Auxiliary Tasks for Efficient Learning of Point-Goal Navigation

Saurabh Satish Desai
desaisau@oregonstate.edu
Oregon State University

Stefan Lee
leestef@oregonstate.edu
Oregon State University

Abstract

Top-performing approaches to embodied AI tasks like point-goal navigation often rely on training agents via reinforcement learning over tens of millions (or even billions) of experiential steps – learning neural agents that map directly from visual observations to actions. In this work, we question whether these extreme training durations are necessary or if they are simply due to the difficulty of learning visual representations purely from task reward. We examine the task of point-goal navigation in photorealistic environments and introduce three auxiliary tasks that encourage learned representations to capture key elements of the task – local scene geometry, transition dynamics of the environment, and progress towards the goal. Importantly, these can be evaluated independent of task performance and provide strong supervision for representation learning. Our auxiliary tasks are simple to implement and rely on supervision already present in simulators commonly used for point-goal navigation. Applying our auxiliary losses to agents from prior works, we observe a $>4\times$ improvement in sample efficiency – in 17 million steps, our augmented agents outperforms state-of-the-art agents trained for 72 million steps.

1. Introduction

There has been a recent surge of research activity on embodied agents operating in simulated, photorealistic environments – navigating to point-goals [1, 2, 5, 13, 9], following instructions [6, 17], and exploring to answer questions [25] based on visual observations. Training agents for these tasks is primarily done through large-scale reinforcement learning [26] leveraging high-throughput simulators [21]. These techniques learn neural agents that map visual observations directly to actions based solely on task reward. Taking point-goal navigation as an example, the state-of-the-art reinforcement learning approach (DD-PPO [26]) achieves near perfect performance on unseen environments. This is achieved at the cost of 2.5 billion simulator steps during training (or approximately 80 years of real-world experience). In this work, we question whether this scale of experience is necessary or a costly side effect of an impoverished setting for visual representation learning.

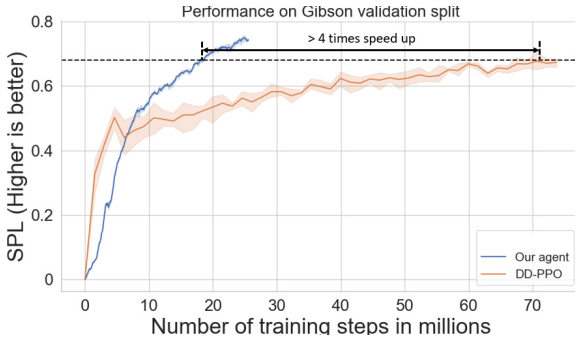

Figure 1. Learning to navigate from visual inputs is challenging. Agents must learn useful visual representations from task-feedback alone. In this work, we introduce auxiliary tasks that significantly improve the sample efficiency of training point-goal navigation agents. In the validation performance plot above, our agents achieve an SPL of 0.68 in 17 million steps which takes DD-PPO almost 72 million – a $\times 4$ improvement in sample efficiency.

We consider the setting of point-goal navigation as defined in [2]. In this task, an agent is spawned at a random location in a never-before-seen environment and must navigate to a goal location specified in relative coordinates. The agent has no access to a map of the environment and must take actions based on its visual observations and pose information alone. Starting from a randomly initialized network, training a competent agent requires overcoming two interconnected problems – learning to represent observations in a useful way and learning to use these representations to make correct decisions. In some sense, this is a chicken-and-egg problem. Without a strong task-relevant representation of observations it is difficult to take correct actions. Without positive reinforcement from the environment for correct actions, it is difficult to learn a strong task-relevant representation of observations.

This challenge is a result of only drawing representation supervision from task performance (i.e. scalar rewards for

navigation success); however, simulated environments are awash in representation-related signals that could be used to provide auxiliary supervision. Simulators provide multiple sensing modalities (e.g. RGB, depth, semantic segmentations); the agent's own actions reveal important information about environment dynamics (e.g. agents collide with obstacles); and intermediate task-level supervision is easy to compute (e.g. distance to goal). In this work, we consider three auxiliary tasks that draw on these signals to guide representation learning. Specifically, we consider:

- **Depth Prediction** – predicting coarse estimates of pixel-wise depth given the visual observation. This encourages capturing local scene geometry – key information to avoid collisions. This is especially important for agents that lack depth as an input. Such 'RGB' agents have historically learned slower and performed worse than their depth-equipped counterparts [21, 26].
- **Inverse Dynamics** – predicting an action taken between two sequential observations. This encourages representations to encode information about transition dynamics of the environment at the current state.
- **Remaining Path Length Prediction** – predicting the geodesic (or shortest-navigable-path) distance to the goal from the current location. Doing so accurately requires reasoning about long-range navigability. For example, a goal that is 8 meters away likely requires a complex path (and thus high geodesic distance) if the agent is in an interior room like a bedroom, but a relatively simpler path if the agent were in a hallway.

We implement each as a small auxiliary network that takes intermediate representations from the agent as input. Task loss is then used to guide these representations in training.

We find each of these tasks significantly improves the sample efficiency of our point-goal navigators. Building off agents from [21], our auxiliary-augmented agents perform as well or better in 15 million steps than the baseline models at 75 million – a $5\times$ reduction in the amount of required experience to reach the same performance! We also observe a $4\times$ speed up over the state-of-the-art DD-PPO algorithm which uses more complex models and a decentralized distributed training regime. This suggests that the extreme training regimes in point-goal navigation may not be necessary if auxiliary supervision is provided.

Contributions. To summarize our contributions, we:

- Introduce three auxiliary tasks to improve representation learning in point-goal navigation agents.
- Provide extensive experiments on agents with different auxiliary losses and input modalities. Including multiple runs to report mean and confidence intervals – training over 30 agents to 25 million steps of experience to provide comprehensive comparisons.
- Our auxiliary task-augmented agents provide significant gains in sample efficiency – achieving the same or greater performance as the baseline agents in $\sim$5x fewer steps. Further, we observe 4x improvement over state-of-the-art architectures (see Fig. 1).

2. Related work

Auxiliary Tasks in Reinforcement Learning. Poor sample efficiency is a common problem for reinforcement learning – especially for model-free agents like those commonly used for pixel-to-action tasks like point-goal navigation. As such, prior work has explored applying auxiliary tasks to boost training efficiency and improve performance. In [14], Jaderberg et al. introduced a suite of unsupervised auxiliary tasks for an RL agent in Atari games and simple maze environments. These unsupervised objectives do not leverage additional signals provided by the simulator – in contrast, our auxiliary losses include predicting additional observation modalities (RGB$\rightarrow$Depth) and aspects of the task itself (remaining path length).

Recent work has also examined auxiliary tasks to promote 'curiosity', improving the exploration abilities of agents. [20] does this by providing intrinsic reward to agents for taking actions which lead to predictable changes in the observation encoding. [20] learns to encode these observations via an inverse dynamics task like ours; however, their inverse dynamics model is separate from the agent and is just used to compute intrinsic 'curiosity' rewards. As such, they do not use this as an auxiliary loss for the agent's observation representation. There are also a number of works focusing on forward-prediction as a means to learn representation encoders [12, 11, 19]. A common challenge in these frameworks is how to determine the quality of future predictions, a task made even more complex in the perceptually rich environments we consider here.

Most relevant to our auxiliary tasks is the work of [18] which studied two auxiliary tasks for navigation in simple 3D maze environments – depth prediction and loop closure detection. As we are not operating in maze-like environments, we do not consider the loop closure tasks. We adapt [18]'s depth prediction task to our setting – significantly altering the architecture to fit the needs of our more visually complex setting. We compare against the original structure as a baseline and significantly outperform it.

Point-Goal Navigation. The point-goal navigation task is a fundamental embodied AI problem on which other complex tasks can be built. As such, it has received significant attention as interest in embodied tasks has grown [21, 22, 8, 26, 2, 1]. A variety of approaches have been proposed including those with strong inductive biases from simultaneous-localization-and-mapping research [8] to methods that draw purely from deep reinforcement learning [21, 26]. [26] demonstrates near-perfect performance on this task by training a neural architecture for over 2.5 billion steps of experience – amounting to over 80 years of continuously ex-

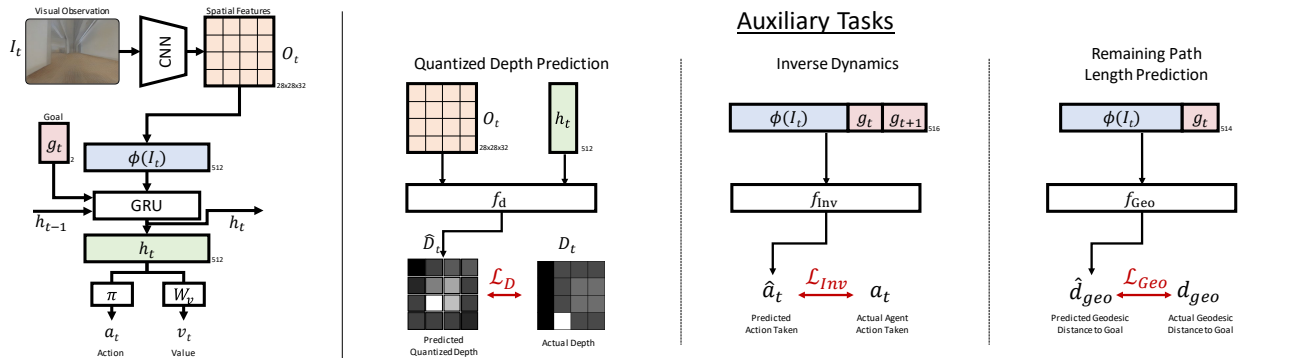

Figure 2. Our agent architecture (left) includes a simple convolutional observation encoder and a GRU-based recurrent policy. We consider three auxiliary tasks (right) – depth prediction, inverse dynamics, and remaining path length prediction – based on representations from the policy. Gradients for auxiliary task losses are propagated to corresponding agent modules, providing additional supervision to learned representation.

ploring simulated environments. We focus instead on improving sample efficiency for these agents. Similar in goal, [22] studies the use of pretrained mid-level representations as input signals for point-goal navigators – demonstrating improvements in sample efficiency. In contrast, we do not assume access to relevant pretrained encoders and instead seek to train a model from scratch efficiently.

3. Auxiliary Tasks for Point-Goal Navigation

Learning to control agents from visual perception presents a chicken-and-egg problem in difficult reinforcement learning settings. Making effective policy decisions requires being able to encode observations in a way that is useful to decision making; however, getting positive feedback from the environment to guide the learning of those representations requires a strong policy – or more often, luck and a mountain of experience. In this work, we leverage information from the training environment to provide auxiliary supervision to these representations to shortcut this dependency. We consider three auxiliary tasks: quantized depth prediction, inverse dynamics, and remaining path length prediction. These focus on enriching the representation with scene geometry, action consequences, and goal-awareness respectively. Refer to Fig. 2 for a schematic of our agent and auxiliary network architectures.

3.1. Preliminaries: Point-Goal Navigation

We consider the problem of point-goal navigation as defined in [21] – an agent is spawned in a never-before-seen environment and must reach a point given in relative coordinates. Agents observe the environment through some set of visual sensors (e.g. RGB or Depth) and are given their exact pose at each time step (e.g. as if they had a perfect GPS+Compass). Based on the goal and the current observation, the agent can turn in place or move forward. More formally, we consider a point-goal agent as a policy π that maps the goal coordinates g and visual observation I_t to a distribution over actions $\{\text{left } 15°, \text{right } 15°, \text{forward } 0.25\text{m}, \text{stop}\}$ at each timestep. A navigation episode ends when the agent calls `stop`.

Point-Goal Agent Models. Common formulations for these agents [21, 26] consider a decomposition of the agent into an observation encoder $\phi(\cdot)$ and a recurrent policy $f(\cdot)$, i.e. we can write

$$h_t = f(\phi(I_t), g_t, h_{t-1}) \quad (1)$$

$$a_t \sim \pi(h_t) = \text{Categorical}(W_a h_t) \quad (2)$$

where g_t is the agent-relative goal position computed from the pose P_t and goal coordinates g. Following [21], we implement $f(\cdot)$ as a Gated Recurrent Unit (GRU) [10] and $\phi(\cdot)$ as a simple CNN[1]. As in [21], we train these agents with PPO with Generalized Advantage Estimation [23]. As such, we also decode a value estimate for the current state as $v_t = W_v h_t$ where W_v is a learned linear layer. See [21] for full details.

Visual Observation. We consider two settings based on the agent's observations – either color images (RGB) or depth (D). We denote observations as I_t in either case. Prior work [21] has shown depth to be an important signal for point-goal navigation – with RGB-only models narrowly outperforming blind agents and falling well below the performance of their depth-enabled counterparts [21]. Fig. 3 demonstrates this trend in terms of SPL [21] – a metric that accounts for both success and path efficiency (see Sec. 4). In Sec. 3.2, we examine depth prediction as an auxiliary task – significantly improving RGB agent performance.

[1] In compact layer notation, the SimpleCNN network architecture is {Conv 8×8, ReLU, Conv 4×4, ReLU, Conv 3×3, ReLU, Linear, ReLU}.

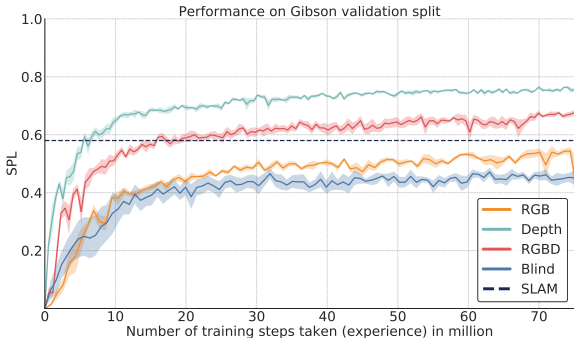

Figure 3. RGB agents perform significantly worse than their depth-equipped counterparts; performing similarly to blind agents. This plot is reproduced from [21] for convenient reference.

3.2. Quantized Depth Prediction

Depth's success as an input modality for point-goal navigation is perhaps unsurprising – depth explicitly represents key aspects of scene geometry for navigation including free-space and the position of potential obstacles. However, depth agents achieving nearly twice the performance of RGB-only agents seems extreme. After all, depth can be inferred from motion of a monocular observer – for example, you or I can likely move about the world with one eye closed without bumping into any of the walls. We then hypothesize that learning point-goal navigation with reinforcement learning is not adequately capturing this structure from RGB observations. We consider a quantized depth prediction task similar to [18]. The agent must predict a representation of the current depth image given the current observation and navigation history. For RGB agents, this amounts to a monocular depth estimation problem. For depth agents, this is a type of autoencoding task.

Depth Quantization. Rather than supervise the model to regress dense, continuous depth values, we consider a low-resolution, quantized representation that focuses on the general structure of the scene. We find this is easier to learn while still providing significant benefit. Given the depth observation at the current agent position, we crop to the central 50% of the frame (128×128 in our setting) and then apply an average pool operation to further reduce to 4×4. Each depth value is then quantized into one of eight bins (classes) of roughly equal size (slightly emphasising near depths over further). We denote this quantized depth observation at time t as $D_t \in [0, \ldots, 7]^{4\times4}$ and treat depth prediction as an 8-way classification task at each spatial location.

Depth Auxiliary Network. We would like representations of the agent state – the current observation and memory – to encode the information about scene geometry. To induce this effect, we introduce an auxiliary network. We denote the final spatial feature representation of $\phi(I_t)$ as $O_t \in \mathbb{R}^{28\times28\times32}$. Given O_t and the policy hidden state $h_t \in \mathbb{R}^{512}$, we add an auxiliary depth prediction task network f_d such that:

$$\hat{D}_t = f_d([O_t + h_t]) \tag{3}$$

where $+$ denotes a spatially-tiled concatenation along the channel dimension – i.e. $[O_t + h_t] \in \mathbb{R}^{28\times28\times544}$. We implement f_d as a series of convolutional layers followed by a fully-connected layer to predict depth-class distributions for each spatial cell. Note that this formulation considers the entire spatial tensor when predicting each depth pixel. See the supplementary - Sec. 2 for more details. The auxiliary loss is simply a cross-entropy between predicted and actual depth-pixel classes, we denote this loss as $\mathcal{L}_D$.

3.3. Inverse Dynamics

In the previous task, we argued for depth's usefulness in terms of reasoning about obstacles and free-space. However, these concepts are primarily useful in estimating how an agent's actions affect its state transitions – e.g. if an agent goes forward when there is an obstacle, its position may not change. Rather than supervising depth, we can instead guide the agent representation in terms of these environment dynamics directly. Following prior work on self-supervised representation learning from video [3, 15], we consider an inverse dynamics formulation – predicting the action given an observed state change in the environment.

Inverse Dynamics Auxiliary Network. Specifically, we consider predicting the action a_t taken by the agent at a given timestep t from the pre-action observation I_t and the pre-and-post action relative goals g_t and g_{t+1} respectively. Note that we do not include h_t as input because state sequences have strong correlation over time that would undermine the need to learn meaningful visual representations. We introduce an inverse dynamics network f_{Inv} such that:

$$\hat{a}_t = f_{\text{Inv}}(\phi(I_t), g_t, g_{t+1}) \tag{4}$$

We implement f_{Inv} as a simple feed-forward network that predicts a distribution over the actions taken given the concatenation of the inputs. For training, we take (a_t, I_t, g_t, g_{t+1}) tuples directly from agent policy rollouts. The auxiliary loss $\mathcal{L}_{\text{Inv}}$ is just the cross-entropy loss of the predicted and actual actions.

3.4. Remaining Path Length Prediction

Beyond obstacle avoidance and short-term path planning, the point-goal navigation task requires reasoning about longer range navigability. For example, avoiding dead ends (e.g. bedrooms) in favor of better-connected areas (e.g. hallways) when the Euclidean distance to the goal is large. Reasoning about these issues requires a sense of room extents or path complexity at different distances in environments. To encourage the observation representation to encode this, we consider predicting the remaining *geodesic* path distance to

	Row		SPL Percentage (↑)					Success Percentage (↑)				
			@5M	@10M	@15M	@20M	@25M	@5M	@10M	@15M	@20M	@25M
RGB	1	Baseline Agent	17.8 ±0.7	41.8 ±0.7	44.2 ±1.2	45.1 ±1.0	44.3 ±1.8	27.0 ±1.2	65.1 ±1.1	69.9 ±1.5	74.6 ±1.1	78.0 ±1.9
	2	+ Depth Pred.	35.6 ±0.6	55.4 ±0.0	63.8 ±1.0	70.6 ±0.7	74.7 ±1.7	56.5 ±1.5	74.8 ±0.3	81.3 ±0.5	84.6 ±0.7	**86.8** ±1.6
	3	+ Inv. Dyn.	37.0 ±0.9	47.6 ±1.5	55.3 ±0.8	59.1 ±2.8	65.5 ±0.8	53.0 ±2.1	66.7 ±1.4	74.9 ±1.0	78.0 ±2.1	81.0 ±1.0
	4	+ Remain. Path.	38.5 ±2.1	53.2 ±2.3	52.7 ±1.4	59.7 ±1.0	66.9 ±0.6	55.5 ±2.4	76.4 ±2.3	78.9 ±2.3	83.9 ±0.8	85.6 ±1.3
	5	+ All	**40.8** ±0.7	**60.5** ±2.5	**68.8** ±1.6	**73.4** ±2.4	**75.1** ±3.0	**61.1** ±0.8	**77.04** ±2.4	**81.7** ±1.0	**85.5** ±2.5	85.6 ±3.5
Depth	6	Baseline Agent	53.7 ±2.0	68.4 ±1.3	72.1 ±0.9	70.9 ±0.6	70.5 ±0.9	68.7 ±1.8	82.1 ±1.2	89.1 ±0.3	90.9 ±1.3	92.1 ±1.0
	7	+ Depth Pred.	54.6 ±1.7	63.6 ±1.6	72.8 ±0.4	78.6 ±0.8	78.8 ±1.0	71.6 ±1.4	80.7 ±1.7	87.0 ±0.7	91.0 ±0.3	90.4 ±0.7
	8	+ Inv. Dyn.	55.2 ±1.7	71.2 ±2.4	77.0 ±0.9	78.2 ±0.8	79.4 ±1.9	73.0 ±1.9	84.3 ±1.9	89.0 ±0.7	91.1 ±1.2	92.4 ±1.4
	9	+ Remain. Path.	**60.0** ±1.1	**72.0** ±0.6	**77.2** ±1.2	**80.8** ±1.1	**83.0** ±2.2	**75.0** ±1.7	**86.9** ±1.1	**91.0** ±0.2	**92.2** ±1.5	**92.6** ±1.0
	10	+ All	58.37 ±1.4	71.46 ±0.7	75.65 ±1.6	80.48 ±1.9	81.9 ±1.0	71.54 ±1.4	85.05 ±1.3	88.08 ±1.6	91.23 ±2.1	91.54 ±1.1

Table 1. Navigation metrics (SPL and Success percentages) on Gibson validation set as a function of training time (5, 10, 15, 20 and 25 million steps). We report means over three runs and 95% confidence intervals. We find our auxiliary task-equipped agents consistently outperform their corresponding baseline agents by a significant margin. Our auxiliary tasks have a stronger affect on the RGB agents.

the goal. That is, the actual distance required to reach the goal from a point by an oracle, shortest-path navigator.

Remaining Path Length Auxiliary Network. At a given time t, we compute the geodesic distance to goal d_{geo} and set an auxiliary task to predict this value based on the current observation I_t and goal g_t. As before, we leave out historical information. Agent history information could enable 'counting' strategies that predict the remaining geodesic distance based on how long the agent has already been navigating rather than using the visual observation. We introduce an auxiliary network f_{Geo} such that:

$$\hat{d}_{\text{geo}} = f_{\text{Geo}}(\phi(I_t), g_t) \tag{5}$$

We implement f_{Geo} as a simple feed-forward neural network and define the auxiliary loss $\mathcal{L}_{\text{Geo}}$ as an L2 regression loss on this geodesic distance prediction.

4. Experimental Setting

PointGoal Navigation. We consider the PointGoal Navigation task in the Habitat Simulator as defined in [21] for the Habitat Challenge 2019. An agent is spawned in a never-before-seen environment and must navigate to a location given in relative coordinates. The agent must do so by relying on a GPS+Compass sensor and visual observations (RGB or Depth). We follow the training and validation splits on the Gibson dataset [27] in accordance with [21].

Training. We train using a combination of supervised learning for auxiliary tasks and reinforcement learning for agent actions. At each step, the environment provides dense reward r_t for an action based on the change in geodesic distance to goal – i.e.

$$r_t = D(P_t, g) - D(P_{t+1}, g) - \lambda \tag{6}$$

where λ is a constant slack penalty to encourage efficient paths and $D(\cdot, \cdot)$ is geodesic distance. An episode ends when the agent takes the `stop` action and an additional +10 reward is accrued if the agent is within 0.2 meters of the goal. We apply Proximal Policy Optimization (PPO)[24] with Generalized Advantage Estimation (GAE)[23]. Auxiliary task loss is also computed at each step during training.

Metrics. We report standard metrics for navigation [4] –

- **Success.** Percentage of episodes in which the agent calls `stop` within 0.2 meters of the goal location.
- **Success weighted by inverse normalized Path Length (SPL).** SPL considers both success and path efficiency – weighting a binary success indicator by the normalized ratio of the agent's path length and the shortest path.

Both are reported as percentages. We report means and 95% confidence intervals over multiple trials.

Implementation Details. For PPO, each batch of observations is generated using 8 rollout workers with rollout length of 128 steps. The PPO trains for 4 epochs on each batch with mini-batch size of 4. We use Adam optimizer with a learning rate of 2.5×10^{-4}. We train agents to 25 million steps of experience on two Tesla V100 GPUs – this takes approximately 21 hours for RGB agents and 15 for depth. To make fair comparisons, we use the agents and training code from the public codebase from [21] – adding our auxiliary losses to their existing model architectures. We set auxiliary-task loss weights to 1.0, 0.2 and 1.0 for quantized depth prediction, inverse dynamics and remaining path length prediction tasks respectively.

5. Results

We apply each of our auxiliary losses and their combination to the baseline agent and train three random initializations – a total of 30 agents (3×{`RGB`, `Depth`}×{`Baseline`,

Row	Depth Pred.	Inv. Dyn.	Path Pred.	SPL Percentage (↑) @5M	@10M	@15M	@20M	@25M	Success Percentage (↑) @5M	@10M	@15M	@20M	@25M
1		✓	✓	**42.9**	53.3	58.4	62.7	63.6	**63.0**	76.5	80.6	84.3	80.3
2	✓	✓		42.1	59.0	67.2	71.8	72.9	61.5	73.9	79.7	83.4	83.3
3	✓		✓	37.0	51.0	58.4	64.4	68.9	55.6	69.7	74.9	80.8	81.5
4	✓	✓	✓	40.8	**60.5**	**68.8**	**73.4**	**75.1**	61.1	**77.04**	**81.7**	**85.5**	**85.6**

Table 2. Navigation metrics (SPL and Success percentages) on Gibson validation set for the RGB agent as a function of training time (5, 10, 15, 20 and 25 million). We present all combinations of auxiliary tasks. Underlined entries denote results for combinations that outperform the independent performance of each individual constituent loss. We find the combination of task often lead to further improvements.

`+DepthPred`, `+InvDyn`, `+RemainPath`, `+All`}). We report results in Tab. 1 for validation performance at fixed points during training; 5, 10, 15, 20 and 25 million steps. For compactness, we will denote performance at these points as Metric@XXM – e.g. SPL@20M for SPL at 20 million steps. We also present these results as curves over the full training run in the supplementary - Sec.3.

Auxiliary tasks improve sample efficiency. We find our auxiliary task-equipped agents outperform the baseline agent across training iterations. This is especially true for RGB agents which see large gains early in training. After only 5M steps, our RGB agents achieve ∼37 SPL on average – over twice that of the baseline at 17.8 SPL (`rows 2-4 vs. 1` in Tab. 1). By 10 million steps, these agents achieve 47.6 to 55.4 SPL – values not reached by the baseline even after 25 million steps in our experiments (`row 1`). By 15 million steps, our best RGB agent (`row 2`) achieves 63.8 SPL, outperforming the ∼57 SPL at 75 million reported in [21] for the baseline agent (see Fig. 3). Likewise, our best depth agent achieves 77.2 SPL@15M (`row 9`) compared to the ∼75.4 SPL@75M reported in [21]. This represents a 5x (75M/15M) improvement in sample efficiency over the baseline agents.

Auxiliary tasks affect Depth and RGB agents differently. We find significant differences in how RGB and Depth agents respond to auxiliary tasks. Depth prediction is extremely effective for the RGB agent (`row 2`), outperforming other auxiliary tasks by a large margin (∼9 SPL). This follows our hypothesis that depth provides important geometric information for navigation that RGB agents have trouble extracting from pixels. In contrast, the depth agent benefits most from the remaining path length prediction task (`row 9`). For depth agents, we find depth prediction and inverse dynamics to be roughly equivalent (`row 7` and `8`). Interestingly, the augmented agents achieve similar success percentages as the baseline, with improvements in navigation efficiency being a strong driver for improvements in the SPL metric. It is worth emphasizing that the depth agent does still benefit from the depth prediction task (`row 6`), improving by 8 SPL@25M over the baseline despite the fact that depth is already provided as input to the agent.

RGB with depth prediction outperforms RGB-D. The RGB agent benefits significantly from the depth auxiliary loss; however, an alternative means of providing depth information is simply to append it as an input. While we do not experiment with such an RGB-D agent here, we compare our RGB+Depth Pred. agent with the RGB-D agent reported in [21] (see Fig. 3). Interestingly, we find that our RGB agent trained with the depth prediction auxiliary task outperforms the RGB-D agent reported in [21] despite this agent already having depth as input (74.6 SPL@25M for our agent and 70 SPL@75M for [21]). This result suggests that the role of auxiliary tasks is not just to add additional information not already provided to the agent, but also to shape representation learning given the existing inputs.

Combining multiple auxiliary tasks. Rows 5 and 10 in Tab. 1 show models trained with all three auxiliary tasks active. We combine task losses by weighting them to empirically normalize their scale. We find these models outperform our best individual models with significantly steeper learning curves for RGB input whereas their performance nearly matches (within variances) our best agents for depth input. This is another instance where auxiliary tasks seem to have different effect for RGB and depth agents. To examine the relationship between auxiliary tasks, we present all possible combinations in Tab. 2 for the RGB agent. We underline any result for a combination that outperforms each individual constituent loss. We find that Inv. Dyn. + Path Pred. (`row 1`) and Depth Pred. + Inv. Dyn. (`row 2`) are complementary to much extent while Depth Pred. + Path Pred. (`row 3`) are not. This suggests depth prediction and remaining path prediction compete when acting on common representation. Despite this, combining all three yields further gains across most time points. We believe the performance of all these combinations can be improved in later stages (>20M steps) by dynamically adjusting their (now static) loss coefficients.

Comparison to state-of-the-art. So far we compared performance of task-equipped agents with the Habitat [21] agents that form our baseline architectures. However, state-of-the-art in point-goal navigation [26] achieves nearly perfect SPL (≥99) in Gibson validation – significantly higher

	Row		SPL Percentage (↑) @5M	@10M	@15M	@20M	@25M	Success Percentage (↑) @5M	@10M	@15M	@20M	@25M
RGB	1	Baseline Agent	17.8 ±0.7	41.8 ±0.7	44.2 ±1.2	45.1 ±1.0	44.3 ±1.8	27.0 ±1.2	65.1 ±1.1	69.9 ±1.5	74.6 ±1.1	78.0 ±1.9
	2	+ Dense Depth Pred. [18]	1.5 ±0.4	13.7 ±1.0	51.3 ±0.8	61.0 ±3.7	64.8 ±2.8	2.14 ±0.5	17.8 ±1.4	68.6 ±0.2	77.4 ±3.6	79.4 ±2.3
	3	+ Reward Regression [14]	27.2 ±0.3	37.3 ±1.0	42.0 ±1.6	41.6 ±1.1	51.1 ±1.7	41.1 ±0.6	57.4 ±1.3	66.6 ±1.9	70.8 ±0.9	80.3 ±2.0
	4	Ours (Best from Tab. 1)	**35.6** ±0.6	**55.4** ±0.0	**63.8** ±1.0	**70.6** ±0.7	**74.7** ±1.7	**56.5** ±1.5	**74.8** ±0.3	**81.3** ±0.5	**84.6** ±0.7	**86.8** ±1.6

Table 3. Navigation metrics (SPL and Success percentages) on Gibson validation set as a function of training time (5, 10, 15, 20 and 25 million steps). We report means over three runs and 95% confidence intervals. We compare the baseline agent with the agents equipped with existing auxiliary tasks for navigation [18, 14]. Agents with [18] are slow to learn whereas those with reward regression only reach 51.1 SPL@25M. One can note that our best task-equipped RGB agent (`row 4`) outperforms all the agents.

than our models and those of [21]. It is however important to consider how this impressive result is achieved. In [26], agents architectures have significantly greater complexity [2] and are trained on an extended dataset (Gibson and Matterport3D [7]) for 2.5 billion steps of experience in a distributed RL framework. This represents an increase in model capacity, training set size, and a training time (100x). Our focus in this work is on training efficiency and our results question whether these extreme training protocols are necessary. Taking the first 75 million steps of the DDPPO agent's training regime, we find our RGB agents achieve the same SPL in only 17 million steps (a >4x improvement) (see supplementary - Sec. 4). An interesting question is at what point our agents would reach optimal performance compared to the 2.5 billion steps required in [26]. Unfortunately, we cannot scale to these extremes. Our existing experiments required 40 days of GPU time – scaling by over $100\times$ is simply infeasible given our infrastructure.

Comparison with prior auxiliary tasks. We examine how existing work on auxiliary tasks in visually-simple settings transfers to photo-realistic environments. Specifically, we examine two auxiliary tasks from prior work, depth prediction [18] and reward regression [14]. We limit our discussion to RGB agents to be consistent with these works.

Depth prediction. We contrast the performance of the depth prediction architecture proposed in [18] with our own. Developed in simple game-like environments, this task requires agents to predict dense pixel-wise depth from non-spatial features (i.e. $\phi(I_t)$) – a difficult task in complex, realistic environments. We find the architecture for depth prediction employed in [18] causes the learning to be delayed and achieves lower performance when compared with our results (Tab. 3 `row 2 vs.` Tab. 1 `row 2`). By 5 million steps, agents augmented with [18] achieve 1.5 SPL on average - which is nearly $1/17$ of the SPL achieved by the baseline agent (Tab. 3 `row 1` vs `row 2`). Over the period of training, we see an improvement in performance of task-equipped agents and it surpasses the baseline at 15 million steps. By 25 million steps, we see their depth-augmented agents outperform the baseline but is still below our results by $\sim$10 SPL. Interestingly, this approach fails to improve the success metric significantly whereas our approach results in an improvement of $\sim$8% success.

[2]Using a SE-ResNeXt50 [28] network for encoding observations

Row		Average collisions per episode (↓) @5M	@10M	@15M	@20M	@25M
1	Baseline Agent	**31.63**	54.97	54.90	62.40	67.01
2	+ Inv. Dyn.	37.03	39.66	42.69	49.45	49.54
3	+ Depth Pred.	44.17	**37.97**	**37.25**	**34.57**	**34.89**
4	+ Remain. Path.	46.75	44.45	56.19	42.58	43.93

Table 4. Average collisions per episode for RGB agents in 250 test episodes. We find auxiliary-augmented agents collide significantly less, with the depth prediction auxiliary task inducing the fewest.

Reward regression. We apply reward regression task proposed in [14] to our problem – predicting the immediate reward using the encoded observation $\phi(I_t)$ and relative goal vector g_t. We find reward regression to be less effective than our approaches in improving either SPL or success metric for the navigator. Initially by 5 million steps, the reward-regression agents show some promise, leading over the baseline agents by $\sim$ 10 SPL (compare Tab. 3 `rows 1` and `3`). But this lead does not extend further into training. By 25 million steps, agents equipped with reward regression achieve 51.1 SPL which is lower than that achieved by any of our individual auxiliary task-equipped agents (see supplementary Sec. 5 for learning curves).

6. Analysis

6.1. Effect on Collision Rate

One hypothesis about the quantized depth prediction loss is that it serves as a proxy for free space – imbuing the agent with a strong sense of whether a forward action might result in a collision. In Tab. 4, we compare average colli-

	SPL (↑)				
	@5M	@10M	@15M	@20M	@25M
Baseline	17.8	41.8	44.2	45.1	44.3
Depth Pred. (Pre.)	23.9	43.9	47.2	46.8	46.2
Depth Pred. (Aux.)	**35.6**	**55.4**	**63.8**	**70.6**	**74.7**
Inv. Dyn. (Pre.)	25.0	40.9	45.2	44.7	46.2
Inv. Dyn. (Aux.)	**37.0**	**47.6**	**55.3**	**59.1**	**65.5**
Remain. Path (Pre.)	27.9	36.3	44.6	48.6	47.1
Remain. Path (Aux.)	**38.5**	**53.2**	**52.7**	**59.7**	**66.9**

Table 5. We compare the agents equipped with auxiliary tasks (Aux.) (Tab. 1) with the agents using a pretrained (Pre.) encoder trained independently for these tasks. The task-equipped agents consistently outperform the agents with pretrained encoders.

sion counts for different auxiliary-augmented RGB agents across 250 test episodes. The baseline agent navigates poorly early on and terminates episodes prematurely, resulting in initially low collision counts that rises significantly as the agent improves at reaching the goal. In contrast, we find all auxiliary tasks keep the collision rate lower throughout training, with depth prediction resulting in the largest reduction. Even when comparing at similar success rates (@25M for Baseline vs. @10M for Depth Pred.), the Depth Pred. augmented agent reduces collision by nearly half.

We note that collisions are not strictly negative. As shown in [16], agents can take advantage of 'sliding' against walls to improve SPL. However, the augmented agents achieve lower collision rates *and* more efficient paths (higher SPL) – implying inefficient collisions are avoided.

6.2. Decoding Position from Agent Representations

While agent's do take the dynamically-updating relative goal coordinate as input, they do not explicitly receive their own position. In this experiment, we examine whether agent representations monitor changing goals to encode agent position. Taking agents trained to 25 million steps and then frozen, we learn a simple fully-connected decoder that predicts agent position in three-dimensional XYZ coordinates from the policy hidden state h_t. After training on 10240 positions from the training set, we measure the proportion of predictions that fall within 4m, 2m, and 1m of the true position for 2400 instances on validation. Due to space constraints, we present the full table in the supplementary - Sec. 6. For accuracy within 1 meter, we find the baseline achieves 48.4%, Depth Pred. 62.5%, Inv. Dyn. 50.6%, and Remain. Path. 74.5%. These results suggest that the remaining path prediction task induces significantly more information about agent position.

Figure 4. Despite of both views in above example having similar Euclidean (L2) distance to the goal, they have different true geodesic distances(Geo). The prediction for geodesic distance (Pred) differs based on the "openness" of the current frame.

6.3. Remaining Path Length Identifies Openness

Given the current observation and the goal in relative coordinates, the remaining path length auxiliary task requires agents to predict how much further an agent must move to reach the goal. The Euclidean distance to the goal coordinate provides a lower bound on the remaining distance – in an open area, this may in fact be the distance to goal. However, if the goal is quite far or the visual observation is of a cluttered space or 'dead end', the prediction should be significantly higher. Fig. 4 shows a pair of qualitative examples for the remaining path length task. In both, the Euclidean distance to goal is around 6.5 meters. However, the "dead end" example on left predicts a significantly higher remaining path length (10.2m) whereas the open area on the right keeps its estimate low (7.6m). Based on random qualitative samples, this seems to be a fairly consistent trend.

6.4. Pre-training vs. Auxiliary Task Learning

As most of our auxiliary tasks focus on enriching the visual encoder, we compare simply pretraining encoders on our auxiliary tasks rather than using them during RL training. Specifically, we collect agent trajectories from our best performing agent in the training environments to create a static dataset of observations and auxiliary task targets. As the depth prediction task also requires a hidden state, we record this as well for the pretrained agent. The models are trained until they reach similar loss levels as those observed during auxiliary learning and then they are used as warm-starts for full RL training. From Tab. 5, we find pretraining improves marginally over the baseline but lags significantly behind the auxiliary training paradigm.

7. Conclusion

In this work, we examine the role of auxiliary training tasks for point-goal navigation in realistic environment. We demonstrate significant improvements (5x) in sample efficiency over baseline models from prior work.

References

[1] Gibson challenge @ CVPR 2020. http://svl.stanford.edu/gibson2/challenge.html.

[2] Habitat Challenge 2019 @ Habitat Embodied Agents Workshop. CVPR 2019. https://aihabitat.org/challenge/2019/.

[3] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. *ICCV*, 2017.

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

[5] 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.

[6] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[7] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pages 667–676. IEEE, 2017.

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

[9] Devendra Singh Chaplot, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Modular visual navigation using active neural mapping.

[10] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*, 2014.

[11] Karol Gregor, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aaron van den Oord. Shaping belief states with generative environment models for rl. *NIPS*, 2019.

[12] Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo A. Pires, and Rémi Munos. Neural predictive belief representations. *arXiv preprint arXiv:1811.06407*, 2018.

[13] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2017.

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

[15] Dinesh Jayaraman and Kristen Grauman. Learning image representations tied to ego-motion. *ICCV*, 2015.

[16] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Are we making real progress in simulated environments? measuring the sim2real gap in embodied visual navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[17] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. 2020.

[18] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[19] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

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

[21] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. *ICCV*, 2019.

[22] Alexander Sax, Bradley Emi, Amir R. Zamir, Leonidas J. Guibas, Silvio Savarese, and Jitendra Malik. Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies. 2018.

[23] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *ICLR*, 2016.

[24] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[25] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6659–6668, 2019.

[26] 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. 2020.

[27] Fei Xia, Amir Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. *CVPR*, 2018.

[28] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.