Auxiliary Tasks Speed Up Learning PointGoal Navigation

Joel Ye[1*] **Dhruv Batra**[2,1] **Erik Wijmans**[1†] **Abhishek Das**[1→2†]

[1]Georgia Institute of Technology [2] Facebook AI Research

Abstract:
PointGoal Navigation is an embodied task that requires agents to navigate to a specified point in an unseen environment. Wijmans et al. [1] showed that this task is solvable in simulation but their method is computationally prohibitive – requiring 2.5 billion frames of experience and 180 GPU-days. We develop a method to significantly improve sample efficiency in learning POINTNAV using self-supervised auxiliary tasks (*e.g.* predicting the action taken between two egocentric observations, predicting the distance between two observations from a trajectory, *etc.*). We find that naively combining multiple auxiliary tasks improves sample efficiency, but only provides marginal gains beyond a point. To overcome this, we use attention to combine representations from individual auxiliary tasks. Our best agent is 5.5x faster to match the performance of the previous state-of-the-art, DD-PPO [1], at 40M frames, and improves on DD-PPO's performance at 40M frames by 0.16 SPL. Our code is publicly available at github.com/joel99/habitat-pointnav-aux.

Keywords: Vision for Robotics, PointGoal Navigation

1 Introduction

Consider a robot tasked with navigating from the bedroom to the kitchen solely from first-person egocentric vision. To do so, it must be able to reason about 1) notions of free space (that doors can be walked through, but not walls), 2) keep regions already visited in memory (so as not to run around in circles), 3) common sense of how houses and objects are typically laid out (that kitchens typically are not inside bedrooms), *etc*. To learn these skills, the agent needs a good environment representation.

The current state-of-the-art method for training a class of such robots (embodied agents) in simulation is Decentralized Distributed PPO (DD-PPO) [1]. Specifically, Wijmans et al. [1] train an agent to autonomously navigate to a point-goal in an unseen environment nearly perfectly (99.9% success rate). However, this comes at a prohibitive computational cost – requiring 2.5 billion frames of experience; 80+ years of experience accrued over *half-a-year* of GPU time, 64 GPUs for 3 days!

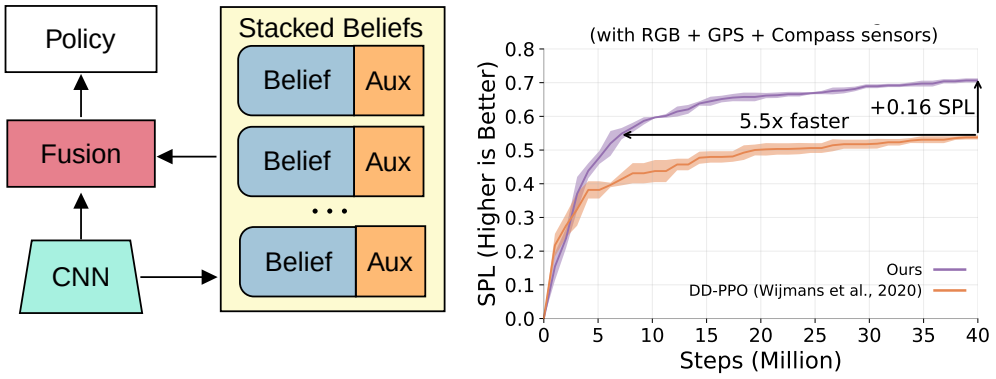
While [1] serves as an excellent 'existence proof' of the learnability of POINTNAV, we believe it should not take 2.5 billion frames of experience and nearly 6 months of GPU time to learn to navigate from point A to B. An existence proof is often the first crack in the wall, enabling subsequent improvements – a non-constructive proof replaced by constructive proof, an improved algorithm, shaving off factors in bounds – until the problem is well-understood. *That* is our goal – to improve sample and time efficiency in learning POINTNAV using self-supervised auxiliary tasks.

In the process of improving sample efficiency, we address several important questions over prior work in auxiliary self-supervised learning, from both the supervised [2–12] and reinforcement learning paradigms [13–21]. First, auxiliary tasks are typically benchmarked in visually simple simulated environments (*e.g.* DeepMind Lab [22], Atari). Do these improvements transfer to realistic environments? Second, it is unclear how these auxiliary objectives interact with each other – can multiple such tasks be combined? Do they lead to positive transfer when combined, or is there interference? Finally, what is the 'right' way to combine them – can they be naively combined by summing the losses, or does combining them necessitate sophisticated weighting mechanisms?

*Correspondence to joel.ye@gatech.edu

†EW and AD contributed equally.

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.



(a) (b)

Figure 1. (a) We use learning signals from multiple self-supervised auxiliary tasks on a recurrent architecture (detailed in Section 4) to speed up learning PointNav. (b) Our best agent achieves the same performance as the DD-PPO [1] baseline 5.5× faster and improves on the baseline's performance at 40M frames by 0.16 SPL.

Concretely, our contributions are the following:

- We *significantly* improve sample- and time-efficiency on PointGoal Navigation over DD-PPO [1].
- We study three self-supervised auxiliary tasks – action-conditional contrastive predictive coding (CPC|A) [14], inverse dynamics [15], and temporal distance estimation – and show that each improves sample efficiency over the baseline agent from [1]. With a fixed computation budget (of 40M steps of experience), our best single auxiliary task CPC|A-4 improves performance from 0.55 to 0.66 SPL (+22%). With a fixed performance level (0.55 SPL), CPC|A-16 achieves a 2.1 reduction in no. of steps required (from 40M to 19M).
- Next, we show that the naive combination (*i.e.* direct addition of losses) of multiple auxiliary tasks can further improve sample efficiency over single tasks. The best combination achieves .70 SPL (+27%) by 40M steps and achieves 0.55 SPL in 12M steps, a 3.3 speedup.
- Finally, we observe that naive summation of losses has diminishing returns on sample efficiency as we further increase the number of auxiliary tasks. We propose a novel attention mechanism to fuse state representations that overcomes these negative effects. Putting it all together, our final model obtains 0.55 SPL in 7M steps of experience, a 5.5 speedup over the baseline from [1].

2 Related Work

Our work relates to prior work in auxiliary objectives for learning representations in reinforcement learning, methods for combining multiple such objectives, and other approaches to PointNav.

Auxiliary Tasks in Reinforcement Learning. Auxiliary tasks provide additional complementary objectives to improve sample efficiency and/or performance on the primary task. Supervised auxiliary tasks expose privileged information to the agent (such as depth [19, 23, 24]). Self-supervised auxiliary tasks, such as next-step visual feature prediction [15], predictive modeling [14, 16], or spatio-temporal mutual information maximization [17, 20] derive supervision from the agent's own experience. In contrast to prior work, which focus on simpler and non-photorealistic environments [14, 16–19], we focus on visually complex, photorealistic environments from the Gibson 3D scans [25].

Closely related to our work is that of Gordon et al. [21] who show that auxiliary tasks can be leveraged to improve transfer to new tasks and new simulation environments (*i.e.* synthetic to photorealistic). In contrast, we focus on improving sample efficiency when learning a task *from scratch*, proposing that the most performant representations should arise by virtue of end-to-end learning.

Combining Multiple Auxiliary Tasks. Combining multiple auxiliary tasks raises the challenges that 1) they have varying affinities with a given primary task, and 2) these affinities can change during the training process as the agent improves. When studying knowledge transfer between multiple tasks, most prior work comes from multi-task learning. There, task affinity has often been taken as a constant, where influence is normalized by task uncertainty [26], or as a prior [27]. In contrast, we propose a formulation that *learns the appropriate influence* of each auxiliary task during training.

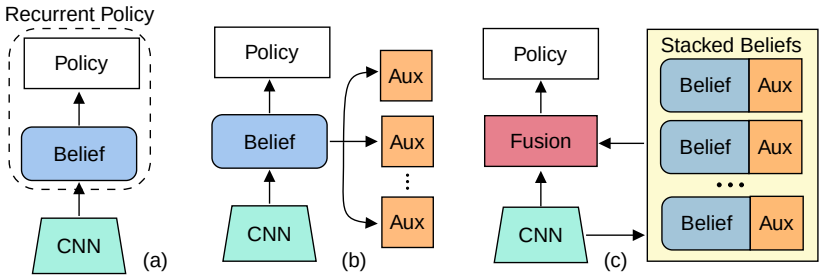


Figure 2. (a) Baseline architecture from DD-PPO [1] with the recurrent policy conceptually separated into a recurrent 'belief' module and a feed-forward policy head. (b) Single-belief, where multiple auxiliary tasks utilize the same shared belief module output. (c) Fused-beliefs, where each auxiliary task is paired with its own separate belief module, and the outputs of the belief modules are fused and fed as input to the policy head.

Lin et al. [18] uses gradient similarity to adaptively weight auxiliary losses. However, this approach is limited to training time, whereas our approach also enables auxiliary task weighting during *evaluation*. *e.g.*, long-horizon predictive modeling ('what room is my goal in?') may be useful overall, but inappropriate when turning a tight corner. While such an ability could be implicit in a loss-driven approach, an explicit weight distribution sidesteps the need for mathematical approximations as in [18] and allows for inference-time visualization of task influence.

PointGoal Navigation. PointGoal Navigation (detailed in Section 3) has progressed remarkably, with several entries to the 2019 Habitat PointNav Challenge exceeding 0.70 SPL in only 10M observations. One leading method is Active Neural Mapping [28], which uses neural environment maps and hierarchical planning modules. Sax et al. [29] transferred visual features from Taskonomy [30], showing no single representation was ideal for multiple embodied tasks and concluding diverse representation sets are best for unknown downstream tasks. Shen et al. [31] presented a vision-conditioned fusion of visual representations that outperformed naive concatenation. Our work operates in the same regime, using dot-product attention [32] to guide fusion. As transferred visual representations have been promising, we briefly compare with [29] in 5.3. However, our work seeks to improve "from-scratch" training of POINTNAV agents, significantly simplifying the training pipeline. Thus our contributions are orthogonal to these approaches *e.g.*, our approach can improve the local planner in [28] (see Section A.7). As for [31], we note their fusion technique is inherently sample inefficient. Each of their policies must be trained individually before fusion can be used, which means samples required scale linearly with the number of tasks fused. Further, their approach uses many large ResNet-50 encoders, while our approach has a footprint smaller by over 100x FLOPs. Our main comparison is with [1], where from-scratch representations effectively solved POINTNAV.

3 Task, Simulation, Agent

PointGoal Navigation. In POINTNAV [33], an agent is initialized in an *unseen* environment and tasked with navigating to a goal location without a map. The goal location is specified with coordinates relative to initial location (*e.g.* 'go to (5, 20)' where units describe distance relative to start in meters). The agent is equipped with an `RGB` camera (providing egocentric `RGB` observations) and a GPS+Compass sensor (providing position and orientation relative to the start location). The agent has access to 4 standard actions: $\{$`move forward` (0.25m), `turn left` $(10°)$, `turn right` $(10°)$, `stop`$\}$.

Metrics. We evaluate the agent on two metrics – 1) Success: whether or not the agent correctly predicted `stop` within 0.2m of the goal, and 2) Success weighted by inverse Path Length (SPL) [33]: which weights success by how efficiently the agent navigated to the goal relative to the shortest path.

Simulation. We simulate our agent on the AI Habitat platform [34], which has been shown to have good Sim2Real transfer [35]. Following the 2019 Habitat Challenge [34], we train and evaluate performance on the higher quality reconstructions from the Gibson dataset [25], *i.e.* 72 houses for training and 14 houses for validation. We test our approach's ability to generalize in Section A.6.

Agent. We divide our agent into three separate components: a convolutional neural network (CNN) encoder that produces an embedding of the visual observation (`RGB`), a 'belief' module that integrates multiple observations to produce an actionable summary representation, and a policy head that determines the agent's action given the belief module output. Note that prior work commonly refers

Figure 3. We study three auxiliary modules. a) Inverse dynamics: decoding action taken from successive visual embeddings $_t$ and $_{t+1}$ and the nal belief stateh_T. b) Temporal distance: decoding the timestep difference between two observation embeddings from nal belief state c) CPC|A: contrasting future observation embeddings ($_{t+1}$; :::; $_{t+k}$) at every timestep from other observation embeddings using a secondary GRU.

to this architecture as consisting of two parts – a CNN encoder and a recurrent policy. We divide this recurrent policy into a recurrent belief module and a feedforward policy head as shown in Fig. 2a. We denote this split as our auxiliary tasks operate on belief module output, as shown in Fig. 2b.

Our modi cations to the baseline architecture are intentionally minimal, isolating the impact of auxiliary tasks. We use ResNet18 [16] as modi ed for on-policy RL by Wijmans et al. [1] for our visual encoder. The belief module is a single layer GRU [17]. Its outputh_t passes to the policy head, a fully-connected layer, to yield a softmax distribution over the action space and a value estimate.

4 Self-Supervised Auxiliary Tasks from Experience

We introduce a set of auxiliary modules, one for each auxiliary task. The auxiliary modules operate on observations, outputs of the belief modules, and actions. Speci cally, the agent receives observation x_t, extracts its CNN representation$_t$, which is fed to the belief module to computeh_t, used to sample an actiona_t from the policy. Auxiliary tasks use a subset of$(x_1; _1; h_1; a_1) : : : (x_T; _T; h_T; a_T)$. Our choice of auxiliary tasks is motivated by providing the agent the ability to learn environment dynamics (which actions separate two observations?, how would the environment look if I moved forward and turned right?, etc.). We speci cally only consider self-supervised tasks (that do not need additional supervision) in Fig. 3, to keep the method generally applicable in simulation and the real world. This disallows tasks like depth prediction, which is known to make POINTNAV much easier to learn [19, 28]. We describe the tasks in detail in Section A.3. In experiments, we consider CPC|A with $k = 1; 2; 4; 8; 16$ (CPC|A-1, CPC|A-2, etc.). The entire family is denoted as CPC|A-{1-16}.

During training, we optimize the parameters of the visual encoder, belief module, and policy head, altogether $_m$, as well as the auxiliary module parameters,$_a$, to jointly minimize the auxiliary loss and the primary POINTNAV objective,L_{RL}, with $_{Aux}$ as a hyperparameter balancing the losses:

$$L_{total}( ) = L_{RL}(_m) + _{Aux}L_{Aux}(_m; _a) \quad (1)$$

We set $_{Aux}$ such that the two losses have roughly equal magnitudes at initialization.

4.1 Leveraging Multiple Auxiliary Tasks

If individual auxiliary tasks help, a natural question to ask is whether their improvements are additive. A simple approach is to apply different tasks to the single belief module (see Fig. 2b), adding all the individual loss terms with the same loss scales as in the individual task setup. Forn_{Aux} such auxiliary tasks, we denote individual auxiliary task-related parameters as$^1_a; :::; ^{n_{Aux}}_a$. The new loss is given by:

$$L(_m; ^1_a : : : ^{n_{Aux}}_a) = L_{RL}(_m) + \sum_{i=1}^{n_{Aux}} {}^i_{Aux}L_{Aux}(_m; ^i_a) \quad (2)$$

4.2 Attention over multiple auxiliary tasks

As we investigate in Section 5, using the method in Section 4.1 to combine auxiliary tasks does improve performance over single tasks, but gains quickly diminish. We hypothesize that this is

	AuC		Best	
1) Baseline	0.422	0.043	0.545	0.010
2) CPC|A-1	0.480	0.032	0.628	0.011
3) CPC|A-2	0.487	0.035	0.641	0.008
4) CPC|A-4	0.512	0.029	0.658	0.014
5) CPC|A-8	0.517	0.027	0.658	0.012
6) CPC|A-16	0.514	0.029	0.663	0.010
7) ID	0.458	0.036	0.588	0.011
8) TD	0.441	0.044	0.564	0.017

(a)

(b)

Figure 4. (a) Auxiliary tasks accelerate learning of POINTNAV. Long-range CPC|A tasks provide more gains; CPC|A-16 provides +0.12 SPL (+22%) by 40M frames. (b) Auxiliary tasks overtake the baseline by 5M frames.

because when multiple auxiliary tasks operate on the same belief module (Fig. 2b), their objectives compete. Additional objectives can thus hinder learning. To better leverage multiple auxiliary tasks, we propose a novel architecture with a shared CNN, separate recurrent belief modules for each auxiliary task, and a `fusion' module to combine belief module outputs into an input for the policy head, depicted in Fig. 2c. With separate belief modules, auxiliary tasks can optimize their respective beliefs for orthogonal objectives without interference, and the fusion module can extract policy-relevant representations. We experiment with several fusion methods (Table 1, details in Section A.4). We further apply an entropy penalty on the attention distribution (denoted $-\mathrm{H}(\alpha)$), to encourage the use of multiple modules (details in Section A.4).

5 Experiments and Results

We aim to answer the following questions:

1. Do auxiliary tasks help on POINTNAV in photorealistic environments?
2. Does combining auxiliary tasks help over individual tasks?
3. What is the best way to fuse representations from multiple auxiliary tasks?

Fusion Method	Description
Fixed [15]	Full weight xed to a single belief module.
Average	Equal weighting on all belief modules.
Softmax Gating [31]	Visually conditioned softmax weighting on belief modules $w = \mathrm{softmax}(f(\ ))$.
Scaled Dot-Product Attention (Attn)[32]	Weights computed as $\mathrm{score}(h_i, k) = \frac{h_i^T k}{\sqrt{n_{\text{Aux}}}}$ with beliefs h_i and $k = g_{\text{key}}(\ )$.

Table 1. All fusion methods are a weighted sum of beliefs. See Section A.4 for further details.

We refer to observations as `frames' of simulation throughout, as done in prior work. Each variant (Section 4) is trained for 40M frames as this corresponds to 1 GPU-week and with 4 random seeds. We report the highest average validation (averaged over three validation runs) SPL achieved by 40M frames. Note that validation is performed on held-out scenes and reward is not available during evaluation. Analyzing success shows similar trends as SPL, so we reserve those results for Section A.1. To analyze sample ef ciency, we compare the area under the learning curves (AuC) over 40M frames, with measurements every 1M frames. Models with higher AuC learn POINTNAV faster, an important skill for more challenging tasks, where slow learning might be intractable. When computing AuC, we rst normalize the x-axis (no. of frames) to $[0, 1.0]$, which normalizes AuC to the same range. In tables, we bold one variant over others with overlapping con dence intervals if it has better performance across validation episodes (paired t-test, $p < 0.05$).

All variants with single auxiliary tasks get higher SPL than the baseline, as shown in Table 4a. CPC|A excels at longer ranges, i.e. rows 4-6 indicate they provide at least $+0.11$ (+20%) SPL at 40M frames, and $+0.09$ (+21%) SPL AuC. The slight edge longer ranges have over CPC|A-{1,2} is consistent with intuitions in [16]. We subsequently use $k = 16$ as reference for the best single task.

All variants, including the baseline, have a ramp-up after which metrics begin to level (0.5 SPL), as seen in Fig. 4b. Intuitively, this in ection point represents when an easier subset of episodes

	AuC	Best
1) Baseline	0.422 ± 0.043	0.545 ± 0.010
2) CPC|A-16 (Best Single)	0.514 ± 0.029	0.663 ± 0.010
3) CPC|A-{1-16}: Add	0.523 ± 0.026	0.687 ± 0.010
4) CPC|A-{1-16}+ID+TD: Add	0.532 ± 0.028	0.696 ± 0.013

(a)

(b)

Figure 5. (a) CPC|A-{1-16} provides +0.02 SPL over CPC|A-16, and adding ID+TD yields only +0.01 SPL more. (b) Using multiple auxiliary tasks improves on a single task by 10M frames at cost to initial ramp-up.

	AuC	Best
1) Baseline	0.422 ± 0.043	0.545 ± 0.010
2) CPC|A-16 (Best Single)	0.514 ± 0.029	0.663 ± 0.010
3) CPC|A-{1-16}+ID+TD: Add	0.532 ± 0.028	0.696 ± 0.013
4) CPC|A-{1-16}+ID+TD: Average	0.539 ± 0.022	0.696 ± 0.008
5) CPC|A-{1-16}+ID+TD: Soft[31]	0.578 ± 0.020	0.696 ± 0.008
6) CPC|A-{1-16}+ID+TD: Attn	0.565 ± 0.027	0.698 ± 0.015
7) CPC|A-{1-16}+ID+TD: Attn+E	0.594 ± 0.019	0.707 ± 0.006

(a)

(b)

Figure 6. (a) Learned fusion (row 7) provides a +0.01 SPL over a single module (row 3). (b) Learning fusion of separate modules in CPC|A-{1-16}+ID+TD: Attn+E reduces ramp-up cost from CPC|A-{1-16}+ID+TD: Add.

is learned, and subsequent episodes provide diminishing returns. The dropoff is softer for variants with auxiliary tasks, indicating where better representations are bene ting primary task learning. Relatedly, we observe a subtle early performance loss for modi ed variants in Fig. 4b. They overtake the baseline as it levels off (40M frames). Initial learning involves the agent's rst successes, requiring understanding the existence of the goal point and its large reward. Intuitively, these navigational auxiliary tasks are distracting from the rare initial success rewards and it does not pay off for the agent to learn better representations before the agent has latched on to the success reward.

The ef cacy of these tasks (+10% training time, +20% SPL) already motivates use in future baselines.

5.1 Adding multiple auxiliary tasks improves over single tasks

Given the improvements from individual auxiliary tasks, we next assess if these improvements are complementary by naively combining these tasks, as given by Eq. (2). These experiments are shown in Table 5a. We experiment with combining similar auxiliary tasks, using all CPC|A variants (CPC|A-{1-16}), and also separately add ID and TD to diversify tasks used. As these variants add task losses (as in Fig. 2b), we refer to them as `Add' in gures.

A sharper early performance loss is clear in Fig. 5b CPC|A-{1-16}: Add and CPC|A-{1-16}+ID+TD: Add, compared to the baseline. Multiple task variants only surpass the baseline at 50M frames, likely due to interference with the primary POINTNAV task. Using all CPC tasks (row 3) does bump SPL by +0.02, though further adding ID and TD (row 4) yields a minor marginal gain. This is surprising, as ID and TD should be providing learning signals distinct from CPC|A tasks. We hypothesize that distinct learning signals interfere with each other when using a single belief module.

5.2 Attention over belief modules outperforms naive summation

To minimize hypothesized task con ict, we next describe experiments with fusing representations from multiple belief modules (as described in Sec. 4.2). Results are shown in Table 6a. Averaging module features (row 4) leads to very similar performance as a single module (row 3), both in AuC and in nal performance. This is expected, as the two variants are similar from the policy head's perspective. However, learned fusion yields signi cant gains in AuC over a single module, softmax fusion (row 5) and attentive fusion (row 6) achieve +0.04 SPL over single module (row 3).

Though these models have similar SPL by 40M frames, speeding up initial learning may be critical for getting off the ground in harder tasks. We propose learned fusion thus enables the use of multiple different signals while mitigating slow initial learning. In fact, we see precisely this in Fig. 6b - both fusion methods match the initial pace of CPC|A-16 (a single task), improving over CPC|A-{1-16}+ID+TD: Add.

In our experiments, we found that the variants using attention quickly (0.5M frames) start attending to just one belief module, preventing the other belief modules from in uencing the policy. This collapsed attention implies the attention variant's improvement at 40M frames are primarily due to an improved visual representation, as unattended belief modules still backpropagate gradients to the CNN. In that case, the belief modules are relying on the shared improved visual representation, rather than forming distinct beliefs as intended. We thus rectify the attention collapse with an entropy penalty on the attention distribution (see Section A.8 for details). With the penalty, the agent consistently attends to all its belief modules, resulting in row 7, the Attn+E variant. With this modi cation, the Attn+E variant edges out the softmax's (row 5) AuC and best performance.

5.3 Comparison with pre-trained weights

We also compare with using Taskonomy [30] weights for the visual encoder, speci cally representations used for depth predictions, known to transfer well to PointNav [29]. We report results with and without ne-tuning the projection layer and nal ResNet block in Table 2.

	AuC		Best	
1) CPC|A-1	0.480	0.032	0.628	0.011
2) Depth (Fine-tuned)	0.451	0.045	0.616	0.033
3) Depth (Frozen)	0.454	0.037	0.612	0.012

Table 2. With an improved visual encoder and without access to a bitmap of past locations [29], an agent with one auxiliary task performs similarly to an agent using Taskonomy depth weights, with and without ne-tuning.

Contrary to [29], we nd that neither variant outperforms our baseline. This has two primary causes. First, our baseline is stronger. Sax et al. [29] used a simple 3-layer CNN [38] for their from scratch baseline, as provided in the Habitat baselines repository [34]. We use ResNet18, improving on [34]'s results for PointNav RGB by 0.1 SPL at 40M. Second, our transfer results are lower than that of [29] due to three differences in training procedure and agent design. 1) Sax et al. [29] use an off-policy version of PPO that trades compute for sample ef ciency, 2) do not learn the stop action, which makes the task easier, and 3) provide the agent with a bitmap of previously visited locations instead of a GRU. This nding strengthens the case for learning representations from scratch rather than transferring from static visual tasks.

6 Model Analysis

Ablative analysis is done in Section A.2. Here, we examine how the agent uses its belief modules. Though different tasks provide quantitative differences, we would like to determine whether they induce characteristic “beliefs” in their modules, i.e. different functional roles. We study a run of our best variant (CPC|A-{1-16}+ID+TD: Attn+E) trained to 40M frames.

To quantify each module's contribution to performance, we mask out select belief modules when computing attention. This experiment is similar to occluding parts of images to identify which features play a causal role in predictions from classi cation CNNs [39]. First, we mask out individual belief modules (Table 3 “Masked out”) and nd that in general, individual module exclusion minimally affects the agent. In fact, SPL slightly rises when CPC|A-1 is excluded. This may indicate these modules provide redundant, generic representations. However, when CPC|A-8 is excluded, the agent's performance drops dramatically below baseline results (0.233 vs. 0.712 on SPL with and without masked CPC|A-8). Surprisingly, correlating attention with actions in Section A.5 reveals little about why CPC|A-8 may be special.

The inverse diagnostic, masking all modules except one, suggests that the agent is not relying entirely on CPC|A-8. For example, if we only use CPC|A-1, the agent can still reach 0.143 SPL, though using CPC|A-8 reaches a higher 0.225 SPL.

Auxiliary task attention distribution based on location in environment. We also qualitatively examine the attention distribution conditioned on location for an environment from the validation set in Fig. 7. We run this with our most sample ef cient model, CPC|A-{1-16}+ID+TD: Attn+E. We randomly sample 200 spawn locations with a xed goal and color-code trajectories according to the auxiliary task belief module maximally attended to. Interestingly, the visualization exhibits

Control	Exclude						Include		
		CPC|A-1	CPC|A-2	CPC|A-4	CPC|A-8	CPC|A-16		CPC|A-1	CPC|A-8
0.712		0.723	0.712	0.706	0.233	0.706		0.143	0.225

Table 3. We mask modules in one run of CPC|A-{1-16}+ID+TD: Attn+E. CPC|A-8 is critical but not sufficient for performance. Excluding ID, TD omitted for brevity, they perform similarly as CPC|A-2.

Figure 7. Top-down map of the Cantwell scene in the Gibson dataset [25]. Colored boxes represent the auxiliary task with maximum attention at the given location. Attention appears to correlate with agent location.

clustering, suggesting the agent associates environment patterns with specific belief modules. This location-characterized activation evokes the notion of `place cells' discovered in rats navigating mazes [40]. We conjecture that these characteristic distributions emerge naturally, analogous to the emergence of specialized kernels in CNNs. Different from transferred specialized representations as in [31], we see from-scratch training can learn specialized features. Indeed, we run a variant with separate belief modules without any auxiliary tasks and again observe specialized distributions.

7 Conclusion

We have shown that auxiliary tasks can greatly accelerate learning in POINTNAV. We systematically disentangle improvements in performance due to 1) individual auxiliary tasks, 2) naive combination of multiple auxiliary tasks by summing losses, and 3) attention over representations from multiple auxiliary tasks, which performs best. Our best model achieves 0.55 SPL in 7M observations — 5:5 better sample-efficiency over the baseline. This speedup suggests auxiliary tasks can be key in training embodied agents in complex environments from scratch within practical computation budgets. Our analysis further reveals agents learn to specialize their modules as specific modules became responsible for driving stopping behavior. In future work, we aim to further study this approach for other embodied tasks (e.g. language-driven navigation [41], question-answering [28]).

8 Acknowledgments

The Georgia Tech effort was supported in part by NSF, AFRL, DARPA, ONR YIPs, ARO PECASE, Amazon. AD was supported in part by fellowships from Facebook, Adobe, and Snap Inc. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government, or any sponsor.

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 *ICLR*, 2020. 1, 2, 3, 4, 11

[2] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 1

[3] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[4] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[5] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[6] C.-Y. Ma, J. Lu, Z. Wu, G. AlRegib, Z. Kira, R. Socher, and C. Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *ICLR*, 2019. 13

[7] P. Bachman, R. D. Hjelm, and W. Buchwalter. Learning representations by maximizing mutual information across views. In *NeurIPS*, 2019.

[8] Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

[9] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

[10] I. Misra and L. van der Maaten. Self-supervised learning of pretext-invariant representations. *arXiv preprint arXiv:1912.01991*, 2019.

[11] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[12] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1

[13] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016. 1

[14] Z. D. Guo, M. G. Azar, B. Piot, B. A. Pires, T. Pohlen, and R. Munos. Neural predictive belief representations. *arXiv preprint arXiv:1811.06407*, 2018. 2

[15] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017. 2, 5

[16] K. Gregor, D. J. Rezende, F. Besse, Y. Wu, H. Merzic, and A. v. d. Oord. Shaping Belief States with Generative Environment Models for RL. In *NeurIPS*, 2019. 2, 5

[17] A. Anand, E. Racah, S. Ozair, Y. Bengio, M.-A. Côté, and R. D. Hjelm. Unsupervised state representation learning in atari. In *NeurIPS*, 2019. 2

[18] X. Lin, H. Baweja, G. Kantor, and D. Held. Adaptive auxiliary task weighting for reinforcement learning. In *NIPS*, 2019. 3

[19] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell. Learning to navigate in complex environments. *CoRR*, abs/1611.03673, 2016. URL http://arxiv.org/abs/1611.03673. 2, 4

[20] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019. 2

[21] D. Gordon, A. Kadian, D. Parikh, J. Hoffman, and D. Batra. Splitnet: Sim2sim and task2task transfer for embodied visual navigation. In *ICCV*, 2019. 1, 2

[22] 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 preprint arXiv:1612.03801*, 2016. 1

[23] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied Question Answering. In *CVPR*, 2018. 2, 8

[24] A. Das, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Neural Modular Control for Embodied Question Answering. In *CoRL*, 2018. 2

[25] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018. Gibson dataset license agreement available at storage.googleapis.com/gibson_material/Agreement GDS 06-04-18.pdf . 2, 3, 8

[26] A. Kendall, R. Cipolla, and Y. Gal. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. pages 7482–7491, 06 2018. doi:10.1109/CVPR.2018.00781. 2

[27] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637, 2005. 2

[28] D. S. Chaplot, S. Gupta, D. Gandhi, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural mapping. In *ICLR*, 2020. 3, 4

[29] A. Sax, B. Emi, A. R. Zamir, L. Guibas, S. Savarese, and J. Malik. Mid-level visual representations improve generalization and sample ef cient learning for visuomotor policies. *arXiv preprint arXiv:1812.11971*, 2018. 3, 7, 12

[30] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018. 3, 7

[31] W. B. Shen, D. Xu, Y. Zhu, L. J. Guibas, L. Fei-Fei, and S. Savarese. Situational fusion of visual representation for visual navigation. In *ICCV*, 2019. 3, 5, 6, 8, 11

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017. 3, 5

[33] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 3

[34] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, et al. Habitat: A platform for embodied AI research. In *ICCV*, 2019. 3, 7, 16

[35] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra. Are we making real progress in simulated environments? measuring the sim2real gap in embodied visual navigation, 2019. 3, 15

[36] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 4

[37] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014. 4

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 7

[39] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 7

[40] N. Burgess. The 2014 nobel prize in physiology or medicine: a spatial model for cognitive neuroscience. *Neuron*, 2014. 8

[41] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 8

[42] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. In *ICLR*, 2018. 13

[43] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. MatterPort3D dataset license available at: http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf. 15

[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 16

[45] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016. 16

[46] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 16

	Area under Curve		Best	
	Success (↑)	SPL (↑)	Success (↑)	SPL (↑)
1) Baseline	0.583 ± 0.061	0.422 ± 0.043	0.714 ± 0.011	0.545 ± 0.010
2) CPC|A-1	0.652 ± 0.040	0.480 ± 0.032	0.796 ± 0.010	0.628 ± 0.011
3) CPC|A-2	0.654 ± 0.040	0.487 ± 0.035	0.797 ± 0.011	0.641 ± 0.008
4) CPC|A-4	0.680 ± 0.030	0.512 ± 0.029	0.815 ± 0.015	0.658 ± 0.014
5) CPC|A-8	0.681 ± 0.034	0.517 ± 0.027	0.815 ± 0.007	0.658 ± 0.012
6) CPC|A-16	0.677 ± 0.039	0.514 ± 0.029	0.819 ± 0.019	0.663 ± 0.010
7) ID	0.655 ± 0.052	0.458 ± 0.036	0.798 ± 0.013	0.588 ± 0.011
8) TD	0.612 ± 0.062	0.441 ± 0.044	0.756 ± 0.017	0.564 ± 0.017
9) CPC|A-{1-16}: Add	0.665 ± 0.028	0.523 ± 0.026	0.831 ± 0.007	0.687 ± 0.010
10) CPC|A-{1-16}+ID+TD: Add	0.682 ± 0.035	0.532 ± 0.028	0.842 ± 0.008	0.696 ± 0.013
11) CPC|A-{1-16}+ID+TD: Average	0.678 ± 0.026	0.539 ± 0.022	0.851 ± 0.003	0.696 ± 0.008
12) CPC|A-{1-16}+ID+TD: Softmax[31]	0.725 ± 0.022	0.578 ± 0.020	0.838 ± 0.008	0.696 ± 0.008
13) CPC|A-{1-16}+ID+TD: Attn	0.724 ± 0.031	0.565 ± 0.027	0.856 ± 0.016	0.698 ± 0.015
14) CPC|A-{1-16}+ID+TD: Attn+E	0.756 ± 0.019	0.594 ± 0.019	0.854 ± 0.008	0.707 ± 0.006
15) Depth (Fine-tuned)	0.584 ± 0.042	0.454 ± 0.037	0.773 ± 0.020	0.612 ± 0.012
16) Depth (Frozen)	0.583 ± 0.051	0.451 ± 0.045	0.772 ± 0.040	0.616 ± 0.033

Table A1. Primary quantitative results summarized. We show 1. individual auxiliary tasks greatly improve PointNav learning ef ciency, (rows 2-8) 2. adding multiple tasks naively yields marginal gains, and (rows 9-10) 3. separating task in uences into separate modules recovers early learning penalties and further improves learning. (row 11-14). Bolded variants dominate their group (denoted with dashed lines).

A Appendix

In this supplement we will perform additional analysis, covering an ablation study Section A.2, a review of trends in Success Section A.1, and model analysis in Section A.5. We provide plots of our method applied to a harder environment, in Section A.6, showing that our main trends still hold. We then consider details omitted in the main paper, describing auxiliary tasks in Section A.3, fusion methods in Section A.4, and training details in Section A.8. We conclude with a few additional gures.

A.1 Success

Success shows similar trends as SPL, as shown by the results in Table A1. We attach success plots reproduced with SPL plots as additional gures. In the following, we provide some observations.

– TD gains 4% and ID gains 10% in success over the baseline (rows 8, 7, 1) by 40M frames, a larger difference than in SPL. This shows the metrics don't strictly trend together. ID and TD agents wander more but still are better able to reach the goal than the baseline.
– The slight SPL edge that CPC|A-{4,8,16} have over CPC|A-{1,2} is still re ected in Success.
– Success is highly similar among agents in rows 9-14. This is expected, as PointNav learning is sharply logarithmic [1]. It takes much longer to improve as the agent matures, and so a small speedup won't move metrics much.
– Depth agents (rows 15, 16) still do not exceed CPC|A-1 in success.

A.2 Ablative Analysis

Given the impressive gains of learned fusion, we conduct additional experiments to decompose its improvements. We use the ve CPC|A-{1-16} tasks to focus on the effects of attention over similar tasks. Speci cally, we address the following:

1. We verify improvements when attention is largely xed on one belief module (a symptom of insuf cient entropy) to be due to improvements in visual representation. To show this, we

	Area under Curve		Best	
	Success(")	SPL(")	Success(")	SPL(")
1) Baseline	0:583 0:061	0:422 0:043	0:714 0:011	0:545 0:010
2) CPC|A-16 (Best Single)	0:677 0:039	0:514 0:029	0:819 0:019	0:663 0:010
3) Weighted CPC|A-16	0:674 0:030	0:527 0:026	0:839 0:015	0:679 0:009
4) CPC|A-{1-16}: Add	0:665 0:028	0:523 0:026	0:831 0:007	0:687 0:010
5) CPC|A-{1-16}+ID+TD: Attn+E	0:756 0:019	0:594 0:019	0:854 0:008	0:707 0:006
6) CPC|A-{1-16}: Attn	0:707 0:025	0:557 0:022	0:847 0:002	0:695 0:010
7) CPC|A-{1-16}: Attn+E	0:712 0:032	0:560 0:030	0:843 0:017	0:692 0:010
8) CPC|A-{1-16}: Fixed Attn	0:691 0:029	0:543 0:027	0:845 0:011	0:698 0:011
9) CPC|A-16 5: Attn	0:694 0:029	0:549 0:023	0:832 0:006	0:683 0:011

Table A2. Performance of attentive fusion and ablations on CPC|A family. CPC|A-{1-16}: Attn (rows 6, 7) increases AuC, as expected. All variants continue to converge to similar metrics at 40M observations.

arti cially x the agent's attention such that it always attends to CPC|A-1 – "Fixed Attn" – even though gradients from all auxiliary tasks in the CPC|A-{1-16} family backpropagate to the visual encoder.

2. Do separate belief modules help when auxiliary tasks are similar? We investigate the effects of attention even when our tasks are all CPC|A variants.
3. Do similar auxiliary tasks offer distinct gains? We apply the same auxiliary task, CPC|A-16, to 5 separate belief modules. We denote this CPC|A-16 5 and compare to CPC|A-16.

Separately, we introduce a "Weighted CPC" task to assess the value of differentiating the CPC tasks.

$$L(\ _m\ ;\ {}^1_a\ ::\ {}^{n_{Aux}}_a) = L_{RL}(\ _m) + \sum_{i=1}^{n_{Aux}} {}_{Aux}^{\ i} L_{Aux}(\ _m\ ;\ {}^i_a) \tag{3}$$

Following Eq. 3 (reproduced for reference) with CPC|A-{1-16} as auxiliary tasks leads to a setting where 1-next step prediction gets counted 5 times in the overall loss function (once each across $k = 1; 2; 4; 8; 16$), 2-step predictions get counted 4 times (once each across $k = 2; 4; 8; 16$), and so on. Since all the CPC|A-{1-16} tasks are structurally similar, we can reduce computation by emulating this total loss in a single auxiliary task. We do this via a single `weighted CPC|A-16', where 1-step prediction is scaled by 5, 2-step prediction is scaled by 4, and so on.

Our results are summarized in Table A2, and we list main findings below.

Weighted CPC|A (row 3) does manage to outperform our best CPC|A task (row 2), capturing the intuition that predicting further timesteps are more valuable for building environmental dynamics. Nonetheless weighting does not reach the performance of separate task modules (row 4).

Improvements are partly due to better visual representations. We find that benefits of multiple auxiliary tasks are partly due to improvement in visual representations (that is, Table A2, row 8 and 6 have similar best results). This mirrors the findings of Sax et al. [28], who use visual encoders pretrained on mid-level vision tasks (e.g. 3D curvature prediction) to improve sample efficiency. Notably, however, though using attention enables slightly improved AuC, indicating attention may enable adaptive learning of some form.

Also, CPC|A-{1-16}: Attn+E (row 7) again improves on CPC|A-{1-16}: Add in Success AuC by almost 7%. This reaffirms separate belief modules help even if auxiliary tasks are from the same family. Adding entropy appears unhelpful in this setting where the auxiliary tasks are highly similar.

Finally, similar tasks are slightly better than identical tasks. We find that a variant using correlated but distinct tasks performs slightly better than applying the same auxiliary task to all belief modules (CPC|A-{1-16}: Attn vs. CPC|A-16 5: Attn, rows 6 and 9 SPL). The slight distinction in signal provided does yield a better policy.

Figure A1. We study three auxiliary modules (re-printed for reference). a) Inverse dynamics: decoding action taken from successive visual embeddings x_t and x_{t+1} and the final belief state h_T. b) Temporal distance: decoding the timestep difference between two observation embeddings from final belief state h_T. c) CPC|A: decoding future observation embeddings x_{t+1} from $(x_t, \ldots, x_{t+k})$ at every timestep from other observation embeddings using a secondary GRU.

A.3 Auxiliary Task Details

Here, we elaborate on the computation done in each auxiliary module.

Inverse Dynamics (ID). As shown in Fig. A1a, given two successive observations x_t and x_{t+1} and the belief module hidden state at the end of the trajectory h_T, the ID task is to predict the action taken at time t, a_t. We include the belief module hidden state to encourage representation of trajectory actions in it.

Specifically, we take the visual embeddings from 1 to T, trim the final timestep to form the "before" batch, and the first timestep to from the "after" batch. We then concatenate each timestep pair with the belief module output from timestep T, and predict action logits. We use cross-entropy loss with the true actions from timesteps 1 to $(T-1)$, and subsample the loss by 0.1.

$$L_{ID} = \sum_{i=1}^{T-1} L_{CE}(I(x_i, x_{i+1}, h_T), a_i) \tag{4}$$

Temporal Distance (TD). As shown in Fig. A1b, given two observations from a trajectory x_i and x_j) and the belief module hidden state at the end of the trajectory h_T, the TD task is to predict $\frac{|i-j|}{T}$. This is similar in spirit to progress estimation in [6] and reachability in [42]. However, rather than Euclidean or geodesic distance, we ask the agent to predict the (normalized) number of steps elapsed between two visual observations. This requires the agent to recall if a location is revisited or similarly viewed, designed to promote understanding of spatio-temporal relations of trajectory viewpoints.

In detail, we select $k = 8$ random pairs of indices, and get their corresponding visual embeddings. We concatenate the pairs' visual embeddings with the belief module's end output h_T, and directly use a linear layer to predict the timestep difference between each pair.

$$L_{TD} = \frac{1}{2}((i-j) - T \cdot (x_i, x_j, h_T))^2 \tag{5}$$

Action-Conditional Contrastive Predictive Coding (CPC|A). As shown in Fig. A1c, given the belief module hidden state h_t, a second GRU is unrolled for k timesteps using future actions $\{a_{t+i}\}_{i=0}^{k-1}$ as input. The output of the second GRU at time $t+i$ is used to distinguish different visual representations. We concatenate the second GRU's output at $t+i$ with a) the ground-truth visual representation at $t+i$, x_{t+i+1}, or b) a "negative" visual feature x_{t+i+1} sampled from other timesteps and trajectories. Then, "contrasting" the different representations can be framed as a classification task, classifying inputs with the ground-truth visual representation as 1, and negatives as 0. This encourages h_t to build long-horizon representations of the environment.

Technically, a CPC|A-specific GRU with hidden size 512 is initialized with output h_t from the belief module. Its k input actions are first fed through a size 4 embedding layer. The CPC|A GRU then outputs $g_1^t, g_2^t, \ldots, g_k^t$. These outputs are concatenated with positive and negative visual embeddings, and fed into a two layer decoder (hidden size 32). The decoder predicts logits for whether the input

Figure A2. Left: Dot-product attention, Right: Softmax gating.

Figure A3. Distribution of which auxiliary tasks are most attended to while taking each action for CPC|A-{1-16}: Attn+E. CPC|A-1 and CPC|A-2 signi cantly affect the STOP action

contained a positive or negative visual embedding, which is fed into a cross entropy loss given targets 1 and 0 respectively. We perform this for all timesteps 1 to $(T - k)$, and subsample the loss by 0.2.

$$L_{CPC|A} = \sum_{k=1}^{K} \sum_{t=1}^{T-k} L_{CE}(c(g_k^t; \quad); 0) + L_{CE}(c(g_k^t; \quad_{t+k}); 1) \tag{6}$$

A.4 Module Fusion Details

All fusion methods are achieved by a form of weighted sum. Fixed and average fusion are achieved by freezing weights as desired. Weight calculation for softmax gating and attention are described by Fig. A2. In softmax gating, a linear layer directly converts the visual embedding into logits that are passed through a softmax layer to create weights. With dot-product attention, the visual representation is passed through a "key" linear layer, outputting a key of size 512. The representations given by the separate belief modules serve as queries, which are multiplied by the key to create our logits. These logits are again put through a softmax layer to create our nal weights.

Entropy We use a variant of scaled dot-product attention with an entropy penalty (denoted `+E'). Given attention distribution $w_{attn} := (p_1; \ldots; p_{n_{Aux}})$, we calculate entropy as $\sum_{i=1}^{n_{Aux}} p_i \log p_i$. Entropy encourages the agent to use multiple belief modules. An agent that quickly learns to use a single module may prevent the other modules from learning about the task (from reduced gradients).

A.5 More Model Analysis

Fig. A3 shows how auxiliary task attention correlates with the action taken for the same run analyzed in Section 3. To compute this, for all agent trajectories in the validation set, we assign credit of +1 to a given action in which an auxiliary task's belief module receives the most attention. All shows the overall count distribution regardless of action taken. In this run, CPC|A-16 is attended very frequently, and is highly correlated with turning actions. The STOP action almost always corresponds with attention to CPC|A-1 and CPC|A-2, which may suggest they play an important role in short-term decisions. However, applying the same analysis to other runs of the same variant, we nd different attended modules for STOP (e.g. corresponding to CPC|A-4). Attention over different belief modules does suggest some functional expertise in POINTNAV, but auxiliary tasks do not consistently determine the expertise.

Figure A4. Action distributions on CPC|A-{1-16}: Attn+E validation episodes, for steps where a given auxiliary task is used (weight $\geq$ 0.25). Again, the STOP action reliably activates CPC|A-1, CPC|A-2.

Auxiliary Tasks have Characteristic Action Distributions Instead of conditioning the task distribution on the action as in the main text, we can condition the action distribution on the task. We generate these plots (Fig. A4) by thresholding all actions taken with conditioned task weight $\geq$ 0.25. In this analysis, it is clear that stopping is infrequent enough that CPC|A-1, CPC|A-2 don't appear particularly associated with the action. Overall, these preliminary analyses suggests attention provides only a shallow explanation for agent behavior.

A.6 MP3D experiments

We run our 3 representative variants on a harder environment to show our method can generalize, showing results in Fig. A5. Namely, we test combining the auxiliary tasks on one module (Single) and our attentive architecture against the baseline. Our new environment is on the Matterport3D dataset [43], with actuation noise and wall sliding turned off, making navigation much more difficult [35]. Nonetheless, using our attentive architecture improves on the baseline, whereas naive application takes much longer to overtake the baseline.

A.7 Improving Local Planners in Hierarchical Navigation

We briefly check that our method improves local navigation. It improves success and SPL when navigation goals are nearby. We plot SPL with respect to geodesic distance to goal at spawn in Fig. A6. Our method improves metrics across the board, with a slight bias towards improving shorter episodes. Thus, our approach could be adopted in improving local planners in hierarchical agents.

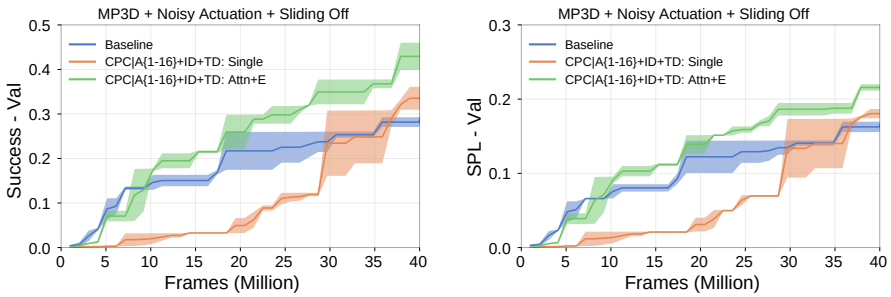


Figure A5. Our method improves on the baseline in Matterport3D.

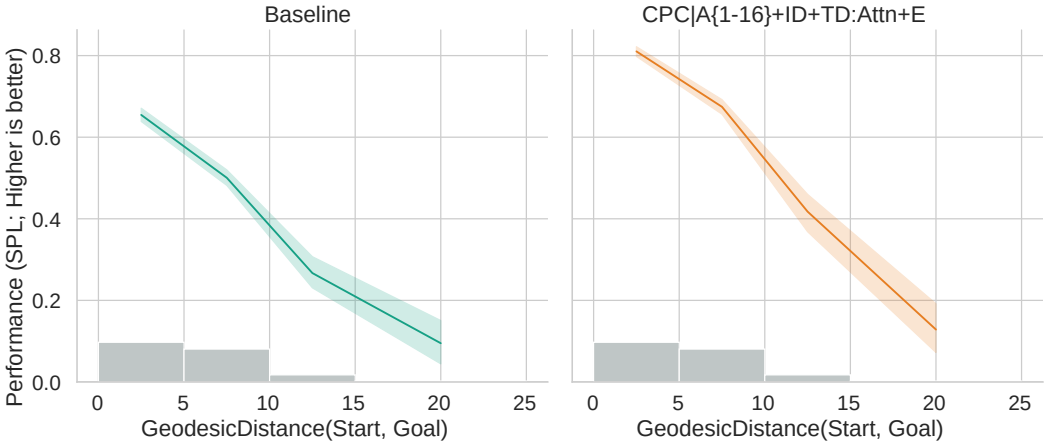


Figure A6. Our method improves navigation at all ranges, thus enabling improvement of local navigation.

A.8 Training Details

Training. We train our agent via Proximal Policy Optimization PPO) [44] with Generalized Advantage Estimation (GAE) [45]. We use 4 rollout workers with rollout length $T = 128$, and 4 epochs of PPO with 2 mini-batches per epoch. We set discount factor to $\gamma = 0.99$ and GAE factor $\tau = 0.95$. We use the Adam optimizer [46] with a learning rate of 2.5 10^{-4} and $\epsilon = 0.1$.s We follow the reward structure in [34]. For goal g, when the agent is in state s_t and executes action a_t (transitioning to s_{t+1}),

$$r_t(s_t, a_t) = \begin{array}{ll} 2.5 & \text{Success} \\ \texttt{GeoDist}(s_t, g) & \texttt{GeoDist}(s_{t+1}, g) \quad \lambda \end{array} \begin{array}{l} \text{if } a_t = \texttt{stop} \\ \text{otherwise} \end{array} \tag{7}$$

where GeoDist is the geodesic distance and λ(=0.01) is a slack penalty. No hyperparameter sweeps were done. Model sizes were all 5.7 0.3 million parameters. This count comprises the visual encoder and the policy networks, but not the decoder networks, though the hidden size is shared throughout the modules. To achieve the uniform model size, single module variants used a GRU with hidden size 512, while multiple module networks had hidden sizes correspondingly reduced to 256 288.

To evaluate models in t-tests, we select the checkpoints with highest average validation metrics (over 3 validation runs) across 4 training seeds.

The belief module receives input of size 514, 512 from the ResNet-18 visual module, and 2 from the GPS-Compass sensor.

Our complete loss is:

$$L_{\text{total}}(\theta_m; \theta_a) = L_{\text{RL}}(\theta_m) \quad \alpha H_{action}(\theta) + L_{\text{Aux}}(\theta_m; \theta_a) \tag{8}$$

$$L_{\text{Aux}}(\theta_m; \theta_a) = \sum_{i=1}^{n_{\text{aux}}} \beta^i L^i_{\text{Aux}}(\theta_m; \theta^i_a) \quad \mu H_{attn}(\theta_m) \tag{9}$$

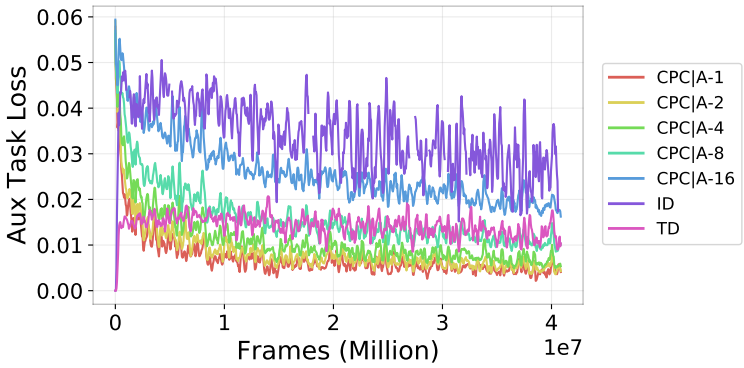


Figure A7. Auxiliary Task loss curves for CPC|A-1-16+ID+TD: Attn+E.

H_{attn} is the entropy of the attention distribution over the different auxiliary tasks. In our experiments, we set $\alpha = 0.01$, and $\mu = 0.01$. We set β^i for ID and CPC|A tasks at 0.1, and 0.4 for TD. These values were determined such that the loss terms were in the same order of magnitude at initialization. Losses for the auxiliary tasks trend stably downward as shown in Fig. A7. The agent does not initially have trajectories sufficiently long (*i.e.* predicting `stop` after a few steps) to appropriately calculate TD and CPC|A tasks, so they start with 0 loss. Other training hyperparameters (some repeated from the main text) are as follows:

$$\text{Rollout Workers: } n = 4 \tag{10}$$
$$\text{Rollout Length: } t = 128 \tag{11}$$
$$\text{PPO Epochs} = 4 \tag{12}$$
$$\text{PPO Mini-batches} = 2 \tag{13}$$
$$\gamma = 0.99 \tag{14}$$
$$\tau = 0.95 \tag{15}$$
$$\epsilon = 0.1 \tag{16}$$
$$\text{lr} = 2.5 \quad 10^{-4} \tag{17}$$
$$\text{Gradient Norm Cap} = 0.5 \tag{18}$$
$$\text{PPO Clip} = 0.1 \tag{19}$$
$$\tag{20}$$

A.9 Additional Figures

Success and SPL Validation Curves We provide the Success validation curves and reproduce the SPL validation curves for reference.

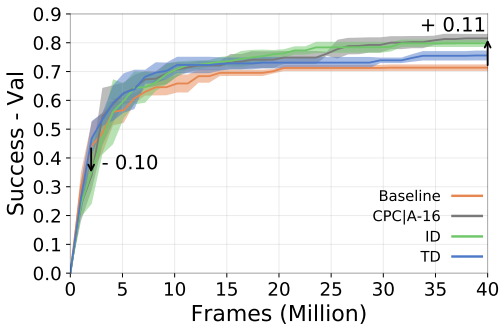


Figure A8. All auxiliary tasks overtake the baseline after 5M frames. CPC|A-4 provides larger gain than other auxiliary tasks.

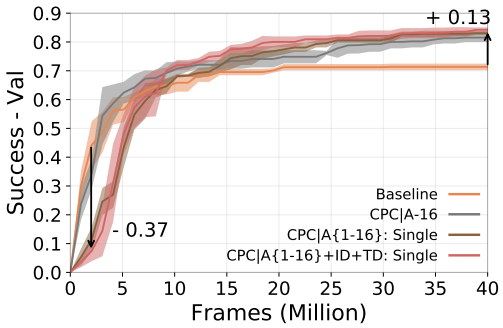


Figure A9. Multiple auxiliary task combinations learn better than a single task as agent matures (20-40M frames) at some cost to initial ramp-up.

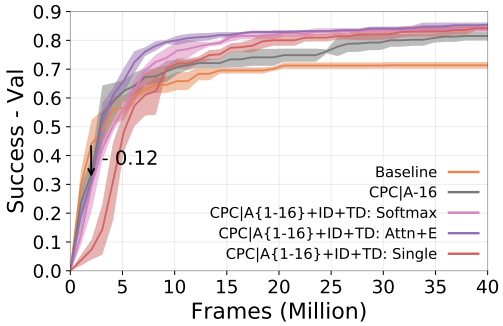


Figure A10. Learning fusion of separate modules in CPC|A-{1-16}+ID+TD: Attn+E recovers the initial ramp-up cost experienced by CPC|A-{1-16}+ID+TD: Single.

Additional Top Down Map Visualizations We also provide top down map visualizations (Fig. A11) from two more Gibson scenes, Quantico and Eastville. Similar trends prevail as before. The TD task appears to be more activated when beds are in the frame.