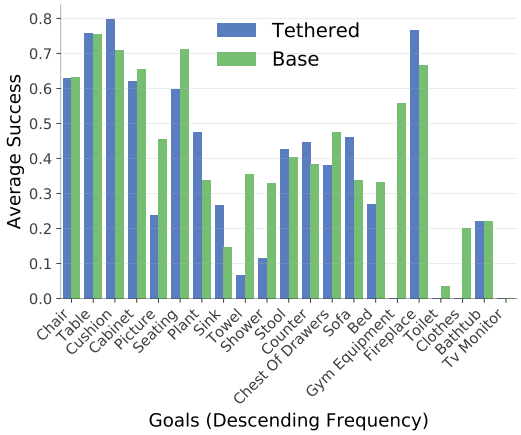

Figure A1. While the base agent succeeds on all tasks but "TV Monitor", the tethered agent struggles to solve 5 categories.

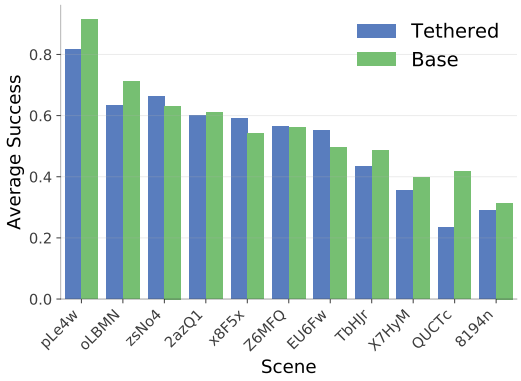

Figure A2. While scene difficulty varies, difficulty distribution is not extremely heavy-tailed. TEST-STD shift could result from 1-2 more difficult scenes.

A. Appendix

We use the appendix to elaborate on agent behavior and methodology.

A.1. Agent Performance against goal distance, category, and scene

Our experiments show large performance variance across OBJECTNAV splits. We check for anomalous agent biases by comparing agent success rates conditioned on several episode attributes. Since the large gap between GT and RedNet performance is already established, we provide these plots with GT segmentation.

The base agent has reasonable performance variance across all categories, succeeding less often on rarer categories (Fig. A1) and more distant goals (Fig. A3). Scene variance (Fig. A2) is wide (0.3 to 0.85), but not particularly heavy-tailed; the TEST-STD drop is likely simply due to increased difficulty. This variance blurs the line that defines state of the art and greatly motivates increasing current OBJECTNAV

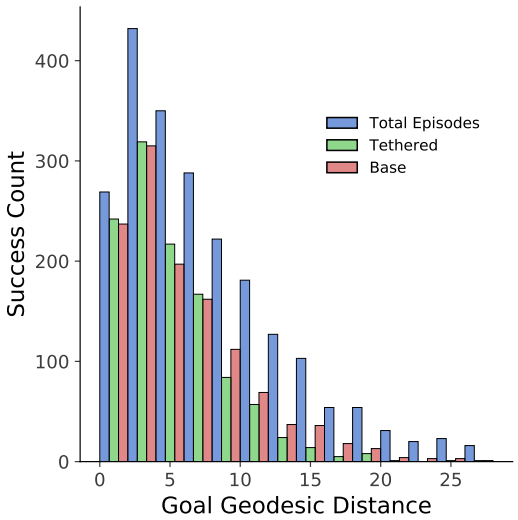

Figure A3. Agent success decreases when goals become more distant, and tethered agent is relatively worse at more distant goals.

dataset sizes.

We also provide the tethered agent's performance; they preferentially succeed on shorter paths relative to the base agent, which can succeed on even very distant goals Fig. A3.

A.2. Sparse Reward Reduces Exploration and Causes Quitting

The tether agent performs worse than the base agent. We note two primary reasons in Fig. A4: reduced exploration (top) and early quitting due to estimation error (bottom). In particular the early quitting is likely a general symptom of using sparse reward. On episodes that are difficult, either due to environmental characteristics (*e.g.* it is outside) or due to goals (*e.g.* clothes, which is difficult at train time), value estimates will occasionally dip below 0 and cause the agent to randomly stop. This may be mitigated by an increased discount factor.

A.3. Train-Val Gap

Though Table 3 suggests virtually no overfitting, our agent does appear to begin moderately overfitting past 0.5 success in Fig. A5.

A.4. 2D vs 3D GPS Comparison

Though we train with the 3D GPS sensor provided by the simulator, all presented results have used a 2D GPS without the vertical dimension at evaluation, to be consistent with Habitat Challenge settings. We compare 2D vs 3D validation scores in Table A1. We do not see a large change in performance; our agents dos not leverage the vertical dimension much.

Figure A4. Top: We plot exploration rate (coverage over time) for tether and base agents. In both success and failure conditions, the exploration rate is worse in the tethered agent. Bottom: We plot value function traces for 100 failure episodes. Tether value predictions approaches 0 quickly, but due to estimation error also dips below 0.

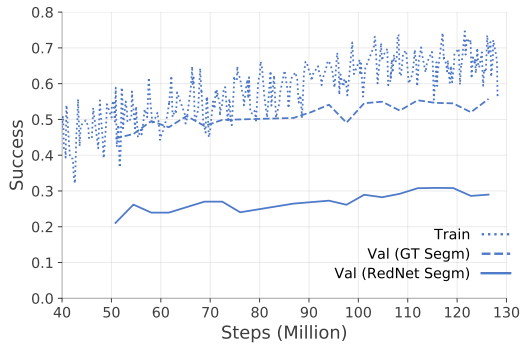

Figure A5. Base 6 Action Agent, training and validation curves. Note: Training curve is taken directly from training logs, *i.e.* it evaluates statistics on the rollout and not on the whole training set.

A.5. Additional Auxiliary Task Ablations

We provide 2 additional ablations: 1. removing SGE by turning it into a feature and 2. removing all auxiliary tasks (and using one large RNN). Results are in Table A2.

SGE works better as a feature than as an auxiliary task (row 2 vs 3). Since SGE is a feature that is easily derived from the semantic input, its large contribution to learning efficiency may feel unintuitive. However, when we encourage SGE by decoding it with an auxiliary task, performance

	Success % (↑)	SPL % (↑)
1) 4-Action	34.4(33.6)	9.58(9.52)
2) 6-Action	30.8(30.4)	7.60(7.28)
3) 6-Action + Tether	26.6(27.5)	9.79(9.23)

Table A1. Comparing VAL scores when providing vertical dimension in GPS. Formatted as 2D (3D). There is little difference in performance.

	Success % (↑)	SPL % (↑)
1) 4-Action	34.4±2.0	9.58±0.75
2) - SGE	20.3±1.7	4.14±0.47
3) Aux SGE	17.7±1.6	4.03±0.48
4) No Aux	30.4±2.0	6.56±0.57

Table A2. Additional ablations on 4-action base for 1. incorporating SGE as an auxiliary task (Aux SGE) and 2. not providing any auxiliary tasks (No Aux).

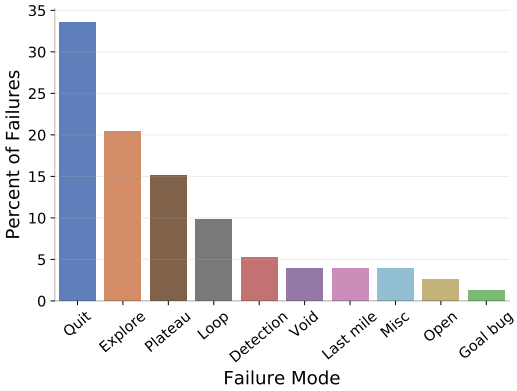

Figure A6. We provide breakdown of tether failure modes.

counterintuitively decreases. The auxiliary task is easily learned, i.e. its loss flatlines near 0 quickly, and module weights suggest this content is stored in sparsely (*i.e.* < 10 peaks in the probe weights). This result may warrant additional investigation to understand what types of priors are best introduced as auxiliary tasks *vs*. as features.

Auxiliary tasks contribute to performance (row 1 vs 4). If we remove all auxiliary tasks, performance drops moderately.

A.6. Behavioral Analysis Details

We provide a failure mode breakdown for the tethered agent (Fig. A6). The tethered agent fails minimally from exploration-reward specific failures (*e.g.* "Commitment"),

Name	Description	Rule
Plateau	Repeated collisions against the same piece of debris causes a plateau in coverage. Includes debris which traps agent in spawn.	Agent collides > 50 times before visiting 2 new voxels.
Loop	Poor exploration due to looping over the same locations or backtracking.	Episode > 250 steps, expected fraction of episode spent in current voxel > 0.15 and collisions < 50.
Detection	Despite positive SGE, the agent does not notice nor successfully navigate to the goal.	Max SGE seen in episode $\in [0.02, 0.1]$.
Commitment	Sees and approaches the goal but passes it.	In first 200 steps, SGE > 0.1 and goal distance < 1.0.
Last mile	Gets stuck near the goal.	On stop, SGE > 0 and distance is $< 1m$.
Open	Explores an open area without any objects.	Coverage > 10 voxels and < 10 collisions.
Quit	Quitting despite lack of obstacles.	Apply if above rules do not apply and episode length < 250.
Void	The goal is seen through a crack in the mesh, which appears to disturb agent behavior.	Qualitative.
Goal Bug	A goal instance has no associated viewpoints where the agent can stop.	Qualitative.
Explore	A generic failure to find the goal despite steady exploration. Includes semantic failures *e.g.* going outdoors to find a bed.	The default if no other modes apply.

Table A3. Description of observed failure modes for 6-action agents and associated heuristics.

but qualitatively fails from a "Quitting" mode often, where the agent stops despite reasonable exploration and no goal in sight. We also provide a full list of observed failure modes (Table A3), along with quantitative heuristics which is consistent with manual annotations $> 80\%$ of the time, to help give a sense of the annotation criteria.

A.7. Action Entropy to quantify agent instability.

Throughout our analysis we have referred to unstable agent behavior. We quantify this instability by measuring action entropy, *i.e.* the entropy of the distribution of actions taken over the course of each episode. Specifically, we measure for each episode the action entropy of a rolling window of 10 steps, averaged across window locations, and show how its distribution differs between agents and episode success or failure in Fig. A7.

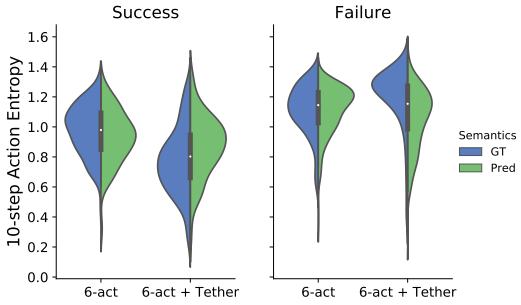

Figure A7. We plot the average action entropy of a rolling window as measured on 300 VAL episodes, for 2 agents and with and without GT segm. Failures tend to correlate with higher action entropy, more strongly for the tethered agent. Predicted segm. pulls action entropy toward an intermediate value for the tethered agent *i.e.* destabilizes successful episodes, and stabilizes failure episodes. GT and predicted segmentation effects are muted for 6-action agent, perhaps due to its wandering behavior.

A.8. Additional Probing Results

We additionally run probes for time spent in location (as in the "loop" failure mode), room ID, and distance to goal, as shown in Fig. A8. These features appear negligibly represented across beliefs.

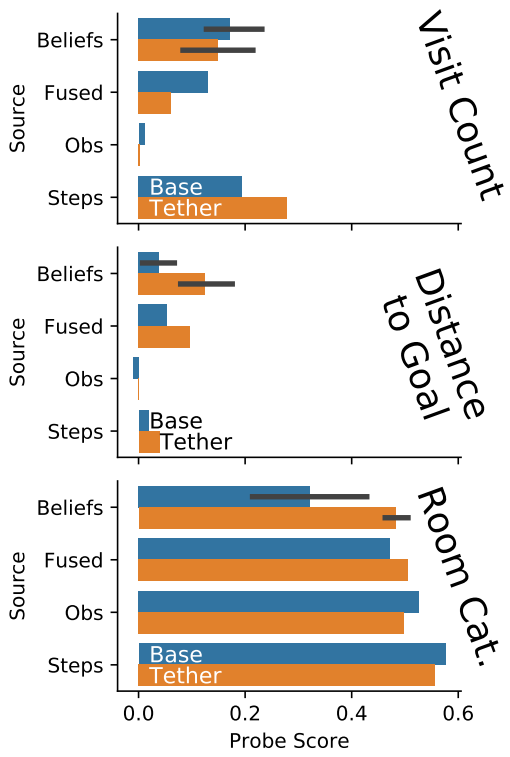

Figure A8. We run the same probing procedure for visit count (number of timesteps spent in current voxel), distance to goal, room category (room category reports classification accuracy instead of R^2). Visit count and room category is better matched by timestep than any belief, and distance to goal R^2 is at best around 0.1.

A.9. Curvature Computation

We use curvature to summarize the stability of a given sequence of representations $x_1 \dots x_t$. We start by calculating normalized displacement vectors $v_i = x_{i+1} - x_i, \hat{v_i} = \frac{v_i}{||v_i||}$. We compute the local discrete curvature as the angle between successive vectors: $c_i = \arccos(\hat{v_i}, \hat{v_{i+1}})$. Then we report the global curvature of the sequence as the mean of these local curvatures. This procedure mirrors [16]. Global curvatures reported in text are averaged over validation episodes.

A.10. Fixed Point Analysis Methodology

Fixed point (FP) analysis of RNNs center around the idea that an RNN's nonlinear dynamics and computation can be understood through its behavior around a set of "slow (fixed) points." These points act as a dynamical skeleton which *e.g.* determines the flow field at other points in hidden space. For example, it has been shown that sentiment analysis RNNs act as line attractors, where the hidden state's position along a line represents the read out sentiment [21].

FP analysis begins with an optimization to identify these slow points, *i.e.* hidden states that satisfy $||\text{RNN}(h, \phi) - h||^2 < \epsilon$, for some input ϕ and threshold ϵ. For this initial analysis we follow [22] and let ϕ be the average input, *i.e.* the average of observation embeddings collected across the 300-episode VAL subset. Note that this average includes an SGE > 0 signal; in case this has large impact on belief dynamics, we verified that fixed point results were qualitatively similar after setting this signal to 0. We optimize 10K fixed points, by sampling 10K random hidden states experienced in the 300-episode subset and performing gradient descent.

We optimize fixed points for each belief, on each agent; by optimization termination most points have an RNN update norm $< 1 \times 10^{-6}$, though we take a subset with norms $< 4 \times 10^{-7}$. From this population of slow points we sample 50 points for Fig. 5. Then, we compute the Jacobian of the RNN update with respect to the hidden state, and compute the Jacobian's eigendecomposition. The Jacobian will likely have complex eigenvalues; to convert these into a term representing memory, we compute a time constant (as in [22]:

$$\tau(\lambda) = |\frac{1}{\log ||\lambda||}| \tag{10}$$

We show an example eigenvalue spectrum and associated time constants in Fig. A9. There does not appear to be large variation in spectra across fixed points.

A.11. High Dimensional Computation in RNNs

Prior works find FPs to be arranged in low-dimensional manifolds and are able to directly link RNN memory structure to, *e.g.*, ring attractors and simplicial structures. We find that the RNN FP subspace for OBJECTNAV is high-dimensional, as measured *e.g.* by the participation ratio of the fixed point subspace. We believe this is due to 1) OBJECTNAV being considerably more complex than previously studied tasks, and 2) unconverged training. This high-dimensionality makes it is difficult to clearly link agent beliefs to any known attractors, *e.g.* inputs project to > 3 dimensions. Though we observe certain behavioral phenomena that we expect to be reflected in RNN dynamics, *e.g.* the agent will do a near 360° when it is blocked, we find this difficult to visualize. We show examples fixed point subspaces for 3 beliefs in Fig. A10. These layouts are not qualitatively consistent across agents, nor beliefs, even though memory structure is.

While 20D fixed point subspaces do frustrate current techniques, we could alternately be surprised that the RNN which has 196 dimensions has such a relatively low-dimensional subspace. It would be valuable to both the vision community and beyond to understand how to scale up this technique to a moderately higher number of dimensions.

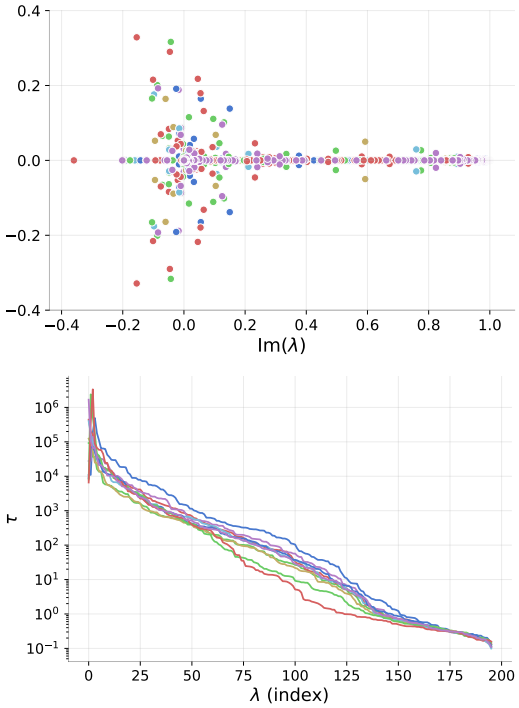

Figure A9. As an example, for the Jacobians of 10 sample fixed points from the PBL belief of the tether agent, we plot Top: their eigenvalues, Bottom: the associated time constants. Different fixed points have different colors.

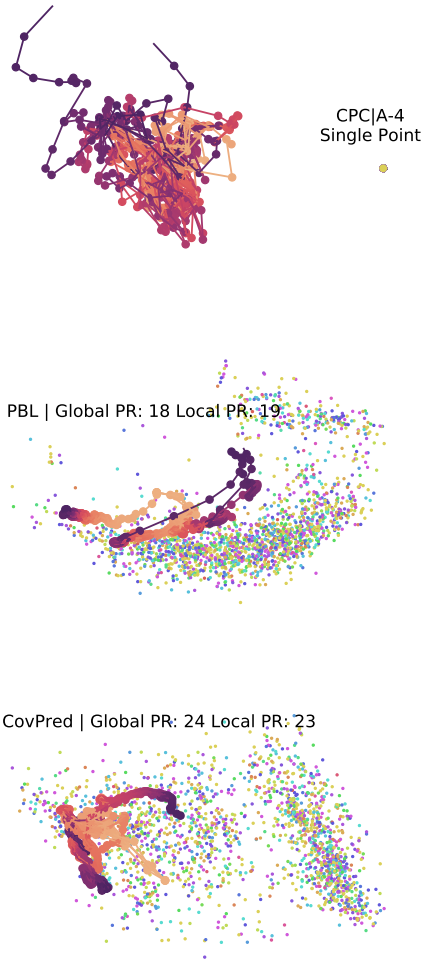

Figure A10. For all beliefs of the tether agent, we plot fixed points (colored by goal category of the state that the fixed point was initialized at) along with sample hidden state trajectories (connected with lines). Plots are in the top-2 PCs of the hidden states. Layouts are different across beliefs, but the top-2 PCs only account for a moderate amount of variance among the fixed points. The RNN corresponding to CPC|A-4 only has 1 unique fixed point, but it is not solely an attractor.

A.12. Agent Embedding Visualization

A common way to gain intuition for agent knowledge is by examining agent embedding layers. Our agent incorporates two semantic embedding layers, one which embeds the semantic frame (for the ResNet), and one which embeds the semantic goal; we visualize the goal embedding with PCA in Fig. A11. It is difficult to draw falsifiable conclusions about these embeddings. For example, though there are promising clusters of (table, chair, cushion) and (toilet, sink, bathtub, shower) goals, this is also confounded as those same categories tend to have high average success or failure, respectively; *i.e.* instead of room semantics, the embeddings may simply imply goal difficulty. We do not show the semantic channel embedding as there does not appear to be any meaningful arrangement in PC-space.

A.13. Tethered Policy Updates

We tether a second policy to the acting policy by sharing a base network (we only split policies at the linear actor-critic heads) and training the second policy with a different reward. The first policy is naturally off-policy for the second policy; we thus incorporate V-Trace-like [10] off-policy importance weighting terms into the standard on-policy PPO update. Specifically, we 1. replace the tethered policy gradient ratio $r = \frac{\pi^{\text{tether}}(a|s)}{\pi^{\text{tether}}_{\text{old}}(a|s)}$ with $\frac{\pi^{\text{tether}}(a|s)}{\pi^{\text{act}}_{\text{old}}(a|s)}$ and 2. add a clipped importance weight term c to the value function target from

$$v_s = V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \tau \delta_t V$$

to

$$v_s = V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \tau \text{clip}(\frac{\pi^{\text{tether}}(a|s)}{\pi^{\text{act}}(a|s)}, a, b) \delta_t V$$

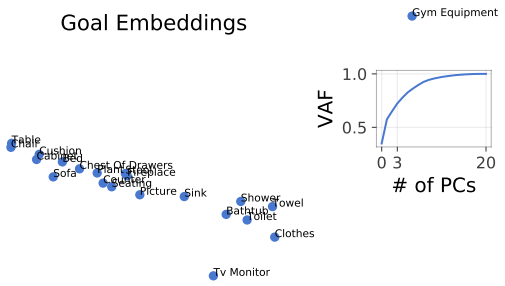

Figure A11. We project goal and semantic channel embeddings into their top-2 PCs. (Inset indicates variance accounted for by PCs, title indicates that participation ratio (approximate dimensionality) of goal is around 10.

With $\delta_t V = r_t + \gamma V_{x_{t+1}} - V_{x_t}$ being a TD for V. We set clip terms $a = 0.01, b = 1.0$. The overall RL loss (both actor and critic losses) are averaged across policies, *i.e.* equal weighting of acting and tethered policy loss.

A.14. Training and Auxiliary Task Details

We train our agent via Proximal Policy Optimization PPO) [31] with Generalized Advantage Estimation (GAE) [30] and using the Adam optimizer [19]. Agent parameter counts were all $5-6$ million parameters, excluding parameters in auxiliary modules. This amounts to GRU hidden sizes of 196 (except for the No Aux ablation, which has 512). General hyperparameters follow [40]:

$$\text{Rollout Workers: } n = 4 \tag{11}$$
$$\text{Rollout Length: } t = 128 \tag{12}$$
$$\text{PPO Epochs} = 4 \tag{13}$$
$$\text{PPO Mini-batches} = 2 \tag{14}$$
$$\text{Discount } \gamma = 0.99 \tag{15}$$
$$\text{GAE } \tau = 0.95 \tag{16}$$
$$\text{lr} = 2.5 \times 10^{-4} \tag{17}$$
$$\text{Adam } \epsilon = 1 \times 10^{-5} \tag{18}$$
$$\text{Gradient Norm Cap} = 0.5 \tag{19}$$
$$\text{PPO Clip} = 0.2 \tag{20}$$

except PPO Clip factor, which was set to 0.2 instead of 0.1, as per recommendations in [38].

Our complete loss is:

$$L_{\text{total}}(\theta_m; \theta_a) = L_{\text{RL}}(\theta_m) - \alpha H_{action}(\theta) + L_{\text{Aux}}(\theta_m; \theta_a) \tag{21}$$

$$L_{\text{Aux}}(\theta_m; \theta_a) = \sum_{i=1}^{n_{\text{Aux}}} \beta^i L^i_{\text{Aux}}(\theta_m; \theta^i_a) - \mu H_{attn}(\theta_m) \tag{22}$$

		mIoU	mRecall	mPrecision	Acc
train	40-class	77.09	86.46	87.22	91.62
	21-class	37.13	92.25	39.23	81.96
val	40-Class	24.95	36.96	39.62	61.88
	21-Class	15.93	39.45	21.65	69.01

Table A4. RedNet [18] performance on the MP3D dataset after tuning on 40 MP3D categories, or after tuning on 21 goal categories. Note that these numbers include accuracy on VOID class which skew numbers (especially accuracy) upward.

H_{attn} is the entropy of the attention distribution over the different auxiliary tasks. We set $\alpha = 0.01$ for 4-action agents (as in [40]) and $\alpha = 0.0075$ for 6-action agents. We set $\mu = 0.075$, which amounts to belief weighting across to be $\approx$ equal; we find little difference when weighting is more flexible. Auxiliary task loss coefficients β were determined such that the loss terms were in the same order of magnitude at initialization.

For extended details, the configurations of our experiments are available in the code release.

A.15. Rednet Tuning

We use RedNet to predict segmentation from RGBD at test time. We finetune the model (which was pretrained on SUN-RGBD) with 100K randomly sampled front-facing views rendered in the Habitat simulator (16K validation views). This procedure is the one used in [3]. We initially trained the model to segment all 40 MP3D categories; we present its accuracies in Table A4. Unlike our agent, this RedNet is greatly overfit. Since our agent did not appear to greatly leverage semantic cues during exploration (through Section 4.1, Section A.12), we reduced the complexity of the RedNet task by asking it to only segment the 21 goal categories (other categories were cast to VOID). This improves performance of segmentation on the goal categories, and resulted in slight improvement in validation scores after 10M frames of agent tuning, so we used it in the main text.

A.16. Negative Results

In the course of our experiments, we found that:

- Adding a curiosity module (ICM [25]) as non-episodic intrinsic reward failed to change performance significantly.
- Forcing agent recall (to fix the "loop" failure mode) with an Action Recall auxiliary task failed to change agent performance. This task was implemented as ADP with a negative

horizon.

- Controlling the agent's sense of time by projecting beliefs out of the probed time dimension was unstable. We could not *e.g.* solve "Commitment" failure modes by setting the time variable to near end-episode without degrading performance.

A.17. RedNet vs GT with CI

We present a full version of Table 3 with 95% CI estimates in Table A5.

	Success % (↑)		SPL % (↑)	
	TRAIN	VAL	TRAIN	VAL
1) 4-Act	50.3±5.7 (36±5.5)	43.3±5.6 (**34.4±2.0**)	18.1±2.7 (12.4±2.3)	12.3±2.1 (**9.58±0.75**)
2) 6-Act	56±5.6 (21.7±4.7)	**58.0±5.6** (30.8±1.9)	21.5±2.9 (8.24±2)	16.9±2.3 (7.60±0.64)
3) 6-Act + Tether	54±5.7 (27.3±5.1)	48.7±5.7 (26.6±1.9)	27.9±3.5 (11.5±2.5)	**19.1±2.7** (**9.79±0.82**)

Table A5. Performance on a 300-episode subset of TRAIN and VAL splits, reported as "with GT segmentation (with RedNet segmentation)". Reproducing Table 3 with 95% CIs included. Bold values are significantly better ($p < 0.05$) than non-bold values in a paired t-test on 300 episodes. Note that the VAL performance with RedNet is reported on the full split, taken from the primary experiments.