INFERRING COGNITIVE PROFILES WITH COGNITION-CONDITIONED DEEP INVERSE PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Bounded rationality proposes that humans think and act as best they can given limited cognitive resources. For example, to solve complex problems, people routinely engage in *subtasking*, in which they chain together solutions to simpler subtasks. But the boundedly rational solution to a task depends on a person's *cognitive profile*, the individual factors that determine what they find rewarding or cognitively effortful. This makes it hard to use bounded rationality to analyze human behavior in naturalistic tasks, as it requires computing optimal solutions under many different candidate profiles. Our method, *cognition-conditioned deep inverse planning* (CCDIP), addresses this by training a single hierarchical policy that takes cognitive parameters as input and yields tractable action likelihoods. By inverting this forward model, we can then recover people's subtasking strategies and cognitive profiles from their behavior. In simulation, CCDIP recovers parameters of agents with known profiles. On a new dataset of participants performing a driving task ($N = 39$, $\sim$345,000 timesteps), CCDIP shows that attentional factors are needed to explain behavior and reveals stable individual differences in cognitive parameters. CCDIP thus offers a route to theory-driven, interpretable cognitive modeling in naturalistic tasks where traditional approaches do not scale.

1 INTRODUCTION

Humans systematically depart from *idealized rationality* (Tversky & Kahneman, 1974), but many of these departures are rational once cognitive costs are taken into account, a principle known as *bounded rationality* (Simon, 1955; Lieder & Griffiths, 2020). Turning this principle into models that can be fit to behavior, however, remains difficult, especially in complex, naturalistic tasks.

The first difficulty is that bounded-rational solutions depend on a person's *cognitive profile*, the unobservable psychological factors that make certain mental operations easier or harder for someone to perform. For instance, working memory capacity varies across people, and people with less available capacity rely on it less (Collins & Frank, 2012). Second, bounded rationality says that people make *meta-cognitive choices* (e.g., choosing what to remember or attend to) that only manifest in observable behaviors indirectly (Gershman et al., 2015; Lieder & Griffiths, 2017). Finally, in all but the simplest of tasks, a person's unobservable cognitive state evolves over time; for example, when driving, a person might initially attend to a car in front of them, but ignore it after passing it. In short, using bounded rationality to model decisions requires dealing with at least three sources of uncertainty: *individual cognitive profiles*, *meta-cognitive choices*, and *cognitive dynamics*.

We propose *cognition-conditioned deep inverse planning* (CCDIP), a method that amortizes the overhead of computing bounded-rational solutions under different cognitive profiles and also provides a likelihood for action trajectories that can be straightforwardly evaluated. Our key idea is to train a single hierarchical policy on a version of a base task (e.g., a driving simulator) augmented with random cognitive profiles (e.g., combinations of attentional costs, heuristic dispositions, and outcome preferences), and then invert this policy for inference. We apply CCDIP to *subtasking*, the sequencing of goals and attention over time, in simulated agents with known cognitive profiles and show that these parameters can be accurately recovered. Finally, we apply this method to human data collected on a driving task ($N = 40$ participants) to make inferences about their subtasking strategies and cognitive profiles. These results suggest that CCDIP offers a promising framework for scaling up interpretable cognitive modeling to complex domains.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 BACKGROUND

Bounded rationality recasts apparent sub-optimalities in human behavior as optimal decisions given realistic cognitive limitations (Gershman et al., 2015). From this perspective, habits (Daw et al., 2005), myopic planning (van Opheusden et al., 2023), and simplified task representations (Ho et al., 2023) are cognitive strategies that enable people to make good decisions while minimizing mental effort. One such strategy, and the focus of this paper, is *subtasking*, the process of repeatedly adopting a local subgoal and scoping planning to a simplified model of the environment. The structuring goal-directed behavior into sequences of subtasks is a classic topic in psychology (Miller et al., 1960; Newell & Simon, 1972) and neuroscience (Botvinick, 2008; Botvinick et al., 2009; Solway et al., 2014), and recent work shows that people form individual subgoals and simplifications according to bounded rational principles by trading off their utility and the cost of the representation (Ho et al., 2022; Correa et al., 2023). Intelligently sequencing subtasks therefore involves making a series of *meta-cognitive decisions*, tacit choices about what to attend to and what to aim for that may be better or worse at producing effective actions in a task. Unsurprisingly, modeling the latent choice dynamics underlying processes like subtasking remain challenging; existing methods apply to small, discrete tasks where candidates can be enumerated or focus on selecting a single subtask for a problem (Ho et al., 2022; 2023; Banerjee et al., 2026).

More broadly, inferring cognitive variables such as goals, rewards, and beliefs is the focus of inverse RL (Abbeel & Ng, 2004; Ziebart et al., 2008) and inverse planning (Baker et al., 2009), in which a model of rational action is inverted to infer cognitive states. These approaches have been used to model humans in interactive settings, where a system can use a model of a person's intentions to guide its actions (Dragan et al., 2013; Russell, 2019). More recent work extends inference beyond rewards to other variables, such as false beliefs and inconsistency (Evans et al., 2016), biases (Shah et al., 2019), bounded search (Zhi-Xuan et al., 2020), and attentional heuristics (Banerjee et al., 2026). These methods share the limitations identified above, namely, that because inference requires repeatedly solving planning problems, they are either confined to small discrete domains or to inference about a single meta-cognitive choice.

CCDIP aims to overcome these limitations by drawing on two ideas from RL. First, we use hierarchical RL (Sutton et al., 1999; Dietterich, 2000) to formalize subtask selection under a cognitive profile as a sequential decision-making problem with a well-defined optimal solution. Second, as in techniques like goal-conditioned RL (Schaul et al., 2015) and contextual MDPs (Hallak et al., 2015), we can recast finding optimal policies for multiple related problems as finding the optimal policy for a single parameterized problem. This allows us to have a single network trained across a distribution of cognitive profiles stand in for a family of bounded rational subtasking strategies. The trained network serves as a forward model of subtasking, which we then invert to perform inference about people's cognitive profiles from their behavior on a task.

3 COGNITION-CONDITIONED DEEP INVERSE PLANNING

This section presents Cognition-Conditioned Deep Inverse Planning (CCDIP), our method for inferring the cognitive profile parameters of boundedly rational agents. We will review preliminaries from reinforcement learning (Sutton & Barto, 1998), formalize subtasking under different cognitive profiles given a base task, and then turn to how the optimal solution to a cognition-conditioned task can be used to perform inference from behavioral data. Throughout, we write $\Delta(X)$ for the probability simplex over a set X and $\mathbb{1}[\cdot]$ for the indicator function.

3.1 PRELIMINARIES

Markov Decision Processes. We model a task (e.g., a driving simulator) as a *Markov Decision Process* (MDP), a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and discount rate $\gamma \in [0, 1]$. A *contextual MDP* (CMDP) is a family of MDPs $\{\mathcal{M}_\theta\}$ indexed by a context $\theta \in \mathbb{R}^n$ (Hallak et al., 2015). We use θ to designate a decision-maker's cognitive profile. This includes outcome preferences (e.g., the perceived penalty for crashing), via a reward function R_θ. The rest of the task MDP $(\mathcal{S}, \mathcal{A}, T, \gamma)$ are assumed fixed for all $\mathcal{M}_\theta$, but in Section 3.2, we elaborate on how θ parameterizes other factors in a cognitive profile. The optimal value function of $\mathcal{M}_\theta$ for each $s \in \mathcal{S}$ is $V^*_\theta(s) =$

$\max_a\{R_\theta(s,a)+\gamma\sum_{s'}T(s'\mid s,a)V^*_\theta(s')\}$, the optimal state-action value function for each $s,a\in\mathcal{S}\times\mathcal{A}$ is $Q^*_\theta(s,a)=R_\theta(s,a)+\gamma\sum_{s'}T(s'\mid s,a)V^*_\theta(s')$, and the randomized optimal policy is $\pi^*_\theta(a\mid s)\propto\mathbb{1}[Q^*_\theta(s,a)=V^*(s)]$.

Factored states with masked variables. We assume a factored, maskable representation of the state space, which allows us to capture human selective attention to task variables (Ho et al., 2022; Banerjee et al., 2026). Given a finite set of *variable names*, Vars, and set of values, $\mathsf{Vals}^\perp$, that includes a designated *masking value* $\perp$, the state space is all assignments of variables to values $\mathcal{S}=\mathsf{Vars}\to\mathsf{Vals}^\perp$. Variables name object properties: a scene with two cars has states over $\mathsf{Vars}=\{\texttt{car_1.position},\texttt{car_1.speed},\texttt{car_2.position},\texttt{car_2.speed}\}$. Given a state s and variable v, we write s_v to denote the value assigned to v in s. Additionally, given a set of variables $c\subseteq\mathsf{Vars}$ and state s, we write $s[c]$ to denote the state with all variables *not* in c set to the masked value $\perp$. That is, $s[c]=\{\mathrm{v}\mapsto s_\mathrm{v}\mid\mathrm{v}\in c\}\cup\{\mathrm{v}\mapsto\perp\mid\mathrm{v}\in\mathsf{Vars}-c\}$.

3.2 Cognition-conditioned subtasking

Subtasks. Bounded rationality proposes that when faced with complex, seemingly intractable tasks, humans adopt cognitive strategies to most effectively utilize their limited cognitive resources (Simon, 1957). One such strategy is to construct and solve simplified, tractable *subtasks* that, when appropriately sequenced, can achieve good performance on the full task (Botvinick, 2008). For instance, when driving, one might adopt a local positional goal, such as to switch to a lane ahead of another car, and scope attention to goal-relevant variables, such as the other car's location and speed, while ignoring irrelevant variables, such as its color. Goals and attentional scope evolve over time: once the goal position is reached or passed, one selects a new goal and scope.

Inspired by work in hierarchical RL (Dietterich, 2000), we formalize a subtask on a task $\mathcal{M}_\theta$ as a tuple $\tau=\langle R_\tau,\beta_\tau,c_\tau\rangle$, where the *goal* of a subtask is represented by the *pseudo-reward function* $R_\tau:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ and *termination function* $\beta_\tau:\mathcal{S}\to[0,1]$, and the *scope* of a subtask is represented by $c_\tau\subseteq\mathsf{Vars}$, a subset of task variables. The optimal subtask value from state $s\in\mathcal{S}$ is:

$$V^*_\tau(s)=\max_a\left\{R_\tau(s[c_\tau],a)+\Big(1-\beta_\tau(s[c_\tau])\Big)\,\mathbb{E}_T\Big[V^*_\tau(s')\Big]\right\},\tag{1}$$

with a corresponding optimal subtask policy $\pi^*_\tau(a\mid s)$. Note that the optimal subtask policy is only directly optimizing subtask pseudo-rewards R_τ over a chosen scope c_τ, and not the ground task rewards, R_θ; although the pseudo-rewards can be designed to reflect the full task rewards.

Cognition-conditioned semi-MDPs. How should a cognitively limited agent select and sequence subtasks? Our central argument is that the right strategy depends on one's cognitive profile, the internal factors that define one's cognitive environment. Here, motivated by work on task simplification in human planning (Ho et al., 2022), we focus on cognitive profiles defined by a person's attentional costs and biases (e.g., the mental effort required to attend to multiple objects), goal-selection biases (e.g., how far ahead to set a local goal), and outcome preferences (e.g., the perceived cost of hitting another car). This combination of factors allows us to go beyond simple forms of inverse RL (Abbeel & Ng, 2004) that do not incorporate bounded rationality and consider sub-optimal behavior resulting from the interaction of preferences, selective attention, and planning.

Formally, we construct a *cognition-conditioned semi-MDP* from a task MDP $\mathcal{M}_\theta$ and subtask set $\mathcal{T}$ by treating subtasks as a temporally extended actions and using the construction in Sutton et al. (1999). The resulting semi-MDP has states $\mathcal{S}$, actions $\mathcal{T}$, and semi-Markov transition and reward functions indexed by a cognitive profile θ. Selecting a subtask $\tau\in\mathcal{T}$ from s executes its optimal policy π^*_τ until the subtask terminates, producing a *sub-episode* $(s_0,a_0,\ldots,s_k)$ of random length k with $s_0=s$. The semi-MDP transition function $P(s',k\mid s,\tau)$ is the distribution over terminal states and durations induced by π^*_τ, T, and β_τ. The semi-MDP reward is the ground reward accumulated over the sub-episode, reweighted by outcome preferences and penalized by cognitive costs:

$$\bar{R}_\theta(s,\tau)=\mathbb{E}_{\pi^*_\tau,T}\left[\sum_{t=0}^{k-1}\gamma^t R_\theta(s_t,a_t)\,\middle|\,s_0=s\right]-C_\theta(s,\tau)\tag{2}$$

where R_θ is the parameterized task reward, and C_θ is the parameterized cognitive cost: an attentional cost that grows with the size of the scope $|c|$ and expected duration of the subtask, and a goal-

selection bias that penalizes goals by distance. The optimal value from a state $s \in \mathcal{S}$ is then:

$$\bar{V}^*_\theta(s) = \max_\tau \left\{ \bar{R}_\theta(s,\tau) + \sum_{s',k} P(s', k \mid s, \tau)\gamma^k \bar{V}^*_\theta(s') \right\}, \tag{3}$$

with corresponding optimal state-action subtasking value $\bar{Q}^*_\theta(s,\tau)$.

Subtasking policies We consider subtasking strategies that depend on two factors: the optimal value under the cognition-conditioned semi-MDP, which captures people's capacity for boundedly rational subtasking; and the effect of attentional heuristics on subtasks, for instance, the tendency to heuristically focus on cars that are moving quickly. Thus, an *attention-biased subtasking policy* is:

$$\mu_\theta(\tau \mid s) \propto \exp\left\{\bar{Q}^*_\theta(s,\tau) + H_\theta(s,\tau)\right\}, \tag{4}$$

where heuristic strategies are represented with a linear *bias function* $H_\theta(s,\tau) = \sum_i \theta_{H,i}\,\phi_i(s,\tau)$, with each $\phi_i : \mathcal{S} \times \mathcal{T} \to \mathbb{R}$ a real-valued feature of the state and $\theta_{H,i}$ the ith heuristic bias weight.

3.3 Inferring cognitive profiles from behavior

Throughout, we have used θ to denote cognitive profile parameters related to a person's outcome preferences (via R_θ), cognitive costs (via C_θ), and attentional heuristics (via H_θ). This highlights the various non-obvious ways in which an individual's cognitive profile can affect the task they are performing as well as the generality of our approach. Moreover, given our formulation of an attention-biased subtasking policy μ_θ and subtask policy π^*_τ, we can now state a likelihood for a cognitive profile given a state-action trajectory $\zeta = \langle s_1, a_1, ..., s_N, a_N \rangle$ as:

$$\mathcal{L}(\theta;\zeta) = \sum_{\tau_{1:N}} \prod_{t=1}^{N} p_\theta(\tau_t \mid \tau_{t-1}, s_t)\pi^*_{\tau_t}(a_t \mid s_t) \tag{5}$$

where the subtask transition kernel is $p_\theta(\tau' \mid \tau, s') = (1 - \beta_\tau(s'))\mathbb{1}[\tau = \tau'] + \beta_\tau(s')\mu_\theta(\tau' \mid s')$ and τ_0 is a fixed null subtask. The inverse problem of estimating cognitive profile parameters from behavioral data ζ can then be expressed in terms of maximum-likelihood estimation:

$$\theta^* = \arg\max_\theta \mathcal{L}(\theta;\zeta) \tag{6}$$

In practice, we solve Equation 6 by pretraining a network that provides the single optimal state-action subtasking value function $\bar{Q}^*_\theta$ over a distribution of cognitive profiles θ (up to a state-dependent constant) and single optimal subtask policy π^*_τ over a distribution of subtasks τ. This effectively amortizes the cost of the most expensive step of inference: computing optimal action-level and subtask-level plans under candidate cognitive profiles.

4 Experiments

To validate our approach, we first measured its accuracy in recovering known cognitive profiles from synthetic data on a 2D driving task. We then used human performance data on the same task to perform an ablation study with our model, and found that attentional factors account for a substantial proportion of human behavior.

4.1 Task

To study behavior in a complex continuous task, we developed a 2D driving simulator (Figure 1). The task involves driving a car (henceforth referred to as the *ego* car) down a road as fast as possible without crashing into other objects, while collecting flags randomly placed along the road.

Task rewards. Task performance was scored using point penalties, each reflecting a different aspect of real-world driving: a 1 point *time penalty* every 2 seconds, to encourage driving faster; a 10 points *collision penalty* per crash; and a 1 point *missed-flag penalty* each time the agent passes a flag without collecting it. Participants began each episode with 500 points and were instructed to retain as many points as possible, which required trading off between penalties (e.g., driving faster saves time but increases collision probability).

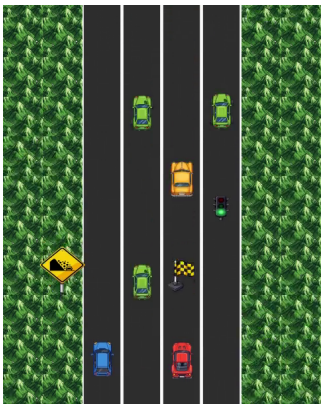


Figure 1: A screenshot of the driving simulator. The red car is the *ego* car (human/agent controlled). The road sign and the traffic light in the scene are distractors. The flag in front is a collectible.

Action space. The task allows five base actions (accelerate, left, right, brake, no-operation) and four composite actions (accelerate-left, accelerate-right, brake-left, brake-right). Left and right actions snap the car to the center of the adjacent lane. For the human task, the actions map to WASD or IJKL key combinations for left-handed and right-handed participants, respectively: W/I accelerates, A/J slides left, D/L slides right, and S/K brakes. Composite actions are executed as key combinations (e.g., W+A for accelerate-left). For the RL task, all nine actions are mapped to a 1D discrete action space. The ego vehicle also always moves at a minimum speed, which prevents participants from stalling and probes attentional capacity by forcing decisions under time pressure.

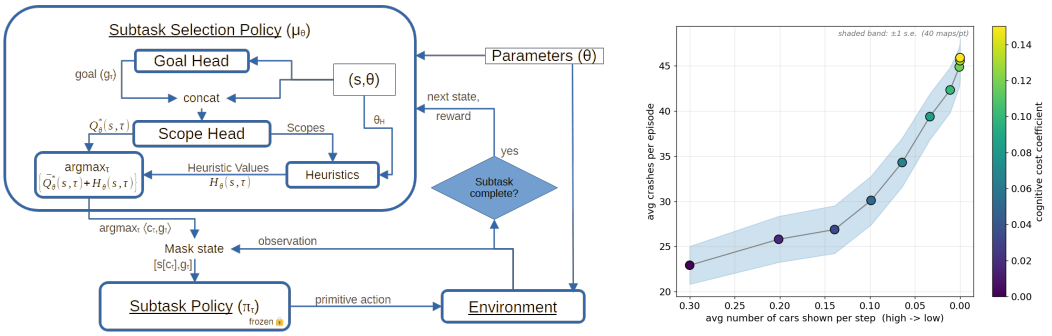
The simulator is a NumPy (Harris et al., 2020) based Python script, enabling fast simulation and integration with standard RL libraries including OpenAI Gymnasium and PufferLib Suarez (2024). For human participants, the script runs in the browser using Pyodide Droettboom et al. (2021), invoked using HTML and JavaScript; for RL policies, it is instantiated using a Gymnasium interface.

4.2 Model Implementation

A parametrized hierarchical deep reinforcement learning policy was trained on the task to model human behavior. Its structure implements the CCDIP approach: the top level is the subtask selection policy $\mu_\theta(\tau \mid s)$, which chooses a subtask τ for the low-level subtask policy π^*_τ to execute (Figure 2a).

Subtask policy. We trained the generalist subtask policy π^* on subtasks defined by randomly sampling goal positions along the map, using full (unmasked) observations spanning all object configurations; construed policies are obtained by masking input states. Masked objects are encoded with the same padding values as absent objects, so construed states fall within the domain of the training distribution of the generalist policy. The goal-conditioned objective included a bonus for reaching the goal, a penalty for passing it unreached, a constant per-step cost encouraging faster goal-reaching, and a collision penalty balanced against the goal-reaching bonus. The collision penalty was carefully chosen to produce a "cautiously obedient" policy: it avoids preventable crashes but prioritizes goal reaching if the goal position overlaps with another object, transferring the responsibility of selecting good goals to the subtask selection policy. Because the agent always moves at a minimum forward speed (see "Action space"), every sub-episode is guaranteed to terminate by either reaching or passing the goal in a finite number of timesteps.

Subtask selection policy. We trained the subtask selection policy on the cognition-conditioned semi-MDP. We define the training context $\theta_{train} = [\theta_R, \theta_C]$. Concretely, θ_C contains the multiplier for attentional cost and the goal proximity cost (subtask planning horizon), while θ_R weighs two behavioral parameters: collision cost (preference for risky behavior) and goal-flag distance cost (priority for flag collection). Each training step is a subtask decision whose reward is the sum of per-timestep returns (task reward R_θ minus cognitive cost C_θ) over the resulting sub-episode, which is the effectively $\bar{R}_\theta(s, \tau)$ (from Equation 2). Subtask-level returns were discounted at $\gamma = 0.5$.



(a) (b)

Figure 2: **Hierarchical policy architecture and attentional cost analysis. (a)** An abstracted representation of the hierarchical policy architecture with a pre-trained (frozen) subtask policy. The goal head inside the *subtask selection policy* first takes the concatenated vector $[s, \theta]$ as input to select a goal, $g_\tau = \langle R_\tau, \beta_\tau \rangle$, which is then passed to the (attentional) scope head along with $[s, \theta]$ to obtain $\tilde{Q}^*_\theta$ for all subtasks. The heuristics module computes heuristic values, H_θ, for all scopes. The subtask policy is then fed the observation vector $[s[c_\tau], g_\tau]$ (based on the highest scoring biased subtask), which generates a primitive action used to step the environment. The environment's output gets fed to the subtask policy to execute the next step. The subtask selection policy again is re-run if the sub-episode has ended. **(b)** Effect of attentional cost on subtask selection and action. Increasing attentional cost C_θ leads the subtask selection policy to pick sparser scopes, which in turn increases the hierarchical policy's crash frequency. Each point shows the mean and standard error for both axes, aggregated across 40 simulated episodes. The leftmost point corresponds to an agent that attends to the full scene ($C_\theta = 0$), and serves as a baseline collision rate.

The active context θ_{train} was supplied through the observation vector, so a single network learned effective subtask selection across the entire range $\Theta_{train} \subseteq \mathbb{R}^4$. Training yielded the unbiased policy μ_θ (Equation 4, with $\theta_H = 0$); heuristic biases were applied at inference.

Training details. The subtask policy is a multi-layer perceptron (MLP), while the subtask selection policy is a combination of an MLP and a DeepSets network (Zaheer et al., 2018). Both policies are actor-critic networks trained with PPO (Schulman et al., 2017). We modified CleanRL's (Huang et al., 2022) implementation of the PPO algorithm, to accommodate the hierarchical and parametrized policy architecture. See the Appendix for further implementation details. Training was done using PufferLib's asynchronous vectorization environment (Suarez, 2025). The subtask policy was trained for 800 million environment steps and the subtask selection policy for 1.5 billion ($\sim$166 million subtask decisions at $\sim$9 environment timesteps per subtask), requiring $\sim$10 and $\sim$22 hours respectively on an Ubuntu HPC node (NVIDIA L40S GPU, 50 CPU cores, and 64GB memory).

4.3 Model Fitting

Fitting the model to an observed trajectory requires evaluating μ_θ at each candidate $\theta \in \Theta$ at every recorded timestep, which by Equations 4 and 5 reduces to reading subtask probabilities from the trained selection network and reweighting them by the heuristic bias. We describe each step below.

Subtask probabilities. Given a task state (s) and context (θ), the subtask selection network outputs a logit vector $u \in \mathbb{R}^{28}$ over goal locations (4 lanes + 24 evenly-spaced forward distances, each bin mapping to a relative goal coordinate (d_x, d_y)) and a logit vector $v \in \mathbb{R}^6$ over masks for vehicles in the scene, giving $|\mathcal{T}| = 4 \times 24 \times 2^6 = 6{,}144$ subtasks per decision. The mask size was chosen because fewer than 1% of simulated steps had more than 6 vehicles in view; when more are present, the closest 6 are retained as candidates for masking and the rest are hidden. We define the unbiased probability of any subtask $\tau \in \mathcal{T}$, given state s and context vector θ, as:

$$\mu_\theta(\tau \mid s) = p_{g,\tau} \prod_{i=1}^{6} p_i^{c_{\tau,i}} (1 - p_i)^{1 - c_{\tau,i}}, \tag{7}$$

where $p_{g,\tau} = \frac{e^{u_{g,\tau}}}{\sum_{\tau'} e^{u_{g,\tau'}}}$, p_i represents the logistic function $\sigma(v_i) = 1/(1+e^{-v_i})$, and $c_{\tau,i} \in \{0,1\}$ is the ith element of the attention mask $c_\tau \in \{0,1\}^6$.

Heuristic features. Given the selection probabilities above, the biased selection policy is obtained by reweighting with the bias function H_θ (Equation 4). We use four heuristic features: (1) mean distance of the ego from all vehicles in the scope; (2) distance of the ego from the closest vehicle in the scope; (3) number of vehicles in the lanes between the ego and the goal, including ego and the goal lanes; and (4) time-to-collision (TTC) with the vehicle directly ahead, which is set to 0, when the scope contains no such vehicle. The final model has eight free parameters— four context and four bias parameters.

Likelihood estimation. To estimate theta for an observed trajectory, we perform joint maximum likelihood estimation (Equation 6) with gradient ascent, and exact gradients obtained by automatic differentiation in PyTorch using Adam (Kingma & Ba, 2017).

4.4 Experimental Data

Synthetic Data. Synthetic data was generated by simulating our trained hierarchical policy in the task, using 100 unique θ-vectors randomly sampled from Θ. Each of the 100 policy profiles, μ_θ, was simulated on the same 6 maps used for the human trials, generating data for 600 simulated episodes.

Human Data. The simulator ran at a constant 30 frames per second (FPS). Data was recorded at a variable rate of $\sim$20 FPS: task state (positions, velocities, and properties of all objects in the scene) and participant actions were recorded every third frame (10 FPS), with additional frames stored at task events (flag collection or miss, collisions) to ensure all important events were captured.

We collected data from 40 participants recruited through Prolific, a platform for recruiting human participants (www.prolific.com). Each participant completed 6 trials, each involving driving through a single map (which corresponds to a single RL episode). Map objects and their positions were generated from a random seed, per trial. The same 6 seeds were used for all participants, with the order of presentation counterbalanced across participants. All participants completed a consent form outlining the risks associated with the experiment. Participants received \$6 USD base compensation and up to \$2 USD as a performance bonus, computed relative to other participants in the batch. This research was reviewed and approved by an IRB (*details of which IRB to be included after blind review*).

Participant data were screened for task compliance prior to analysis. Trials scoring below 100 points—two standard deviations below the grand mean (M = 255.24, SD = 65.97)—were manually reviewed; those revealing extended inactivity were excluded. One participant scored below 100 on every trial and with video suggesting random key presses, so their data was excluded. In total, 10 out of 240 trials (6 trials $\times$ 40 participants; 4.16%) were excluded.

5 Results

Before fitting our model to behavioral data, we confirmed that our policy did, in fact, adapt its behavior to different parameter values, through learned associations between context parameter values and context-dependent rewards. Figure 2b, for example, shows how our model behaves under different attentional costs. When the policy observes a high attentional cost θ_C, it selects subtasks with sparser attentional scopes, even when such scopes lead to more collisions. This demonstrates that the context parameters give direct, interpretable control over the policy's meta-cognitive tradeoffs, which justifies our use of μ_θ as a generative model of individual differences. To ensure the subtask selection model represents a wide range of behavior, we trained the agent to optimize on the following range of parameter values: [0,10] for collision cost, [0,0.15] the cognitive cost, [0,6] for goal proximity cost, [0,1.5], and for goal-flag distance cost.

Simulation Studies. For the synthetic data, we measured the correlation between true (θ used to generate the data) and the mean (over 6 maps) estimated parameter values. Among the context parameters, the goal-flag distance ($r = 0.75$, 95% CI $[0.64, 0.83]$), goal proximity cost ($r = 0.70$, $[0.61, 0.78]$), and cognitive cost ($r = 0.67$, $[0.56, 0.76]$) were recovered well, while, collision penalty was recovered moderately ($r = 0.43$, $[0.25, 0.58]$). Recovery of the heuristic weights (θ_H) showed more variation, where the weight on the lead vehicle's inverse TTC was the best-recovered ($r = 0.84$, $[0.79, 0.88]$); mean vehicle distance ($r = 0.44$, $[0.25, 0.60]$) and closest vehicle distance ($r = 0.33$, $[0.12, 0.51]$) showed moderate recovery; and only the weight for the number vehicles in the goal path ($r = 0.25$, $[0.06, 0.42]$) recovered weakly. It should be noted that heuristics such as number of vehicles in the goal path and closest vehicle distance are critical for goal reaching with collision avoidance, and the policy network may have already learned these heuristics during training, leaving little variability for these parameters to explain, which could be a reason for their relatively low recoverability.

To further show that attentional factors are necessary for explaining attention limited behavior, we compared the fit (log-likelihood) of our biased and attention limited model to a full-attention model (a variant of the model with the heuristics and the cognitive cost parameters frozen at 0), on the synthetic data. we found that accounting for attentional factors improved fit by 67.2 points on average (95% CI $[59.1, 75.8]$), across the 100 simulated driver profiles.

Human Data Experiments. We further extended this analysis to human data, where we compared the fit of the fully parametrized (biased) model to two ablated variants (Figure 3): a *no-heuristics* model, which sets the 4 heuristic bias parameters to 0, and a *no-scopes* model, which additionally freezes the cognitive cost parameter to 0. We scored each model's fit per episode using the Bayesian Information Criterion (BIC), as it effectively balances goodness-of-fit against model complexity (parameter count). To test if the BIC differences were reliable, we averaged the per-episode BIC difference between each pair of models within participants and tested whether it differed from zero with a Wilcoxon signed-rank test, Holm-corrected across the three comparisons; confidence intervals were obtained by bootstrapping participants. The fully parameterized model outperformed the no-heuristics model ($\Delta\text{BIC} = 80.7$, 95% CI [71.1, 90.3]; Holm-adjusted $p < .001$) and the no-scopes model ($\Delta\text{BIC} = 153.5$, 95% CI [133.1, 173.']; $p < .001$), and the no-heuristics model outperformed the no-scopes model ($\Delta\text{BIC} = 72.8$, 95% CI [59.6, 86.5]; $p < .001$). All three comparisons were significant, suggesting that the cost of attention and the heuristic biases each explain a distinct component of resource-rational behavior in our task.

Additionally, to evaluate our model's ability to capture human behavior, we used CCDIP to estimate parameter values from human episodes, simulated agents initialized with the fitted values across several maps, and compared the average collision rate for each human across all episodes with that of the fitted agent's. We found a strong correlation ($r = 0.57$) between the human and simulated collision behavior, demonstrating that our approach also reliably captures human behavioral trends.

Finally, to assess how well each model parameter captures individual differences, we measure and report parameter intraclass correlation coefficients ICC(1,k) (Shrout & Fleiss, 1979), the reliability of a participant's parameter estimate averaged over their episodes (route means removed; $k \approx 5.9$ episodes each; 95% CIs from a bootstrap over participants). Three of the four reward parameters presented good to moderate reliability (Koo & Li, 2016): collision penalty (ICC(1,k) = 0.82, 95% CI [0.64, 0.89]) and goal proximity cost (0.81 [0.70, 0.88]) showed good reliability, while cognitive cost (0.71 [0.51, 0.82]) had moderate reliability. Goal-flag distance showed no between-participant variation because estimates for the parameter were close to 0 in 206 of 230 episodes, demonstrating a low preference for collecting flags across all participants. The heuristic weights, were not reliable participant-level measures: the lead-vehicle inverse TTC reached 0.45 [0.00, 0.66], 0.23 [-0.28, 0.48] for number of vehicles in the goal path, 0.24 [-0.15, 0.47] for closest-vehicle distance, and 0.25 [-0.22, 0.50] for mean vehicle distance. The reward and cognitive cost parameters therefore characterize individual participants, whereas the heuristic weights either varied across episodes or could not be estimated precisely enough to distinguish participants, with the later being more likely given the relatively low recoverability observed for heuristic weights in the simulation study.

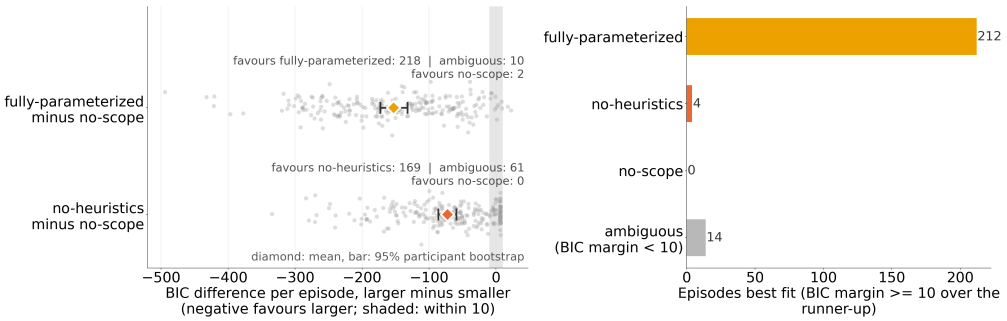


Figure 3: Top plot: Distribution of difference in BIC scores for the full and no-heuristics models, with the no-scopes model. Individual dots represent BIC score for a specific trial, diamonds represent the mean values, with horizontal whiskers showing ± 1 standard error. A lower value indicates better fit. Current plots suggest that our fully-parametrized model best fit the data on 212 of the 230 episodes. A model is considered as "best fitting" if it shows at least a $\Delta BIC \geq 10$ compared to the next best model, which is considered "very strong" evidence in favor of the winning model Kass & Raftery (1995). Ambiguous fits are episodes where $\Delta BIC < 10$ between the top two models.

6 Limitations and Future Work

Although our approach shows that accounting for attentional limits and heuristics, along with other behavioral parameters such as preference for risky behavior explains a significant chunk of boundedly rational behavior in a complex continuous task, several key limitations remain. First, we use a 2D driving simulator with a discrete action space which may have limited real world ecological validity. The task was intentionally designed to strike a balance between complexity and experimenter control, and future studies can identify and address challenges for scaling up to real-world driving scenarios. Secondly, here, we focus on inferring agent cognitive profiles from demonstrated behavior, therefore we account for cognitive parameters such as risk preference and attention capacity. However, human behavior is also constrained by motor limitations, and models that include factors such as reaction-time delay (Ma & Andréasson, 2006; Firoiu et al., 2018) can be more informative in capturing the interactions between attention and reaction time (Strayer et al., 2014). Finally, we did not compare CCDIP to other purely data-driven baseline models such as behavioral cloning. This is because, although we expect such models to better predict human actions (given enough data), unlike CCDIP, such models cannot produce human-interpretable psychometric parameter estimates, that help us understand human cognitive profiles and characterize individual differences.

7 Conclusion

The goal of this paper is to demonstrate how various reinforcement learning theories such as hierarchical policy design (Sutton et al., 1999; Levy et al., 2017) and contextual models of decision making (Hallak et al., 2015) can be leveraged to operationalize and scale psychological principles of human decision making (Simon, 1957; de Clippel & Rozen, 2024; Griffiths et al., 2015) to high-dimensional naturalistic settings, such as driving. Building on previous work on modeling attentional strategies in attention limited agents (Ho et al., 2022; Banerjee et al., 2026), we present CCDIP, an approach for dynamic inference of cognitive profiles for attention limited agents, from behavior. To validate our approach, we first showed that CCDIP can accurately estimate cognitive profiles from simulated behavioral data on a 2D driving task, with known ground truth. We then performed an ablation study using human data, and demonstrated that attentional factors that drive biased and boundedly rational behavior are necessary for explaining human behavior in our task. Finally, we show that several of our cognitive profile parameters reliably capture inter-individual differences among human participants. These results suggest that CCDIP can be used to scale up theory-driven and interpretable cognitive modeling to complex and naturalistic domains.

AI USE STATEMENT

We used generative AI tools to search for relevant related work (looking up literature for specific topics), aid and polish writing, and write experiment code. We have not used generative AI tools for drafting sections of the paper, generating synthetic datasets, or providing mathematical claims.

We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by the first author. Test suites and extensive documentation for the code are part of the repository that will be made available upon acceptance of this manuscript. AI was only used to review human writing for clarity and not drafting text.

REPRODUCIBILITY STATEMENT

All implementation and training (hyperparameter) details for the hierarchical deep reinforcement leaning model have been provided in the appendix. Additionally, the complete code-base and the human data will be made available upon acceptance of the manuscript.

REFERENCES

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, pp. 1, Banff, Alberta, Canada, 2004. Association for Computing Machinery. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015430.

Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, December 2009. ISSN 0010-0277. doi: 10.1016/j.cognition.2009.07.005.

Sounak Banerjee, Daphne Cornelisse, Deepak Gopinath, Emily Sumner, Jonathan DeCastro, Guy Rosman, Eugene Vinitsky, and Mark Ho. Estimating cognitive biases with attention-aware inverse planning. *Advances in Neural Information Processing Systems*, 38:14524–14550, 2026.

Matthew M. Botvinick. Hierarchical models of behavior and prefrontal function. *Trends in Cognitive Sciences*, 12(5):201–208, May 2008. ISSN 13646613. doi: 10.1016/j.tics.2008.02.009.

Matthew M. Botvinick, Yael Niv, and Andrew G. Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009. ISSN 0010-0277. doi: https://doi.org/10.1016/j.cognition.2008.08.011. URL `https://www.sciencedirect.com/science/article/pii/S0010027708002059`. Reinforcement learning and higher cognition.

Anne GE Collins and Michael J Frank. How much of reinforcement learning is working memory, not reinforcement learning? a behavioral, computational, and neurogenetic analysis. *European Journal of Neuroscience*, 35(7):1024–1035, 2012.

Carlos G. Correa, Mark K. Ho, Frederick Callaway, Nathaniel D. Daw, and Thomas L. Griffiths. Humans decompose tasks by trading off utility and computational cost. *PLOS Computational Biology*, 19:1–31, 2023.

Nathaniel D. Daw, Yael Niv, and Peter Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12):1704–1711, 2005.

Geoffroy de Clippel and Kareen Rozen. Bounded rationality in choice theory: A survey. *Journal of Economic Literature*, 62(3):995–1039, 2024.

Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

Anca D. Dragan, Kenton CT Lee, and Siddhartha S. Srinivasa. Legibility and predictability of robot motion. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 301–308. IEEE, 2013.

Michael Droettboom, Hood Chatham, Roman Yurchak, Dexter Chua, Gyeongjae Choi, Henry Schreiner, Marc Abramowitz, Will Lachance, Madhur Tandon, Jan Max Meyer, et al. pyodide/pyodide. *Zenodo*, 2021.

Owain Evans, Andreas Stuhlmüller, and Noah Goodman. Learning the preferences of ignorant, inconsistent agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Vlad Firoiu, Tina Ju, and Josh Tenenbaum. At human speed: Deep reinforcement learning with action delay. *arXiv preprint arXiv:1810.07286*, 2018.

Samuel J. Gershman, Eric J. Horvitz, and Joshua B. Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278, 2015. doi: 10.1126/science.aac6076.

Thomas L. Griffiths, Falk Lieder, and Noah D. Goodman. Rational Use of Cognitive Resources: Levels of Analysis Between the Computational and the Algorithmic. *Topics in Cognitive Science*, 7(2):217–229, April 2015. ISSN 1756-8757, 1756-8765. doi: 10.1111/tops.12142.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual Markov Decision Processes, February 2015.

Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Mark K. Ho, David Abel, Carlos G. Correa, Michael L. Littman, Jonathan D. Cohen, and Thomas L. Griffiths. People construct simplified mental representations to plan. *Nature*, 606(7912):129–136, May 2022. ISSN 1476-4687. doi: 10.1038/s41586-022-04743-9.

Mark K. Ho, Jonathan D. Cohen, and Thomas L. Griffiths. Rational Simplification and Rigidity in Human Planning. *Psychological Science*, 34(11):1281–1292, November 2023. ISSN 0956-7976, 1467-9280. doi: 10.1177/09567976231200547.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Robert E. Kass and Adrian E. Raftery. Bayes Factors. *J. Am. Stat. Assoc.*, June 1995. ISSN 1047-6572. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1995.10476572.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Terry K. Koo and Mae Y. Li. A Guideline of Selecting and Reporting Intraclass Correlation Coefficients for Reliability Research. *Journal of Chiropractic Medicine*, 15(2):155, March 2016. doi: 10.1016/j.jcm.2016.02.012.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

Falk Lieder and Thomas L Griffiths. Strategy selection as rational metareasoning. *Psychological review*, 124(6):762, 2017.

Falk Lieder and Thomas L. Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and brain sciences*, 43:e1, 2020.

Xiaoliang Ma and Ingmar Andréasson. Estimation of driver reaction time from car-following data: Application in evaluation of general motor–type model. *Transportation research record*, 1965(1): 130–141, 2006.

George A Miller, Galanter Eugene, and Karl H Pribram. *Plans and the Structure of Behaviour*. Holt, Rinehart & Winston, 1960.

Allen Newell and Herbert Alexander Simon. *Human Problem Solving*, volume 104. Prentice-hall Englewood Cliffs, NJ, 1972.

Stuart Russell. *Human compatible: AI and the problem of control*. Penguin Uk, 2019.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Rohin Shah, Noah Gundotra, Pieter Abbeel, and Anca Dragan. On the Feasibility of Learning, Rather than Assuming, Human Biases for Reward Inference. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 5670–5679. PMLR, May 2019.

P. E. Shrout and J. L. Fleiss. Intraclass correlations: uses in assessing rater reliability. *Psychol. Bull.*, 86(2):420–428, March 1979. ISSN 0033-2909. doi: 10.1037//0033-2909.86.2.420.

Herbert A Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, pp. 99–118, 1955.

Herbert A. Simon. *Models of Man: Social and Rational*. John Wiley & Sons, New York, NY, 1957.

Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G. Barto, Yael Niv, and Matthew M. Botvinick. Optimal Behavioral Hierarchy. *PLOS Computational Biology*, 10(8): e1003779, August 2014. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1003779.

David L Strayer, Jonna Turrill, James R Coleman, Emily V Ortiz, and Joel M Cooper. Measuring cognitive distraction in the automobile ii: Assessing in-vehicle voice-based. *Accident Analysis & Prevention*, 372:379, 2014.

Joseph Suarez. PufferLib: Fast and sane simplifying reinforcement learning for complex game environments. https://github.com/PufferAI/PufferLib, 2024. GitHub repository.

Joseph Suarez. Pufferlib 2.0: Reinforcement learning at 1m steps/s. In *Reinforcement Learning Conference*, 2025.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge, 1998.

Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Amos Tversky and Daniel Kahneman. Judgment under Uncertainty: Heuristics and Biases: Biases in judgments reveal some heuristics of thinking under uncertainty. *Science*, 185(4157):1124–1131, September 1974. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.185.4157.1124.

Bas van Opheusden, Ionatan Kuperwajs, Gianni Galbiati, Zahy Bnaya, Yunqi Li, and Wei Ji Ma. Expertise increases planning depth in human gameplay. *Nature*, 618(7967):1000–1005, June 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06124-2.

David Winkel, Niklas Strauß, Maximilian Bernhard, Zongyue Li, Thomas Seidl, and Matthias Schubert. Autoregressive policy optimization for constrained allocation tasks, 2024. URL https://arxiv.org/abs/2409.18735.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets, 2018. URL https://arxiv.org/abs/1703.06114.

Tan Zhi-Xuan, Jordyn Mann, Tom Silver, Josh Tenenbaum, and Vikash Mansinghka. Online bayesian goal inference for boundedly rational planning agents. *Advances in neural information processing systems*, 33:19238–19250, 2020.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Association for the Advancement of Artificial Intelligence*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A Appendix

A.1 Environment

The task involves driving a car down a road that contains (1) a finish line at the end of each map; (2) obstacles, including other cars, deer crossing the road, and landslides; (3) ice patches, which cause the agent to slip unpredictably for up to 2 seconds; (4) flags to be collected; and (5) attentional distractors such as railroads, traffic signals near intersections, and road signs . The task was designed to ensure that each observation frame would have both of task-relevant and irrelevant objects, forcing attention-limited agents to choose a subset of objects relevant to the task at any point in time.

Other cars and deer are semi-intelligent: they accelerate and decelerate (or completely halt) to prevent collisions with objects in their path, with rates determined by time-to-collision, but cannot adapt if approached laterally or from behind. Ice patches and distractors increase task difficulty by increasing attentional demands. The flags provide intermediate, short-horizon goals in an otherwise long-horizon task ($\sim$1500 environment steps per episode).

Each map (corresponding to an individual episode in the task environment) is represented as a 2D tile grid, where the number of rows represent the length of the maps, and the lane and boundary configurations are specified by the columns. For our experiments, we used a 1000×8 matrix, where the 8 columns consist of 4 road and 4 boundary (2 on each side) lanes. At any point in time, the agent can only observe a small slice of the map (a 12×8 tile window) around its position: 10 tiles ahead, and 1 tile behind. Dynamic objects that fall within 22 tile-rows of the agent vehicle are animated.

The environment produces an observation matrix of shape 137×7, where the 7 feature dimensions provide the following information (in order): the x and y location, the v_x and v_y velocity components, crashable and interactable flags, and an integer index identifying the object type for the 40 different objects (car type, road type, deer, road signs, etc.). The crashable and interactable flags indicate the effect of colliding with an object. Collision with crashable objects results in a crash (other cars, deer, and landslides), while interactable objects are associated with interaction events (collecting flags).

The first row of the 137 observation rows, contains the observation vector for the agent controlled (ego) vehicle. The next 96 rows contain information about the road and boundary objects in each tile of the 12×8 window. The last 40 rows are dynamically populated based on the number of objects inside the active window. Our decision to reserve 40 rows for objects was based on empirical evidence from millions of simulation steps, where we found that a maximum of about 30 objects was present in any frame. However, we kept 10 additional buffer slots as precautionary measure. Rows that do not contain objects contain '0'-padded values. All object positions and velocities are expressed relative to the ego.

A.2 Policy Architecture

The hierarchical agent is a combination of two independently trained *actor-critic* networks: a subtask selection policy (SSP) and a low-level subtask policy (SP). Training was carried out using the CleanRL (Huang et al., 2022) implementation of the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017).

A.2.1 Subtask Selection Policy

The SSP is a θ conditioned actor-critic network, which itself contains two Multi-Layer Perceptron (MLP) actor heads (one for choosing the subgoal coordinates, and one for choosing the construal mask) and a single MLP critic head. The subgoal coordinates are chosen on a 4×24 grid, one column for each of the four lanes, and the forward distance divided into 24 bins.

The current subsection contains the following subsubsections, in order: subsection A.2.1 contains implementation details of the SSP that were crucial for training stability, subsection A.2.1 contains the actual algorithms for the forward pass and the training loop, subsection A.2.1 contains function descriptions for functions used in the algorithms, and finally, subsection A.2.1 specifies the network and variable dimensions.

Notation. The environment observation is $s \in \mathbb{R}^{N\times F}$, where N=137 and F=7. The N rows are indexed by $\mathcal{T}$ (tile rows) and $\mathcal{O}$ (object rows). The SSP, denoted μ, attends to K=6 car slots (chosen based on empirical evidence from millions of simulation steps) and is conditioned on a reward-coefficient vector $\theta \in \mathbb{R}^P$, P=2; the SP is denoted π, and H is the sub-episode horizon. The per-block reward for the SSP is expressed as

$$R \;=\; -\,\theta_{\text{coll}}\, n_{\text{coll}} \;-\; \theta_{\text{cog}}\, n_{\text{steps}} \textstyle\sum_k m_k v_k, \tag{8}$$

where $\theta = (\theta_{\text{coll}}, \theta_{\text{cog}})$, with θ_{coll} the collision-cost and θ_{cog} the cognitive-cost parameter; n_{coll} is the number of collisions accumulated over the block and n_{steps} the number of steps during which the environment was still accumulating (its first sub-episode, before any episode end); v_k indicates whether construal slot k is assigned to a vehicle, and m_k whether that slot is set to visible. This ensures the agent is penalised only for mask slots that are valid and visible.

Implementation details

θ-conditioning. θ is resampled per environment episode (not per block), appended to the observation normalised to $[-1, 1]$ per coefficient over its training range — raw magnitudes ($\sim 10^{-2}$) are otherwise ignored by the network — and simultaneously used as the coefficients of the block reward, yielding a single policy conditioned on its own reward function.

PPO updates. The rollout buffer stores each block's observation, the sampled action lane, dy-bin and mask (ℓ, b, m), its log-probability under the behaviour policy, and the goal offset δ derived from the sampled goal g. During the update, the observation is passed through the network again under the current parameters, but no part of the action is resampled: the stored action (ℓ, b, m) are supplied to the forward pass, which recomputes their log-probabilities and the policy entropy. The stored δ is likewise supplied rather than recomputed, so the mask head is conditioned on the goal that was actually executed. This is necessary because the action is factorised as $p(\ell \mid s)\, p(b \mid s)\, p(m \mid s, g)$; recomputing δ from a goal preferred by the updated parameters would evaluate the stored mask under a goal that was never taken, and the resulting importance ratio would not correspond to any sampled action. Only the old log-probability need be retained, as the corresponding logits are reconstructed exactly from the stored observation and actions.

Identity remap. When the visible object set changes mid-block, mask bits are re-associated with objects by persistent identity rather than by slot position, without a new SSP forward pass; objects entering the view default to hidden. An object that leaves and re-enters within one block recovers its original bit.

Masking semantics. A car is visible to the SP if it occupies a valid slot and its Bernoulli bit is 1; invalid (empty) slots are forced visible so they contribute no gradient; cars beyond the K nearest are excluded from candidacy and always hidden. Zeroed rows are indistinguishable from the padded empty object slots the SP saw throughout its own training.

Train/inference slot allocation. One of the challenges with constrained allocation tasks is selection bias in early slots (Winkel et al., 2024). In our case, if construal slots are allocated to vehicles in the order of slot indices, slot 1 would be trained on the highest number of samples, slot 2 would be trained on fewer samples, while slot 6 would rarely be used because the occurrence of 6 vehicles in a single frame is a rare event. This would cause the network to incorrectly associate lower cognitive penalties with lower slot indices instead of the number of active slots. To get around this issue, mask slots were randomly assigned to vehicles at each SSP decision step. This was only used as a training-time debiasing device. At inference, slots are assigned to vehicles by index, so $p(a \mid s, \theta)$ is a well-defined function — required for likelihood-based fitting.

Permutation equivariance. Under random slot allocation, the visible cars constitute a set whose ordering varies arbitrarily across decisions. A fixed-order MLP head would therefore have to learn that the slot index is uninformative, and no structural constraint binds a given logit to the car features in its slot rather than to the slot's position in the input. This motivates an architecture in which the required symmetry is enforced by construction rather than acquired during training: permuting

the slots should permute the logits identically (permutation equivariance). DeepSets (Zaheer et al., 2018) satisfies this requirement by applying a shared encoder to each element of the set and aggregating the resulting embeddings through a symmetric pooling operation. In the present model, every slot is encoded by the same network, so the head can respond only to a car's features and not to its position in the input. The mean and maximum of the embeddings of the occupied slots are then concatenated with each slot's own embedding prior to scoring, so that each car is evaluated relative to the others present rather than in isolation.

Control flow. Three boolean flags drive control flow in the training algorithms: *subdone* marks the end of a sub-episode (goal reached or missed, or the horizon H expired); *envdone* marks the end of an environment episode (collision or map completion), after which the environment resets automatically; and *objchange* marks a change in the observation's object-identity column. The per-block *done* flag written to the buffer is *envdone* latched over that block.

Algorithms

The algorithms for SSP single-step inference (Algorithm 1) and for training the network through backpropagation (Algorithm 2) are provided in this subsection. Functions were abstracted away to improve readability of the algorithms. The function descriptions are provided in subsection A.2.1.

Algorithm 1 SSPFORWARD: one θ-conditioned goal–construal decision, $p(\ell, b, m \mid z) = p(\ell \mid z)\, p(b \mid z)\, p(m \mid z, \delta)$

Require: base input z, slot features c, player position (x, y)

1: $(\ell, b) \sim \textsc{GoalHead}(z)$ {lane and forward-distance categoricals; argmax if deterministic}
2: $g \leftarrow \textsc{GoalPosition}(\ell, b, x, y)$; $\quad \delta \leftarrow g - (x, y)$ {world goal and its ego-relative offset}
3: $u \leftarrow \textsc{MaskHead}\big(c,\ \textsc{Context}(z, \delta)\big)$ {permutation-equivariant per-slot logits}
4: $m_k \sim \text{Bernoulli}(\sigma(u_k))$ for $k = 1 \dots K$ {m_k=1: show car k to the SP}
5: $V \leftarrow \textsc{Critic}_\mu(z)$ {goal-free value}
6: **return** $\big(g,\ \delta,\ \ell,\ b,\ m,\ \log p(\ell, b, m),\ V\big)$

Algorithm 2 TRAINSSP: PPO over sub-episode blocks against a frozen SP

Require: B vectorised envs, stepped simultaneously with independent episode boundaries; frozen SP π; SSP μ; buffers $(T, B, \cdot)$

for block $n = 1 \dots T$ **do**
  resample $\theta_j \sim \mathcal{U}[\theta^{\min}, \theta^{\max}]$ for each env j whose episode ended during the previous block {per-episode, not per-block: θ is the reward being optimised}
  $(z, c) \leftarrow$ BUILDINPUT(s, θ); $(x, y) \leftarrow$ PLAYERPOS(s)
  $(g, \delta, \ell, b, m, \log p, V) \leftarrow$ SSPFORWARD(z, c, x, y) {snapshot; fixed for the whole block}
  for $t = 1 \dots H$ **do**
    $s' \leftarrow$ APPLYMASK(s, m) {hide masked-off and non-selected cars}
    $a \leftarrow \arg\max \pi\big($AUGMENT$(s', g, \cdot)\big)$; $s \leftarrow$ step envs with a
    ACCUMULATEOUTCOMES: collisions, for envs still in their first sub-episode of this block
    $(\text{subdone}, \text{objchange}) \leftarrow$ CHECKTERMINATION(s, g)
    $m_j \leftarrow$ REMAPBYIDENTITY(m_j) for each env j with objchange_j {mask bits follow objects; entering objects hidden}
    $(z', \cdot) \leftarrow$ BUILDINPUT(s_j, θ_j); $V_j^{\text{next}} \leftarrow$ CRITIC$_\mu(z')$ for each env j where subdone_j fires for the first time {bootstrap at that env's own sub-episode end}
    if every env has completed a sub-episode **then**
      break
    end if
  end for
  $R_n \leftarrow$ BLOCKREWARD$(\theta, \text{outcomes})$ {Eq. 8}
  write $(z, \ell, b, \delta, m, \log p, V, R_n, \text{done}, V^{\text{next}})$ to buffer slot n
end for
compute advantages by GAE, bootstrapping from V^{next}; update μ with clipped PPO on the stored (z, ℓ, b, m), with δ conditioning the mask head

Function glossary

GOALHEAD(z) Multilayer perceptron mapping the base input z to $L + D$ logits, split into the lane and dy-bin categoricals.

GOALPOSITION(ℓ, b, x, y) World coordinates of the lane-ℓ centre and forward-distance bin-b centre, clamped to the track.

MASKHEAD$\big(\{c_k\}_{k=1}^{K},\ [\, z_{\setminus c};\ \delta\,]\big)$ DeepSets head (see subsection A.2.1) taking the per-slot car features c_k and, as context, the base input with the car-slot block removed, concatenated with the goal offset δ. A shared encoder is applied to every slot, and each slot is scored against mean- and max-pooled embeddings of the occupied slots together with the encoded context, yielding K logits. Permutation-equivariant by construction.

CRITIC$_\mu(z)$ Multilayer perceptron mapping the base input z to a scalar state value; goal-free, so that V does not condition on the action.

BUILDINPUT(s, θ) Selects the K nearest cars, encodes the observation (normalise positions and velocities, expand the kind index to a one-hot), gathers the selected slots' features with empty slots zeroed, and concatenates these with the nearest-flag features and θ rescaled to $[-1, 1]$, giving z.

APPLYMASK(s, m) Zeroes the observation rows of all cars not shown (masked-off slots and cars beyond the K nearest) — indistinguishable to the SP from empty object slots.

ACCUMULATEOUTCOMES Adds this step's collision count to the block's running totals, for environments that have not yet completed a sub-episode or ended an episode within the block.

CHECKTERMINATION(s, g) Goal reached, goal missed, or horizon exceeded $\rightarrow$ *subdone*; any change in the observation's object-identity column $\rightarrow$ *objchange*.

REMAPBYIDENTITY(m) Re-associates mask bits with persistent object identities (useful for keeping track of objects within sub-episodes) after the visible set changes; entering objects default to hidden.

CONTEXT(z, δ) Returns z with the slot block removed, concatenated with the goal offset δ.

PLAYERPOS(s) Player world position, read from the observation's ego row.

AUGMENT$(s', g, \cdot)$ The SP's trained input format, defined in subsection A.2.2. The elided arguments — the running sub-episode step, the previous action, and the blend coefficient, which is held fixed during SSP training — are maintained by the block loop.

BLOCKREWARD$(\theta, \textbf{outcomes})$ Per-block reward expressed in Equation 8.

Network and variable dimensions

Table 1: SSP network flow. B is the batch (one row per environment per decision); $K{=}6$ car slots, $E{=}46$ encoded features per entity, $N{=}137$ observation rows, $P{=}2$ reward coefficients. Activations: R = ReLU, T = Tanh.

Stage	Operation	Output shape	Params
Input assembly (BUILDINPUT)			
raw observation	from environment	$(B, 137, 7)$ / $(B, 959)$	—
encode (p)	normalise + one-hot kind	$(B, 137, 46) \to (B, 6302)$	—
car slots (c)	gather K nearest car rows, zero invalid	$(B, 6, 46) \to (B, 276)$	—
flag features (f)	nearest-flag $[\mathbb{1}, \Delta x, \Delta y]$	$(B, 3)$	—
θ	normalised to $[-1, 1]$	$(B, 2)$	—
concatenate	$[\, p \mid c \mid f \mid \theta \,]$	$(\mathbf{B}, \mathbf{6583})$	—
Goal head (lane + forward-distance categoricals)			
goal.0	Linear $6583 \to 128$, R	$(B, 128)$	842,752
goal.2	Linear $128 \to 128$, R	$(B, 128)$	16,512
goal.4	Linear $128 \to 28$	$(B, 28) = 4 \oplus 24$	3,612
	$\mathrm{Cat}(\cdot_{1:4})$, $\mathrm{Cat}(\cdot_{5:28}) \to$ goal offset δ	$(B, 2)$	—
subtotal			**862,876**
Mask head (MASKHEAD, DeepSets; permutation-equivariant)			
enc.0	Linear $46 \to 128$, R (shared, per slot)	$(B, 6, 128)$	6,016
enc.2	Linear $128 \to 128$, R (shared, per slot)	$(B, 6, 128)$	16,512
pool	mean and max over *valid* slots	$2 \times (B, 128)$	—
ctx.0	Linear 6309 $(6583 - 276 + 2) \to 128$, R	$(B, 128)$	807,680
ctx.2	Linear $128 \to 128$, R	$(B, 128)$	16,512
score.0	Linear $512 \to 128$, R $([e_k \mid \mathrm{mean} \mid \mathrm{max} \mid \mathrm{ctx}])$	$(B, 6, 128)$	65,664
score.2	Linear $128 \to 1$	$(B, 6, 1) \to (B, 6)$	129
	$\mathrm{Bernoulli}(\sigma(\cdot))$	$(B, 6)$	—
subtotal			**912,513**
Critic (goal-free)			
critic.0	Linear $6583 \to 256$, T	$(B, 256)$	1,685,504
critic.2	Linear $256 \to 256$, T	$(B, 256)$	65,792
critic.4	Linear $256 \to 256$, T	$(B, 256)$	65,792
critic.6	Linear $256 \to 1$	$(B,)$	257
subtotal			**1,817,345**
Buffers (lane centres 4, dy-bin centres 24)			28
SSP total			**3,592,762**

Table 2: Model and environment dimensions (production checkpoint).

Symbol	Description	Value
Observation		
N	observation rows (1 ego + 96 tiles + M objects)	137
M	object slots	40
F	raw features per row	7
C	object-kind classes (one-hot)	40
E	encoded features per row, $(F-1)+C$	46
SSP		
K	car slots (Bernoulli mask outputs)	6
P	reward coefficients in θ	2
$\dim z$	base input, $(N+K)\,E+3+P$	6583
L	lanes (goal categorical 1)	4
D	forward-distance bins (goal categorical 2)	24
	goal-head output, $L+D$	28
	mask-head context, $\dim z - KE + 2$	6309
	goal-/mask-head hidden width	128
	critic hidden width (3 layers)	256
Training and rollout		
	SP input (raw obs + 6 scalars), $N \times F + 6$	965
$\lvert\mathcal{A}\rvert$	SP discrete actions	9
H	sub-episode horizon (env steps)	100
T	blocks per training iteration	200
S	rollout length (steps per training iteration)	256
B	parallel environments	432

A.2.2 SUBTASK POLICY

The SP is a goal-conditioned, discrete-action, actor-critic MLP policy trained with PPO against sub-episode goals issued by a hardcoded high-level policy, later replaced by the learned SSP.

The current subsection contains the following subsubsections, in order: subsection A.2.2 contains the actual algorithms for the SP forward pass and the training loop, subsection A.2.2 contains function descriptions for functions used in the algorithms, and finally, subsection A.2.2 specifies the network and variable dimensions.

Notation (SP). A goal $g = (x_g, y_g)$ is a lane centre and a forward target position; each sub-episode lasts at most H env steps. c_{env} denotes the per-sub-episode environment-reward coefficient, a^- the previous action, and t the step index within the current sub-episode. The per-step reward is expressed as

$$r_t \;=\; \underbrace{\rho_{\text{reach}}\,\mathbb{1}[\text{reach}] - \rho_{\text{miss}}\,\mathbb{1}[\text{miss}]}_{r_t^{\text{goal}}} \;+\; c_{\text{env}}\, r_t^{\text{env}} \;-\; c_{\text{const}} \;-\; c_{\text{switch}}\,\textsc{SwitchDist}(a_{t-1}, a_t), \tag{9}$$

where *reach* is entry into the goal radius, *miss* is crossing the goal's y-coordinate outside that radius, *timeout* is reaching H without either (which never occurred, because the agent always had a forward velocity). c_{env} is uniformly sampled at random (in the range $[0.1, 1.0]$) at every sub-episode boundary and exposed to the policy in its observation and $r_t^{\text{env}} \in \{0, -1\}$ is the environment reward, a sparse penalty signal: -1 at each collision step and -1 if the episode times out before the map is completed, 0 otherwise, including on goal completion. Although the SP was trained on a range of c_{env} values, it was frozen to 0.1 for SSP training and rollout to ensure the SP was cautiously obedient. c_{const} was a small constant per-step cost added to ensure the policy remained motivated to reach the goal sooner rather than later. Finally, $c_{\text{switch}}\,\textsc{SwitchDist}(a_{t-1}, a_t)$ was added to reduce action noise.

Algorithms

The algorithms for SP single-step inference (Algorithm 3) and for training the network through backpropagation (Algorithm 4) are provided in this subsection. Functions were abstracted away to improve readability of the algorithms. The function descriptions are provided in subsection A.2.2.

Algorithm 3 SPFORWARD: goal-conditioned action selection

Require: augmented input $x = [\, x_{\text{obs}} \mid x_{\text{aux}} \,]$

1: $\tilde{s} \leftarrow \textsc{Encode}(x_{\text{obs}})$ {N entity rows, normalised and kind-expanded}
2: $h \leftarrow \big[\textsc{EncodeEgo}(\tilde{s}_0);\ \textsc{EncodeTiles}(\tilde{s}_{\mathcal{T}});\ \textsc{EncodeObjects}(\tilde{s}_{\mathcal{O}});\ \textsc{GoalFeatures}(x_{\text{aux}})\big]$ {shared per-entity encoders, max-pooled: permutation-invariant}
3: $a \sim \text{Cat}\big(\textsc{Actor}(h)\big)$ {argmax at evaluation / under the SSP}
4: $V \leftarrow \textsc{Critic}_\pi(h)$
5: **return** $\big(a,\ \log p(a),\ V\big)$

Algorithm 4 TRAINSP: PPO over sub-episodes with externally sampled goals

Require: B vectorised envs (staggered start positions) with initial observation s; agent π; buffers $(S, B, \cdot)$

$g \leftarrow$ SAMPLEGOAL(s); $c_{\text{env}} \leftarrow$ SAMPLEBLEND(); $t \leftarrow 0$; $a^- \leftarrow$ IDLE {trainer-side goal state; env is goal-agnostic}
for iteration $= 1, 2, \ldots$ **do**
  for step $= 1 \ldots S$ **do**
    $x \leftarrow$ AUGMENT$(s, g, t, a^-, c_{\text{env}})$ {ego-position-masked obs + 6 scalar features}
    $(a, \log p, V) \leftarrow$ SPFORWARD(x); $s \leftarrow$ step envs with a; $t \leftarrow t + 1$
    $(r^{\text{goal}}, \text{subdone}, \text{reached}) \leftarrow$ GOALREWARD(s, g, t) {reach / miss / horizon-timeout}
    $r \leftarrow r^{\text{goal}} + c_{\text{env}}\, r^{\text{env}} - c_{\text{const}} - c_{\text{switch}}$ SWITCHDIST(a^-, a) {Eq. 9}
    $a^- \leftarrow a$, reset to IDLE for envs with envdone {previous action persists across sub-episode boundaries}
    $d \leftarrow \text{subdone} \vee \text{envdone}$ {the PPO/GAE done mask — no bootstrapping across sub-episodes}
    store $(x, a, \log p, V, r, d)$ in buffer slot
    if subdone or envdone **then**
      $g \leftarrow$ SAMPLEGOAL(s); $t \leftarrow 0$; $c_{\text{env}} \leftarrow$ SAMPLEBLEND() {env state persists across sub-episode boundaries}
    end if
  end for
  compute advantages by GAE using d as the terminal mask; update π with clipped PPO
end for

Function glossary

ENCODE(x_{obs}) Reshapes the observation component of the input into N entity rows, normalises positions and velocities, and expands the kind index into a one-hot vector, giving $\tilde{s} \in \mathbb{R}^{N \times E}$.

ENCODEEGO$(\tilde{s}_0)$ Multilayer perceptron applied to the ego row alone; no pooling is required for a single entity.

ENCODETILES$(\tilde{s}_{\mathcal{T}})$ Shared multilayer perceptron applied independently to each tile row, followed by max-pooling over $\mathcal{T}$.

ENCODEOBJECTS$(\tilde{s}_{\mathcal{O}})$ Shared multilayer perceptron applied independently to each object row, followed by max-pooling over $\mathcal{O}$; permutation-invariant by construction.

GOALFEATURES(x_{aux}) The six appended scalars, with the goal offset rescaled to visible-window units to match the encoder's normalization.

SAMPLEGOAL(s) Hardcoded high-level policy: a uniformly random lane centre x_g and $y_g = y - \Delta y$ with $\Delta y \sim \mathcal{U}[d_{\min}, d_{\max}]$, clamped to the track.

SAMPLEBLEND() Draws $c_{\text{env}} \sim \mathcal{U}[c_{\text{env}}^{\min}, c_{\text{env}}^{\max}]$ per sub-episode.

AUGMENT$(s, g, t, a^-, c_{\text{env}})$ Zeroes the ego's absolute position in s and appends

$$[\Delta x_g,\ \Delta y_g,\ t/H, \text{lat}(a^-), \text{lon}(a^-),\ c_{\text{env}}],$$

where $\text{lat}, \text{lon} \in \{-1, 0, +1\}$ are the previous action's lateral and longitudinal components.

GOALREWARD(s, g, t) Terminal goal outcome (Eq. 9, first term) and the sub-episode termination flag (reach, miss, horizon, or episode end).

SWITCHDIST(a, a') $|\text{lat}(a) - \text{lat}(a')| + |\text{lon}(a) - \text{lon}(a')| \in \{0, \ldots, 4\}$ — penalises oscillation between opposing actions.

ACTOR, CRITIC$_\pi$ MLP heads over the concatenated representation.

IDLE The no-op action of the nine-action set; used as the previous-action value before any action has been taken and after an environment reset.

Network and variable dimensions

Table 3: SP network flow. The entity encoder is shared across tiles and across objects and max-pooled, giving permutation invariance within each group. Activations: R = ReLU, T = Tanh. Layers use orthogonal initialisation ($\sqrt{2}$ gain; 1.0 on the value output, 0.01 on the policy output).

Stage	Operation	Output shape	Params
Input assembly (AUGMENT)			
masked observation	ego absolute x, y zeroed	$(B, 959)$	—
appended scalars	$[\Delta x_g, \Delta y_g, t/H, \mathrm{lat}(a^-), \mathrm{lon}(a^-), c_{\mathrm{env}}]$	$(B, 6)$	—
	concatenate	$(\mathbf{B}, \mathbf{965})$	—
Entity encoder (operates on the 959-dim observation slice)			
reshape / expand	$(B, 137, 7) \to$ normalise + one-hot	$(B, 137, 46)$	—
ego.0/.2	Linear $46 \to 64$, R; Linear $64 \to 64$ (row 0)	$(B, 64)$	7,168
tile.0/.2	Linear $46 \to 256$, R; Linear $256 \to 256$ (rows 1..96)	$(B, 96, 256)$	77,824
	max-pool over tile rows	$(B, 256)$	—
obj.0/.2	Linear $46 \to 256$, R; Linear $256 \to 256$ (rows 97..136)	$(B, 40, 256)$	77,824
	max-pool over object rows	$(B, 256)$	—
concat	$64 + 256 + 256$	$(B, 576)$	—
re-append	+ 6 scalars (goal offset rescaled by $8.0, 11.0$)	$(\mathbf{B}, \mathbf{582})$	2
subtotal			**162,818**
Actor			
actor.0	Linear $582 \to 256$, T	$(B, 256)$	149,248
actor.2	Linear $256 \to 256$, T	$(B, 256)$	65,792
actor.4	Linear $256 \to 9$	$(B, 9)$	2,313
	$\mathrm{Cat}(\cdot)$	$(B,)$	—
subtotal			**217,353**
Critic			
critic.0	Linear $582 \to 256$, T	$(B, 256)$	149,248
critic.2	Linear $256 \to 256$, T	$(B, 256)$	65,792
critic.4	Linear $256 \to 1$	$(B,)$	257
subtotal			**215,297**
SP total			**595,468**

Table 4: SP dimensions and training constants (production checkpoint).

Symbol	Description	Value
Observation		
	raw obs, $N \times F$ flattened	959
	tile rows / object rows	96 / 40
	augmented input, raw + 6 scalars	965
Networks		
	ego encoder ($E \to \cdot$)	64
	tile encoder, shared, max-pooled ($E \to \cdot$)	256
	object encoder, shared, max-pooled ($E \to \cdot$)	256
	encoder output ($64 + 256 + 256$), +6 scalars	582
	actor/critic hidden width (2 layers, tanh)	256
$\lvert\mathcal{A}\rvert$	discrete actions	9
Sub-episodes and reward		
H	sub-episode horizon (env steps)	100
	goal radius (m)	1.0
$[d_{\min}, d_{\max}]$	goal forward-distance range (m)	$[0.2, 12]$
$[c_{\mathrm{env}}^{\min}, c_{\mathrm{env}}^{\max}]$	env-reward blend range	$[0.1, 1.0]$
c_{const}	per-step living penalty	0.005
c_{switch}	action-switch coefficient	0.007
ρ_{reach}	reach reward	0
ρ_{miss}	miss penalty	0.3