretized action space in this work (see App. C). The discretized action space might not be efficient and realistic in many real-world scenarios where the agent must act in a timely manner.

Cross-embodiment. In this paper, we demonstrate that we can train POLIFORMER using LoCoBot and Stretch RE-1. However, we have not yet explored training a single POLIFORMER for both embodiments. We leave this interesting research direction as future work.

Further Scaling. Our training and validation curves strongly suggest that even further scaling of model parameters and training time may lead to even more masterful models than those we have trained in this work. This perspective is exciting and we hope to enable further scaling with more computation resources and better visual foundation models in the near future.

Failure Analysis. The main mode of failure for POLIFORMER is the agent's limited memory. POLIFORMER clearly demonstrates memorization capabilities and is able to perform long-horizon tasks by exploring large indoor scenes without access to explicit mapping. However, as the trajectories get longer (specifically after visiting more than 4 rooms in an environment), the agent's recollection of the rooms it has explored deteriorates and the robot might re-visit rooms that it has explored previously.

Extensive Use of GPUs and Significant Training Time. We acknowledge that PoliFormer requires a large number of GPUs to train efficiently. Our main focus in this work is on pushing the performance of navigation agents to its limit without constraints on computing resources. Given the trend of GPU improvements over the last 10 years, we believe that the resources used in this work will become relatively commonplace in the near future. Empirically, we found that training with a smaller number of GPUs (e.g., 8 A6000 in a single host) could yield similar final performance, with the downside of slower training speed ($\sim 3.5\times$ longer to train compared to 32 A6000 in four nodes). Moreover, there are many interesting research directions in improving training speed/efficiency; for example, since our submission, we have found that removing the average pooling from PoliFormer's visual encoder can increase convergence speed by up to $2\times$. E.g., for the LoCoBot agent, we keep the number of tokens (e.g., 256 tokens) from DINOv2, instead of spatially average pooling them to 7×7 (resulting in 49 tokens). This implies that a better visual encoder with a larger receptive view over the visual observation could even further speed up training. Furthermore, we believe that incorporating more advanced techniques such as mixed precision training, flash attention, and advanced GPUs optimized for transformer models could reduce the training resource requirements.

Sample Efficiency. We hypothesize that an "IL pretraining +RL finetuning" paradigm could further improve sample efficiency. Early in on-policy RL training, policies trained from scratch are effectively random and thus produce a vast amount of marginally useful experiences. Pretraining with IL may improve sample efficiency by providing helpful priors on navigation and thus produce meaningful RL gradients faster. While it is nontrivial to implement this two-stage pipeline, we hypothesize that training PoliFormer using IL on expert data (e.g., the shortest path trajectories from SPOC) first and finetuning using on-policy RL could be an option to improve the sample efficiency. Since we focus on obtaining SoTA performance in this work, we believe that improving sample efficiency, by an IL+RL pipeline or with new RL algorithms such as SAPG [93], is an exciting research direction for future work.