CoFineLLM: Conformal Finetuning of Large Language Models for Language-Instructed Robot Planning

Jun Wang JUNW@WUSTL.EDU
Yevgeniy Vorobeychik YVOROBEYCHIKE@WUSTL.EDU
Yiannis Kantaros IOANNISK@WUSTL.EDU
Washington University in St. Louis, 1 Brookings Drive St. Louis, MO 63130

Editors: G. Sukhatme, L. Lindemann, S. Tu, A. Wierman, N. Atanasov

Abstract

Large Language Models (LLMs) have recently emerged as planners for language-instructed agents, generating sequences of actions to accomplish natural language tasks. However, their reliability remains a challenge, especially in long-horizon tasks, since they often produce overconfident yet wrong outputs. Conformal Prediction (CP) has been leveraged to address this issue by wrapping LLM outputs into prediction sets that contain the correct action with a user-defined confidence. When the prediction set is a singleton, the planner executes that action; otherwise, it requests help from a user. This has led to LLM-based planners that can ensure plan correctness with a user-defined probability. However, as LLMs are trained in an uncertainty-agnostic manner, without awareness of prediction sets, they tend to produce unnecessarily large sets, particularly at higher confidence levels, resulting in frequent human interventions limiting autonomous deployment. To address this, we introduce CoFineLLM (Conformal Finetuning for LLMs), the first CP-aware finetuning framework for LLM-based planners that explicitly reduces prediction-set size and, in turn, the need for user interventions. We evaluate our approach on multiple language-instructed robot planning problems and show consistent improvements over uncertainty-aware and uncertainty-agnostic finetuning baselines in terms of prediction-set size, and help rates. Finally, we demonstrate robustness of our method to out-of-distribution scenarios in hardware experiments.

Keywords: Finetuning of LLMs, Conformal Prediction, Language-instructed Robot Planning

1. Introduction

Task planning is a fundamental capability in robotics, enabling autonomous agents to design sequences of actions to accomplish high-level missions. Such missions are typically specified using formal languages (Sahin et al., 2019; Kantaros and Zavlanos, 2020; Ye et al., 2024; Kamale and Vasile, 2024; Vlahakis et al., 2025) or optimization-based formulations (Chen et al., 2017; Kiran et al., 2021; Zhou et al., 2023; Sachdeva et al., 2024; Wang et al., 2025a). These approaches, while powerful, require significant user expertise and manual effort to craft correct task specifications.[1]

In contrast, natural language (NL) offers a more intuitive interface for specifying robot missions. Motivated by the strong generalization abilities of large language models (LLMs), recent work has introduced a new paradigm of high-level planning from NL instructions (Ahn et al., 2022; Singh et al., 2023; Shah et al., 2023; Ding et al., 2023; Liu et al., 2023; Yang et al., 2024; Rana et al., 2023; Garg et al., 2024; Wang et al., 2024; Li and Zhou, 2025; Zhang and Kantaros, 2025; Shi et al., 2025; Ravichandran et al., 2025; Schwartz et al., 2025; Strader et al., 2025; Xu et al., 2025).

1. Our software implementation of CoFineLLM is available at https://cofinellm.github.io.

Despite impressive empirical results, LLM-based planners often hallucinate and generate overconfident yet wrong actions. To mitigate this, recent works have leveraged Conformal Prediction (CP) (Angelopoulos and Bates, 2023; Shafer and Vovk, 2008) to quantify uncertainty in LLM-based planners (Ren et al., 2023; Wang et al., 2025b). The planning problem is cast as a sequence of multiple-choice question–answering (MCQA) tasks, where the LLM is queried to select an action from a finite set. Instead of selecting the action with the highest model's confidence score, these works apply CP to produce a prediction set guaranteed, with user-specified confidence, to contain the correct action. When the set is a singleton, the planner selects the action included in that set; otherwise, it queries a human to disambiguate. These approaches ensure that the designed plans are correct with a user-defined probability. However, the larger this probability is, the larger the prediction sets are resulting in frequent human queries. The resulting help rate, i.e., the fraction of steps requiring user input, can become prohibitively high as LLMs are trained in an uncertainty-agnostic manner, with CP applied only post hoc (Huang et al., 2025; Chen et al., 2024; Zeng et al., 2025).

We address this limitation by introducing CoFineLLM (Conformal Finetuning for LLMs), the first uncertainty-aware fine-tuning framework for LLM-based planners that integrates CP during training. Our method aims to reduce prediction set sizes while preserving CP's coverage guarantees. The key idea is to "simulate" CP during training by incorporating both calibration and prediction steps within mini-batches. We propose a new loss function that combines a cross-entropy term with a CP-based term penalizing non-singleton prediction sets. To optimize this objective, we employ Low-Rank Adaptation (LoRA) (Hu et al., 2022) with a curriculum-based training scheme; however, other fine-tuning methods are compatible. During deployment, CP is applied to the fine-tuned model as in prior works to enable probabilistically correct planning (Ren et al., 2023; Wang et al., 2025b). We show empirically that this approach significantly outperforms baselines in terms of help rates.

Related Works: *(i) Conformal LLM-based Decision Making:* Prior works have applied CP to quantify uncertainty of LLMs used to address planning, translation, and general MCQA problems (Ren et al., 2023; Kumar et al., 2023; Su et al., 2024; Cherian et al., 2024; Vishwakarma et al., 2025, 2024; Liang et al., 2024; Campos et al., 2024; Lin et al., 2025; Wang et al., 2025b,c,d). However, CP in these methods is used exclusively at test time, leading to frequent human interventions when prediction sets are non-singleton. A comprehensive survey on how CP has been used to quantify uncertainty of various AI-enabled components can be found in (Lindemann et al., 2024). *(ii): Conformal Training:* Several CP-aware training methods have been proposed for neural network classifiers (Colombo and Vovk, 2020; Bellotti, 2021; Stutz et al., 2022; Noorani et al., 2024; Yan et al., 2024). Efforts towards developing conformal finetuning methods for graph neural networks are presented in (Wang et al., 2025e). These works design differentiable loss functions to minimize prediction set size while preserving the CP coverage guarantees. However, unlike this paper, they focus on models addressing single-step tasks (e.g., classification) and, therefore, target single-step prediction-set efficiency. Moreover, our empirical analysis suggests that existing CP-based loss functions, such as those proposed in (Stutz et al., 2022), can be sensitive to the quality of the initial model and may not transfer effectively to LLMs. In particular, we observed empirically that when a pre-trained LLM (e.g., Gemma3-1B) is both incorrect and overconfident, these losses can produce near-zero gradients, effectively halting learning. As a result, such approaches may be less suitable for fine-tuning large models that begin far from convergence. Building on these insights, we introduce a new CP-aware fine-tuning framework for sequential, multi-step decision-making tasks with LLMs. Our formulation explicitly mitigates the gradient sensitivity issue by defining a loss function that combines the standard cross-entropy objective with a novel CP regularizer.

Contribution: The contribution of this paper can be summarized as follows: (i) We propose CoFineLLM, the first CP-aware fine-tuning framework for LLMs in sequential decision-making tasks. (ii) We instantiate this framework for robot planning and demonstrate substantial reductions in help rates from users compared to both uncertainty-agnostic and conformal-aware fine-tuning methods. (iii) We empirically show that the proposed framework is robust to the performance of the initial model, making it well suited for cases where the model is far from convergence. (iv) We validate the method on hardware experiments, demonstrating robustness in out-of-distribution settings where test-time tasks and environments differ from those used during finetuning and calibration.

2. Problem Formulation

Robot Agent and Environment. We consider a single robot equipped with a known skill set $\mathcal{S} = \{s_1, \dots, s_S\}$ (e.g., 'move', 'turn', 'pick up'). The robot operates in a known environment $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, containing $M > 0$ semantic objects or regions o_m, $m \in \{1, \dots, M\}$. Each object o_m is defined by its location and a semantic label (e.g., 'box', 'key'). Applying a skill $s_j \in \mathcal{S}$ to an object o_m yields an action $a(s_j, o_m)$; when it is clear from the context, it is simply denoted by a. The set of all valid actions a is denoted by $\mathcal{A}$ and can be designed offline by enumerating admissible skill–object combinations.

NL-based Planning Problem: The agent is assigned a mission ϕ, expressed in NL, that can be completed if the robot executes a plan $\tau = \tau(1), \dots, \tau(t), \dots, \tau(H)$, defined as a finite sequence of actions $\tau(t) \in \mathcal{A}$, $t \in \{1, \dots, H\}$ for some horizon H. We formalize this setup by assuming a distribution $\mathcal{D}$ over mission scenarios $\xi_i = \{\phi_i, \mathcal{S}_i, H_i, \Omega_i\}$, where each scenario bundles together a mission ϕ_i, the available skill set $\mathcal{S}_i$, the mission horizon H_i, and the environment Ω_i. The distribution $\mathcal{D}$ is unknown, but we assume that i.i.d. scenarios can be drawn from it. When clear from context, we omit the subscript i.

Conformal NL-based Planners: Existing approaches can be employed to design plans τ that are correct with user-specified probability $1 - \alpha \in (0, 1)$ (Ren et al., 2023; Wang et al., 2025b). These works frame the planning problem a sequence of MCQA problems solved by an LLM denoted by g. The MCQA problem at time step t consists of a 'question', denoted by $\ell(t)$, providing a textual description of the scenario ξ along with past actions (if any) the agent has taken, and the 'choices' refer to the available actions $a \in \mathcal{A}$ of the robot. The solution to this MCQA problem results in $\tau(t)$. To solve it, the confidence scores, denoted by $g(a|\ell(t))$, of the employed LLM for each $a \in \mathcal{A}$ are extracted. Using these scores, CP is applied to construct a prediction set defined as

$$\mathcal{C}(t) = \{a \in \mathcal{A} \mid g(a|\ell(t)) \geq \delta\}, \tag{1}$$

where δ is the threshold computed offline (during a calibration phase; see Section 3.1) that depends on $1 - \alpha$ and the model. If $\mathcal{C}(t)$ is a singleton, the corresponding action is executed; otherwise, the system queries a human to select the correct action if it lies in $\mathcal{C}(t)$, or halts otherwise. Repeating this procedure for all steps yields a plan τ that satisfies ϕ with probability at least $1 - \alpha$.

Conformal Finetuning Problem: Our goal is to minimize the help rate, i.e., the fraction of time steps requiring human assistance, in conformal LLM-based planners through finetuning. Specifically, our goal is to develop an uncertainty-aware fine-tuning framework that explicitly reduces the size of the sets $\mathcal{C}(t)$ generated at test time without compromising their coverage guarantees. During deployment, CP is applied to the finetuned model exactly as in standard conformal planners.

Algorithm 1 `ComputeConformalThreshold`

1: **Input:** Calibration Dataset $\mathcal{M}_{\text{cal}}$; LLM g; Coverage level $1-\alpha$.
2: Compute nonconformity scores $\{\bar{r}_i\}_{i=1}^{D}$ on $\mathcal{M}_{\text{cal}}$ using LLM confidence scores g.
3: Compute $\hat{q} = \text{Quantile}_{1-\alpha}(\{\bar{r}_i\})$ and define $\delta = 1 - \hat{q}$.
4: **Output:** Threshold δ.

3. Method

In this section, we present CoFineLLM, our conformal-aware finetuning framework for LLM-based planners. The key idea is to fine-tune the model using a novel loss function that penalizes prediction sets, generated via CP, that are non-singleton and fail to include the correct action. Constructing these sets (and consequently the loss) depends on the threshold δ, introduced in (1), which varies as the model parameters are updated. Inspired by Stutz et al. (2022), we simulate the conformalization process during fine-tuning to dynamically adjust δ. The subsections below provide a detailed overview of this process, summarized in Algorithm 2. Section 3.1 first describes how prediction sets $\mathcal{C}(t)$ in (1) are constructed for a fixed model. Section 3.2 then presents our fine-tuning method, which optimizes the proposed loss function while periodically updating δ, by invoking the procedure in Section 3.1, to account for changes in the model parameters.

3.1. Constructing Local Prediction Sets using CP

In this section, we describe how the prediction sets $\mathcal{C}(t)$ are constructed for a given mission scenario $\xi \sim \mathcal{D}$, following the procedure in (Ren et al., 2023; Wang et al., 2025b); see also Alg. 1. Constructing these sets requires two components: (i) a calibration dataset comprising mission scenarios sampled from $\mathcal{D}$ along with their corresponding correct robot plans, and (ii) a nonconformity score (NCS) that quantifies the "error" of the LLM-based planner with respect to the ground truth. We first outline the construction of the calibration dataset, followed by the definition of the NCS.

Calibration dataset. We sample $D > 0$ i.i.d. scenarios $\xi_i \sim \mathcal{D}$. As discussed in Sec. 2, generating a robot plan is framed as solving a sequence of MCQA problems. At each time step t, we construct the corresponding prompt associated with ξ_i, denoted by $\ell_{i,\text{cal}}(t)$, and identify the correct action $\tau_{i,\text{cal}}(t)$. This gives rise to the following sequence of prompts $\bar{\ell}_{i,\text{cal}} = \ell_{i,\text{cal}}(1), \ldots, \ell_{i,\text{cal}}(H_i)$ and the corresponding sequence of correct actions (the ground-truth plan) $\tau_{i,\text{cal}} = \tau_{i,\text{cal}}(1), \ldots, \tau_{i,\text{cal}}(H_i)$. Repeating this process for all scenarios ξ_i yields the calibration dataset $\mathcal{M}_{\text{cal}} = \{(\bar{\ell}_{i,\text{cal}}, \tau_{i,\text{cal}})\}_{i=1}^{D}$.

Non-conformity score (NCS). Next, we define the NCS, which models the error of the LLM-based planner in generating a correct plan. This is defined as the worst-case error of the LLM-based planner across all decision steps of a mission scenario. Specifically, for a calibration scenario ξ_i, the NCS is given by

$$\bar{r}_i = 1 - \min_{t \in \{1,\ldots,H_i\}} g(\tau_{i,\text{cal}}(t) \mid \ell_{i,\text{cal}}(t)) \,, \tag{2}$$

where $g(\tau_{i,\text{cal}}(t) \mid \ell_{i,\text{cal}}(t))$ denotes the confidence score assigned by the LLM g to the correct action $\tau_{i,\text{cal}}(t)$ given the corresponding prompt $\ell_{i,\text{cal}}(t)$, as defined in Section 2. We compute the NCS $\bar{r}_i$ for all calibration scenarios ξ_i [line 2, Alg. 1]. The $1-\alpha$ empirical quantile $\hat{q} = \text{Quantile}_{1-\alpha}(\bar{r}_i)$ yields the threshold $\delta = 1 - \hat{q}$ used to form prediction sets [line 3, Alg. 1].

Constructing prediction sets. Consider an unseen test scenario $\xi_{\text{test}} \sim \mathcal{D}$ with planning horizon H_{test} and unknown true plan. As discussed in Section 2, planning is formulated as a sequence of MCQA problems. At each time step t, given the prompt $\ell_{\text{test}}(t)$ (associated with an MCQA problem of time step t), we define the prediction set $\mathcal{C}(t)$ introduced in (1) which collects all actions whose

confidence scores exceed the threshold δ. This is done for all time steps $t \in \{1, \dots, H_{\text{test}}\}$. As shown in Ren et al. (2023); Wang et al. (2025b), it holds that $P(\tau_{\text{test}} \in \bar{\mathcal{C}}) \geq 1 - \alpha$, where τ_{test} is the true plan and $\bar{\mathcal{C}} = \mathcal{C}(1) \times \cdots \times \mathcal{C}(H_{\text{test}})$. A formal proof of this guarantee, along with a discussion of the case with multiple feasible solutions, can be found in Ren et al. (2023).

3.2. CoFineLLM: Conformal Fine-Tuning for LLM-based Planners

In this section, we present our conformal-aware fine-tuning algorithm; see Algorithm 2. The algorithm takes as input: (i) an initial model g; (ii) a training dataset $\mathcal{M}_{\text{train}}$ (defined below); (iii) a calibration dataset $\mathcal{M}_{\text{cal}}$; (iv) a desired coverage level $1 - \alpha$; and (v) hyperparameters λ and K, introduced later. The calibration dataset $\mathcal{M}_{\text{cal}} = \{(\bar{\ell}_{i,\text{cal}}, \tau_{i,\text{cal}})\}_{i=1}^{D}$ consists of $D > 0$ pairs of prompt sequences and their corresponding ground-truth plans, constructed as described in Section 3.1 by sampling mission scenarios from $\mathcal{D}$. Note that this calibration dataset is not necessarily the same as the one used at test time. The training dataset $\mathcal{M}_{\text{train}} = \{(\bar{\ell}_{i,\text{train}}, \tau_{i,\text{train}})\}_{i=1}^{T}$ is generated in the same manner as the calibration dataset, using $T > 0$ mission scenarios sampled from $\mathcal{D}$[2].

The finetuning process runs until convergence where at each training epoch the model parameters, denoted by θ, are updated to minimize a loss function $L(\theta, \delta)$ that will be introduced later. Given a threshold δ, the model parameters θ are updated using stochastic gradient descent. Specifically, at each epoch, we sample from $\mathcal{M}_{\text{train}}$ a mini-batch $\mathcal{B}$ consisting of B pairs $(\ell_{i,\text{train}}(t), \tau_{i,\text{train}}(t))$, i.e., prompts $\ell_{i,\text{train}}$ (i.e., questions in MCQA tasks) and their corresponding true actions $\tau_{i,\text{train}}(t) \in \mathcal{A}$, [line 4, Alg. 2]. These individual prompts and actions can be drawn from different sequences i in $\mathcal{M}_{\text{train}}$ and do not need to correspond to the same time step t. This process repeats across epochs. As discussed earlier, the threshold δ used in the loss function $L(\theta, \delta)$ depends on the model parameters which evolve across epochs. To account for that, every K epochs, Algorithm 1 is called to update δ using $\mathcal{M}_{\text{cal}}$ and the current parameters θ [line 5, Alg. 2].

Next, we define the loss function $L(\theta, \delta)$ that augments the standard cross-entropy objective with a conformal regularizer as follows:

$$L(\theta, \delta) = L_{\text{CE}}(\theta) + \lambda\, L_{\text{CP}}(\theta, \delta), \qquad \lambda > 0, \tag{3}$$

where λ controls the trade-off between optimizing task accuracy and prediction set efficiency. The cross-entropy term $L_{\text{CE}} = -\frac{1}{B}\sum_{j=1}^{B} \log g(y_j|\ell_j)$ increases the probability assigned to the ground-truth action y across the datapoints ℓ included in $\mathcal{B}$. The conformal term $L_{\text{CP}}(\theta, \delta)$ is designed to minimize prediction set size only when the ground-truth action is already contained in the set. Intuitively, this encourages the model to first focus on improving prediction accuracy, when it is 'wrong', and to optimize prediction set efficiency once it is 'right'. This structure avoids conflicts between the two objectives, particularly during early training when the model's predictions may be incorrect, and leads to a more stable finetuning process. We define the conformal loss as below:

$$L_{\text{CP}}(\theta, \delta) = \frac{1}{|\mathcal{B}|} \sum_{(\ell, y) \in \mathcal{B}} \psi(p_y, \delta)\, [p_{\max} - \delta]_+, \tag{4}$$

where (i) $p_y = g(y \mid \ell)$ is the confidence score of the LLM assigned to the correct action y associated with the prompt ℓ; (ii) $p_{\max} = \max_{a \in \mathcal{A} \setminus \{y\}} g(a \mid \ell)$ is the largest probability assigned to any other action (called hereafter the 'strongest rival action') [line 6-7, Alg. 2]; (iii) $[\cdot]_+$ is the ReLU function,

2. To construct the training dataset, we select for each task the plan $\tau_{i,\text{train}}$ with the shortest mission horizon among all plans that satisfy the corresponding NL instruction.

Algorithm 2 CoFineLLM: Conformal Finetuning for LLM-based Planners

1: **Input:** LLM g; Distribution $\mathcal{D}$; Coverage level $1-\alpha$; Weight λ; Period K.
2: Sample scenarios from $\mathcal{D}$ and construct training dataset $\mathcal{M}_{\text{train}}$ and calibration dataset $\mathcal{M}_{\text{cal}}$
3: **repeat**
4: Sample minibatch $\mathcal{B}$ of step-level pairs $(\ell_{i,\text{train}}(t), \tau_{i,\text{train}}(t))$ drawn randomly from $\mathcal{M}_{\text{train}}$.
5: Every K epochs, update $\delta \leftarrow \textsc{ComputeConformalThreshold}(g, \mathcal{M}_{\text{cal}}, 1-\alpha)$.
6: Compute $p_y = g(y \mid \ell)$ and $p_{\max} = \max_{a \in \mathcal{A}\setminus\{y\}} g(a \mid \ell)$ for all pairs $(\ell, y) \in \mathcal{B}$.
7: Compute $L_{\text{CP}}(\theta, \delta) = \frac{1}{|\mathcal{B}|} \sum_{(\ell,y)\in\mathcal{B}} \psi(p_y, \delta)\,[p_{\max} - \delta]_+$.
8: Update the model's parameters θ by minimizing $L(\theta, \delta) = L_{\text{CE}}(\theta) + \lambda\, L_{\text{CP}}(\theta, \delta), \qquad \lambda > 0.$
9: **until** training converges
10: **Output:** Fine-tuned model g.

i.e., $[a]_+ = \max\{a, 0\}$; and (iv) $\psi(p_y, \delta) = \mathbf{1}(p_y \geq \delta)$, i.e., $\psi(p_y, \delta) = 1$ if $p_y \geq \delta$ and, therefore, the correct action is included in the prediction set, and $\psi(p_y, \delta) = 0$ otherwise. By definition of $\psi(p_y, \delta)$, we have that L_{CP} does not introduce penalties for those training examples $(\ell, y) \in \mathcal{B}$ where $\psi(p_y, \delta) = 0$, i.e., y is not in the prediction set. Similarly, L_{CP} introduces penalties only when (a) $\psi(p_y, \delta) = 1$, i.e., y is included in the prediction set and (b) $[p_{\max} - \delta]_+ \geq 0$ meaning that the strongest rival action is included in the set (and, therefore, the set is non-singleton). In other words, as discussed earlier in this section, L_{CP} introduces penalties only for non-singleton sets containing the true action. Then θ is update with the loss computed in (3) [line 8, Alg. 2].

A challenge in the above definition of L_{CP} is that it is not differentiable due to ψ. Thus, to enable gradient-based optimization, we use a smooth sigmoid approximation:

$$\psi(p_y, \delta) = \sigma(T(p_y - \delta)), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Observe that as $T \to \infty$, $\psi(p_y, \delta)$ converges to the above-mentioned step function $\mathbf{1}(p_y \geq \delta)$.

4. Experiments

In this section, we evaluate the effectiveness of CoFineLLM on language-instructed robot planning problems. Section 4.1 describes the setup of our algorithm and baselines. Section 4.2 presents comparative experiments showing that our method outperforms both uncertainty-agnostic and uncertainty-informed (Stutz et al., 2022) baselines across various metrics, such as user help rates. Section 4.3 provides ablation studies demonstrating that CoFineLLM generalizes well to unseen test-time coverage thresholds differing from the one used during finetuning. Finally, Section 4.4 empirically demonstrates the robustness of our method in out-of-distribution (OOD) hardware experiments.

4.1. Experiment Setup: Distribution of Mission Scenarios, Baselines, & Evaluation Metrics

Distribution: We build on the BabyAI-Text environment (Carta et al., 2023), a language-conditioned grid-world for grounded instruction following. The distribution $\mathcal{D}$ generates scenarios ξ as follows. *(a) Environment:* The distribution $\mathcal{D}$ produces random grid layouts using the BabyAI (Chevalier-Boisvert et al., 2018) procedural generator. Each environment is an 8×8 grid world composed of a 6×6 navigable workspace surrounded by a one-cell-thick wall boundary (see Fig. 1). The outer wall cells are fixed and impassable, while the inner 6×6 region contains the agent and objects. The environment is populated with objects belonging to four semantic categories: keys, balls, boxes, and doors. The number, color, and placement of objects for each category are sampled uniformly. Both the agent and the objects are initialized at random collision-free positions at the start of each episode. *(b) Action Space*: The action space of the agent consists of six actions, i.e., $\mathcal{A} = \{a_i\}_{i=1}^{6}$, where

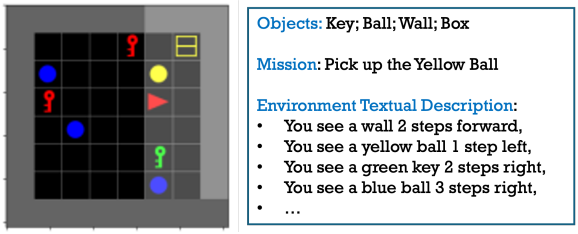

Figure 1: Example environment from the BabyAI-Text simulator. The agent (red triangle) operates in a grid world with colored keys, balls, boxes, and walls, and receives NL mission (e.g., "pick up the yellow ball"). The simulator provides textual descriptions listing objects and their relative positions (e.g., "You see a yellow ball 1 step left") for decision-making.

$a_1 = \texttt{turn left}$, $a_2 = \texttt{turn right}$, $a_3 = \texttt{go forward}$, $a_4 = \texttt{pick up}$, $a_5 = \texttt{drop}$, $a_6 = \texttt{toggle}$. These six primitive actions enable the agent to move, manipulate objects, and interact with the environment. *(c) Tasks*: Tasks are sampled uniformly from four categories of increasing complexity: (i) *GoTo* — navigate to a randomly selected object in the environment; (ii) *PickUp* — navigate to and pick up a randomly selected object; (iii) *PickUpThenGoTo* — pick up one randomly selected object and then navigate to another randomly selected object; (iv) *PutNext* — pick up a randomly selected object and place it next to another randomly selected goal object. Each mission is expressed in natural language (e.g., "put the yellow ball next to the red key").

Prompt Description: As discussed in Section 2, the prompt $\ell(t)$, corresponding to the question of the MCQA problem at time t, includes the following components: (1) *System description:* the symbolic action space and the decision objective; (2) *Environment description:* the relative position of the agent with respect to semantic objects at time t (e.g., 'You see a yellow ball 1 step left'); (3) *Task description:* the natural-language mission ϕ; (4) *Response format:* instructions specifying how the LLM should output its decision; (5) *Action history:* the sequence of past decisions up to time $t-1$; and (6) *Step index:* the current time step t. Appendix A shows an example prompt.

Training, Calibration, and Validation Datasets: The training ($\mathcal{M}_{\text{train}}$), calibration ($\mathcal{M}_{\text{cal}}$), and validation datasets are constructed by sampling $4{,}000$, 400, and 400 mission scenarios, respectively, from $\mathcal{D}$. The training and calibration datasets are used during fine-tuning by Alg. 2, while the validation dataset is reserved exclusively for evaluating the fine-tuned model.

Models and Training Setup: In our evaluation, we consider the Gemma3-1B model (Team et al., 2025) finetuned using LoRA (Hu et al., 2022) adapters while keeping the pretrained backbone frozen. This setup preserves the base model's language understanding while enabling lightweight adaptation for language-instructed robot planning tasks. To enhance the stability of fine-tuning, we also employ a *phase-based curriculum learning* strategy that progressively introduces more complex mission categories while retaining a subset of simpler ones to mitigate catastrophic forgetting. This enables the LLM-based planner to first master short-horizon tasks before progressing to longer-horizon, compositional missions. All implementation details—including training hardware, LoRA configuration, optimizer settings, learning rate, batch sizes, and phase schedules—are provided in Appendix B. This fine-tuning framework is used to optimize our proposed loss function (3). Alternative fine-tuning frameworks could also be adopted.

Evaluation Metrics. We evaluate the models using 4 metrics: *(i) Set size:* the average size of prediction sets $\mathcal{C}(t)$ across all decision steps and validation scenarios. *(ii) Help rate:* the fraction of individual decision steps, aggregated across all validation missions, at which human assistance is requested (i.e., $|\mathcal{C}(t)| > 1$); *(iii) Coverage rate:* the percentage of validation scenarios in which the

Table 1: Comparative evaluations using $1-\alpha$=0.95 during both finetuning and validation.

Method	**Set Size** $\downarrow$	**Help Rate** $\downarrow$	**Coverage**	**Verification Rate** $\uparrow$
Pre-trained	6.000	100%	100%	0%
UI	6.000	100%	100%	0%
UA	1.138	12.7%	98.7%	41.2%
Ours (λ=0.1)	**1.109** (-2.54%)	**9.7%** (-23.67%)	98.7% (+0%)	**49.8%** (+21.05%)

LLM-based planner generates a correct plan while asking for help from a user only when $|\mathcal{C}(t)| > 1$; *(iv) Verification rate:* the fraction of validation scenarios for which the planner designs the entire plan without asking for help, i.e., all prediction sets are singletons. All metrics are averaged over 3 independent runs. In cases where multiple feasible plans exist at test time, we apply CP following the approach proposed in Ren et al. (2023).

Baselines: We consider two baselines to evaluate the effectiveness of our method. To ensure fair comparisons, both baselines share the same training setup and fine-tuning process described in Algorithm 2; they differ from our approach only in the loss function they optimize. *Uncertainty-agnostic (UA) baseline:* This baseline optimizes only the cross-entropy loss, ignoring prediction set efficiency entirely. It can be recovered from our formulation in (3) by setting $\lambda = 0$. *Uncertainty-informed (UI) baseline:* This baseline optimizes the ConfTr loss introduced in (Stutz et al., 2022) for finetuning image classifiers. For each option (or label) k, a soft set-membership score is defined as $C_k(\theta) = \sigma((p_k - \delta)/\beta)$, where p_k is the model's predicted probability for option k (that depends on the model's parameters θ), δ is the confidence threshold generated by Alg. 1, β controls the sharpness of the sigmoid $\sigma(\cdot)$, and $C_y(\theta)$ denotes the score for the ground-truth option y. Then, the ConfTr loss function is defined as $L_{\text{UI}}(\theta, \delta) = \log\big((1 - C_y(\theta)) + \sum_{k \neq y} C_k(\theta) + \lambda \max(0, \sum_k C_k(\theta) - 1)\big)$ encouraging inclusion of the true class while penalizing unnecessarily large prediction sets. In our experiments, following (Stutz et al., 2022), we set $\beta = 0.1$. Observe, that unlike our formulation, this loss does not include a cross-entropy term; instead, it simultaneously penalizes large prediction set sizes, even when the correct action is absent, and prediction sets that fail to contain the correct action. We also report the performance of the *pre-trained* model.

4.2. Comparative Evaluation in In-Distribution Mission Scenarios

In this section, we present comparative numerical evaluations of our method and both baselines for a target coverage level of $1 - \alpha = 0.95$ during fine-tuning and validation; see Table 1. The loss functions of our method and the UI baseline are both configured using this coverage level. In our loss function (eq. 3), we set $\lambda = 0.1$. Before fine-tuning, the pre-trained model performs poorly on the considered planning tasks. For instance, its help rate is 100%, i.e., human assistance is required at every decision step across all validation missions to ensure a coverage rate of 95%; see the first row in Table 1. We also observed that the predictions of the pre-trained model are overconfident yet wrong in the sense that it assigned a large confidence score to a wrong action across all validation MCQA problems. After fine-tuning, CoFineLLM yields consistent improvements over both baselines across all evaluation metrics. Notably, relative to the UA baseline, the help rate decreases by 23.67% while the verification rate increases by 21.05%, indicating a larger fraction of missions completed without human intervention; see Table 1. The UI baseline failed to learn any useful pattern and is stuck at 100% help rate with an average prediction size of 6 (i.e., $|\mathcal{A}|$). We observed empirically that this issue is not solely due to low initial accuracy but rather to the pretrained LLM producing overconfident yet incorrect predictions early in training. When a confidence score p_k

Table 2: Evaluation of finetuned model on unseen test-time coverage thresholds $1-\alpha$.

$1-\alpha$	Method	**Set Size**↓	**Help Rate**↓	**Coverage**	**Verification Rate**↑
85%	UA	1.037	3.7%	96.1%	76.3%
	Ours	**1.010** (-2.71%)	**1.0%** (-72.45%)	**97%** (+0.95%)	**92.2%** (+20.87%)
90%	UA	1.077	7.2%	97.3%	60%
	Ours	**1.046** (-2.76%)	**4.4%** (-39.20%)	**97.6%** (+0.26%)	**70.3%** (+17.22%)
96%	UA	1.159	14.38%	98.8%	37.8%
	Ours	**1.154** (-0.45%)	**13.3%** (-7.36%)	**99%** (+0.17%)	**38.3%** (+1.10%)

Table 3: Evaluation under various values for λ with $1-\alpha$=0.95 during finetuning and validation.

Method	**Set Size**↓	**Help Rate**↓	**Coverage**	**Verification Rate**↑
UA	1.138	12.7%	98.7%	41.2%
λ=0.5	**1.131** (-0.63%)	**11.5%** (-9.82%)	**98.8%** (+0.17%)	**46.2%** (+12.15%)
λ=0.3	**1.110** (-2.54%)	**10.1%** (-20.55%)	98.7% (+0%)	**48.7%** (+18.27%)
λ=0.1	**1.109** (-2.56%)	**9.7%** (-23.67%)	98.7% (+0%)	**49.8%** (+21.05%)

is either close to 1 or 0 , the corresponding C_k quickly saturates near 1 or 0 under small changes in the model's parameters. Consequently, the loss landscape becomes nearly flat with respect to θ, yielding vanishing gradients and preventing effective learning. This issue does not arise in our setup particularly because of the cross-entropy term in (3). The coverage rate for all fine-tuned models remains close to the desired $1-\alpha = 0.95$ across all baselines as expected due to the theoretical guarantees in (Ren et al., 2023; Wang et al., 2025b).

4.3. Ablation Studies: Effect of Test-time Coverage Threshold $1-\alpha$ and Parameter λ

Generalization to unseen test-time coverage level $1-\alpha$. Here we evaluate the fine-tuned model from Section 4.2 (trained with $1-\alpha$=0.95 and λ=0.1) under test-time coverage targets $1-\alpha \in \{0.85, 0.90, 0.96\}$, which differ from the value used during fine-tuning. Comparative results against the finetuned model obtained in Section 4.2 by the UA baseline are summarized in Table 2. Our method consistently outperforms the UA baseline across all coverage levels. For example, at a lower coverage threshold ($1-\alpha$=0.85), the help rate decreases by 72.45% and the verification improves by 20.87%. At a stricter coverage level of $1-\alpha$=0.96, it reduces help rate by 7.36% and increases verification rate by 1.1%. These results demonstrate that the benefits of conformal-aware fine-tuning persist even when the test-time coverage requirement differs from the one used during training, highlighting strong generalization of uncertainty calibration.

Sensitivity to λ. We fix the coverage level to $1-\alpha$=0.95 during both finetuning/calibration and validation, and vary the penalty coefficient $\lambda \in \{0.1, 0.3, 0.5\}$. As shown in Table 3, our method outperforms the UA baseline across all settings. The best performance is attained at λ=0.1, which reduces prediction-set size by 2.56%, lowers help rate by 23.67%, and boosts verification rate by 21.05%. An intermediate penalty (λ=0.3) also yields strong improvements, reducing help rate by 20.55% and increasing verification rate by 21.05%. A larger penalty (λ=0.5) still improves verification but provides slightly weaker overall gains. Overall, these results show that our approach is robust to the choice of λ and consistently improves prediction-set size and help rates.

Figure 2: Evaluation in a physical environment with OOD scenarios sampled from $\mathcal{D}'$. The robot receives the NL instruction 'put the fire hydrant next to the red car.' The left panel shows the 3×5 grid abstraction used for planning, where each cell represents a discrete navigation step. Using this representation, the LLM planner generates τ directing the robot to turn, move to the hydrant, pick it up, move to the red car, and place the hydrant next to it.

Table 4: Evaluation in OOD scenarios using $1-\alpha$=0.95 during finetuning and validation.

Method	**Set Size** ↓	**Help Rate** ↓	**Coverage**	**Verification Rate** ↑
UA	1.131	11.7%	95.1%	58.9%
Ours (λ=0.1)	**1.072** (-5.22%)	**6%** (-48.67%)	**98.1%** (+3.16%)	**71.7%** (+21.70%)

4.4. Hardware Experiments & Comparative Evaluations in OOD Mission Scenarios

OOD Validation Scenarios: In this section, we evaluate the finetuned model obtained in Section 4.2 on OOD mission scenarios. Specifically, we consider cases where validation scenarios are sampled from a distribution $\mathcal{D}'$ that differs from the distribution $\mathcal{D}$, used during calibration and finetuning, in the environments it generates. The distribution $\mathcal{D}'$ generates 3×5 grid-world environments populated with traffic cones, fire hydrants, and cars, where cars may appear in one of six possible colors. Both $\mathcal{D}$ and $\mathcal{D}'$ generate tasks from the same four mission categories, but the tasks are defined over different objects. The distribution $\mathcal{D}'$ is not accessible and thus cannot be used for calibration or finetuning. Thus, calibration data during both finetuning and test time, the LLM-based planner are sampled from $\mathcal{D}$ while validation mission scenarios are generated by $\mathcal{D}'$. The coverage threshold during both finetuning and test time is set to 0.95.

Comparative Results: The comparative results over 80 validation mission scenarios generated by $\mathcal{D}'$ against the UA baseline are summarized in Table 4. Using the threshold δ calibrated using data from $\mathcal{D}$, CoFineLLM substantially improves help and verification rates. Specifically, the verification rate rises by +21.7% and help rate drops by 48.67% compared to the UA baseline. Interestingly, models finetuned by both methods preserve the $1-\alpha$ coverage level despite the distribution shift.

Hardware Demonstration: The plans generated by the finetuned LLM-based planner are demonstrated on real robot platform TurtleBot3 Waffle-Pi equipped with an OpenManipulator-X arm (OpenRobotics, 2022) in a miniature urban environment modeled as 3×5 grid world, populated with traffic cones, fire hydrants, and cars. Snapshots from such a demonstration are provided in Fig. 2.

5. Conclusion

This paper introduced the first conformal-aware finetuning framework for LLMs used for language-instructed robot planning. Our method reduces the size of prediction sets generated by CP at test time, thereby decreasing the need for user intervention. Our experiments demonstrate that the proposed method consistently outperforms baselines in terms of prediction-set size and help rate.

Acknowledgments

This work was supported in part by the NSF award CNS #2231257.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *Conference on Robot Learning (CoRL)*, 2022.

Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *Foundations and Trends in Machine Learning*, 2023. doi: 10.1561/2200000101.

Anthony Bellotti. Optimized conformal classification using gradient descent approximation. *arXiv preprint arXiv:2105.11255*, 2021.

Margarida Campos, António Farinhas, Chrysoula Zerva, Mário AT Figueiredo, and André FT Martins. Conformal prediction for natural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 12:1497–1516, 2024.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023.

Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. In *The International Conference on Learning Representations (ICLR)*, 2024.

John Cherian, Isaac Gibbs, and Emmanuel Candes. Large language model validity via enhanced conformal prediction methods. *Advances in Neural Information Processing Systems*, 37:114812–114842, 2024.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

Nicolo Colombo and Vladimir Vovk. Training conformal predictors. In *Proceedings of the Ninth Symposium on Conformal and Probabilistic Prediction and Applications*, volume 128 of *Proceedings of Machine Learning Research*, pages 55–64. PMLR, 09–11 Sep 2020.

Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. Task and motion planning with large language models for object rearrangement. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2086–2092. IEEE, 2023.

Kunal Garg, Jacob Arkin, Songyuan Zhang, Nicholas Roy, and Chuchu Fan. Large language models to the rescue: Deadlock resolution in multi-robot systems. *CoRR*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Lianghuan Huang, Sagnik Anupam, Insup Lee, Shuo Li, and Osbert Bastani. Rapid: An efficient reinforcement learning algorithm for small language models. *arXiv preprint arXiv:2510.03515*, 2025.

Disha Kamale and Cristian-Ioan Vasile. Optimal control synthesis with relaxed global temporal logic specifications for homogeneous multi-robot teams. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 250–256. IEEE, 2024.

Yiannis Kantaros and Michael M Zavlanos. Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems. *The International Journal of Robotics Research*, 39(7):812–836, 2020.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

Bhawesh Kumar, Charlie Lu, Gauri Gupta, Anil Palepu, David Bellamy, Ramesh Raskar, and Andrew Beam. Conformal prediction with large language models for multi-choice question answering. In *ICML 2023 Workshop on Neural Conversational AI (TEACH)*, 2023. arXiv:2305.18404.

Peihan Li and Lifeng Zhou. Llm-flock: Decentralized multi-robot flocking via large language models and influence-based consensus. *arXiv preprint arXiv:2505.06513*, 2025.

Kaiqu Liang, Zixu Zhang, and Jaime F Fisac. Introspective planning: Aligning robots' uncertainty with inherent task ambiguity. *Advances in Neural Information Processing Systems*, 37:71998–72031, 2024.

Zhexiao Lin, Yuanyuan Li, Neeraj Sarna, Yuanyuan Gao, and Michael von Gablenz. Domain-shift-aware conformal prediction for large language models. *arXiv preprint arXiv:2510.05566*, 2025.

Lars Lindemann, Yiqi Zhao, Xinyi Yu, George J Pappas, and Jyotirmoy V Deshmukh. Formal verification and control with conformal prediction. *arXiv preprint arXiv:2409.00536*, 2024.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. In *NeurIPS 2023 Workshop on Instruction Tuning and Language-Guided Robotics*, 2023.

Sima Noorani, Orlando Romero, Nicolo Dal Fabbro, Hamed Hassani, and George J Pappas. Conformal risk minimization with variance reduction. *arXiv preprint arXiv:2411.01696*, 2024.

OpenRobotics. Turtlebot e-manual, 2022. URL https://emanual.robotis.com.

Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning. In *7th Annual Conference on Robot Learning*, 2023.

Zachary Ravichandran, Fernando Cladera, Jason Hughes, Varun Murali, M Ani Hsieh, George J Pappas, Camillo J Taylor, and Vijay Kumar. Deploying foundation model-enabled air and ground robots in the field: Challenges and opportunities. In *ICRA 2025 Workshop on Field Robotics*, 2025. arXiv:2505.09477.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners. *Conference on Robot Learning*, 2023.

Rewat Sachdeva, Raghav Gakhar, Sharad Awasthi, Kavinder Singh, Ashutosh Pandey, and Anil Singh Parihar. Uncertainty and noise aware decision making for autonomous vehicles-a bayesian approach. *IEEE Transactions on Vehicular Technology*, 2024.

Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. Multirobot coordination with counting temporal logics. *IEEE Transactions on Robotics*, 36(4):1189–1206, 2019.

David Schwartz, Kota Kondo, and Jonathan P How. Efficient navigation in unknown indoor environments with vision-language models. In *IROS 2025 Workshop on Open-World Navigation (OWN)*, 2025. arXiv:2510.04991.

Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on Robot Learning*, pages 492–504. PMLR, 2023.

Guangyao Shi, Yuwei Wu, Vijay Kumar, and Gaurav S Sukhatme. Pip-llm: Integrating pddl-integer programming with llms for coordinating multi-robot teams using natural language. *arXiv preprint arXiv:2510.22784*, 2025.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530, 2023.

Jared Strader, Aaron Ray, Jacob Arkin, Mason B Peterson, Yun Chang, Nathan Hughes, Christopher Bradley, Yi Xuan Jia, Carlos Nieto-Granda, Rajat Talak, et al. Language-grounded hierarchical planning and execution with multi-robot 3d scene graphs. In *International Symposium on Experimental Robotics (ISER)*, 2025. arXiv:2506.07454.

David Stutz, Ali Taylan Cemgil, and Arnaud Doucet. Learning optimal conformal classifiers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Jiayuan Su, Jing Luo, Hongwei Wang, and Lu Cheng. Api is enough: Conformal prediction for large language models without logit-access. *arXiv preprint arXiv:2403.01216*, 2024.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Harit Vishwakarma, Alan Mishler, Thomas Cook, Niccolo Dalmasso, Natraj Raman, and Sumitra Ganesh. Improving decision-making in open-world agents with conformal prediction and monty hall. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.

Harit Vishwakarma, Alan Mishler, Thomas Cook, Niccolo Dalmasso, Natraj Raman, and Sumitra Ganesh. Prune'n predict: Optimizing llm decision-making with conformal prediction. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Eleftherios E Vlahakis, Lars Lindemann, and Dimos V Dimarogonas. Conformal data-driven control of stochastic multi-agent systems under collaborative signal temporal logic specifications. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2025. arXiv:2504.04615.

Jun Wang, Hosein Hasanbeig, Kaiyuan Tan, Zihe Sun, and Yiannis Kantaros. Mission-driven exploration for accelerated deep reinforcement learning with temporal logic task specifications. In *Proceedings of the 7th Annual Learning for Dynamics & Control Conference*, volume 283 of *Proceedings of Machine Learning Research*, pages 763–776. PMLR, 04–06 Jun 2025a.

Jun Wang, Guocheng He, and Yiannis Kantaros. Probabilistically correct language-based multi-robot planning using conformal prediction. *IEEE Robotics and Automation Letters*, 10(1):160–167, 2025b. doi: 10.1109/LRA.2024.3504233.

Jun Wang, David Smith Sundarsingh, Jyotirmoy V Deshmukh, and Yiannis Kantaros. Conformalnl2ltl: Translating natural language instructions into temporal logic formulas with conformal correctness guarantees. *arXiv preprint arXiv:2504.21022*, 2025c.

Jun Wang, Jiaming Tong, Kaiyuan Tan, Yevgeniy Vorobeychik, and Yiannis Kantaros. Conformal temporal logic planning using large language models. *ACM Transactions on Cyber-Physical Systems*, September 2025d. ISSN 2378-962X. doi: 10.1145/3769111.

Shu Wang, Muzhi Han, Ziyuan Jiao, Zeyu Zhang, Ying Nian Wu, Song-Chun Zhu, and Hangxin Liu. Llm^3: Large language model-based task and motion planning with motion failure reasoning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

Ting Wang, Zhixin Zhou, and Rui Luo. Enhancing trustworthiness of graph neural networks with rank-based conformal training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 21261–21268, 2025e.

Shaojun Xu, Xusheng Luo, Yutong Huang, Letian Leng, Ruixuan Liu, and Changliu Liu. Nl2hltl2plan: Scaling up natural language understanding for multi-robots through hierarchical temporal logic task specifications. *IEEE Robotics and Automation Letters*, 2025.

Ge Yan, Yaniv Romano, and Tsui-Wei Weng. Provably robust conformal prediction with improved efficiency. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Zejun Yang, Li Ning, Haitao Wang, Tianyu Jiang, Shaolin Zhang, Shaowei Cui, Hao Jiang, Chunpeng Li, Shuo Wang, and Zhaoqi Wang. Text2reaction: Enabling reactive task planning using large language models. *IEEE Robotics and Automation Letters*, 2024.

Bowen Ye, Jianing Zhao, Shaoyuan Li, and Xiang Yin. Prioritize team actions: Multi-agent temporal logic task planning with ordering constraints. In *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, pages 2840–2845. IEEE, 2024.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *Proceedings of the Conference on Language Modeling (COLM)*, 2025.

Yuqing Zhang and Yiannis Kantaros. Namo-llm: Efficient navigation among movable obstacles with large language model guidance. *IEEE Robotics and Automation Letters*, 2025.

Weitao Zhou, Zhong Cao, Nanshan Deng, Kun Jiang, and Diange Yang. Identify, estimate and bound the uncertainty of reinforcement learning for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 24(8):7932–7942, 2023.

Appendix A. Example Prompt and LLM Scoring Function g

For completeness, we show an example of the complete prompt used to query the LLM planner during fine-tuning and evaluation. Each prompt encodes (i) the action space and response format, (ii) the mission goal, (iii) the current textual observation, and (iv) the action history up to the current step. This structure transforms each decision point into a multiple-choice question where the LLM selects one symbolic action.

```
Select an action by its corresponding number:
0:  turn left
1:  turn right
2:  go forward
3:  pick up object
4:  drop object
5:  toggle

Goal of the agent:  go to a red ball after you pick up the grey key

Observation:  You see a wall 2 steps back and 1 step left.  You see
a wall 1 step back and 1 step left.  You see a wall 1 step left.
You see a wall 1 step forward and 1 step left.  You see a wall 2
steps forward and 1 step left.  You see a wall 3 steps forward and
1 step left.  You see a wall 4 steps forward and 1 step left.  You
see a wall 5 steps forward and 1 step left.  You see a wall 2 steps
back.  You see a wall 5 steps forward.  You see a wall 2 steps back
and 1 step right.  You see a red ball 4 steps forward and 1 step
right.  You see a wall 5 steps forward and 1 step right.  You see
a wall 2 steps back and 2 steps right.  You see a yellow key 1 step
forward and 2 steps right.  You see a yellow ball 2 steps forward
and 2 steps right.  You see a grey key 3 steps forward and 2 steps
```

```
right.  You see a wall 5 steps forward and 2 steps right.  You see
a wall 2 steps back and 3 steps right.  You see a red ball 1 step
forward and 3 steps right.  You see a green key 4 steps forward and
3 steps right.  You see a wall 5 steps forward and 3 steps right.
You see a wall 2 steps back and 4 steps right.  You see a red key
1 step back and 4 steps right.  You see a wall 5 steps forward and
4 steps right.  You see a wall 2 steps back and 5 steps right.  You
see a blue box 5 steps right.  You see a green box 2 steps forward
and 5 steps right.  You see a red ball 3 steps forward and 5 steps
right.  You see a wall 5 steps forward and 5 steps right.  You see
a wall 2 steps back and 6 steps right.  You see a wall 1 step back
and 6 steps right.  You see a wall 6 steps right.  You see a wall
1 step forward and 6 steps right.  You see a wall 2 steps forward
and 6 steps right.  You see a wall 3 steps forward and 6 steps right.
You see a wall 4 steps forward and 6 steps right.  You see a wall
5 steps forward and 6 steps right.

Previous Steps:

Your next action (choose number):
Action:
```

This example corresponds to a mid-episode decision in the *PickUpThenGoTo* mission family. We apply a similar idea to extract scores as proposed in Ren et al. (2023). At each time step t, the LLM-based planner is formulated as a multiple-choice question-answering problem. Given the current mission description, environment state, and action history, the LLM is prompted to select the next action from a fixed discrete action set. Each action is represented in the prompt by a numeric identifier (e.g., '0', '1', ...), forming an action-number pair (e.g., '0: turn left') in the prompt. Specifically, the model does not generate free-form text; instead, we extract the model's logits at the final token position, and only the logits corresponding to the discrete action tokens (e.g., "0", "1", ..., representing the available actions) are retained. A Softmax operation is then applied over these selected logits to obtain a probability distribution over the action set. These normalized probabilities define the confidence scores g used by conformal prediction. During training, only the observation and history fields evolve over time, while the system description and response format remain constant across steps.

Appendix B. Implementation Hyperparameters

Table 5 summarizes the core hyperparameters used for CP-aware fine-tuning. All models were trained with an RTX A100 80GB graphics card. We train with Adam at a learning rate of 5×10^{-5} and a batch size of 4, using LoRA adapters (rank 16, scaling 128). We target 95% conformal coverage and introduce the CP loss in phases, gradually increasing the number of calibration samples retained during training.

Table 5: Key hyperparameters used in CP-aware fine-tuning.

Hyperparameter	**Value**
Temperature (T)	Hard Indicator $\mathbf{1}(p_y \geq \delta)$
Period between updates of δ (K)	10
Batch Size	4
Learning rate	5e–5
Optimizer	Adam
LoRA rank (r)	16
LoRA scaling	128
Coverage level ($1 - \alpha$)	0.95
Phase start epochs	[1, 6, 11, 21]
Retained samples per phase	[100, 100, 500, 1000]