RAHF: Reward-Amplified Human Feedback for Closed-Loop Policy Fine-Tuning

Haoyuan Cai, Seth Z. Zhao, Jason Zhang, Bolei Zhou
Department of Computer Science,
University of California, Los Angeles

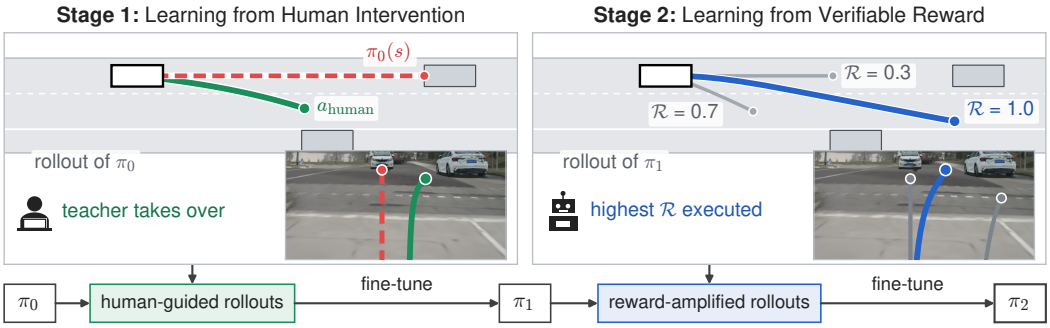


Figure 1: Overview of RAHF. RAHF fine-tunes the pretrained policy through two stages: human intervention at safety-critical states (Stage 1) and reward-amplified rollouts (Stage 2). The agent proposal of DrivoR [1] fails in a closed-loop simulation setting due to distribution shift, so we collect human corrections in Stage 1 and use them to fine-tune the base policy. The verifiable reward in Stage 2 evaluates the policy's candidate proposals and selects the best one for execution. We further fine-tune the policy on these reward-selected trajectories to obtain the final policy.

Abstract

Policies trained on offline logs can perform well in open-loop evaluation but poorly in closed-loop settings because they never learn to recover from their mistakes and compounding errors. Fine-tuning these policies typically requires either many human demonstrations or large-scale reinforcement learning rollouts, both of which are expensive to collect. We propose RAHF (Reward-Amplified Human Feedback), a two-stage closed-loop policy fine-tuning framework that reduces human effort by amplifying a small number of human corrections with a verifiable reward. In Stage 1, a human operator monitors the policy during closed-loop rollout and intervenes when the policy reaches dangerous states, producing a small but targeted correction dataset while keeping the training process safe. In Stage 2, the improved policy operates without human intervention. At each step, a verifiable reward scores the policy's candidate proposals. We use these reward scores to construct group-normalized advantages and fine-tune the policy toward the higher-advantage proposals. To prevent the policy from forgetting its pretrained knowledge, we propose a Policy Adaptation Module (PAM) that freezes the pretrained weights and learns closed-loop corrections through a separate trainable branch. On BridgeSim-NavHard, RAHF with a human teacher reaches a driving score of 78.90, surpassing all baselines including DAgger, HG-DAgger, GRPO, and AWR. In addition, PAM reduces forgetting and enables zero-shot transfer to HUGSIM without any additional fine-tuning[1].

[1]Demo video and code are available at `https://github.com/metadriverse/RAHF`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Enabling policies trained on offline demonstrations to perform well in closed-loop settings remains difficult across many domains, from autonomous driving to robotic manipulation and embodied agents [2–5]. Behavior cloning (BC) from expert logs can achieve strong open-loop performance, but these policies only learn from states visited by the expert. Expert trajectories alone cannot teach the policy to recover from mistakes or handle out-of-distribution states, leading to failures in closed-loop evaluation.

Current research typically addresses this gap through interactive imitation learning or reinforcement learning. DAgger [6] and HG-DAgger [7] rely on human interventions to generate corrective demonstrations. LEAD [8] and RoaD [9] replace the human with an automated expert. Although effective, these methods require a strong teacher and many rounds of data collection. Reinforcement learning methods fine-tune the policy with a task reward [10–12], where the exploration data enables recovery behaviors beyond expert demonstrations. These methods do not require a teacher but need large-scale on-policy rollouts to learn from. Moreover, none of these methods explicitly use the domain-specific knowledge already available in the target environment, such as safety constraints and traffic rules, which can provide cheap and dense feedback for policy improvement.

More broadly, closed-loop fine-tuning can be viewed as a post-training problem. Recent advances in RLHF show that limited human feedback, when combined with a reward signal, can produce policies that exceed the human's own performance. A similar opportunity exists in closed-loop policy improvement. Most environments already provide rule-based metrics such as collision detection and lane keeping that can deterministically score any candidate action without human involvement. These metrics are a form of *verifiable reward* [13]. They provide closed-loop safety and progress signals that are missing from open-loop pretraining and interactive imitation learning. Existing post-training methods use such metrics only for evaluation. We instead use them to amplify a small number of human corrections into high-quality training data, reducing the human effort required for closed-loop fine-tuning.

Motivated by this insight, we propose RAHF (Reward-Amplified Human Feedback), a two-stage algorithm that combines human intervention with a verifiable reward for closed-loop policy improvement (see Figure 1). A human correction is expensive, but scoring an action with the verifiable reward is cheap. After evaluating all action proposals from the policy, we can select the best ones as additional supervision without further human effort. Thus, RAHF is composed of the following two stages: In **Stage 1 (Learning from Human Intervention)**, we deploy the pretrained policy π_0 in a closed-loop setting, with a human operator intervening when π_0 makes mistakes. The human's presence also keeps data collection safe. We fine-tune π_0 on the resulting rollouts to obtain an improved policy π_1. In **Stage 2 (Learning from Verifiable Reward)**, we deploy π_1 without any human intervention. At each step, π_1 generates multiple proposals, and then the verifiable reward picks the best one for execution. We fine-tune π_1 on these rollouts to obtain the final policy π_2.

Closed-loop fine-tuning also brings the risk of catastrophic forgetting and environment-specific overfitting. After fine-tuning in a single environment, the policy's generalizability obtained from offline pretraining might decrease. That's because the resulting policy overfits to the environment's visual rendering and simplified traffic patterns. We mitigate this by designing a Policy Adaptation Module (PAM). PAM does not modify any pretrained weights in the base model π_0. Instead, we add a parallel trainable branch that learns the action corrections on top of the frozen policy. The frozen path preserves the general behavior learned from offline pretraining. The trainable branch incorporates closed-loop corrections from the new domain, such as recovery behaviors and reward-guided action selection. This design helps the fine-tuned model preserve the generalizability from pretraining and achieve zero-shot transfer to unseen environments (see Figure 3).

We evaluate RAHF on an end-to-end autonomous driving task, where closed-loop performance must balance safety, comfort, and progress across heterogeneous driving data. Using 6K closed-loop samples collected over two rounds, the baseline model DrivoR [1] fine-tuned by RAHF achieves a driving score of 69.27 on BridgeSim-NavHard with a neural teacher, and 78.90 with a real human teacher. The fine-tuned policy also transfers zero-shot to HUGSIM, a photorealistic environment based on 3D Gaussian Splatting, suggesting that the gains are not limited to the training environment (see Table 3). Our contributions are:

- We introduce RAHF, a two-stage closed-loop fine-tuning algorithm. Stage 1 collects a small set of human correction trajectories that keep training safe. Stage 2 uses the verifiable reward to select high-quality actions from the improved policy's proposals, eliminating further human involvement.
- We propose PAM, a policy adaptation module that freezes the pretrained policy and learns closed-loop corrections through a parallel trainable branch. PAM prevents catastrophic forgetting and enables zero-shot transfer to unseen environments.
- We evaluate RAHF with both neural and human teachers on BridgeSim-NavHard, and show consistent closed-loop improvement in both settings. We additionally compare against DAgger, HG-DAgger, and RL-style post-training baselines. We also evaluate RAHF in the Table Wiping task (see Table 5(b) and Figure 7) and show its effectiveness in robotics.

2 Related Work

Human-in-the-Loop Imitation Learning. Behavior cloning from expert trajectories only trains the agent on states visited by the expert, while the agent may fail to recover from its own mistakes at test time. DAgger addresses this by iteratively querying an expert on states visited by the learner and aggregating the resulting data to better match the induced distribution [6]. Follow-up works such as HG-DAgger [7] let the human decide when to take over based on the perceived risk of the current state. ThriftyDAgger [14] allocates a fixed query budget across an episode using novelty and risk estimates. PPL [15] moves beyond per-state action labels by eliciting preferences over forecasted future trajectories, helping reduce the cognitive burden on human supervisors. However, the final performance of these methods is fundamentally bounded by the quality of the human expert, and they require the human to participate across multiple rounds of demonstration until the student reaches near-optimal performance. In contrast, RAHF requires the human for only a single round of interaction, after which the verifiable reward fully replaces the human as the source of training supervision.

Policy Optimization via Reinforcement Learning. An alternative to expert supervision is to optimize policies using reward signals. RWR [16] and AWR [17] use advantage-weighted regression to improve policies from offline data. More recently, GRPO [13] introduces group-relative normalization for on-policy optimization without a learned critic. In autonomous driving, DiffusionDriveV2 [10] and AutoVLA [11] adopt GRPO-style objectives to fine-tune pretrained planners. These methods eliminate the need for human demonstrations. However, they require many on-policy rollouts for exploration, wasting samples when a pretrained base policy is already available. RAHF takes a different approach: instead of treating the reward as an optimization objective that requires exploration, we use it to select the best trajectory from a set of candidates already generated by the policy. This removes the need for extensive exploration and allows training with only a small number of closed-loop rollouts.

Policy Adaptation. Fine-tuning a pretrained policy in a new environment may result in catastrophic forgetting and reduce transferability [5]. LoRA [18], ControlNet [19], and recent robot navigation models S2E [20] keep the original weights frozen and learn task-specific corrections through lightweight parallel modules. Our Policy Adaptation Module (PAM) uses a frozen trajectory decoder combined with a trainable copy and zero-initialized layers. This ensures that the fine-tuned policy learns from human interventions and verifiable rewards without losing knowledge learned from pretraining. We show that PAM also helps the fine-tuned model generalize to unseen environments (see Section 5.4).

3 Preliminaries

We use the Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, d_0 \rangle$ to model the environment. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \to [R_{\min}, R_{\max}]$ is the reward function, $\gamma \in (0, 1)$ is the discount factor, and $d_0 : \mathcal{S} \to [0, 1]$ is the initial state distribution. Reinforcement learning (RL) aims to learn a policy $\pi(a|s)$ that maximizes

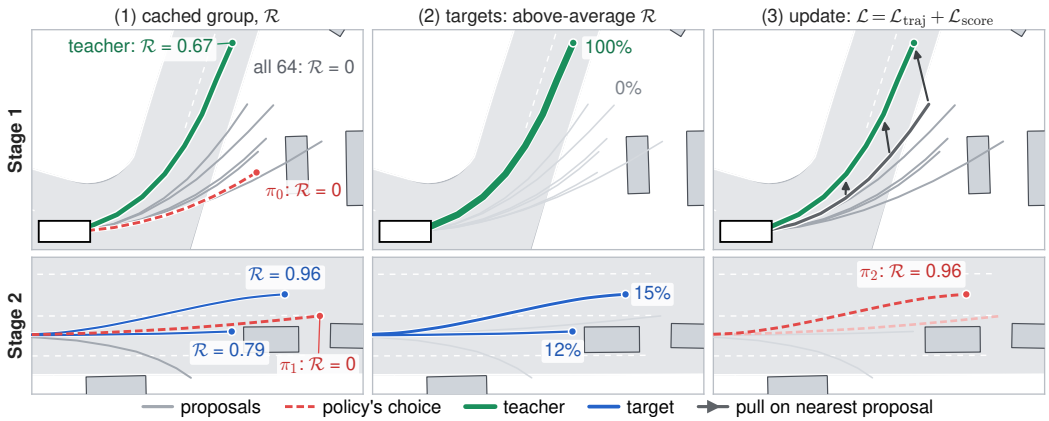


Figure 2: Fine-tuning objective of RAHF. (1) The verifiable reward $\mathcal{R}$ scores the cached candidates at each state, forming a group. In Stage 1, the human trajectory (green) is added to this group when takeover occurs; Stage 2 uses only policy proposals. (2) Candidates with above-average reward become training targets. Percentages show their normalized weights w_i in $\mathcal{L}_{\text{traj}}$, proportional to their reward above the group mean. (3) Both stages minimize $\mathcal{L} = \mathcal{L}_{\text{traj}} + \mathcal{L}_{\text{score}}$ (Eq. (1)-(2)). For each target, $\mathcal{L}_{\text{traj}}$ pulls the nearest current proposal toward it using a weighted L_1 loss. $\mathcal{L}_{\text{score}}$ trains the scorer f to predict $\mathcal{R}$ for all K proposals. These losses improve proposal generation and selection.

the expected cumulative return $J(\pi) = \mathop{\mathbb{E}}_{\tau \sim P_\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$, wherein $\tau = (s_0, a_0, s_1, a_1, ...)$ is the trajectory sampled from trajectory distribution P_π induced by π, d_0 and $\mathcal{P}$.

Multi-Proposal Policy Parameterization. We consider policies that generate a set of K candidate actions at each state and select one for execution. At each state s, the policy outputs action candidates $\{a_1, \ldots, a_K\}$ and a scoring function $f(s, a)$. We execute the action with the highest score, i.e., $\pi(s) = \arg\max_{i=1,2,\cdots,K} f(s, a_i)$. This multi-proposal formulation is common in recent end-to-end autonomous driving systems [1, 21, 22]. QT-Opt [23] and V-GPS [24] use the same structure in robotic manipulation. Also, the action space can be either low-level controls or higher-level plans such as future trajectories. Our method applies to any policy with this multi-proposal design.

Problem Setting. In our setting, we have access to a pretrained base policy as π_0, trained on a dataset $\mathcal{D}_{\text{pre}}$ collected from offline logs. Although π_0 performs well on $\mathcal{D}_{\text{pre}}$ under open-loop evaluation, it suffers from covariate shift and compounding errors in closed-loop deployment. Our goal is to improve the closed-loop performance of π_0 while preserving the knowledge learned from the pretraining data $\mathcal{D}_{\text{pre}}$.

We assume access to two additional sources of supervision during closed-loop fine-tuning. First, a *verifiable reward function* $\mathcal{R}(s, a) \in \mathbb{R}$ evaluates candidate actions using environment information (e.g., collision checking and distances to lane boundaries). Second, a human operator can intervene using their policy $\mathcal{T} : \mathcal{S} \to \mathcal{A}$ during closed-loop rollouts and provide corrective actions when the agent acts dangerously. We do not assume that the human teacher is perfect, and they may provide jerky demonstrations in safety-critical states. Our goal is to combine these two sources to efficiently fine-tune π_0 for closed-loop deployment, improving robustness while minimizing the number of human interventions and closed-loop rollouts required.

In our driving experiments, we use the EPDMS metrics [25] as the verifiable reward $\mathcal{R}$ (see Fig. 6 for illustration and Appendix A for full definitions). At test time, the policy selects actions with its learned scorer f, without accessing $\mathcal{R}$. RAHF also supports neural experts as a substitute for the human operator (see Sec. 4.1).

4 Method

We present RAHF (Reward-Amplified Human Feedback), a two-stage fine-tuning procedure for pretrained multi-proposal policies (see Figure 2). Starting from a base policy π_0, we first collect

Algorithm 1 RAHF: Reward-Amplified Human Feedback

Require: Pretrained policy π_0, pretraining data $\mathcal{D}_{\text{pre}}$, verifiable reward $\mathcal{R}$, human operator $\mathcal{T}$

$\mathcal{D} \leftarrow \mathcal{D}_{\text{pre}}$
// Stage 1: Learning from Human Intervention
for each state s visited by π_0 **do**
  Execute a_{human} if human takes over, otherwise execute $\pi_0(s)$.
  Add the group $\{(s,\ \{a_1, \ldots, a_K, a_{\text{human}}\})\}$ to $\mathcal{D}$ if human takes over,
  otherwise add the group $\{(s,\ \{a_1, \ldots, a_K\})\}$ to $\mathcal{D}$.
end for
$\pi_1 \leftarrow \texttt{Fine-Tune}(\pi_0,\ \mathcal{D})$

// Stage 2: Learning from Verifiable Reward ▷ No human involvement needed here
for each state s visited by π_1 **do**
  Generate K proposals $\{a_1, \ldots, a_K\}$ from $\pi_1(s)$, execute $\arg\max_i \mathcal{R}(s, a_i)$.
  Add the group $\{(s,\ \{a_1, \ldots, a_K\})\}$ to $\mathcal{D}$.
end for
$\pi_2 \leftarrow \texttt{Fine-Tune}(\pi_1,\ \mathcal{D})$
return π_2

human intervention data when the base policy makes mistakes in Stage 1. In Stage 2, we run the improved policy π_1 without any human involvement. Instead, we use the verifiable reward $\mathcal{R}$ to select the best proposal at each step, generating high-quality training data automatically. The full procedure is summarized in Algorithm 1.

Stage 1: Learning from Human Intervention. In the first stage, we run the base policy π_0 with a human operator $\mathcal{T}$ monitoring the rollouts. The human intervenes and takes control when π_0 is about to fail. When there is no intervention, π_0 selects the candidate with the highest score using its own scoring function $f(s, a)$. At takeover states, we add the human action to the K policy proposals. We fine-tune π_0 on these groups using normalized reward advantages to obtain π_1, rather than directly cloning the takeover action.

This stage serves two purposes. The human's interventions produce corrective demonstrations at failure states, teaching π_1 how to recover from situations not covered by offline pretraining. The human's presence also keeps data collection safe, preventing the agent from repeatedly crashing during rollouts.

Stage 2: Reward-Amplified Rollouts. In the second stage, no human involvement is needed. At each step, we use π_1 to generate K proposals. Instead of using the policy's own scoring function $f(s, a)$, we use the verifiable reward $\mathcal{R}$ to pick the best one and execute. We retain the candidate groups from these rollouts to train both the proposals and the learned scorer, obtaining π_2. Thus, reward-guided selection supplies a new state distribution, while the shared objective improves the policy on the collected data.

Discussion. We find both stages necessary for closed-loop fine-tuning. Stage 1 provides corrective demonstrations when the base policy makes mistakes, helping it recover from dangerous or out-of-distribution states. The human may be imperfect and can only provide a limited number of corrections, but the policy still learns recovery from failure cases. Stage 2 then uses the verifiable reward to choose safe and comfortable actions from the improved proposal set, producing reward-guided rollout data. We show in Section 5.5 that removing either stage leads to a significant performance drop.

4.1 Implementation Details

Training losses. During fine-tuning, we train the proposal model to produce better candidates. For each state $s \in \mathcal{D}$, we cache the proposals generated during rollout together with their verifiable reward values. Inspired by GRPO [13], we treat the cached candidates at each state as a group. Let $\mathcal{A}_s = \{a_1, \ldots, a_{N_s}\}$ denote this group, where $N_s = K$ in normal rollout states, and $N_s = K + 1$ only when a human intervention adds a_{human} to the policy's K proposals in Stage 1. We first compute per-group normalized advantages: $\hat{A}_i = (r_i - \bar{r}_s)/\sigma_{r,s}$, where $r_i = \mathcal{R}(s, a_i)$ and $\bar{r}_s, \sigma_{r,s}$ are the mean

and standard deviation of the N_s reward values at state s. We retain only candidates with positive advantage and normalize their weights: $w_i = \max(0, \hat{A}_i) / \sum_{j=1}^{N_s} \max(0, \hat{A}_j)$. The trajectory loss is:

$$\mathcal{L}_{\text{traj}} = \sum_{i=1}^{N_s} w_i \cdot \min_{j=1,\cdots,K} \|a_i^{\text{cache}} - a_j^{\theta}\|_1, \tag{1}$$

where a_i^{cache} is the i-th cached proposal from the rollout and a_j^{θ} is the j-th proposal from the current model. This loss encourages the model to cover all high-quality proposals rather than collapse onto a single target. Proposals with negative advantage receive zero weight and do not affect training, which helps preserve the diversity of the K proposals. We note that standard GRPO requires multiple rounds of on-policy explorations, while RAHF uses only cached proposals from two rounds of data collection.

For policies that include a learned scoring head f (Section 3), we additionally fine-tune f to better approximate the verifiable reward $\mathcal{R}$. Using the same K cached proposals at state s:

$$\mathcal{L}_{\text{score}} = \sum_{i=1}^{K} \left(f(s, a_i^{\text{cache}}) - \mathcal{R}(s, a_i^{\text{cache}}) \right)^2. \tag{2}$$

The total loss is $\mathcal{L} = \mathcal{L}_{\text{traj}} + \mathcal{L}_{\text{score}}$. In practice, $\mathcal{R}$ contains multiple components, and we train f to predict each component separately. We provide the details in Appendix A.2.

Human and neural teachers. RAHF is designed around human intervention in Stage 1, but also supports neural policy teachers as an alternative. Human teachers decide when to take over based on their own judgment. They monitor the agent on screen and use a gamepad to intervene whenever they anticipate a dangerous outcome. Their corrective trajectories are recorded as the teacher's actions.

When a human operator is not available, RAHF also supports neural policy teachers, such as OpenPilot [26]. The neural network teacher shares the same sensor inputs with the student, but is pretrained on large-scale real-world driving data together with closed-loop fine-tuning. For neural network teachers, the takeover is triggered when the student's selected proposal scores below a threshold under the verifiable reward, indicating an imminent safety violation.

Policy Adaptation Module (PAM). Fine-tuning a pretrained policy in a new environment may cause catastrophic forgetting. To prevent this, we freeze the perception backbone in the base policy π_0, and we propose the Policy Adaptation Module (PAM) to fine-tune its action planning head in Fig. 3. PAM adds a trainable copy of the planning head in parallel, connected via zero-initialized linear projections W_{in} and W_{out}. The output $\mathbf{y}$ is the sum of the frozen part's output and the trainable copy's output, i.e., $\mathbf{y} = \pi_0(\mathbf{x}) + W_{\text{out}} \cdot \pi_{\text{train}}(W_{\text{in}} \cdot \mathbf{x})$. Since W_{out} is initialized to zero, the trainable branch has no effect at the start of training, so the model begins exactly as the pretrained policy. As training progresses, the trainable branch learns how to correct the frozen copy's actions. Because the perception backbone remains frozen, PAM avoids overfitting to the visual characteristics of the training environment, which is essential for cross-environment generalization (see Section 5.4).

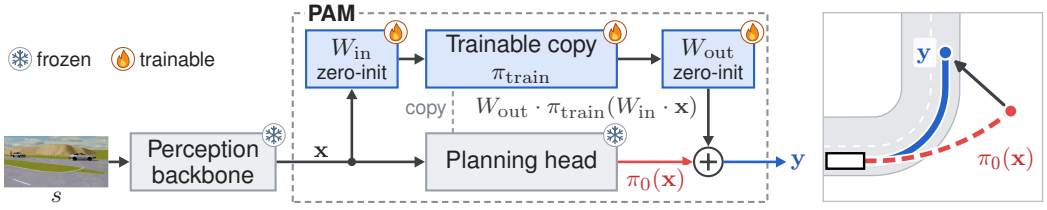


Figure 3: Illustration of Policy Adaptation Module (PAM). We freeze the perception backbone, while a parallel trainable branch of the action planner learns closed-loop corrections.

5 Experiments

5.1 Experimental Setup

We evaluate RAHF on closed-loop autonomous driving tasks in simulation. Many recent end-to-end driving models are pretrained on NAVSIM logs in an open-loop manner. However, in open-loop

evaluation, their future trajectory prediction does not affect their future states. Our goal is to adapt these pretrained policies to a closed-loop environment while preserving their open-loop knowledge.

We use BridgeSim [2], built on the MetaDrive [27] engine, for closed-loop training and evaluation. NAVSIM [25] provides large-scale real-world driving logs for pretraining and 421 held-out NavHard scenarios for testing. BridgeSim imports NAVSIM scenarios into MetaDrive and reconstructs them with consistent sensor configurations, vehicle dynamics, and evaluation protocols. This allows policies pretrained on NAVSIM to be evaluated in closed loop under the same scene distribution. We run MetaDrive at 10 Hz with a simulation timestep of Δt=0.1s. The policy replans at 2Hz (every 5 simulation frames), and each episode runs for at most 8 seconds. We provide implementation details of BridgeSim in Appendix A.

For RAHF, we use 500 NAVSIM training scenarios to collect closed-loop data across both stages. Stage 1 (Learning from Human Intervention) supports both human and neural teachers (see Sec. 4.1). Stage 2 (Reward-Amplified Rollouts) uses EPDMS [25] as the verifiable reward. We collect 3000 samples in each stage, giving 6K closed-loop samples in total. We train each interactive imitation learning baseline five times using distinct random seeds. Then we evaluate the final policies on 421 unseen NavHard test scenarios and report the average results.

Our main experiments use DrivoR [1] as the base policy π_0. DrivoR is a multi-proposal policy with a learned scoring head pretrained on NAVSIM. Results on additional base policies (LTF [28], DiffusionDrive [21]) are provided in Appendix B.1. In this section, we present (1) in-domain closed-loop evaluation on NavHard with both neural and human teachers (Table 1); (2) preservation of pretraining knowledge on open-loop NAVSIM evaluation (Table 2); (3) zero-shot cross-environment transfer to HUGSIM [29] (Table 3); and (4) ablation studies (Table 4).

All reward-based baselines use the same EPDMS reward, pretrained checkpoint, and 6K-sample budget. AWR and RWR also use the human-intervention data from Stage 1. At test time, action selection uses the learned scorer f. No evaluated policy accesses the verifiable reward $\mathcal{R}$.

Evaluation metrics. The primary closed-loop metric is the BridgeSim Driving Score (DS), defined as the product of Route Completion (RC) and the temporal average of the Extended Predictive Driver Model Score (EPDMS) [25]. EPDMS multiplies critical safety constraints (e.g., drivable area compliance (DAC)) with a weighted sum of driving quality features (lane keeping (LK), time-to-collision (TTC), history comfort (HC)). A high DS requires the agent to complete the route with a safe and comfortable trajectory. We refer the reader to Appendix A for detailed metric definitions. For open-loop evaluation, we report the standard NAVSIM PDMS score on 2000 NavTest held-out samples.

5.2 Main Results

Table 1 reports the main closed-loop results on BridgeSim-NavHard using DrivoR as the pretrained backbone. All fine-tuning methods start from the same pretrained checkpoint and use 6K closed-loop samples. The whole experiment of RAHF takes about half an hour on a desktop computer with an Nvidia GeForce RTX 4090 GPU.

The pretrained DrivoR policy only achieves a driving score (DS) of 43.02 in closed-loop evaluation. Imitation learning baselines' performance is bounded by their teacher's quality. DAgger reaches 61.72 DS with OpenPilot supervision, below the OpenPilot teacher's 65.36. Even with real human supervision, all the imitation learning baselines still fall far below the human teacher's performance (84.31). These methods learn only from teacher demonstrations and do not use the verifiable reward from the target environment, making them less effective under the limited 6K-sample budget. The RL post-training baselines use the verifiable reward to improve the base policy. GRPO collects on-policy rollouts using only the policy's own scoring function and does not require a teacher, reaching a DS of 63.62. AWR and RWR can additionally incorporate teacher demonstrations into their replay data, reaching 65.34 and 59.53 DS, respectively.

With OpenPilot as the teacher policy, RAHF reaches 69.27 DS, outperforming all baselines and surpassing the OpenPilot teacher itself. Compared with AWR, the strongest post-training baseline, RAHF improves DS from 65.34 to 69.27 even with a weaker teacher. Since AWR also weights its regression loss by the reward, this gap suggests that the gain comes mainly from executing the highest-reward proposal during Stage 2 rollouts, which yields better training data.

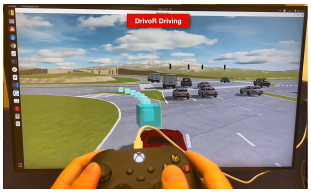
Table 1: Closed-loop evaluation on BridgeSim-NavHard (DrivoR backbone). All methods start from the same pretrained checkpoint and use 6K closed-loop samples. Our method RAHF achieves the highest Driving Score (DS) compared with all the baselines.

Method	Teacher-Used	DS	EPDMS	RC	DAC	TTC	LK	HC
Reference								
DrivoR (pretrained π_0)	–	43.02	66.54	64.66	76.34	95.41	94.37	96.99
OpenPilot [26]	–	65.36	79.50	82.21	94.29	92.58	84.43	92.12
Real Human	–	84.31	86.49	97.47	99.24	94.95	92.39	99.25
Imitation Learning Baselines (reward-free)								
Behavioral Cloning	Human	50.96	66.44	76.71	86.78	93.39	83.45	73.34
DAgger [6]	OpenPilot	61.72	73.90	83.51	88.91	90.10	90.35	88.32
HG-DAgger [7]	Human	62.92	72.44	86.87	82.12	94.28	94.57	81.88
PVP [30]	Human	64.04	76.64	83.55	94.49	88.34	89.09	90.07
RL Post-Training Baselines (reward-based)								
GRPO [13]	None	63.62	76.79	82.85	94.77	88.53	87.96	88.86
RWR [16]	Human	59.53	74.04	80.40	90.02	88.50	89.78	89.89
AWR [17]	Human	65.34	74.55	87.65	91.26	89.45	90.55	87.56
RAHF (Ours)								
RAHF	OpenPilot	**69.27**	**79.08**	87.60	93.95	92.73	93.70	88.22
RAHF	Human	**78.90**	**83.04**	**95.02**	**95.65**	93.23	**94.87**	**95.84**

Table 2: Open-loop performance on NavTest evaluated in NAVSIM open-loop protocol (PDMS). Full sub-metric breakdowns are in Appendix B.2.

Model	PDMS
DrivoR (native)	85.46
RAHF w/ PAM	**84.09**
RAHF w/o PAM	79.05
RAHF w/ full backbone FT	71.57

Figure 4: Illustration of our human interface. The human expert uses a gamepad to provide corrections.



With a real human teacher, RAHF reaches 78.90 DS, the highest score among all fine-tuned policies, but remains below the human teacher (84.31). The gain of closed-loop fine-tuning is especially large in Route Completion (RC) and Drivable Area Compliance (DAC). Compared with the pretrained DrivoR, our method improves RC from 64.66 to 95.02, and improves DAC from 76.34 to 95.65. These gains indicate that RAHF mainly improves closed-loop failure modes where the pretrained policy leaves drivable areas or fails to complete the route.

5.3 Preservation of Pretraining Knowledge

A practical concern with closed-loop fine-tuning is that it might degrade performance on the pretraining data distribution. Table 2 evaluates our RAHF-finetuned models under the NAVSIM open-loop protocol. RAHF with our Policy Adaptation Module (PAM) retains most of the original open-loop performance (84.09 vs. 85.46). Removing PAM drops the score to 79.05, and fine-tuning the full backbone including perception layers drops it further to 71.57. This confirms the effectiveness of PAM's residual design: the frozen planning head preserves general behavior, and the zero-initialized parallel branch learns closed-loop corrections without overwriting pretrained representations.

5.4 Generalization to Unseen Environment

We evaluate cross-simulator transfer in Table 3 and Figure 5. We train our RAHF agent in MetaDrive with a human teacher, and then deploy it directly in HUGSIM [29], a photorealistic closed-loop simulator based on 3D Gaussian Splatting. No HUGSIM data is used for fine-tuning. HUGSIM uses different rendering, physics, and evaluation protocols from MetaDrive. As shown in Table 3, RAHF improves over native DrivoR across all difficulty levels, with the largest gain on hard scenarios (+10.9). Removing PAM or fine-tuning the full backbone hurts transfer performance. This suggests that the PAM design avoids the closed-loop fine-tuning overfitting to MetaDrive-specific patterns.

Table 3: Zero-shot transfer to HUGSIM (3DGS-based photorealistic environment, HD-Score). RAHF-trained models generalize without any in-domain fine-tuning. Full sub-metric breakdowns are in Appendix B.3.

Model	Easy	Medium	Hard
DrivoR (native)	81.1	31.8	19.4
RAHF (DrivoR)	**84.2**	**33.8**	**30.3**
RAHF (w/o PAM)	66.5	28.5	20.7
RAHF (w/ full backbone FT)	57.3	26.7	14.5
Δ (RAHF vs. native)	+3.1	+2.0	+10.9

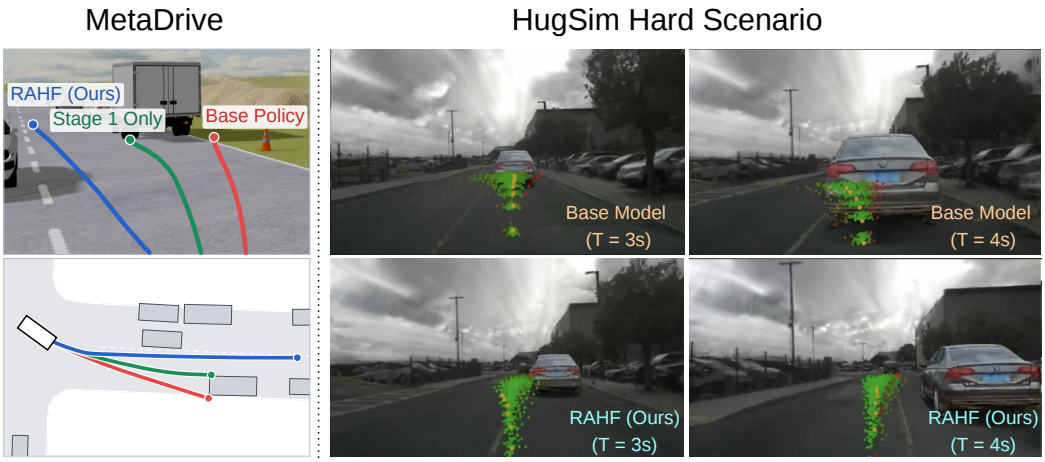


Figure 5: Qualitative results on MetaDrive (left) and HUGSIM (right). In MetaDrive, the Base Policy (red) and Stage 1 Only (green) crash into the other truck. RAHF (blue) follows a smooth and safe path. The BEV below shows the same scenario with lane context. In HUGSIM, the base model collides with the front vehicle at $T = 4s$. RAHF agent passes it safely from the left. RAHF is trained only in MetaDrive and transferred to HUGSIM without fine-tuning. The full episode is visualized in our demo video.

5.5 Ablation Studies

Stage decomposition. Table 4 isolates each stage's contribution. Applying reward-amplified rollouts directly on the base model π_0 (Stage 2 only) gains only a slight improvement (45.51 vs. base model's 43.02). Without Stage 1 fine-tuning, all K proposals from the base policy may lead to collisions in safety-critical states, so even the best-scoring one is not safe. Applying human intervention alone (Stage 1 only) reaches 58.31. The teacher's corrections help the policy recover from dangerous states, but the corrections can be imperfect (safe but uncomfortable or jerky), and the limited intervention budget of 3K samples cannot cover all failure modes. Combining both stages yields the best result (78.90): Stage 1 introduces recovery behaviors and improves the proposal set in hard scenarios, and Stage 2 selects the best actions from this improved set.

5.6 Human Effort and Robotic Manipulation

We next quantify human effort and evaluate the performance of RAHF on a robot manipulation task Table Wiping. Table 5 summarizes the results.

Human effort. In Table 5(a), we define Human-Involved Steps as the total number of steps when the human monitors the rollout, and Takeover Steps as the number of steps when the human intervenes and directly controls the agent. RAHF requires monitoring only for the 3K Stage 1 samples, halving the monitoring budget of HG-DAgger and PVP. Human takeover occurs in 27.9% of Stage 1 samples (approximately 0.83K); Stage 2 requires no human involvement. Although RAHF has a higher takeover rate than PVP, it requires fewer takeover steps overall because humans monitor only half

Table 4: Stage decomposition ablation on BridgeSim-NavHard (human teacher).

Configuration	DS	EPDMS	RC
Expert BC only (Human demos)	50.96	66.44	76.71
Stage 1 only (no reward amplification)	58.31	69.14	84.33
Stage 2 only (no human intervention)	45.51	68.15	66.78
RAHF (Stage 1 + Stage 2)	**78.90**	**83.04**	**95.02**

Table 5: Selected results on (a) human effort in BridgeSim-NavHard, with a 6K-sample budget, and (b) RoboSuite Table Wiping, with 5K samples per trainable method.

(a) Human effort

Method	Human-Involved Steps	Takeover Steps	DS
HG-DAgger	6K	1.58K	62.92
PVP	6K	1.29K	64.04
RAHF	**3K**	**0.83K**	**78.90**

(b) RoboSuite Table Wiping

Method	Success Rate	Task Completion
Human Expert	0.96	0.98
RWR	0.81	0.94
AWR	0.79	0.93
RAHF (Ours)	**0.93**	**0.97**

as many steps. The higher rate reflects a difference in data collection: RAHF keeps the base policy fixed throughout Stage 1, while HG-DAgger and PVP update their policies during collection. Human actions are recorded automatically during takeover, so no additional annotation is needed.

Robotic manipulation. In RoboSuite [31] Table Wiping, the robot must remove 100 markings within 500 steps. The policy maps a 34-dimensional state to four candidate 6-dimensional end-effector actions and uses a learned scorer. All trainable methods use 5K samples. In Table 5(b), Success Rate measures the fraction of episodes in which all markings are removed, while Task Completion measures the average fraction of markings removed. Over 100 evaluation episodes with distinct seeds, RAHF reaches a Success Rate of 0.93, compared with 0.81 for the strongest baseline, RWR, and 0.96 for the human expert. The Task Completion of RAHF is 0.97. This shows the effectiveness of RAHF beyond driving tasks. We provide the visualization of Table Wiping task and qualitative examples in Fig. 7.

Reward robustness. We perturb training rewards independently with uniform noise $\epsilon_i \sim \mathcal{U}[-\sigma, \sigma]$. Evaluation uses the original noise-free EPDMS, and the policy has no access to this reward at test time. With a human teacher, DS decreases from 78.90 to 72.40 at $\sigma = 0.25$, still above AWR's unperturbed 65.34, and it drops to 65.25 at $\sigma = 0.50$. Appendix B.5 reports full results of reward robustness, different takeover-threshold, and different proposal-count in Table 9, 10, 11, respectively.

6 Conclusion

We introduced RAHF, a two-stage algorithm for sample-efficient closed-loop policy fine-tuning. Stage 1 collects a small set of human corrections when the base policy makes mistakes, keeping the training process safe. Stage 2 uses a verifiable reward to select high-quality actions from the improved policy's proposals, amplifying the effect of the human corrections without any further human effort. Combined with PAM for adaptation with reduced forgetting, RAHF preserves open-loop capabilities and achieves zero-shot generalization across environments. Our results show that combining limited human intervention with a cheap verifiable reward is an effective path for closed-loop fine-tuning.

Limitations. RAHF requires a verifiable reward during training. In our driving experiments, reward computation uses privileged simulator information, while the policy relies only on its learned scorer at test time. Incomplete reward specifications may limit the learned policy's performance, and proposal selection cannot recover actions absent from the candidate set. RAHF targets small rollout budgets with human guidance, and conventional RL may be preferable when large amounts of on-policy data can be collected inexpensively. Extending RAHF to settings without verifiable rewards remains an important direction for future work.

Acknowledgments and Disclosure of Funding

This work was supported by NSF grants CCF-2344955, CNS-2235012, and IIS-2339769. Seth Z. Zhao was supported by Qualcomm Innovation Fellowship.

References

[1] Ellington Kirby, Alexandre Boulch, Yihong Xu, Yuan Yin, Gilles Puy, Eloi Zablocki, Andrei Bursuc, Spyros Gidaris, Renaud Marlet, Florent Bartoccioni, Anh-Quan Cao, Nermin Samet, Tuan-Hung Vu, and Matthieu Cord. Driving on registers. *preprint*, 2026.

[2] Seth Z Zhao, Luobin Wang, Hongwei Ruan, Yuxin Bao, Yilan Chen, Ziyang Leng, Abhijit Ravichandran, Honglin He, Zewei Zhou, Xu Han, et al. Bridgesim: Unveiling the ol-cl gap in end-to-end autonomous driving. *arXiv preprint arXiv:2604.10856*, 2026.

[3] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[4] Honglin He, Yukai Ma, Brad Squicciarini, Wayne Wu, and Bolei Zhou. Learning sidewalk autopilot from multi-scale imitation with corrective behavior expansion. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2026.

[5] Peter Karkus, Maximilian Igl, Yuxiao Chen, Kashyap Chitta, Jef Packer, Bertrand Douillard, Ran Tian, Alexander Naumann, Guillermo Garcia-Cobo, Shuhan Tan, et al. Beyond behavior cloning in autonomous driving: a survey of closed-loop training techniques. *Authorea Preprints*.

[6] Stéphane Ross, Geoffrey Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

[7] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[8] Long Nguyen, Micha Fauth, Bernhard Jaeger, Daniel Dauner, Maximilian Igl, Andreas Geiger, and Kashyap Chitta. Lead: Minimizing learner-expert asymmetry in end-to-end driving, 2026. URL `https://arxiv.org/abs/2512.20563`.

[9] Guillermo Garcia-Cobo, Maximilian Igl, Peter Karkus, Zhejun Zhang, Michael Watson, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Road: Rollouts as demonstrations for closed-loop supervised fine-tuning of autonomous driving policies. *arXiv preprint arXiv:2512.01993*, 2025.

[10] Jialv Zou, Shaoyu Chen, Bencheng Liao, Zhiyu Zheng, Yuehao Song, Lefei Zhang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Diffusiondrivev2: Reinforcement learning-constrained truncated diffusion modeling in end-to-end autonomous driving. *arXiv preprint arXiv:2512.07745*, 2025.

[11] Zewei Zhou, Tianhui Cai, Seth Z. Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[12] Chaojun Ni, Guosheng Zhao, Xiaofeng Wang, Zheng Zhu, Wenkang Qin, Xinze Chen, Guanghong Jia, Guan Huang, and Wenjun Mei. Recondreamer-rl: Enhancing reinforcement learning via diffusion-based scene reconstruction. *arXiv preprint arXiv:2508.08170*, 2025.

[13] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[14] Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S. Brown, and Ken Goldberg. ThriftyDAgger: Budget-aware novelty and risk gating for interactive imitation learning. In *5th Annual Conference on Robot Learning*, 2021. URL `https://openreview.net/forum?id=KKBfrCzCVOn`.

[15] Haoyuan Cai, Zhenghao Peng, and Bolei Zhou. Predictive preference learning from human interventions. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ErEaq1UNaQ.

[16] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007.

[17] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[18] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[19] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023.

[20] Honglin He, Yukai Ma, Wayne Wu, and Bolei Zhou. From seeing to experiencing: Scaling navigation foundation models with reinforcement learning. *arXiv preprint arXiv:2507.22028*, 2025.

[21] Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, and Xinggang Wang. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. pages 12037–12047, 2025. doi: 10.1109/CVPR52734.2025.01124. URL https://openaccess.thecvf.com/content/CVPR2025/html/Liao_DiffusionDrive_Truncated_Diffusion_Model_for_End-to-End_Autonomous_Driving_CVPR_2025_paper.html.

[22] Lan Feng, Yang Gao, Eloi Zablocki, Quanyi Li, Wuyang Li, Sichao Liu, Matthieu Cord, and Alexandre Alahi. Rap: 3d rasterization augmented end-to-end planning, 2025. URL https://arxiv.org/abs/2510.04333.

[23] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2018.

[24] Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Conference on Robot Learning (CoRL)*, 2024.

[25] Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Pseudo-simulation for autonomous driving. In *Conference on Robot Learning (CoRL)*, 2025.

[26] comma.ai. openpilot. https://github.com/commaai/openpilot, 2026.

[27] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *TPAMI*, 2022.

[28] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[29] Hongyu Zhou, Longzhong Lin, Jiabao Wang, Yichong Lu, Dongfeng Bai, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. Hugsim: A real-time, photo-realistic and closed-loop simulator for autonomous driving. *arXiv preprint arXiv:2412.01718*, 2024.

[30] Zhenghao Peng, Wenjie Mo, Chenda Duan, Quanyi Li, and Bolei Zhou. Learning from active human involvement through proxy value propagation. *Advances in Neural Information Processing Systems*, 2023.

[31] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

[32] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

[33] Jieru Mei, Alex Zihao Zhu, Xinchen Yan, Hang Yan, Siyuan Qiao, Liang-Chieh Chen, and Henrik Kretzschmar. Waymo open dataset: Panoramic video panoptic segmentation. In *European Conference on Computer Vision*, pages 53–72. Springer, 2022.

[34] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[35] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[36] Yuxin Liu, Zhenghao Peng, Xuanhao Cui, and Bolei Zhou. Adv-bmt: Bidirectional motion transformer for safety-critical traffic scenario generation, 2025. URL `https://arxiv.org/abs/2506.09485`.

A Extended Experimental Details

A.1 BridgeSim Details

In this section, we briefly recap the components of BridgeSim [2] used in our evaluations, and refer the reader to the original paper for full details.

BridgeSim Platform Description. BridgeSim [2] is a cross-simulator evaluation platform built on MetaDrive [27] that unifies OL and CL testing of end-to-end driving policies pretrained from different simulators. It supports multiple driving datasets (nuPlan [32], WOMD [33], nuScenes [34]), heterogeneous traffic modes (log-replay, IDM [35], and adversarial policies [36]), and configurable execution frequency and simulation horizon. BridgeSim supports two evaluation modes: *non-reactive open-loop* and *reactive closed-loop* simulation. In open-loop mode, the ego-vehicle's state transitions are forced to follow the logged trajectory regardless of the policy's output, isolating planner reasoning from cumulative control error. In closed-loop mode, the policy controls the ego-vehicle while background traffic either responds via Intelligent Driver Models [35] (*fully reactive*) or replays the original logs (*semi-reactive*).

Metrics. BridgeSim adopts open-loop metrics for non-reactive open-loop simulations and closed-loop metrics for closed-loop simulations.

1) Open-loop Metric: Extended Predictive Driver Model Score (**EPDMS**). The **EPDMS** score is implemented as a weighted combination of safety-critical constraints $\mathcal{C}$ and driving quality features $\mathcal{F}$. Unlike L2 displacement errors, **EPDMS** penalizes physically infeasible or unsafe plans even if they mimic the log. The final score $S_{\text{EPDMS}} \in [0, 1]$ for a given frame is defined as the product of critical boolean constraints $\mathcal{C}$ and a weighted sum of soft metrics $\mathcal{F}$:

$$S_{\text{EPDMS}} = \left(\prod_{c \in \mathcal{C}} \mathbb{I}_c \right) \cdot \left(\frac{\sum_{f \in \mathcal{F}} w_f \cdot v_f}{\sum_{f \in \mathcal{F}} w_f} \right), \tag{3}$$

where $\mathbb{I}_c \in \{0, 1\}$ is the indicator function for constraint compliance, $v_f \in [0, 1]$ is the normalized feature score, and w_f is the corresponding weight. The constraints $\mathcal{C}$ comprise **NC** (No At-Fault

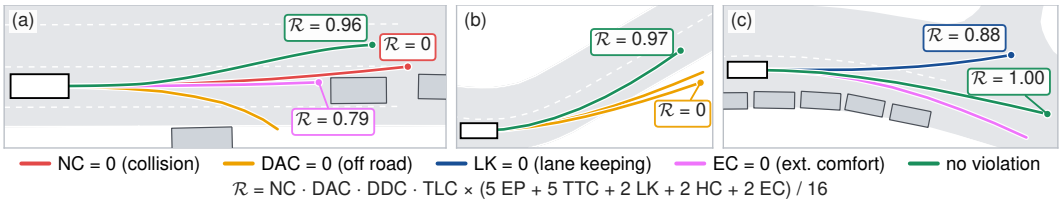


Figure 6: Illustration of the EPDMS-based verifiable reward $\mathcal{R}$. Candidate trajectories from the same state receive different rewards according to safety constraints and driving quality. (a) A collision sets NC to zero, while reduced extended comfort (EC) lowers the reward without setting it to zero. (b) Leaving the drivable area sets DAC to zero. (c) A lane-keeping violation reduces the LK contribution. The formula below combines multiplicative constraints with weighted driving-quality terms. Green trajectories satisfy the illustrated constraints, but their rewards can still differ due to the remaining term EP (Ego Progress).

Collisions), **DAC** (Drivable Area Compliance), **TLC** (Traffic Light Compliance), and **DDC** (Driving Direction Compliance); the quality features $\mathcal{F}$ comprise **EP** (Ego Progress), **LK** (Lane Keeping), **TTC** (Time-to-Collision), **HC** (History Comfort), and **EC** (Extended Comfort).

2) Closed-loop Metric: BridgeSim Driving Score. For reactive closed-loop simulation, BridgeSim defines the final driving score S_{CLS} as the product of the agent's global Route Completion (**RC**) and the temporal average of the frame-level scores ($S^{(t)}_{EPDMS}$):

$$S_{CLS} = R_c \times \frac{1}{T} \sum_{t=1}^{T} S^{(t)}_{EPDMS}, \tag{4}$$

where $R_c \in [0, 1]$ represents the percentage of the route completed by the agent relative to the expert driver's path or the goal destination, T is the total number of frames in the simulation episode, and $S^{(t)}_{EPDMS}$ is the EPDM score at time step t.

BridgeSim-NavHard Leaderboard. BridgeSim-NavHard is derived from the NAVSIM benchmark [28], whose logs originate from the nuPlan dataset [32] at 2 Hz. Agent trajectories are up-sampled to 10 Hz via linear interpolation, and HD maps are extracted from the nuPlan map API with a large spatial buffer to cover the full ego trajectory. The converted scenes contain 421 scenarios. For simulation settings, all models are evaluated under simulation operating at a default timestep (`sim_dt=0.1s`) and the simulation horizon spans for 8 seconds. Inference is executed at a specified `replan_rate` (default set to 2hz), with cached waypoints consumed between replan intervals. An initial replay phase is performed before evaluation begins to provide historical temporal buffers for agents.

A.2 Scoring Head Details

In practice, the verifiable reward $\mathcal{R}$ may decompose into multiple components $\mathcal{R}_c \in [0, 1]$ (e.g., drivable area compliance, lane keeping, history comfort). And in the implementations of pretrained base policies like DrivoR [1] and DiffusionDrive [21], the scoring head f predicts each component separately as f_c. In this case, the score loss in Eq. 2 will be implemented as:

$$\mathcal{L}_{\text{score}} = \sum_{c} \sum_{i=1}^{K} \text{BCE}\big(f_c(s, a_i^{\text{cache}}),\ \mathcal{R}_c(s, a_i^{\text{cache}})\big). \tag{5}$$

B Additional Experiments

B.1 Additional Base Model Results

Table 6 reports closed-loop results for RAHF applied to LTF [28] (regression-based) and DiffusionDrive [21] (diffusion-based) base policies. We use OpenPilot or real human experts as the Stage 1 teacher. RAHF consistently improves all base models.

Table 6: Closed-loop evaluation on BridgeSim-NavHard with additional base policies.

Model	DS	EPDMS	RC	DAC	TTC	LK	HC
LTF [28] (NeurIPS'24)	48.95	69.29	70.64	78.80	88.61	90.28	95.02
DiffusionDrive [21] (CVPR'25)	42.73	69.70	61.30	81.48	87.47	91.57	91.64
OpenPilot [26] (Teacher)	65.36	79.50	82.21	94.29	92.58	84.43	92.12
Real Human (Teacher)	84.31	86.49	97.47	99.24	94.95	92.39	99.25
RAHF (LTF)	64.53	76.45	84.40	93.92	90.51	92.13	83.60
RAHF (LTF) w/ Human Teacher	70.62	74.38	94.94	88.85	90.58	90.32	88.39
RAHF (DiffusionDrive)	65.34	76.54	85.38	95.72	91.47	90.93	81.60
RAHF (DiffusionDrive) w/ Human Teacher	74.28	78.13	95.08	95.35	87.04	94.66	87.22

B.2 Full Sub-metrics of Table 2 (Pretraining Knowledge Preservation)

Table 7 provides the full sub-metric breakdown for the open-loop preservation experiment reported in Table 2.

Table 7: Open-loop performance on 2000 NavTest scenarios evaluated in NAVSIM open-loop protocol with full sub-metrics.

Model	PDMS	NC	DAC	DDC	EP	TTC
DrivoR (native)	85.46	96.63	98.65	97.30	82.92	96.15
RAHF (DrivoR) w/ PAM	**84.09**	93.90	97.05	96.93	89.39	93.15
RAHF (DrivoR) w/o PAM	79.05	92.50	96.87	99.93	81.23	90.50
RAHF (DrivoR) w/ full backbone FT	71.57	89.50	93.87	98.90	76.23	90.73

B.3 Full Sub-Metric Breakdown of Table 3 (Zero-shot transfer to HUGSIM)

Here we provide the full submetric-breakdown of Table 3. HUGSIM uses a driving score (HD-S) defined as the product of Route Completion (RC) and PDMS, where PDMS is a weighted combination of safety constraints (NC, DAC) and driving quality features (TTC, Comfort). This is analogous to BridgeSim's DS = RC $\times$ EPDMS, but computed with HUGSIM's own evaluation protocol and physics engine.

Table 8: Closed-loop performance on HUGSIM scenarios, grouped by difficulty, with full sub-metrics.

Model	HD-S	NC	DAC	TTC	Comf.	RC	PDMS
Easy (n=18)							
DrivoR (native)	81.11	94.25	98.44	92.76	97.72	89.09	90.68
RAHF (DrivoR)	**84.22**	97.22	99.41	95.57	98.70	88.70	95.29
RAHF (DrivoR) w/o PAM	66.49	87.57	93.06	79.81	97.06	85.99	75.76
RAHF (DrivoR) w/ full backbone FT	57.34	79.32	96.56	73.75	98.49	81.07	72.55
Medium (n=34)							
DrivoR (native)	31.80	50.87	98.74	43.37	96.80	49.71	43.77
RAHF (DrivoR)	**33.83**	66.59	99.62	62.85	94.82	48.56	61.21
RAHF (DrivoR) w/o PAM	28.48	47.00	95.91	40.09	90.74	48.01	37.89
RAHF (DrivoR) w/ full backbone FT	26.66	52.50	99.35	45.83	98.62	46.42	47.11
Hard (n=18)							
DrivoR (native)	19.38	40.04	98.82	32.55	95.14	38.25	32.47
RAHF (DrivoR)	**30.30**	55.86	99.58	49.41	90.88	42.73	50.74
RAHF (DrivoR) w/o PAM	20.70	43.90	96.74	35.03	92.00	41.58	34.31
RAHF (DrivoR) w/ full backbone FT	14.48	38.28	99.86	31.98	93.99	33.49	33.34

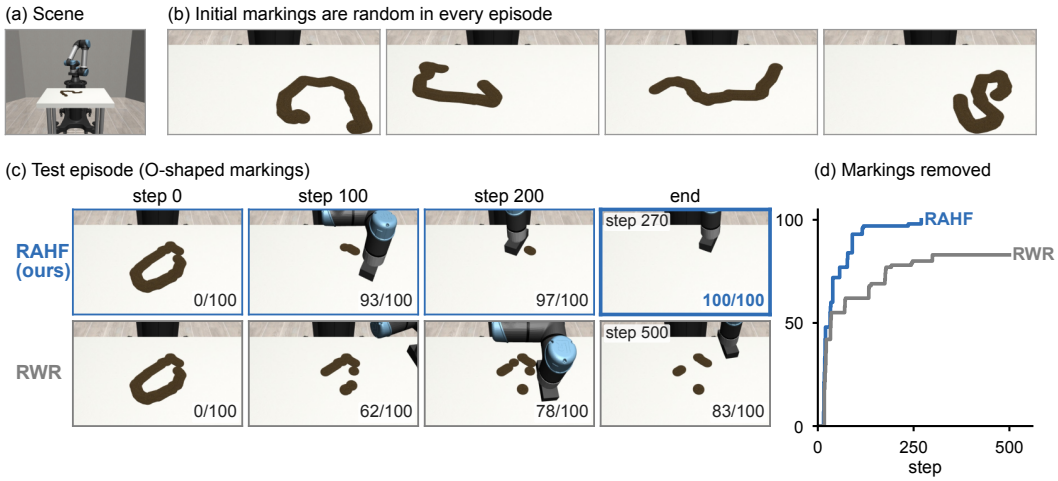


Figure 7: Table Wiping task and qualitative comparison. (a) Robot and table setup. (b) Examples of randomly initialized marking patterns. (c) Snapshots of RAHF and RWR on a test episode; labels indicate the number of markings removed out of 100. RAHF completes the task at step 270, while RWR removes 83 markings within the 500-step budget. (d) Cumulative markings removed during the same episode.

B.4 Table Wiping Experiments

Task and evaluation. We evaluate RAHF on RoboSuite Table Wiping, where a robot must remove 100 markings from a table within 500 steps. The initial marking pattern is randomized in each episode, as illustrated in Figure 7(a,b). The policy maps a 34-dimensional state to four candidate 6-dimensional end-effector actions and uses a learned scorer to select an action. All trainable methods use 5K samples, and evaluation is performed over 100 episodes with distinct seeds. Success Rate measures the fraction of episodes in which all 100 markings are removed within the step budget. Task Completion measures the average fraction of markings removed by the end of an episode. Table 5(b) reports the quantitative results. Figure 7(c,d) compares RAHF with RWR, the strongest baseline in Table 5(b), on a test episode. Starting from the same marking pattern, RAHF removes 93 markings by step 100 and 97 by step 200, completing the task at step 270. RWR removes 62 and 78 markings at the corresponding steps and finishes with 83 markings removed at the 500-step limit. In this example, RAHF makes faster initial progress and clears the remaining markings, whereas RWR leaves part of the table unwiped.

B.5 Reward and Hyperparameter Sensitivity

Reward noise. During training in the human-teacher setting, we perturb each normalized proposal reward independently as $\widetilde{\mathcal{R}}(s, a_i) = \mathcal{R}(s, a_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{U}[-\sigma, \sigma]$. Evaluation uses the original noise-free EPDMS, and the policy has no access to this reward at test time. Table 9 shows degradation with increasing noise.

Table 9: Training-reward noise with a human teacher and DrivoR.

σ	DS	EPDMS	RC
0	78.90	83.04	95.02
0.25	72.40	79.92	90.59
0.50	65.25	76.67	85.11
0.75	62.19	73.56	84.54
1.00	59.37	72.60	81.79

Neural-teacher takeover threshold. In Stage 1, the student selects $a_{\text{sel}} = \arg\max_{a_i} f(s, a_i)$, and OpenPilot takes over when $\mathcal{R}(s, a_{\text{sel}}) < \tau$. Table 10 reports the fraction of Stage 1 steps with

takeover. The default $\tau = 0.60$ gives the highest DS among the tested thresholds, and more frequent intervention does not necessarily improve the final policy.

Table 10: OpenPilot takeover-threshold sensitivity with DrivoR and a 6K-sample budget.

τ	Takeover rate (%)	DS	EPDMS	RC
0.50	18.21	64.76	77.45	83.62
0.60 (default)	30.27	69.27	79.08	87.60
0.70	64.86	63.72	75.16	84.79
0.80	85.76	61.80	74.94	82.47

Proposal count. Table 11 varies K for DrivoR with an OpenPilot teacher, keeping the remaining settings and the 6K-sample budget fixed. Gains become smaller beyond $K = 16$. Reward evaluation and scorer supervision require $O(K)$ candidate evaluations per state. The pairwise trajectory matching in Eq. (1) requires $O(K^2)$ comparisons for fixed action dimension. At inference, candidate scoring scales linearly in K.

Table 11: Proposal-count sensitivity with an OpenPilot teacher.

K	DS	EPDMS	RC
1	48.64	62.49	77.84
4	65.50	77.59	84.42
8	67.36	77.81	86.57
16	68.22	78.47	86.94
32	68.65	78.71	87.23
64 (default)	69.27	79.08	87.60

C Demo Video

We have attached our demo video to this paper. This video shows how we conduct human experiments and contains qualitative results of our method RAHF. This video includes four sections:

1. The first section gives an overview of open-loop to closed-loop gap. We show how the base DrivoR [1] policy fails in MetaDrive simulator.
2. The second section shows how we conduct real human experiment, where the human expert monitors the screen uses a gamepad to provide corrective demonstrations. We also include how we can use a neural teacher Openpilot to replace human.
3. The third section shows how the model learns from verifiable reward, where we no longer require human interventions in this stage.
4. The fourth section shows that our fine-tuned model has performance improvement in held-out NavHard scenarios converted from BridgeSim [2]. We also show some qualitative examples of how our fine-tuned model generalizes to unseen simulator HUGSIM [29].

D Human Subject Research Protocol

Human Participants. The study recruited three volunteer university students (ages 20–30), all of whom held valid driver's licenses and reported prior video gaming experience. We thoroughly went over with these participants on the experimental procedures and let participants sign informed consent forms under an IRB-approved protocol. Before experiment officially starts, we explained how predicted trajectories were rendered on screen and allowed each participant to practice with the control interface and learning environments.

Main Experiment. Each participant began with one or two fully manual episodes to gain familiarity with the tools. Participants were instructed to intervene only when the agent's predicted trajectory appeared unsafe, illegal, or inconsistent with their desired actions. They were directed to prioritize safe task completion and then to guide the agent toward their personal driving or manipulation preferences.

E Ethics Statement

RAHF offers an accessible human-in-the-loop training pipeline that boosts automation while keeping expert effort to a minimum, contributing to more capable AI systems with a lighter human workload. Since all experiments were carried out in simulation, participants were exposed to no physical risk. Each volunteer provided informed consent, received compensation above prevailing local rates, and was free to pause or exit the study at any moment without any penalty. We capped individual sessions at under one hour and required a minimum three-hour break before a participant could take part again. No personal or sensitive information was gathered or shared, and the project was conducted under IRB approval. Although RAHF may enhance societal impact through smoother human-AI collaboration, it could also induce overreliance on automated systems or carry forward behavioral biases in the expert demonstrations it draws upon.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our Method and Experiments sections accurately address the claims in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in the Conclusion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: We do not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provided our implementation details in Experiment part.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will release the code to reproduce the results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We clearly specify all the details in the first paragraph in Experiment part.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Every trainable method was run five times with distinct random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the type of GPU and the time of execution needed to reproduce the experiments in Experiment part.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss both positive and negative social impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not include pretrained language models, image generators, or scraped datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have credited BridgeSim [2] in our paper, the creator of assets used in our work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not include new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: We provide the instructions to the human participants in the supplemental material.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: We have obtained IRB approval to conduct this project.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Our method does not involve LLM.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.