# 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 but their method is computationally prohibitive – requiring 2.5 billion frames and 180 GPU-days.

In this work, we develop a method to significantly increase sample and time 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 learnt from individual auxiliary tasks. Our best agent is 5x faster to reach 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

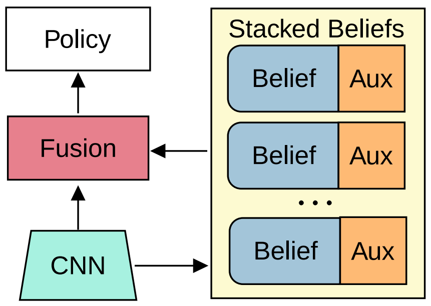
Consider a robot tasked with navigating from the living room to the kitchen solely from first-person egocentric vision. In order to do so, it must be able to reason about 1) notions of free space (that corridors and doors can be walked through, but not walls), 2) keep regions already visited in memory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

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!

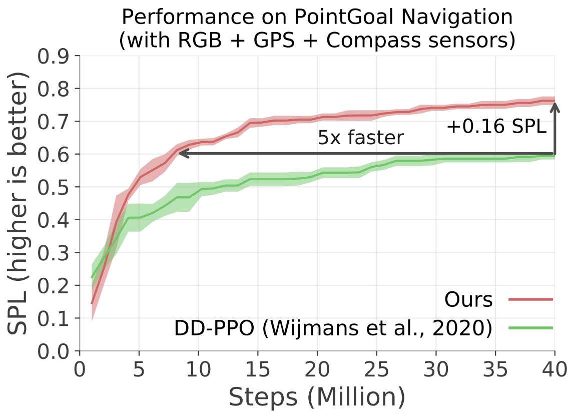
While [1] serves as an excellent 'existence proof' of the learnability of PointNav, we believe it should not take 2.5 billion frames of experience and nearly 6 months of GPU time to learn to navigate from point A to B. In mathematics and theoretical computer science, an existence proof is often the first crack in the wall, frequently followed soon thereafter by improvements to the underlying techniques – a non-constructive proof replaced by constructive proof, an improved algorithm, shaving off factors in bounds – till more barriers come falling down and the problem is well-understood. *That* is the goal of this work – specifically, to improve sample and time efficiency in learning PointNav using 'self-supervised auxiliary' tasks.

These tasks (*e.g.* predicting the action taken between two egocentric observations, predicting the distance between two observations from a trajectory, predicting future observations in a trajectory from current observation) are 'self-supervised' – *i.e.* make use of information already available to the

*Correspondence to `joel.ye@gatech.edu`
†EW and AD contributed equally.



(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× faster and improves on the baseline's performance at 40M frames by 0.16 SPL.

agent – and 'auxiliary' – *i.e.* requiring core competencies independent of any particular downstream task (such as PointGoal Navigation).

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, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] and reinforcement learning paradigms [16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. First, auxiliary tasks are typically benchmarked in visually simple simulated environments (*e.g.* DeepMind Lab [26], 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?

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) [17], inverse dynamics (predicting the action taken between two egocentric observations) [18], and temporal distance estimation – and show that each of these objectives 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 0.61 to 0.74 SPL (+21%). With a fixed performance level (0.61 SPL), CPC|A-4 achieves a 2.4× reduction in no. of steps required (from 40M to 17M).
– 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 .76 SPL (+24%) by 40M steps and achieves 0.61 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 0.61 SPL in 8M steps of experience, a 5× speedup over the baseline from [1].

## 2 Related Work

Our work is related to and builds on prior work in auxiliary objectives for learning representations in reinforcement learning, combining and weighing multiple such objectives, and prior approaches to PointGoal navigation.

**Auxiliary Tasks in Reinforcement Learning.** Intuitively, 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 [22, 27, 28, 29]). Self-supervised auxiliary tasks, such as next-step visual feature prediction [18], predictive modeling [17, 19], or spatio-temporal mutual information maximization [20, 23] derive supervision from the agent's own experience. In contrast to prior work, which focus on simpler and non-photorealistic environments [17, 19, 20, 21, 22], we focus on visually complex, photorealistic environments from the Gibson 3D scans [30].

Closely related to our work is that of Gordon et al. [24] 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 [31], or as a prior [32]. In contrast, we propose a formulation that *learns the appropriate influence* of auxiliary tasks during training.

Another similarly motivated approach, that of Lin et al. [21], adaptively manipulates weights of auxiliary losses. However, this approach is limited to training time. In comparison, our approach enables different auxiliary tasks to have different influences during training *and 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, we propose an explicit approach using adaptive weights over multiple 'experts' (representations learnt via separate auxiliary tasks). An explicit weight distribution sidesteps the need for mathematical approximations as in [21] and allows for inference-time visualization of task influence.

Use of multiple visual representations derived from static-vision tasks appear frequently in visual navigation literature, driven largely by studies of relations between visual representations [33]. Sax et al. [34] showed that no single visual transfer task was ideal for multiple embodied tasks, and combinations of diverse representations are best when the downstream navigation task is unknown. Shen et al. [35] demonstrated that fusing representations based on the current observation outperforms naive combination. Our work operates in the same regime and makes use of dot-product attention [36] to guide this fusion.

**PointGoal Navigation.** PointGoal Navigation in photorealistic environments has seen remarkable progress in the last few years, with several entries to the 2019 Habitat PointNav Challenge exceeding 0.70 SPL in only 10M steps of experience. One leading method is Active Neural Mapping [37], which uses explicit neural maps of the environment in hierarchical planning modules. Another approach [34] transfers mid-level visual features from Taskonomy [33], freezing visual representation encoders. 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 be used to improve the local planner in [37]. Nonetheless we briefly compare with [34] in 5.4, which uses off-the-shelf Taskonomy representations. As for [35], 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], as from-scratch representations are demonstrably effective at learning navigation. Given 180 GPU-days, agents can be trained from scratch to solve PointNav [1]. We build on this baseline architecture and restrict ourselves to a realistic compute budget (1 GPU-week). We briefly compare with Taskonomy representations [33, 34] to see whether resource-constrained from-scratch learning can overtake SoTA transferred representations.

## 3 Task, Simulation, Agent, and Training

**PointGoal Navigation.** In PointNav [38], 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`}.

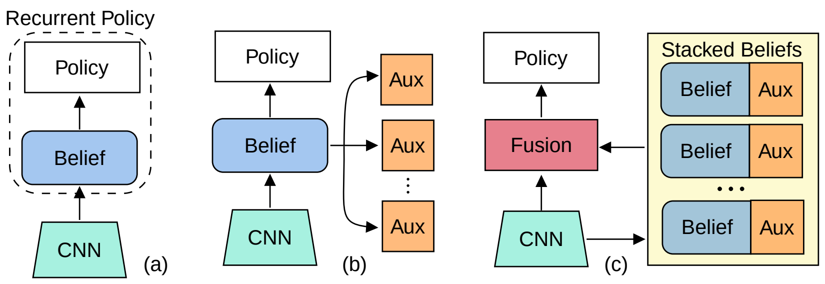


**Figure 2.** (a) Baseline architecture from DD-PPO [1] with the recurrent policy conceptually separated into a recurrent 'belief' module and a feed-forward policy head. (b) Single-belief, where multiple auxiliary tasks utilize the same shared belief module output. (c) Fused-beliefs, where each auxiliary 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.

**Metrics.** We evaluate the agent on two metrics – 1) success: whether not not the agent correctly predicted `stop` within 0.2m of the goal, and 2) Success weighted by inverse Path Length (SPL) [38]: which weights success by how efficiently the agent navigated to the goal relative to the shortest path.

**Simulation.** We utilize the AI Habitat platform [39] to simulate our agent. Following the 2019 Habitat Challenge [39], we train and evaluate performance on the Gibson dataset [30]. As in [39], we utilize higher-quality reconstructions, namely 72 houses for training and 14 houses for validation.

**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 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 use this split so our auxiliary tasks can operate on belief module output, as shown in Fig. 2b.

Our modifications to the baseline architecture are intentionally minimal, allowing us to isolate the impact of auxiliary tasks. We use ResNet18 [40] as modified for on-policy RL by Wijmans et al. [1] for our visual encoder.

The belief module is a single layer GRU [41]. Its output $h_t$ is passed to the policy head, which is a fully-connected layer, producing a softmax distribution over the action space, and a value estimate.

**Training.** We train our agent via Proximal Policy Optimization PPO) [42] with Generalized Advantage Estimation (GAE) [43]. 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 [44] with a learning rate of $2.5 \times 10^{-4}$ and $\epsilon = 0.1$.s We follow the reward structure in [39]. For goal $g$, when the agent is in state $s_t$ and executes action $a_t$ (transitioning to $s_{t+1}$),

$$r_t(s_t, a_t) = \begin{cases} 2.5 \cdot \text{Success} & \text{if } a_t = \texttt{stop} \\ \texttt{GeoDist}(s_t, g) - \texttt{GeoDist}(s_{t+1}, g) - \lambda & \text{otherwise} \end{cases} \quad (1)$$

where `GeoDist` is the geodesic distance and $\lambda$(=0.01) is a slack penalty.

## 4 Self-Supervised Auxiliary Tasks from Experience

We introduce a set of auxiliary modules, one for each auxiliary task. While the policy head is trained to maximize rewards for the primary task – PointNav (Eq. 1) – these auxiliary modules provide additional learning signals and operate on the observations, outputs of the belief module, and actions. Specifically, the agent receives observation $x_t$, extracts its CNN representation $\phi_t$, which is fed to the belief module to compute $h_t$, based on which an action $a_t$ is sampled from the policy. Auxiliary tasks use a subset of $\{(x_1, \phi_1, h_1, a_1) \dots (x_T, \phi_T, h_T, a_T)\}$. 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 specifically only consider self-supervised tasks, to keep the method generally applicable across

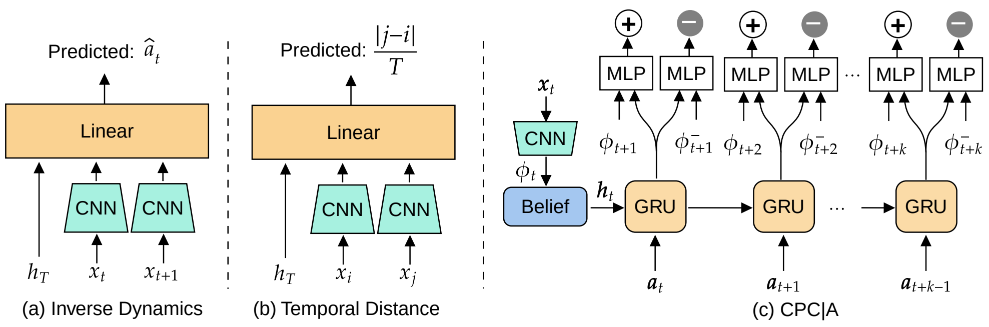


**Figure 3.** We study three auxiliary modules. a) Inverse dynamics: decoding action taken from successive visual embeddings $\phi_t$ and $\phi_{t+1}$ and the final belief state $h_T$. b) Temporal distance: decoding the timestep difference between two observation embeddings from final belief state $h_T$. c) CPC|A: decoding future observation embeddings $(\phi_{t+1}, ..., \phi_{t+k})$ at every timestep from other observation embeddings using a secondary GRU.

settings. This disallows tasks like depth prediction, which is known to make POINTNAV much easier to learn [22, 37]. We next describe the tasks in detail.

**Inverse Dynamics (ID).** As shown in Fig. 3a, 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.

**Temporal Distance (TD).** As shown in Fig. 3b, 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 $\frac{|j-i|}{T}$. This is similar in spirit to progress estimation in [8] and reachability in [45]. 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.

**Action-Conditional Contrastive Predictive Coding (CPC|A).** As shown in Fig. 3c, 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 $t+i$, $\phi_{t+i+1}$, or b) a "negative" visual feature $\phi^-_{t+i+1}$ 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. In our experiments, we consider the instances of CPC|A with $k = 1, 2, 4, 8, 16$ (denoted CPC|A-1, CPC|A-2 *etc.*), which leads to a family of 5 *similar* tasks, each being optimized for representations that make predictions at different timescales. We denote the entire family as CPC|A-{1-16}.

During training, we optimize the parameters of the belief module, the policy, and the visual encoder, altogether $\theta_m$, as well as the auxiliary module parameters, $\theta_a$, to jointly minimize the auxiliary loss and the primary POINTNAV objective, $L_{\text{RL}}$.

$$L_{\text{total}}(\theta) = L_{\text{RL}}(\theta_m) + \beta_{\text{Aux}} L_{\text{Aux}}(\theta_m, \theta_a) \tag{2}$$

where $\beta_{\text{Aux}}$ is a hyperparameter that balances the two losses. We set $\beta_{\text{Aux}}$ such that the two losses have roughly equal gradient 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 naive implementation of this (see Fig. 2b) is to apply different tasks to the same recurrent module, adding all the individual loss terms with the same loss scales as in the individual task setup. For $n_{\text{Aux}}$ such auxiliary tasks, we denote individual auxiliary task-related parameters as $\theta_a^1 \dots \theta_a^{n_{\text{Aux}}}$. Then, the new loss is given by:

$$L(\theta_m; \theta_a^1 \dots \theta_a^{n_{\text{Aux}}}) = L_{\text{RL}}(\theta_m) + \sum_{i=1}^{n_{\text{Aux}}} \beta_{\text{Aux}}^i L_{\text{Aux}}(\theta_m; \theta_a^i) \tag{3}$$

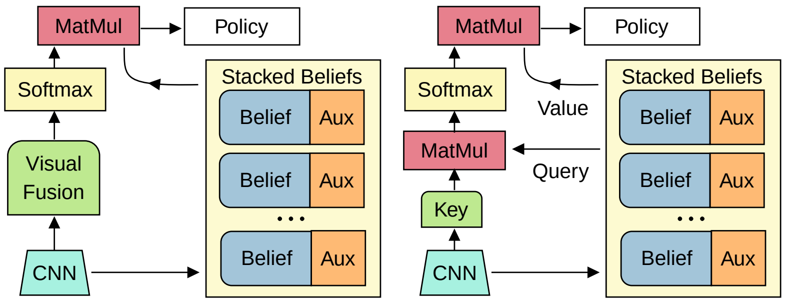


**Figure 4.** We use two learned weightings to combine different belief representations. Left: Softmax gating predicts weights conditioned on the visual representation. Right: Scaled dot-product attention computes scores by taking dot product of belief module outputs with a key vector computed from the visual representation, normalized by $\sqrt{n_{\text{Aux}}}$.

**Weighted CPC|A-16.** Following Eq. 3 with CPC|A-$\{1\text{-}16\}$ 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-$\{1\text{-}16\}$ 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.

### 4.2 Attention over multiple auxiliary tasks

As we investigate in Section 5, we find that using the setup described in Section 4.1 to combine auxiliary tasks does improve performance over individual tasks, but the gains are marginal beyond a point.

| Fusion Method | Description |
|---|---|
| Fixed [18] | Full attention fixed to a single belief module. |
| Average | Equal weighting on all belief modules. |
| Softmax Gating [35] | Softmax weighting on belief modules predicted directly from the visual representation, $\mathbf{w} = \text{softmax}(f(\phi))$. |
| Scaled Dot-Product Attention (Attn)[36] | Weights computed as $\text{score}(h_i, k) = \frac{h_i^T k}{\sqrt{n_{\text{Aux}}}}$ with belief outputs $h_i$ and key vector $\mathbf{k} = g_{\text{key}}(\phi)$. |

**Table 1.** Fusion methods. See Appendix for details.

We hypothesize that this is because when multiple auxiliary tasks operate on the same belief module (Fig. 2b), their objectives can potentially compete or be orthogonal. Adding additional objectives thus hinders learning. To improve on this and 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 the outputs of the belief modules to be fed as input to 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 attend to policy-relevant representations.

We experiment with several fusion methods (Table 1). Softmax gating and scaled dot-product attention are shown in Fig. 4, with details in Section 9.3. We additionally consider a variant of scaled dot-product attention with an entropy penalty (denoted '+E'). Given attention distribution $w_{\text{attn}} := (p_1, \dots, p_{n_{\text{Aux}}})$, we calculate entropy as $\sum_{i=1}^{n_{\text{Aux}}} -p_i \log p_i$. 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).

## 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 auxiliary tasks?
3. What is the best way to fuse representations from multiple auxiliary tasks?

|  | Area under Curve |  | Best |  |
|---|---|---|---|---|
|  | Success (↑) | SPL (↑) | Success (↑) | SPL (↑) |
| 1) Baseline | $0.602 \pm 0.074$ | $0.441 \pm 0.054$ | $0.776 \pm 0.014$ | $0.612 \pm 0.022$ |
| 2) CPC\|A-1 | $0.689 \pm 0.045$ | $0.516 \pm 0.037$ | $0.864 \pm 0.017$ | $0.701 \pm 0.013$ |
| 3) CPC\|A-2 | $0.698 \pm 0.041$ | $0.527 \pm 0.039$ | $0.861 \pm 0.015$ | $0.721 \pm 0.022$ |
| 4) CPC\|A-4 | $0.725 \pm 0.032$ | $0.554 \pm 0.034$ | $\mathbf{0.883 \pm 0.013}$ | $\mathbf{0.739 \pm 0.011}$ |
| 5) CPC\|A-8 | $0.723 \pm 0.039$ | $0.554 \pm 0.032$ | $0.879 \pm 0.020$ | $0.735 \pm 0.017$ |
| 6) CPC\|A-16 | $0.720 \pm 0.043$ | $0.556 \pm 0.037$ | $\mathbf{0.878 \pm 0.007}$ | $\mathbf{0.739 \pm 0.009}$ |
| 7) ID | $0.688 \pm 0.061$ | $0.487 \pm 0.045$ | $0.856 \pm 0.012$ | $0.646 \pm 0.023$ |
| 8) TD | $0.647 \pm 0.066$ | $0.472 \pm 0.051$ | $0.810 \pm 0.017$ | $0.622 \pm 0.017$ |

**Table 2.** A variety of auxiliary tasks accelerate learning of POINTNAV. Bolded variants dominate others in a paired t-test ($p < 0.05$). Mid-range CPC|A tasks provide the most gain, for example CPC|A-4 provides +0.12 SPL (+19%) gain by 40M frames.

**Learning PointNav with a Compute Budget** We train each variant described in Section 4 for 40 million observations over 4 random seeds and report the highest average validation (averaged over three validation runs) Success rate and SPL achieved by 40M observations. We select 40M as it corresponds to a reasonable compute threshold of 1 GPU-week. Since we are interested in sample efficiency, we also compare the area under the metric learning curves (AuC) over 40M observations, with measurements at 1M observation intervals. Models with higher AuC learn POINTNAV faster, and are especially important for more challenging tasks, where slow learning might become intractable. When computing AuC, we first normalize the x-axis (number of observations) to $[0.0, 1.0]$, making the maximum possible AuC 1.0.

Throughout experiments, we refer to observations as 'frames' of simulation, as done in prior work. In tables, we bold one variant over others with overlapping confidence intervals if it has better performance across validation episodes (paired t-test, $p < 0.05$). Note that we use separate runs of the checkpoints with highest metrics for t-test evaluations, naturally biasing t-test metrics to be lower than values in tables, and occasionally resulting in bolded runs with lower early-stopping metrics.

## 5.1 Individual auxiliary tasks help

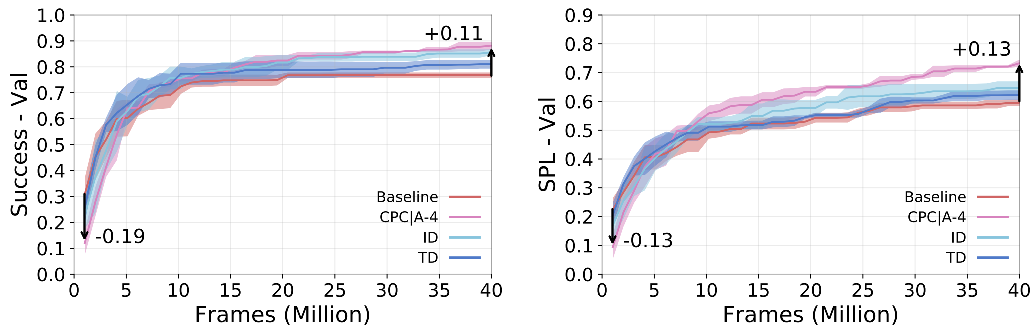


**Figure 5.** All auxiliary tasks overtake the baseline after 5M frames. CPC|A-4 provides larger gain than other auxiliary tasks.

All variants with single auxiliary tasks obtained higher SPL and success rates than the baseline, as shown in Table 2. CPC|A excels at longer ranges, for example, with rows 4, 5, and 6 indicating they provide at least $+0.12$ SPL at 40M frames, a $+19.6\%$ gain. This is also reflected in AuC, where $+0.11$ SPL is a $25\%$ gain. TD and ID also provide gains, of $\sim 4\%$ and $\sim 10\%$ respectively, over baseline success.

| | Area under Curve | | Best | |
|---|---|---|---|---|
| | Success (↑) | SPL (↑) | Success (↑) | SPL (↑) |
| 1) Baseline | $0.602_{\pm 0.074}$ | $0.441_{\pm 0.054}$ | $0.776_{\pm 0.014}$ | $0.612_{\pm 0.022}$ |
| 2) CPC\|A-4 (Best individual task) | $0.725_{\pm 0.032}$ | $0.554_{\pm 0.034}$ | $0.883_{\pm 0.013}$ | $0.739_{\pm 0.011}$ |
| 3) Weighted CPC\|A-16 | $0.716_{\pm 0.033}$ | $0.567_{\pm 0.031}$ | $0.890_{\pm 0.005}$ | $0.741_{\pm 0.007}$ |
| 4) CPC\|A-{1-16}: Single | $0.708_{\pm 0.033}$ | $0.563_{\pm 0.034}$ | $\mathbf{0.893}_{\pm \mathbf{0.008}}$ | $\mathbf{0.760}_{\pm \mathbf{0.010}}$ |
| 5) CPC\|A-{1-16}+ID+TD: Single | $0.725_{\pm 0.042}$ | $0.574_{\pm 0.039}$ | $\mathbf{0.915}_{\pm \mathbf{0.014}}$ | $\mathbf{0.765}_{\pm \mathbf{0.010}}$ |

**Table 3.** Impact of multiple auxiliary tasks. CPC|A-{1-16} adds another 0.02 SPL over CPC|A-4. We see sharp diminishing returns, as adding ID+TD provides $\approx$ +0.005 SPL further. Using more tasks does not affect AuC, implying early learning is slowed.

Interestingly, we observe a slight loss in performance early on in variants with auxiliary tasks in Fig. 5. They overtake the baseline at around $4-5$M frames. This is highly prominent in the CPC|A-4 curve. This initial learning phase involves the agent's first 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.

All of the variants, including the baseline, have an initial ramp-up period where success quickly reaches around $0.6-0.7$ success, as seen in Fig. 5, after which point learning slows significantly. Intuitively, this inflection point represents when an easier subset of episodes is learned, and subsequent episodes provide diminishing returns. The dropoff is softer for variants with auxiliary tasks, indicating better representations are already making it easier to learn.

Overall, the cheap effectiveness of these tasks ($+10\%$ training time, $+20\%$ performance gain) already suggests future baselines should incorporate them.

**CPC|A tasks perform similarly well**. The five CPC|A variants (with different horizon lengths $k$) have subtly differing performances. Shorter horizons ($k=1,2$) seem to lack $0.02$ success over longer ones $k=4,8,16$, in terms of best performance by 40M frames. The picture is clearer when observing the average performance in rows 2-6, where longer horizon variants achieve $+0.02$ SPL over shorter ones. This means that longer horizon predictions lead to better representations, consistent with intuitions in [19]. However, these are minor gains, and in general, the tasks give similar benefits despite intuitively distinct signals.

With similar performance among $k=4,8,16$, we choose $k=4$ to represent the best single task in subsequent comparisons.

## 5.2 Multiple auxiliary task combinations improve over single

Given the improvements from individual auxiliary tasks, we next assess if these improvements are complementary by naively combining these tasks, as given by Eq. (3). These experiments are shown in Table 3. First, we combine all CPC|A variants (CPC|A-{1-16}). These are multiple similar auxiliary tasks. Next, we combine CPC|A-{1-16} with ID and TD. So these are multiple diverse auxiliary tasks. Finally, we also compare to Weighted CPC|A-16 (described in Section 4.1), which emulates CPC|A-{1-16} in a single auxiliary objective. As all these variants still operate on a single belief module (as in Fig. 2b), we refer to them as 'Single' in tables and plots.

Weighted CPC|A (row 3), performs similarly to our best CPC|A task (row 2). In that light, the gain of row 4 is more significant – we gain $+0.02$ SPL at 40M while using the same penalties. The distinction is that CPC|A-{1-16} has separate belief modules for each CPC|A time horizon.

However, an early performance loss is clear in Fig. 6 CPC|A-{1-16}: Single and CPC|A-{1-16}+ID+TD: Single, compared to the baseline. The initial learning inhibition noted earlier is sharper, with our multiple task variants only surpassing the baseline at ~5M frames, likely due to interference with the primary POINTNAV task.

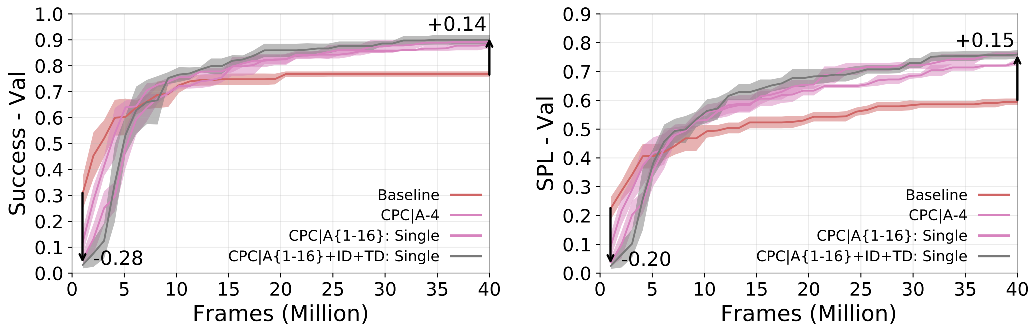


**Figure 6.** Multiple auxiliary task combinations learn better than a single task as agent matures (20-40M frames) at some cost to initial ramp-up.

| | Area under Curve | | Best | |
|---|---|---|---|---|
| | Success (↑) | SPL (↑) | Success (↑) | SPL (↑) |
| 1) Baseline | 0.602±0.074 | 0.441±0.054 | 0.776±0.014 | 0.612±0.022 |
| 2) CPC\|A-4 (Best Individual) | 0.725±0.032 | 0.554±0.034 | 0.883±0.013 | 0.739±0.011 |
| 3) CPC\|A-{1-16}+ID+TD: Single | 0.725±0.042 | 0.574±0.039 | **0.915±0.014** | 0.765±0.010 |
| 4) CPC\|A-{1-16}+ID+TD: Average | 0.718±0.030 | 0.580±0.031 | **0.911±0.008** | 0.765±0.020 |
| 5) CPC\|A-{1-16}+ID+TD: Softmax[35] | 0.766±0.027 | 0.621±0.029 | **0.912±0.008** | **0.791±0.017** |
| 6) CPC\|A-{1-16}+ID+TD: Attn | 0.771±0.032 | 0.612±0.033 | **0.919±0.014** | 0.784±0.006 |
| 7) CPC\|A-{1-16}+ID+TD: Attn+E | 0.794±0.024 | 0.631±0.025 | **0.905±0.005** | 0.770±0.003 |

**Table 4.** Performance of separate belief modules. Fusion methods match a single module in best success, but learned fusion (row 5) provides a +.035 SPL over a single module (row 3). Further, attentive fusion (row 7) yields a +.057 (+10%) SPL AuC over a single module. All variants converge to similar best metrics at 40M observations.

Adding on the additional tasks of ID and TD (row 5) performs slightly better than CPC|A-4 in the 10M to 25M frame range, but yields minimal final performance gain. This is somewhat surprising, as ID and TD should be providing learning signals distinct from CPC|A tasks. We hypothesize that these distinct learning signals interfere with each other when operating on a single belief module. We explore this further in the next section. The convergence of best metrics by 40M frames across rows 2-5 is likely due to the sharply logarithmic learning observed in PointNav agents as observed in [1].

### 5.3 Attention over belief modules outperforms naive summation

To prevent multiple auxiliary objectives conflicting over a single belief representation, we next describe experiments with fusing representations from multiple belief modules (as described in Sec. 4.2). Results are summarized in Table 4.

We first note that final success is equal across the board in Table 4. Intuitively, this means that these auxiliary tasks are not providing distinct enough signals to solve the hardest episodes. Also, averaging representations (row 4) across modules leads to very similar performance as a single module (row 3), both in AuC and in final performance. This is expected, as the two variants are similar from the policy head's perspective. However, learned fusion yields significant gains in AuC over a single module, *i.e.* softmax fusion (row 5) and attentive fusion (row 6) achieve $+0.04$ SPL over single module (row 3).

Though these models have similar metrics 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. 7 - both fusion methods match the initial pace of CPC|A-4 (a single task), improving over CPC|A-{1-16}+ID+TD: Single.

The choice of fusion method does not seem to matter much. The more expressive fusion method – dot-product attention – does equally well as softmax fusion in best performance (row 6 vs. 5 in Table

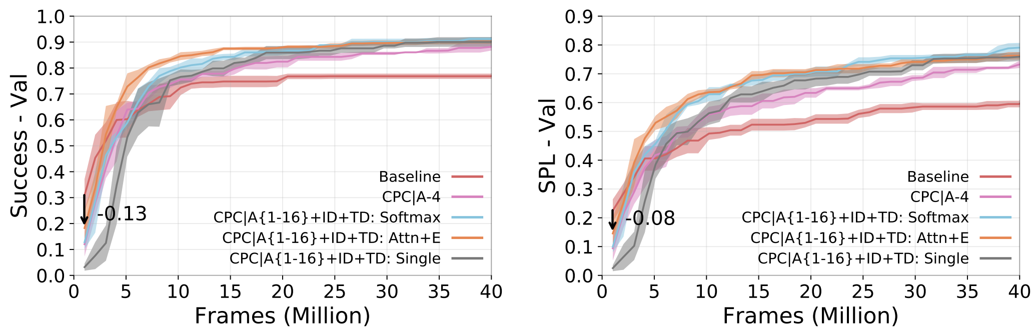


**Figure 7.** Learning fusion of separate modules in CPC|A-{1-16}+ID+TD: Attn+E recovers the initial ramp-up cost experienced by CPC|A-{1-16}+ID+TD: Single.

4). This trend may only hold in PointNav, a relatively simple navigation task. However, the Attn+E variant improves over softmax on AuC. As shown in Fig. 7, this gain is largely in the initial learning phase, where the Attn+E variant largely mitigates the early sample inefficiency of using multiple auxiliary tasks seen in CPC|A-{1-16}+ID+TD: Single.

**Addressing peaky attention.** In our experiments, we found that the variants using attention quickly (in $< 0.5$M frames) start attending to just one belief module, preventing the other belief modules from influencing the policy. This collapsed attention implies the attention variant's improvements 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 9.1 for details). With the penalty, the agent consistently attends to all its belief modules, resulting in row 7, the Attn+E variant. While adding this entropy term hurts final performance a little, the benefits to AuC in early learning suggest this variant is worth investigating.

## 5.4 Comparison with pre-trained weights

We also compare with an agent using Taskonomy [33] weights for its visual encoder, as proposed by Sax et al. [34]. We use representations learned from the depth prediction task, shown to be a highly transferable representation for PointNav [34]. We compare results with and without finetuning the compression layer and final ResNet block in Table 5.

Contrary to the results in [34], we find that both variants do not outperform our from scratch baseline. This has two primary causes. First, our baseline is more competitive. [34] used a simple 3-layer CNN [46] for their from scratch baseline while ours uses ResNet18. ResNet18 improves performance for PointNav RGB agents considerably. For example, the Habitat baselines repository [39] uses the same 3-layer CNN, and our baseline improves upon those results by 0.1 SPL at 40M. Second, our Taskonomy transfer results are lower than that of [34] due to three differences in training procedure and agent design. [34] 1) use an off-policy version of PPO, which trades compute for sample efficiency, 2) do not learn stop, which makes the task slightly easier, and 3) provide the agent with a bitmap of previously visited locations instead of a GRU. This finding strengthens the case for training representations from scratch rather than transferring from static visual tasks.

| | Area under Curve | | Best | |
|---|---|---|---|---|
| | Success (↑) | SPL (↑) | Success (↑) | SPL (↑) |
| 1) Baseline | $0.602{\pm}0.074$ | $0.441{\pm}0.054$ | $0.776{\pm}0.014$ | $0.612{\pm}0.022$ |
| 2) Depth (Fine-tuning) | $0.584{\pm}0.042$ | $0.454{\pm}0.107$ | $0.766{\pm}0.017$ | $0.611{\pm}0.009$ |
| 3) Depth (Frozen) | $0.584{\pm}0.050$ | $0.451{\pm}0.044$ | $0.770{\pm}0.029$ | $0.619{\pm}0.028$ |

**Table 5.** With an improved visual encoder and without access to a bitmap of past locations [34], a baseline agent training a visual encoder from scratch performs comparably with an agent using Taskonomy depth weights, with and without fine-tuning.

| | Area under Curve | | Best | |
|---|---|---|---|---|
| | Success (↑) | SPL (↑) | Success (↑) | SPL (↑) |
| 1) Baseline | $0.602_{\pm 0.074}$ | $0.441_{\pm 0.054}$ | $0.776_{\pm 0.014}$ | $0.612_{\pm 0.022}$ |
| 2) CPC\|A-4 (Best Individual) | $0.725_{\pm 0.032}$ | $0.554_{\pm 0.034}$ | $0.883_{\pm 0.013}$ | $0.739_{\pm 0.011}$ |
| 3) CPC\|A-{1-16}: Single | $0.704_{\pm 0.038}$ | $0.563_{\pm 0.034}$ | $0.913_{\pm 0.008}$ | $0.760_{\pm 0.010}$ |
| 4) CPC\|A-{1-16}+ID+TD: Attn+E | $0.794_{\pm 0.024}$ | $0.631_{\pm 0.025}$ | $\mathbf{0.905_{\pm 0.005}}$ | $\mathbf{0.770_{\pm 0.003}}$ |
| 5) CPC\|A-{1-16}: Attn | $0.751_{\pm 0.030}$ | $0.599_{\pm 0.030}$ | $0.909_{\pm 0.007}$ | $0.779_{\pm 0.014}$ |
| 6) CPC\|A-{1-16}: Attn+E | $0.789_{\pm 0.026}$ | $0.629_{\pm 0.027}$ | $\mathbf{0.911_{\pm 0.005}}$ | $\mathbf{0.777_{\pm 0.006}}$ |
| 7) CPC\|A-{1-16}: Fixed Attn | $0.733_{\pm 0.034}$ | $0.585_{\pm 0.033}$ | $\mathbf{0.908_{\pm 0.008}}$ | $0.768_{\pm 0.004}$ |
| 8) CPC\|A-16×5: Attn | $0.737_{\pm 0.034}$ | $0.591_{\pm 0.029}$ | $0.902_{\pm 0.007}$ | $0.752_{\pm 0.009}$ |

**Table 6.** Performance of attentive fusion and ablations on CPC|A family. CPC|A-{1-16}: Attn+E provides the same benefits as CPC|A-{1-16}+ID+TD: Attn+E. All variants continue to converge to similar metrics at 40M observations.

## 6 Model Analysis

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

1. We verify improvements when attention is largely fixed on one belief module to be due to improvements in visual representation. To show this, we artificially fix 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? To answer this, 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.

Our results are summarized in Table 6, and we list main findings below.

**Improvements are partly due to better visual representations.** We find that benefits of multiple auxiliary tasks are partly due to improvement in visual representations (Table 6, row 7 and 5). This mirrors the findings of Sax et al. [34], who use visual encoders pretrained on mid-level vision tasks (*e.g.* 3D curvature prediction) to improve sample efficiency. There's a slight gain in AuC by allowing entropy rather than fixing it to 0.

Also, CPC|A-{1-16}: Attn+E (row 6) again improves on CPC|A-{1-16}: Single in Success AuC, a +12% gain. This means **separate belief modules help**, even if auxiliary tasks are from the same family. Moreover, comparing CPC|A-{1-16}: Attn+E to CPC|A-{1-16}+ID+TD: Attn+E (row 4), we maintain equivalent performance. That is, despite ID and TD introducing gains when the tasks were naively combined( Table 4.1), learned fusion does **not** find independent gains.

Finally, **similar tasks are only marginally better than identical tasks** on average. We find that a variant using correlated but distinct tasks (row 4) performs only slightly better than applying the same auxiliary task to all belief modules (CPC|A-{1-16}: Attn *vs*. CPC|A-16×5: Attn in Table 6 offer no advantages). We believe this to be due to the general worse performance of CPC|A-{1,2}.

### 6.1 Examining usage of belief modules

Though different tasks lead to quantitatively different performances, that does not determine whether they induce characteristic "beliefs" in their modules. To understand whether different auxiliary tasks assist in learning "expert" behavior in the respective belief modules they operate on, we study each task's induced behavior and the situations where each belief module has maximum attention. We perform this analysis with different runs of one of our best models – CPC|A-{1-16}: Attn+E at 40M.

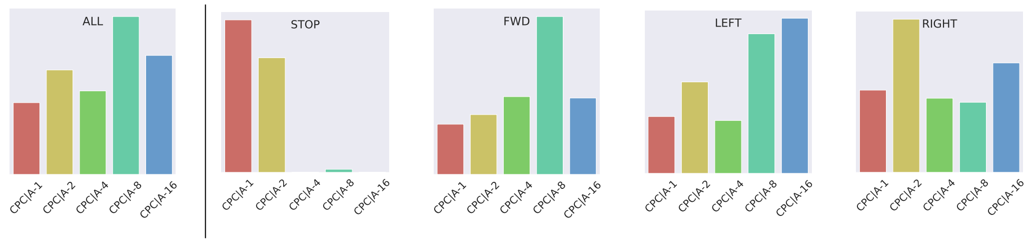


**Figure 8.** Distribution of which auxiliary tasks are most attended to while taking each action for CPC|A-{1-16}: Attn+E. CPC|A-1 and CPC|A-2 significantly affect the STOP action

| | | Control | CPC\|A-1 | CPC\|A-2 | CPC\|A-4 | CPC\|A-8 | CPC\|A-16 | CPC\|A-{1,2} |
|---|---|---|---|---|---|---|---|---|
| Masked out | Success | 0.911 | **0.416** | 0.815 | 0.901 | 0.878 | 0.906 | 0.058 |
| | SPL | 0.777 | **0.224** | 0.711 | 0.800 | 0.772 | 0.793 | 0.012 |
| All others masked out | Success | 0.911 | **0.419** | 0.074 | 0.043 | 0.078 | 0.005 | 0.772 |
| | SPL | 0.777 | **0.320** | 0.042 | 0.020 | 0.054 | 0.002 | 0.690 |

**Table 7.** Masking CPC|A-{1-16}: Attn+E modules. CPC|A-1, CPC|A-2, associated with stopping are critical.

Fig. 8 shows how different auxiliary task attentions correlate with the action to be taken. 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.

The STOP action almost always corresponds with attention to CPC|A-1 and CPC|A-2, which might suggest they play an important role in short-term decisions. However, when applying the same analysis to other runs of the same variant, we find that they can rely on different modules (*e.g.* the one corresponding to CPC|A-4) for the same stop decision. Additionally, LEFT prefers to attend to the belief of CPC|A-16, but not RIGHT. Attention over different belief modules does induce some functional expertise in POINTNAV, but auxiliary tasks alone do not make it do so consistently. Additional plots are in the appendix.

Next, to quantify the contribution of each belief module to overall performance, we selectively mask out belief modules when computing attention. We continue to use the CPC|A-{1-16}: Attn+E variant. This experiment is similar to occluding parts of images to identify which features play a causal role in predictions from classification CNNs [47]. First, we mask out individual belief modules (Table 7 "Masked out") and find that for most belief modules, the agent is unaffected by their exclusion. However, when CPC|A-1 or CPC|A-2 are excluded, the agent's performance drops dramatically below baseline results (0.416 *vs.* 0.911 on Success with and without masked CPC|A-1). This is consistent with the previous analysis that the agent learns heavy dependence on CPC|A-1 and CPC|A-2 for stopping, a critical task behavior. When both CPC|A-1 and CPC|A-2 are excluded (rightmost column in Table 7), the agent is unable to recover. The lack of performance drop when masking other modules may indicate that they all offer reasonable general navigation representations. This would support the earlier hypothesis that agent gains are largely in improved visual representation rather than improvements in the belief modules. Nevertheless, the agent does not stop successfully without specific modules.

Additionally, we conduct a diagnostic where we mask out all belief modules except one – Table 7, "All others masked out". Most settings show immense drop in performance, suggesting that individual belief modules are largely not useful by themselves. However,the CPC|A-1 module retains reasonable performance in isolation (0.419 Success vs 0.911 Success of the baseline), confirming that it is an essential module. As a converse to the previous conclusion, providing the agent access to two essential belief modules from the previous experiment, CPC|A-1 and CPC|A-2, is sufficient to regain most of the agent's performance.

**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. 9. We run this with our most sample efficient model, CPC|A-{1-16}+ID+TD: Attn+E. We randomly sample 200 spawn locations, make the agent navigate to a single fixed goal location, and color-code trajectories according to the auxiliary task belief module maximally attended to.

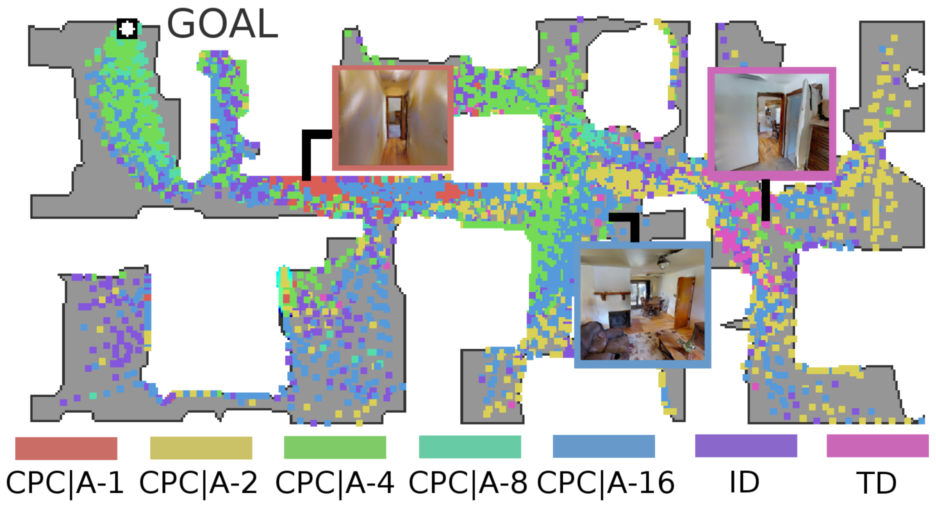


**Figure 9.** Top-down map of the Cantwell scene in the Gibson dataset [30]. Colored boxes represent the auxiliary task with maximum attention at the given location. We see a natural clustering of auxiliary tasks based on where the agent is in the environment.

Interestingly, the visualization exhibits natural clustering, which suggests that the agent is able to recognize patterns in its observations and rely on specific belief modules depending on where it is in the environment. This location-characterized activation is reminiscent of the behavior of 'place cells' neuroscientists have discovered in rats navigating mazes [48]. We conjecture that these characteristic distributions emerge naturally, analogous to the emergence of specialized kernels in CNNs. Indeed, we run a variant with separate belief modules without any auxiliary tasks and again observe specialized distributions. Instead of providing fixed specialized representations as in [35], we see from-scratch training can learn specialized features.

## 7 Conclusion

We have shown that auxiliary tasks can greatly accelerate learning in PointNav. We systematically disentangle the 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 $0.61$ SPL in $8$M observations – $5\times$ better in sample efficiency over our baseline. This speedup suggests auxiliary tasks can be key in training embodied agents in complex environments from scratch within reasonable computation budgets.

Our analysis further reveals insightful task and data relationships – attention extracts information from CPC|A-$\{1\text{-}16\}$ that renders ID and TD unhelpful, even though the tasks provide orthogonal gains when naively combined. Further, attention agents learn to specialize their modules – CPC|A-1 and CPC|A-2 are largely responsible for driving stopping behavior. In future work, we aim to study this relationship in more detail, and explore auxiliary objectives for other embodied tasks (*e.g.* language-driven navigation [49], question-answering [27, 50], *etc.*).

## 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, 9

[2] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 2

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

[4] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2

[5] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016. 2

[6] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 2

[7] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2

[8] 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. 2, 5

[9] O. J. Hénaff, A. Razavi, C. Doersch, S. Eslami, and A. v. d. Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019. 2

[10] P. Bachman, R. D. Hjelm, and W. Buchwalter. Learning representations by maximizing mutual information across views. In *NeurIPS*, 2019. 2

[11] Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 2

[12] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019. 2

[13] I. Misra and L. van der Maaten. Self-supervised learning of pretext-invariant representations. *arXiv preprint arXiv:1912.01991*, 2019. 2

[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 2

[15] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 2

[16] 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. 2

[17] 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, 3

[18] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017. 2, 6

[19] 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, 3, 8

[20] 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, 3

[21] X. Lin, H. Baweja, G. Kantor, and D. Held. Adaptive auxiliary task weighting for reinforcement learning. In *NIPS*. 2019. 2, 3

[22] 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, 3, 5

[23] 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, 3

[24] D. Gordon, A. Kadian, D. Parikh, J. Hoffman, and D. Batra. Splitnet: Sim2sim and task2task transfer for embodied visual navigation. In *ICCV*, 2019. 2, 3

[25] A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020. 2

[26] 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. 2

[27] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied Question Answering. In *CVPR*, 2018. 2, 13

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

[29] E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. Batra. Embodied Question Answering in Photorealistic Environments with Point Cloud Perception. In *CVPR*, 2019. 2

[30] 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`. 3, 4, 13

[31] 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. 3

[32] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637, 2005. 3

[33] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018. 3, 10

[34] A. Sax, B. Emi, A. R. Zamir, L. Guibas, S. Savarese, and J. Malik. Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies. *arXiv preprint arXiv:1812.11971*, 2018. 3, 10, 11

[35] 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, 6, 9, 13

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017. 3, 6

[37] D. S. Chaplot, S. Gupta, D. Gandhi, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural mapping. In *ICLR*, 2020. 3, 5

[38] 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, 4

[39] 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. 4, 10

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

[41] 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

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4

[43] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016. 4

[44] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 4

[45] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. In *ICLR*, 2018. 5

[46] 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. 10

[47] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 12

[48] N. Burgess. The 2014 nobel prize in physiology or medicine: a spatial model for cognitive neuroscience. *Neuron*, 2014. 13

[49] 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. 13

[50] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. IQA: Visual Question Answering in Interactive Environments. In *CVPR*, 2018. 13

# 9 Appendix

In this supplement we will describe the architectural specifics and provide a few additional figures. Randomly sampled agent trajectories from the baseline and CPC|A-{1-16}+ID+TD: Attn+E variants are attached with this submission, which demonstrate a variety of shared weaknesses and different behaviors.

## 9.1 Training Details

No hyperparameter sweeps were done. Model sizes were all $5.7 \pm .3$ 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 $\approx 256 - 288$.

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:

$$L_{\text{total}}(\theta_m;\theta_a) = L_{\text{RL}}(\theta_m) - \alpha H_{action}(\theta) + L_{\text{Aux}}(\theta_m;\theta_a) \tag{4}$$

$$L_{\text{Aux}}(\theta_m;\theta_a) = \sum_{i=1}^{n_{\text{Aux}}} \beta^i L^i_{\text{Aux}}(\theta_m;\theta^i_a) - \mu H_{attn}(\theta_m) \tag{5}$$

$H_{attn}$ 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 $\beta^i$ 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. 10. 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:

$$\text{Rollout Workers: } n = 4 \tag{6}$$

$$\text{Rollout Length: } t = 128 \tag{7}$$

$$\text{PPO Epochs} = 4 \tag{8}$$

$$\text{PPO Mini-batches} = 2 \tag{9}$$

$$\gamma = 0.99 \tag{10}$$

$$\tau = 0.95 \tag{11}$$

$$\epsilon = 0.1 \tag{12}$$

$$\text{lr} = 2.5 \times 10^{-4} \tag{13}$$

$$\text{Gradient Norm Cap} = 0.5 \tag{14}$$

$$\text{PPO Clip} = 0.1 \tag{15}$$

$$\tag{16}$$

## 9.2 Auxiliary Task Implementation Details

The full implementation will be publicly released. More detailed descriptions and loss equations are provided below.

### 9.2.1 Inverse Dynamics

We take the visual embeddings from $1 - T$, trim the final timestep to form the "before" batch, and the first timestep to from the "after batch. We then concatenate each timestep $t$ and $t + 1$ pair with

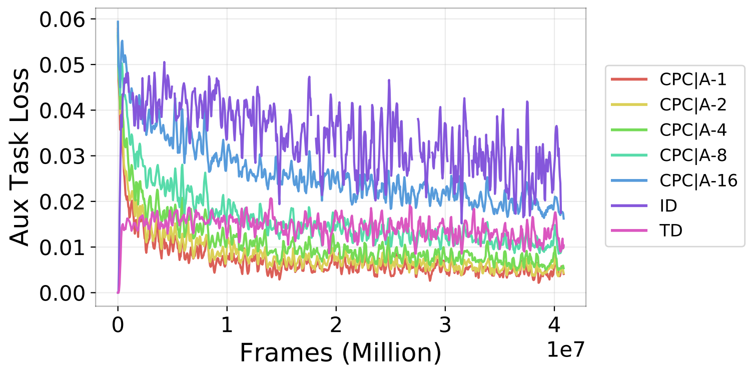


**Figure 10.** Auxiliary Task loss curves for CPC|A-1-16+ID+TD: Attn+E.

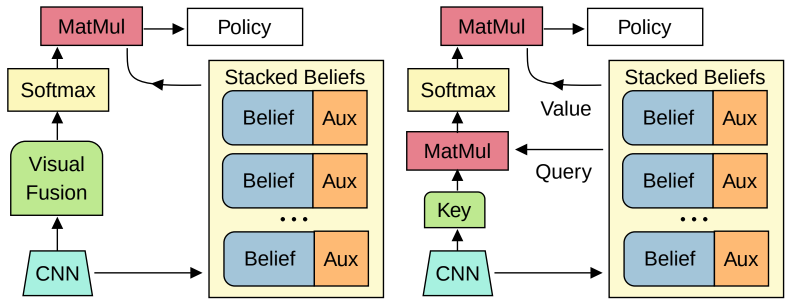


**Figure 11.** Left: Dot-product attention, Right: Softmax gating.

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.

$$L_{ID} = \sum_{i=1}^{T-1} L_{CE}(I(\phi_i, \phi_{i+1}, h_T), a_i) \tag{17}$$

### 9.2.2 Temporal Distance

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, (*i.e.* $h_T$), and directly use a linear layer to predict the timestep difference between each pair.

$$L_{TD} = \frac{1}{2}((i-j) - \mathcal{T}(\phi_i, \phi_j, h_T))^2 \tag{18}$$

### 9.2.3 CPC|A

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 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.

$$L_{CPC|A} = \sum_{k=1}^{K} \sum_{t=1}^{T-k} L_{CE}(c(g_k^t, \phi^-), 0) + L_{CE}(c(g_k^t, \phi_{t+k}), 1) \tag{19}$$

## 9.3 Module Fusion Implementation Details

All fusion methods are achieved by weighted sum. Fixed and average fusion are achieved by freezing weights as desired. Weight calculation for softmax gating and attention are described by Fig. 11,

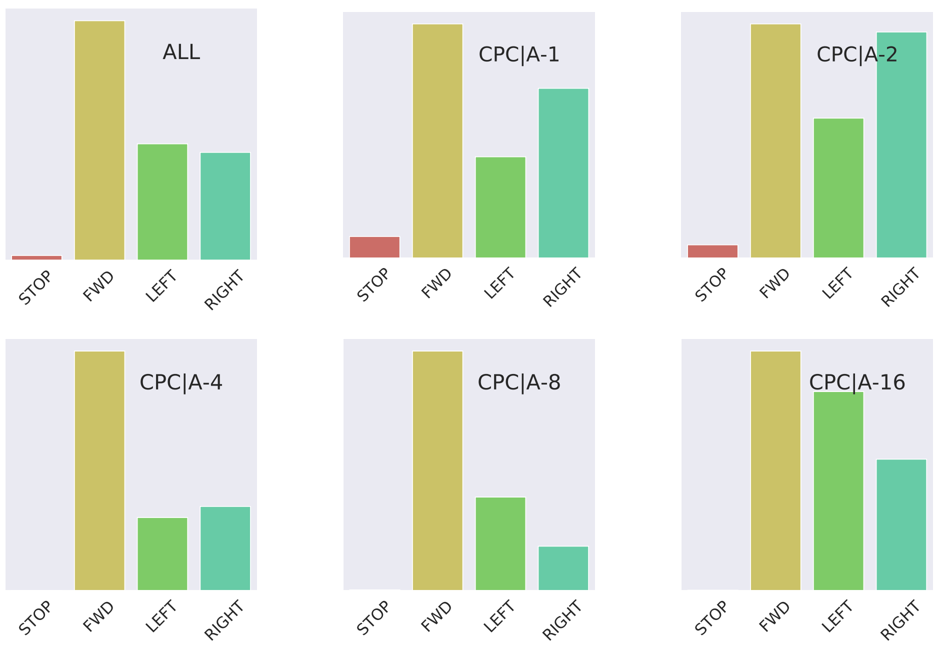


**Figure 12.** Action distributions on CPC|A-{1-16}: Attn+E validation episodes, for steps where a given auxiliary task is used (weight $> 0.25$). Again, the STOP action reliably activates CPC|A-1, CPC|A-2

re-printed for convenience. 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 final weights.

## 9.4 Additional Figures

**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. 12) by thresholding all actions taken with conditioned task weight $> 0.25$. The same STOP bias towards CPC|A-1, CPC|A-2 is reflected.

**Training and Validation** Though validation scores are roughly on par with training scores as the agent learns basic navigational abilities (*i.e.* discerning clear walls), agents overfit the training scenes after a few million frames. Training and validation curves are shown for the baseline and CPC|A-{1-16} to demonstrate our proposed architecture does not affect this.

**Additional Top Down Map Visualizations**

We also provide top down map visualizations (Fig. 14) 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.

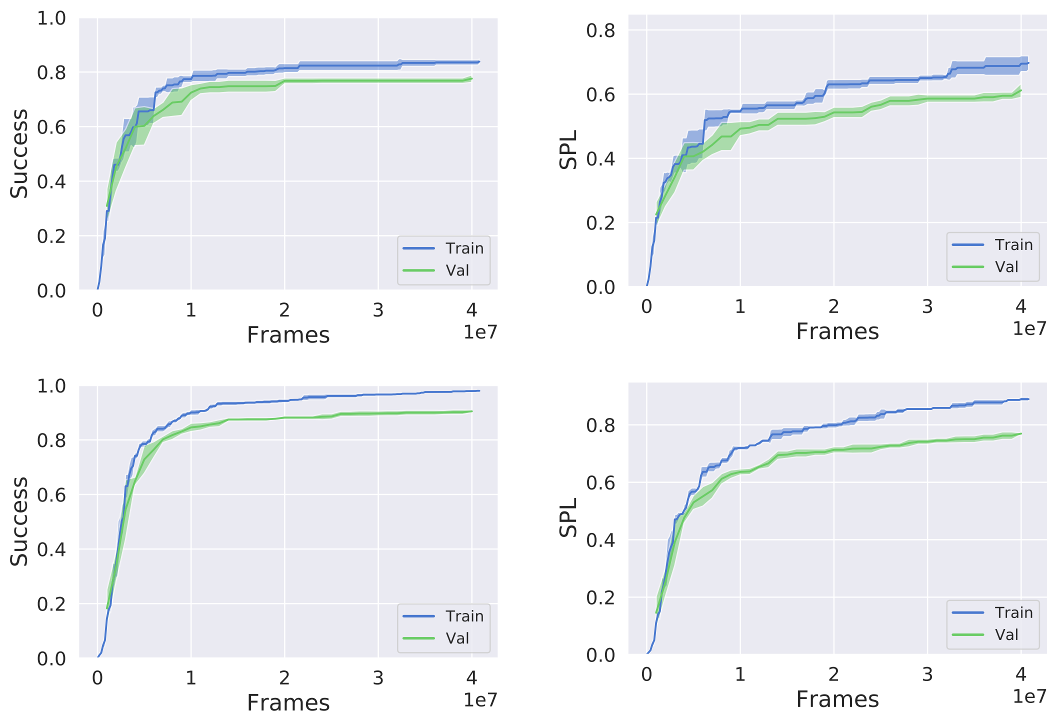


**Figure 13.** Training and validation curves for the baseline (top) and CPC|A-{1-16}+ID+TD: Attn+E (bottom) (Window-averaged, $w = 3$). Overfitting begins at 0.5 SPL, or 0.66 Success

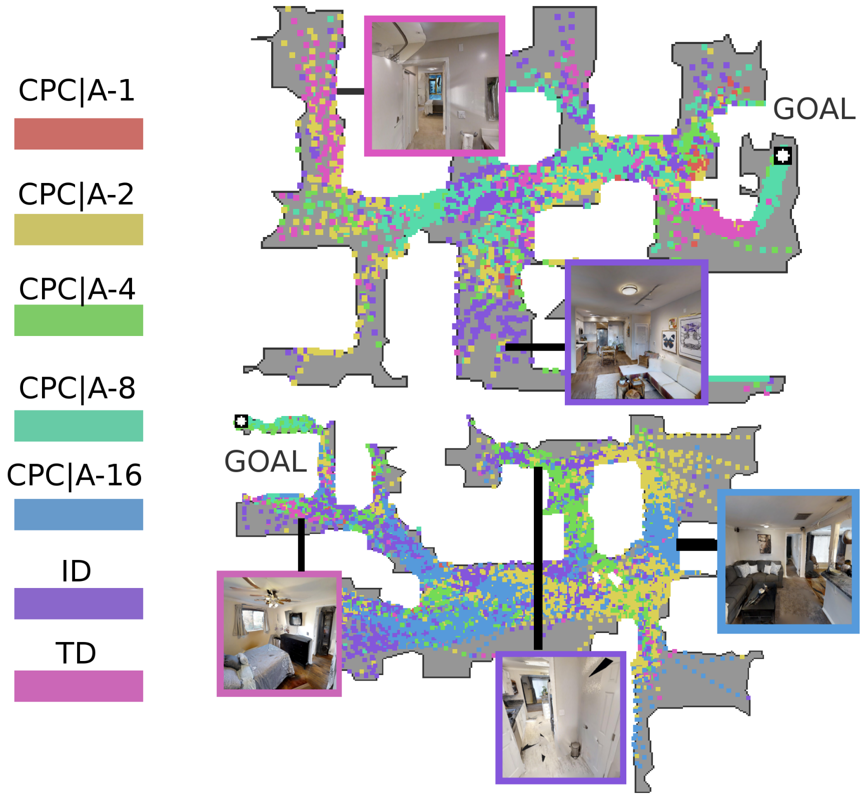


**Figure 14.** Additional location-based attention weighting visualization. Top: Quantico scene. Bottom: Eastville scene.