Improving Agent Behaviors with RL Fine-tuning for Autonomous Driving

Zhenghao Peng[1*], Wenjie Luo[2], Yiren Lu[2], Tianyi Shen[2], Cole Gulino[2], Ari Seff[2], and Justin Fu[2]

[1]UCLA, [2]Waymo

Abstract. A major challenge in autonomous vehicle research is modeling agent behaviors, which has critical applications including constructing realistic and reliable simulations for off-board evaluation and forecasting traffic agents motion for onboard planning. While supervised learning has shown success in modeling agents across various domains, these models can suffer from distribution shift when deployed at test-time. In this work, we improve the reliability of agent behaviors by closed-loop fine-tuning of behavior models with reinforcement learning. Our method demonstrates improved overall performance, as well as targeted metrics such as collision rate, on the Waymo Open Sim Agents challenge. Additionally, we present a novel policy evaluation benchmark to directly assess the ability of simulated agents to measure quality of autonomous vehicle planners and demonstrate the effectiveness of our approach on this new benchmark.

Keywords: Autonomous Driving · Reinforcement Learning · Policy Evaluation · Behavior Prediction

1 Introduction

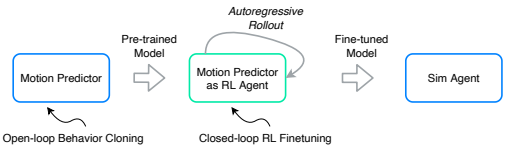

Fig. 1: We propose to fine-tune a pre-trained motion prediction model with closed-loop reinforcement learning.

Transformer-based architectures have demonstrated state-of-the-art performance in a variety of tasks in language [24], vision [26], and robotics [43]. The success of these models is credited to a widely adopted "pre-training and fine-tuning" scheme [40]. In the pre-training phase, the model acquires knowledge from a very large amount of training data; during fine-tuning, the model behaviors are rectified to align with human preferences and expectations. While supervised learning can be used for fine-tuning, previous work has shown superior performance with reinforcement learning (RL) fine-tuning in language tasks [25] and text-to-image generation [3]. In autonomous driving

*Work done as an intern at Waymo.

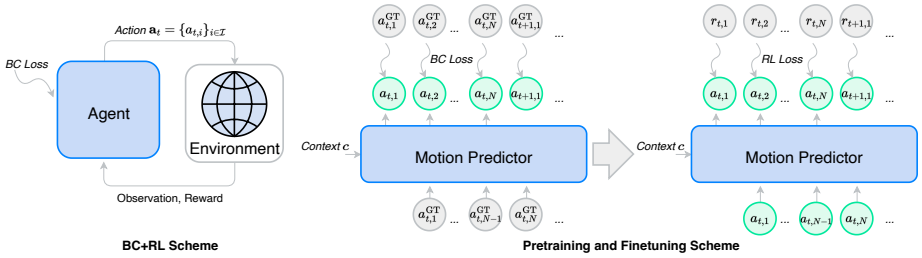

Fig. 2: Left: The agent is trained from scratch using a combined Behavioral Cloning (BC) and Reinforcement Learning (RL) approach. Without pre-training on large datasets, the agent must simultaneously explore the environment and develop its capabilities from scratch. **Right**: The agent undergoes a two-phase training scheme. Agent acquires a foundational skill set from aligning its actions (green) with ground truth data in pre-training (gray). The fine-tuning through RL refines the agent behaviors in the autoregressive rollout.

(AD), given the abundance of human driving data, a natural question arises: Can we leverage the popular "pre-training and RL fine-tuning" strategy to effectively model agent behaviors?

In this paper, we investigate the viability of applying the "pre-training and RL fine-tuning" paradigm to model the behaviors of traffic agents in AD scenarios. Such models can be applied in critical AD tasks such as simulation agents (sim agents) [20], enabling high-fidelity simulation systems for off-board evaluation, and behavior prediction for surrounding traffic participants, facilitating onboard planning.

Behavioral cloning (BC), or training an imitative model using supervised learning on demonstrations [1], has been the predominant approach for learning driving agents [5, 34]. While BC provides supervision for modeling realistic behavior, during closed-loop simulation, the agent behaviors can deviate from the training distribution, known as the "covariate shift" issue [27]. Moreover, BC lacks the ability to explicitly incorporate human preferences, expectations and constraints. For example, safety-critical events, such as collisions, are only implicitly discouraged in the BC loss due to their rarity in human driving datasets. In situations where a collision is likely to occur, the model may only have limitated examples to learn from. RL fine-tuning can address these limitations. Firstly, RL learns from closed-loop synthetic rollouts, addressing the covariate shift problem as the reward function penalizes actions leading to future trajectories that diverge from ground-truth. Secondly, explicit objectives can be incorporated into the reward function so the agents can learn to align with human preferences and expectations.

Inspired by the success of fine-tuning large language models to align with human preferences, we apply the "pre-training and RL fine-tuning" scheme to training behavior models for sim agents. As demonstrated in Fig. 1, we can fine-tune a pre-trained model via a simple on-policy RL approach with autoregressive

rollouts. We propose a simple reward function that not only enables the model to satisfy human preferences on the agent behaviors, but also maintains human likeness.

Our experiments on the Waymo Open Sim Agent Challenge (WOSAC) [20] demonstrate that RL fine-tuning significantly improves the reliability of the agent behaviors, especially in terms of collision avoidance. An important application of the learned behaviors is in actuating the traffic agents in AD simulation. We study the reliability of the learned models in a novel planner evaluation benchmark. The intuition is that a simulator with more realistic sim agents model should provide more reliable evaluation across ego AD planners. With this insight, we use different sim agents model to control the traffic agents and assess the performance of a predefined set of AD planners. By comparing the planners' estimated performance as evaluated by the sim agents model against their known performance ranking, we find that our fine-tuned models provide more accurate planner evaluations. This indicates that our approach is beneficial for testing autonomous driving planners. The main contributions of this work are:

1) We propose to apply the popular "pre-training and RL fine-tuning" paradigm commonly used for large language models (LLMs) to the autonomous driving behavior modeling problem, demonstrating the effectiveness of closed-loop fine-tuning a Transformer-based architecture on the Waymo Open Motion Dataset (WOMD) [15].

2) We demonstrate that an on-policy RL algorithm with a simple reward function can successfully preserve the realism in the dataset while aligning human preferences on safety and reliability.

3) To better evaluate the performance of sim agents models, we propose a novel planner evaluation task and demonstrate that our method can significantly improve the performance of the sim agents models in terms of its capability to assess the AD planners.

2 Related Work

2.1 Pre-training and Fine-tuning of Transformer-based Models

Transformer-based models have been applied to various domains such as text generation [4], image generation [26], robotics [43], drug discovery [18], disease diagnose [42], and generalist medical AI [22]. Many large Transformer-based models are trained in the "pre-training then fine-tuning" manner, where supervised fine-tuning [40] or reinforcement learning with human feedback [25] holds the promise to align the model behaviors to human preferences. In the autonomous driving domain, similar Transformer-based architectures have been applied to various tasks, ranging from perception [19], motion prediction [23], self-driving policies [10] and simulation [31,38,39]. In this work, we focus on the motion prediction problem, where predictors forecast the future trajectories of the target agents by observing history information [23, 29, 30]. Unlike foundational models in other domains such as large language models [24] and vision

language models [16], motion prediction models are most commonly trained via supervised learning and rarely fine-tuned to better boost alignment with human preferences.

2.2 Behavior Modeling for Autonomous Driving

Modeling the behavior of traffic participants is a critical task in many autonomous driving systems, particularly for constructing realistic simulation to test the AD planners. Most existing simulators [6, 12, 41] rely on hand-crafted rules for traffics and maps generation. However, the data distributions for the map structure, traffic flow, the interaction between traffic participants and other elements do not realistically represent the real world. Modern data-driven simulators [9,13,14,33] address this by replaying the behaviors of the traffic participants from real-world scenarios recorded by an autonomous vehicle (log-replay). Yet, the downside of log-replay is that re-simulation may become unrealistic when the planner behavior diverges from the original logged behavior. For example, if an AD planner is more cautious than the human driver and brakes earlier, the trailing vehicle might collide into it, leading to a false positive collision.

In this work, we mainly focus on the simulation agents task and evaluate our solutions on the Waymo Open Sim Agent Challenge (WOSAC) [20]. Many existing WOSAC submissions apply the marginal motion prediction models [2, 7, 29], which typically take initial states and predict the positions of traffic participants at all future steps in a single inference (one-shot). Those marginal models do not explicitly model interactions between agents during the prediction horizon. The autoregressive (AR) models naturally fit to the driving behavior modeling, especially in the context of closed-loop simulation [11, 28, 31, 38]. AR decoding [28] allows the interactions between agents to be modeled via a self-attention mechanism at each step of the decoding process. However, closed-loop training of the AR behavior prediction models remains an understudied area. We propose to improve a pre-trained AR model with closed-loop fine-tuning and evaluate the performance on the WOSAC benchmark.

In contrast to prior research on combining behavior cloning and reinforcement learning [17, 36, 37], our approach eliminates the need for an external simulator and a dynamics model. Since our model predicts actions for all agents, it functions as a simplified simulation environment itself. From an algorithmic perspective, we avoid back-propagation through time (BPTT) used in existing works [11,36]. Instead, we propose that RL can be conducted with a minimalistic policy gradient algorithm [35]. This allows us to use non-differentiable rewards (such as a boolean collision indicator) and non-differentiable models with discrete outputs which would not be possible with BPTT.

3 Preliminaries

Motion Prediction. A driving scenario includes static information such as the map topology and dynamic information such as the states of traffic participants

and traffic lights. At each time step, the state of a traffic participant is represented by a feature vector containing the position, velocity, and heading angle in global frame, and the object type (vehicle, cyclist, pedestrian). For traffic lights, the feature vector contains their position and state (green, yellow, red or unknown). Given the history states of N traffic participants with indices $\mathcal{I} = [1...N]$, the goal of motion prediction is to predict future trajectories, i.e. the positions in future steps, of these agents.

Behavior Modeling as a Multi-Agent RL Problem. We consider driving behavior modeling as a Multi-agent Markov Decision Process (Multi-agent MDP). The Multi-agent MDP is defined by the tuple $\langle \mathcal{I},\ \mathcal{S},\ \{\mathcal{A}_i\},\ \mathcal{T},\ \{\mathcal{R}_i\},\ \Omega,\ \{\mathcal{O}_i\},\ \gamma \rangle$, where $\mathcal{S}$ is the joint state space, $\mathcal{A} = \times_i \mathcal{A}_i$ is the joint action space, the transition function is $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, the reward functions are $\mathcal{R}_i(s_t, a_{t,i}, s_{t+1}), \forall i \in \mathcal{I}$, the observation space is $\Omega = \times_i \Omega_i$, the observation functions $\mathcal{O}_i$, and the discount factor is γ. In this Multi-agent MDP, the goal is to learn action policies $\pi_i : \mathcal{A}_i \times \Omega_i \to [0, 1]$ for each agent. Each agent aims to maximize its expected cumulative return: $\pi_i = \arg\max_\pi \mathbb{E}_{\tau \sim P_{\pi_j, \forall j \in \mathcal{I}}}[\sum_{t=1}^{T_{\text{future}}} \gamma^t r_{t,i}]$, where $\tau = (\mathbf{s}_0, \mathbf{a}_0, ..., \mathbf{s}_T, a_T)$ is the joint future rollout obtained by executing the learned policy model conditioned on the initial state. Here, $\mathbf{a}_t = \{a_{t,i}\}_{i \in \mathcal{I}}$ is the joint actions.

Autoregressive Encoder-Decoder Architecture. We use MotionLM [28], an encoder-decoder transformer-based autoregressive motion prediction model. The model's encoder takes a set of tokens representing the initial states of the scenarios as input and generates a scene embedding. These initial states include the traffic lights states, map topology, and the history information of all traffic participants. During inference, we run the decoder for T_{pred} prediction steps to autoregressively generate the prediction of all agents. At each prediction step, the decoder takes a set of motion tokens as well as the scene embedding as input and generates a distribution of N output tokens. The decoder consists of multiple layers, each applying self-attention among input tokens and cross-attention to the scene embedding. All N motion tokens at step t can attend to each other and all previous tokens as shown in Fig. 3, where each row represents a query token and each column a key token and green blocks indicate key tokens that the query can attend. After running T_{pred} prediction steps, the $T_{\text{pred}} \times N$ output motion tokens form complete trajectories for N agents. This autoregressive approach ensures that each agent's action is based on a temporally causal relationship with the previous actions of all traffic participants, leading to improved modeling of interaction between agents within the prediction horizon.

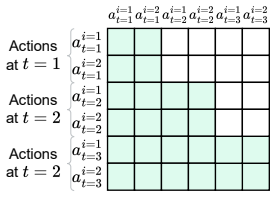

Fig. 3: The causal mask in the decoder.

We modify the original MotionLM model by adopting a scene-centric input format and predicting the motion of all agents, rather than using a pre-selected

subset. The scene-centric representation and the encoder-decoder architecture enable more computationally efficient encoding of the scene context and prediction of all agents' motion.

4 Method

As shown in Algorithm 1, our method has two stages. In the pre-training stage, we reconstruct the ground truth actions from the data and use the maximum likelihood objective to match the joint action distribution of observed behaviors in the dataset:

$$\max_{\pi_\theta} \mathop{\mathbb{E}}_{\mathcal{D}} \sum_{t=1}^{T_{\text{pred}}} \sum_{i \in \mathcal{I}} \log \pi_\theta(a_{t,i}^{\text{GT}} | o_{t,i}). \tag{1}$$

The second stage of our learning process fine-tunes the model using reinforcement learning (RL). We formalize the problem as a Multi-agent Markov Decision Process (MDP) for our behavior modeling task as follows:

Action. The action space $\mathcal{A} = \times_i \mathcal{A}_i$: The action space for each agent $\mathcal{A}_i$ is a Verlet-wrapped delta action space [28], where each action represents the X, Y acceleration in scene coordinates. To reconstruct the ground truth action targets, we first infer the accelerations by differentiating the observed positions in the data. These accelerations are then discretized into a 13x13 uniformly spaced grid, where outliers are clipped to the minimum and maximum values of 6 m/s^2.

State. The state space $\mathcal{S}$ contains map features, the joint state of all objects and traffic lights.

Transition Dynamics. Agents transit to new positions computed by adding previous positions with an offset: $pos_{t+1,i} = (a_{t,i}\Delta + vel_{t,i})\Delta + pos_{t,i}$ wherein the velocity $vel_{t,i}$ and the position $pos_{t,i}$ are in s_t and Δ is the time interval between steps.

Observation. We define observations to consist of a historic context c, previous actions of all objects and the agent identity: $o_{t,i} = (c, \mathbf{a}_1, ..., \mathbf{a}_{t-1}, i)$. Here, the context $c = (\{m_i\}_M, s_{T_{\text{prev}}}, ..., s_0)$ is a set containing M map features and the object and traffic light states for history steps $t = T_{\text{prev}}, ..., 0$.

4.1 RL Fine-tuning

We propose to fine-tune a pre-trained autoregressive motion predictor with RL. The reward function for each agent at each step is defined as:

$$r_{t,i} = -||Pos_{t,i} - GT_{t,i}||_2 - \lambda Coll_{t,i}, \tag{2}$$

where $Pos_{t,i}$ is the position of agent i in step t and $GT_{t,i}$ is the corresponding position in the logged trajectory. $Coll_{t,i}$ is a Boolean indicator and will be 1 if the bounding box of agent i intersects with others' bounding boxes. This reward function, while simple, captures the key objectives of preserving the behavioral realism as well as satisfying the safety constraint of collision avoidance.

Algorithm 1: Pre-train and fine-tune an autoregressive motion predictor.

input : Large-scale driving dataset $\mathcal{D}$.
output: A Sim Agent policy π_θ.

Initialize model parameters θ and model π_θ.
for *pre-training iterations* $j = 1, ...$ **do**
  Retrieve $\mathbf{s}_{\text{history}}, \mathbf{s}_{\text{GT}}$ from $\mathcal{D}$.
  Construct target actions $\mathbf{a}_{\text{GT}} = \{a^{\text{GT}}_{t,i}, \forall t, i\}$ and construct the observation $o_{t,i}$ with GT actions.
  Run the model with the observation and get the predicted actions $\{a_{t,i}, \forall t, i\}$.
  Update π_θ via Eq. 1.
for *fine-tuning iterations* $j = 1, ...$ **do**
  Retrieve $\mathbf{s}_{\text{history}}$ from $\mathcal{D}$; Set $\mathbf{a}_0 \leftarrow$.
  for $t = 1, ..., T_{pred}$ **do** ▷ `Autoregressive Rollout.`
    $\mathbf{o}_t \leftarrow (\mathbf{s}_{\text{history}}, \mathbf{a}_0, ..., \mathbf{a}_{t-1}, \mathcal{I})$. ▷ `Get obs.`
    $\mathbf{a}_t \leftarrow \pi_\theta(\cdot|\mathbf{o}_t)$. ▷ `Decode next actions.`
  Reconstruct predicted states $\mathbf{s}_{\text{pred}} = \{\hat{s}_{t,i}, \forall t, i\}$ by translating actions $a_{t,i}, \forall t, i$.
  Compute per-agent per-step $r_{t,i}$ via Eq. 2
  Compute normalized return via Eq. 4 and update π_θ via Eq. 5

During fine-tuning, we run the model for T_{pred} prediction steps. The encoder first encodes the scene context c as a shared scene embedding, before the autoregressive decoding. At each prediction step t, the fixed scene embedding and the $t \times N$ tokens are fed to the autoregressive decoder and sample N new actions. Specifically, at prediction step $t = 1$, we project the agents' current positions through a MLP and get the agent embeddings: $id_i = \text{MLP}(Pos_{0,i}), i = 1, ..., N$. The agent and scene embeddings serve as the input tokens to the decoder. After several layers of self-attention and cross-attention, N actions are sampled from the categorical distributions constructed from the output of the decoder. The embeddings of those sampled actions will be added with corresponding id_i and concatenated with the tokens in previous steps to form the input tokens for the next step. Compared to the decoding process of a language model, we output N tokens concurrently at each prediction step instead of one token. Our model autoregressively rolls out the actions in T_{pred} time steps. After collecting the rollout trajectories, we translate the actions to sequences of 2D positions for computing the rewards following Eq. 2. The return (*i.e.* the "reward-to-go"), for step t and each agent i is:

$$R_{t,i} = \sum_{t'=t}^{T_{\text{pred}}} \gamma^{t'-t} r_{t',i}. \tag{3}$$

We normalize the return across the whole training batch, here $Mean$ and Std are the average and the standard deviation computed across all time steps in all

scenarios for all agents in the training batch:

$$\tilde{R}_{t,i} = (R_{t,i} - Mean(R))/Std(R). \tag{4}$$

We then apply the REINFORCE [35] method to compute a policy gradient for optimizing the model by differentiating the following surrogate objective:

$$\max_{\pi_\theta} \mathop{\mathbb{E}}_{\mathcal{D}} \sum_{t=1}^{T_{\text{pred}}} \sum_{i \in \mathcal{I}} \log \pi_\theta(a_{t,i}|o_{t,i}) \tilde{R}_{t,i}. \tag{5}$$

4.2 Policy Evaluation for Sim Agents

A key limitation of common "imitative" metrics (such as ADE), which compare model rollouts to ground truth trajectories, is the weak connection between the metric and the actual goal of assessing the AD planner performance. A low ADE metric does not guarantee good driving behaviors. Log-replay, for example, has a perfect ADE of zero but would be a poor choice for sim agents because it is non-reactive. To create an evaluation that has a direct connection to measuring the performance of the AD planners, we propose a new policy evaluation framework for sim agents, inspired by the RL policy evaluation literature [32].

Our policy evaluation framework involves ranking and scoring the performance of a predetermined collection of AD planner policies. This is analogous to a real-world use case where one must decide which planner to deploy from a collection of candidate software releases. A better sim-agent model will give a more accurate signal on which policy would be best when deployed in the real world. As shown in Fig. 4, we first prepare a batch of AD planner policies with known performance ranking. Then, we evaluate the performance of these AD planners when the traffic agents in the scenario are controlled by a sim agent model. Therefore, we will generate the estimated performance for those AD planners for the specific sim agent. We then measure the discrepancy between the estimated performance and the ground truth performance of those planners. This discrepancy becomes the measurement of the sim agents model's ability to assess the performance of the planners. Policy evaluation covers two important use cases for the sim agent models in the deployment of autonomous vehicles: *evaluation*, where we wish to estimate the performance of agents in the simulation, and *selection*, where we wish to determine a ranking or order between different deployment candidates.

Choice of policies. In order to perform policy evaluation, we must have a fixed set of policies on hand to rank or evaluate. To generate a large variety of planning policies with both good and bad performance, we propose to use a random shooting search-based policy family, parameterized by the number (J) and depth (D) of trajectories sampled. We compute a "ground truth" score for each policy by evaluating it with log playback agents. Note that the choice of ground truth is an important design decision. Any sim agent could serve as a ground truth,

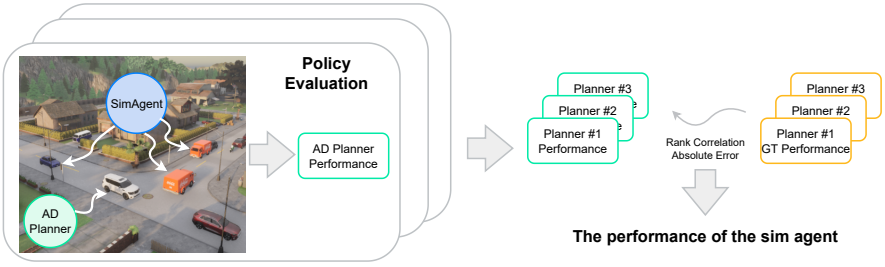

Fig. 4: We evaluate the sim agent by its ability to correctly assess the AD planner.

but we need to pick one that is the most fair to all models and we believe log playback to be the most neutral.

Our random shooting policy operates in a model-predictive control (MPC) fashion: at each time step, our random shooting policy samples from a fixed library of J trajectories, which are generated by maintaining a single steering wheel angle and acceleration for D steps. Note that this action specification is different from the action space of the MotionLM architecture we described in Sec. 4. We found this simple strategy to work much better than randomly selected actions. The trajectories are then scored by a reward function and the first step of the best scoring trajectory is executed. This process repeats for the entirety of the rollout. We used 16 different settings of J, ranging between 9 to 81, and we used 4 values of depth $D \in [6, 8, 12, 16]$. We then used the product of these two sets, for a total of 64 different policies evaluated.

Reward Function. The reward function used for selecting actions from a set of candidate trajectories is a linear combination of collisions, as well as off-road and route-following infractions. We use a modified reward function from Eq. 2 by replacing the L2 norm from the ground truth (which is not available to the planner at execution time) with additional terms for following a reasonable path. We instead give the planner a high-level route in the form of waypoints, and we use a weighted sum of $-10C - O - R + 10^{-4}P$, where $C \in \{0, 1\}$ denotes the collision indicator and is 1 when a collision between AV and another object happens, $O \in \{0, 1\}$ denotes the offroad indicator and is 1 when the AV is too close to the road edge, $R \in \{0, 1\}$ denotes the off-route indicator which is 1 when the AV's lateral distance to the GT trajectory exceeds a threshold, and P is the projection of the AV's displacement between two time steps when projected onto the logged trajectory and measures the route-following behavior. We use Waymax [9]'s utility function to compute those metrics.

5 Experiments

We now describe our method's experimental results on the Waymo Open Sim Agents Challenge (WOSAC) [20] and on the Policy Evaluation task introduced in Sec. 4.2. Our experiments are designed to answer the following questions:

	Lin. Speed ↑	Lin. Acc. ↑	Ang. Speed ↑	Ang. Acc. ↑	Dist. to Obj. ↑	Collision ↑	TTC ↑	Dist. to Road Edge ↑	Offroad ↑	Composite ↑	ADE ↓	MinADE ↓
Random	0.002	0.044	0.074	0.120	0.000	0.000	0.734	0.178	0.287	0.155	50.739	50.706
Constant Velocity	0.074	0.058	0.019	0.035	0.208	0.345	0.737	0.454	0.455	0.287	7.923	7.923
Wayformer	0.331	0.098	0.413	0.406	0.297	0.870	0.782	0.592	0.866	0.575	**2.498**	2.498
MVTE	0.445	0.222	0.535	0.481	0.383	0.893	0.832	0.664	0.908	**0.645**	3.859	**1.674**
Logged Oracle	0.561	0.330	0.563	0.489	0.485	1.000	0.881	0.713	1.000	0.722	0.000	0.000
Pre-trained (1M)	0.390	0.235	0.504	0.447	0.348	0.544	0.803	0.582	0.525	0.490	6.332	3.177
RL-only (1M)	0.257	0.115	0.487	0.429	0.244	0.239	0.759	0.456	0.164	0.320	7.785	6.918
Fine-tuned (1M)	0.412	0.219	0.451	0.420	0.348	0.863	0.814	0.637	0.804	0.597	2.436	1.867
Pre-trained (10M)	0.439	0.241	0.502	0.454	0.371	0.673	0.811	0.625	0.655	0.549	4.508	2.274
Fine-tuned (10M)	0.433	0.220	0.455	0.423	0.361	0.877	0.819	0.647	0.825	**0.608**	**2.428**	**1.706**

Table 1: Results on the Waymo Open Sim Agents Challenge (WOSAC) benchmark. Metrics marked with (↑) are better if higher, while metrics marked with (↓) are better if lower. Fine-tuned agents score better than pre-trained agents on safety-critical metrics such as collision and offroad, which results in a significantly higher composite metric score. We bold the best results for the baseline agents and best results from the model we trained for this work.

1. Does RL fine-tuning improve the overall sim agents behavior?
2. Can fine-tuning be used to improve targeted metrics through reward engineering?
3. Does the fine-tuned sim agents model provide better evaluation of AD planner performance?

Dataset. We train our method and baselines on the Waymo Open Motion Dataset (WOMD) [15] and evaluate on the Waymo Open Sim Agents Challenge (WOSAC) benchmark [20]. WOMD contains scenarios recorded at 10Hz including one second of history (11 discrete time steps), and 8 seconds of future states (80 time steps) to predict. In total, there are 486k training scenarios, 44k validation scenarios, and 45k test scenarios. Up to 128 agents are simulated in each scenario.

Model. We use a pre-trained autoregressive motion prediction model MotionLM [28]. For the MotionLM model with 10M parameters, we use 4 encoder and 4 decoder layers. The hidden size is 256. The number of attention heads is 4. The activation is ReLU. The feed-forward network intermediate size is 1024.

Training. We pre-train a MotionLM model on the WOMD training set with the objective in Eq. 1. The model is then used for RL fine-tuning. The encoder and the decoder of the model are fine-tuned jointly. 1M steps of updates are conducted both in pre-training and fine-tuning. At each training step, 128 scenarios are sampled from the dataset to form a batch. During RL fine-tuning, the learning rate is set to 5e-6 and the discount factor is set to 0.95.

5.1 Waymo Open Sim Agents Challenge

Baselines. We include several notable baselines reported by [20] on WOSAC. The "random" and "constant velocity" agents are included to provide a reasonable performance lower bound. The "logged oracle" represents the ground truth future

behavior that is not visible to other baselines and represents an upper-bound on performance. Wayformer [23] is a recent transformer-based model which shares the same encoder structure as our model. MVTE [34] is another transformer-based architecture which is the current state-of-the-art on the benchmark. Both Wayformer and MVTE adopt the agent-centric input representation. We also report the performance of our pre-trained 1M parameter and 10M parameter models, which are based on the MotionLM [28] architecture.

Evaluation Metrics. The Waymo Open Sim Agents challenge evaluates agents on a wide range of *likelihood*-based metrics. These metrics are designed to measure realistic simulation in aggregate over the entire dataset, while allowing agents to have enough flexibility to deviate from the exact logged ground truth in each scenario. Each benchmarked method samples 32 rollouts for each WOMD test scenario. Metrics (such as velocity and heading angle) are then measured on these samples, but binned into discrete histograms, and the log-likelihood of the ground-truth data is measured under these histograms. The individual scores are then weighted and averaged to produce a final composite metric. In addition, following WOSAC [21] we also report the mean average displacement error (ADE) over 32 rollouts and the minimum average displacement error (minADE) over 32 rollouts.

Results. We report our results on WOSAC in Table 1. The results provide strong evidence that closed-loop fine-tuning from a pre-trained model can significantly improve performance of the model as a sim agent. We see that both the "Fine-tuned 1M" and "Finedtuned 10M" models perform better than "Pre-trained 1M" and "Pre-trained 10M", respectively. When we look at the breakdown of the constituent metrics, we see that most of the gains come from improved safety-critical metrics such as collision and offroad. As a concrete example, Fig. 5 illustrates a single scenario comparing a rollout from a pre-trained 1M parameter model with a fine-tuned model. The pre-trained model is prone to slowly drifting away from the ground truth trajectory, a distribution shift problem commonly impacting pure imitative and teacher forcing [27] methods. By training the model in closed-loop, we can mitigate this distribution shift issue.

The composite performance of "Fine-tuned 10M" is still lower than the state-of-the-art MVTE [34] model. We believe this can be mostly attributed to the choice of our pre-trained baseline model, which was our re-implemented version of the MotionLM [28] architecture with the encoder architecture of Wayformer [23]. We leave the fine-tuning of MVTE using the same methodology described in this work as future work, which we hypothesize would lead to a sizeable performance gain.

To measure the effect of reward engineering and the ability to target specific metrics with fine-tuning, we ran an ablation study varying the relative weight λ of the collision metric in Eq. 2, with results reported in Table 2. We fine-tuned the 1M parameter model with 4 different collision weights, and reported the collision score versus the ADE of the predictions. We can clearly see that adding some amount of collision penalty improves the collision metric at the cost

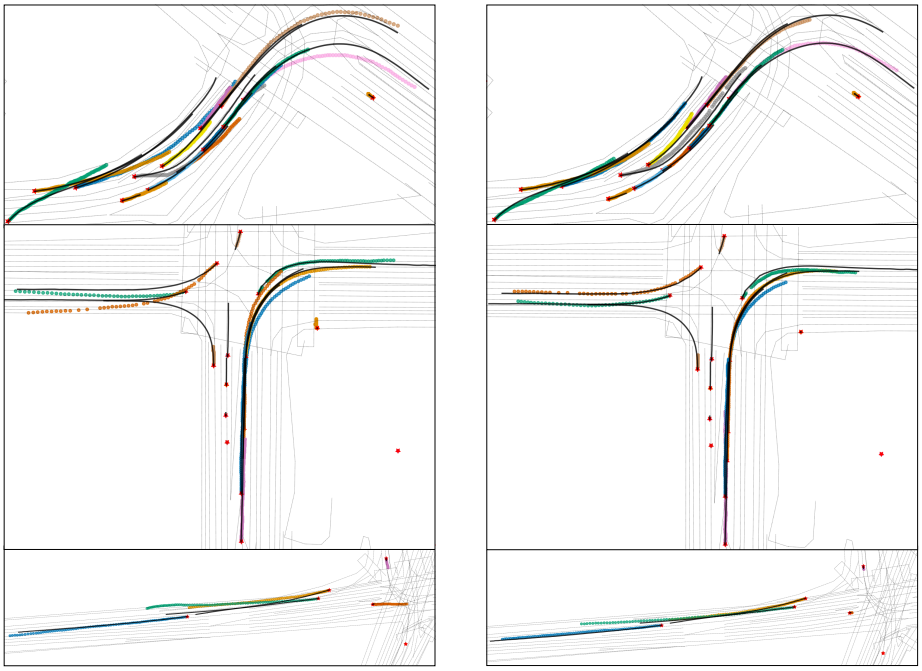

Fig. 5: Visualization of scenario rollouts using a pre-trained and a fine-tuned model. The start locations of vehicles are marked with a red star, the ground truth futures are marked with a solid black line, and the sampled trajectory is marked with circles of different colors. **Left**: The pre-trained model suffers from drifting due to distributional shift between training (with teacher forcing) and testing (with an autoregressive rollout). **Right**: The fine-tuned model is able to follow the ground truth much more precisely, which is quantitatively demonstrated by the better ADE metric.

Collision Weight	Collision ↑	TTC ↑	Composite ↑	ADE ↓	MinADE ↓
0	0.834	0.810	0.590	**2.405**	1.871
2	**0.863**	0.814	**0.597**	2.436	**1.867**
5	0.844	**0.817**	0.595	2.838	1.980
10	0.831	**0.817**	0.594	3.058	2.023

Table 2: WOSAC benchmark scores for different values of the collision fine-tuning weight λ with the 1M parameter model. Increasing the collision weight improves the collision score at the cost of decreasing imitative behavior metrics such as ADE. We find a good balance at an intermediate value.

of degrading the ADE metric. This generally makes sense, as the optimization must trade-off the collision penalty versus the displacement error terms in the reward function. However, we also see that at very high values of the collision

weight, all metrics tend to degrade, and the best result is with an intermediate cost weight. We hypothesize this is the case because displacement error is a very dense and rich reward signal, whereas collision is a more sparse and noisy signal.

5.2 Policy Evaluation

As described in Sec. 4.2, we also propose to evaluate sim agents through the lens of policy evaluation. In particular, we follow the methodology proposed by [8] and report two metrics used in their benchmark: Spearman's rank correlation and absolute error. Rank correlation is a metric for "selection" and measures the ability for the simulation to discern between good and bad policies (AD planners). This is useful for the problem setting where one must select the best policy to deploy from a set of candidates. On the other hand, absolute error is a metric for "evaluation" and measures how closely a simulation comes close to estimating the true cost or reward accrued by the policy. This is useful if one is interested in concrete performance numbers such as estimating the rate of a particular event of interest. The sim agent causing less absolute error and higher rank correlation is better for evaluating different planners.

Given a set of planners with known performance ranking, we assess sim agents by measuring how well they estimate the value function of each planner. We generate a Monte-Carlo estimate of each planner's value by running it on each scenario, where all traffic participants are controlled by the sim agent, and computing the empirical returns of the planner. The returns are then averaged across all scenarios in the WOMD test set to form the estimate of the planner. The value estimates are then compared to a "ground-truth value", and we show two quantities, *rank correlation* and *absolute error* in Table 3 and Table 4, respectively. Each row in these tables represents the policy evaluation results of a sim agent. A sim agent will generate $K = 64$ estimated returns of K policies (AD planners). Due to the absence of real-world simulation, there is no a universal ground truth sim agent that can fully replicate the realistic behaviors. We can not know the ground truth returns when running these K policies in the real world. In this work, we picked the log-playback sim agent as the "ground truth sim agent" and considered the average returns of these K policies when running with the log-playback sim agent as the ground truth value. These form the `Log` column in Table 3 and Table 4. We also show rank correlation and absolute error relative to each other sim agent we considered for completeness.

Results. Our results are reported in Table 3 and Table 4. We report results on a total of 64 candidate policies created by varying the depth (D) and the number of sampled trajectories (J). According to the `Log` column, we see that with fine-tuned models, a higher rank correlation and lower absolute error relative to the log-playback sim agent can be achieved, indicating that the fine-tuned models are more accurate at measuring a planner's performance and deciding whether the planner is better than another.

Sim Agent	Log	Pre. 1M	Fine. 1M	Pre. 10M	Fine. 10M
Pre-trained 1M	0.859	-	0.954	0.953	0.911
Fine-tuned 1M	0.865	0.954	-	0.959	0.925
Pre-trained 10M	0.845	0.953	0.959	-	0.932
Fine-tuned 10M	**0.866**	0.911	0.925	0.932	-

Table 3: Policy evaluation **rank correlation** results for pre-trained and fine-tuned models (higher is better). Each cell corresponds to a ranking correlation of estimated returns of a set of predefined AD planners between two sim agent models. For example, the highlighted `Fine-tuned 10M - Log` means the ranking correlation of predefined AD planners when using the Fine-tuned 10M and the log-playback sim agents.

Sim Agent	Log	Pre. 1M	Fine. 1M	Pre. 10M	Fine. 10M
Pre-trained 1M	10.518	-	0.557	0.551	1.013
Fine-tuned 1M	10.101	0.557	-	0.426	0.729
Pre-trained 10M	10.014	0.551	0.426	-	0.602
Fine-tuned 10M	**9.509**	1.013	0.729	0.602	-

Table 4: Policy evaluation **rank absolute error** results for pre-trained and fine-tuned models (lower is better).

6 Conclusion

We studied the viability of applying the popular "pre-training and fine-tuning" scheme to modeling traffic agents for AD simulation. We drew the connection between a multi-agent driving behavior model and a simulation environment – the multi-agent behavior model itself can be used to perform rollouts for closed-loop training. By using an on-policy RL algorithm with a simple reward, we are able to fine-tune a pre-trained large multi-agent behavior model to effectively align the traffic agent behaviors with human expectations, such as collision avoidance. The experimental results show that our method can significantly improve the performance of the pre-trained model on the Waymo Open Sim Agent Challenge (WOSAC) [20]. We also proposed a novel policy evaluation task and demonstrated that the model fine-tuned by our method can achieve more reliable AD testing result.

Limitations. There are several limitations to the approach we have discussed in this paper. We use a simple transition and action model (based on predicting accelerations and integrating them to estimate positions) as the environment dynamics model, which could produce kinematically unrealistic behaviors during a rollout. A more realistic solution would be to embed a low-level controller into simulation that attempts to reach the positions predicted by the model. In addition, we studied a reward function (Eq. 2) that encourages collision avoidance and minimizes divergence in closed-loop simulation. The reward function can be extended to induce various driving behaviors, such as encouraging adversarial behavior (e.g. using the negative of the ego vehicle's reward) to stress-test challenging scenarios.

References

1. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and autonomous systems **57**(5), 469–483 (2009)
2. Bergamini, L., Ye, Y., Scheel, O., Chen, L., Hu, C., Del Pero, L., Osiński, B., Grimmett, H., Ondruska, P.: Simnet: Learning reactive self-driving simulations from real-world observations. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 5119–5125. IEEE (2021)
3. Black, K., Janner, M., Du, Y., Kostrikov, I., Levine, S.: Training diffusion models with reinforcement learning. arXiv preprint arXiv:2305.13301 (2023)
4. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
5. Codevilla, F., Müller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end driving via conditional imitation learning. In: 2018 IEEE international conference on robotics and automation (ICRA). pp. 4693–4700. IEEE (2018)
6. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)
7. Feng, L., Li, Q., Peng, Z., Tan, S., Zhou, B.: Trafficgen: Learning to generate diverse and realistic traffic scenarios. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 3567–3575. IEEE (2023)
8. Fu, J., Norouzi, M., Nachum, O., Tucker, G., Novikov, A., Yang, M., Zhang, M.R., Chen, Y., Kumar, A., Paduraru, C., et al.: Benchmarks for deep off-policy evaluation. In: International Conference on Learning Representations (2020)
9. Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., Harb, J., Pan, X., Wang, Y., Chen, X., et al.: Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. arXiv preprint arXiv:2310.08710 (2023)
10. Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., et al.: Planning-oriented autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17853–17862 (2023)
11. Kamenev, A., Wang, L., Bohan, O.B., Kulkarni, I., Kartal, B., Molchanov, A., Birchfield, S., Nistér, D., Smolyanskiy, N.: Predictionnet: Real-time joint probabilistic traffic prediction for planning, control, and simulation. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 8936–8942. IEEE (2022)
12. Leurent, E.: An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env` (2018)
13. Li, Q., Peng, Z., Feng, L., Liu, Z., Duan, C., Mo, W., Zhou, B.: Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling. Advances in Neural Information Processing Systems (2023)
14. Li, Q., Peng, Z., Feng, L., Zhang, Q., Xue, Z., Zhou, B.: Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. IEEE transactions on pattern analysis and machine intelligence (2022)
15. LLC, W.: Waymo open dataset: An autonomous driving dataset (2019)
16. Lu, J., Batra, D., Parikh, D., Lee, S.: Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. Advances in neural information processing systems **32** (2019)

17. Lu, Y., Fu, J., Tucker, G., Pan, X., Bronstein, E., Roelofs, R., Sapp, B., White, B., Faust, A., Whiteson, S., et al.: Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios. In: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 7553–7560. IEEE (2023)
18. Mendez-Lucio, O., Nicolaou, C.A., Earnshaw, B.: Mole: a molecular foundation model for drug discovery. In: NeurIPS 2022 Workshop on Learning Meaningful Representations of Life (2022)
19. Min, C., Zhao, D., Xiao, L., Nie, Y., Dai, B.: Uniworld: Autonomous driving pre-training via world models. arXiv preprint arXiv:2308.07234 (2023)
20. Montali, N., Lambert, J., Mougin, P., Kuefler, A., Rhinehart, N., Li, M., Gulino, C., Emrich, T., Yang, Z., Whiteson, S., et al.: The waymo open sim agents challenge. arXiv preprint arXiv:2305.12032 (2023)
21. Montali, N., Lambert, J., Mougin, P., Kuefler, A., Rhinehart, N., Li, M., Gulino, C., Emrich, T., Yang, Z., Whiteson, S., et al.: The waymo open sim agents challenge. arXiv preprint arXiv:2305.12032 (2023)
22. Moor, M., Banerjee, O., Abad, Z.S.H., Krumholz, H.M., Leskovec, J., Topol, E.J., Rajpurkar, P.: Foundation models for generalist medical artificial intelligence. Nature **616**(7956), 259–265 (2023)
23. Nayakanti, N., Al-Rfou, R., Zhou, A., Goel, K., Refaat, K.S., Sapp, B.: Wayformer: Motion forecasting via simple & efficient attention networks. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 2980–2987. IEEE (2023)
24. OpenAI: Gpt-4 technical report (2023)
25. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems **35**, 27730–27744 (2022)
26. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125 **1**(2), 3 (2022)
27. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 627–635. JMLR Workshop and Conference Proceedings (2011)
28. Seff, A., Cera, B., Chen, D., Ng, M., Zhou, A., Nayakanti, N., Refaat, K.S., Al-Rfou, R., Sapp, B.: Motionlm: Multi-agent motion forecasting as language modeling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8579–8590 (2023)
29. Shi, S., Jiang, L., Dai, D., Schiele, B.: Motion transformer with global intention localization and local movement refinement. Advances in Neural Information Processing Systems **35**, 6531–6543 (2022)
30. Shi, S., Jiang, L., Dai, D., Schiele, B.: Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. arXiv preprint arXiv:2306.17770 (2023)
31. Suo, S., Regalado, S., Casas, S., Urtasun, R.: Trafficsim: Learning to simulate realistic multi-agent behaviors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10400–10409 (2021)
32. Uehara, M., Shi, C., Kallus, N.: A review of off-policy evaluation in reinforcement learning. arXiv preprint arXiv:2212.06355 (2022)

33. Vinitsky, E., Lichtlé, N., Yang, X., Amos, B., Foerster, J.: Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. Advances in Neural Information Processing Systems **35**, 3962–3974 (2022)
34. Wang, Y., Zhao, T., Yi, F.: Multiverse transformer: 1st place solution for waymo open sim agents challenge 2023. arXiv preprint arXiv:2306.11868 (2023)
35. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**, 229–256 (1992)
36. Zhang, C., Tu, J., Zhang, L., Wong, K., Suo, S., Urtasun, R.: Learning realistic traffic agents in closed-loop. In: 7th Annual Conference on Robot Learning (2023)
37. Zhang, Q., Gao, Y., Zhang, Y., Guo, Y., Ding, D., Wang, Y., Sun, P., Zhao, D.: Trajgen: Generating realistic and diverse trajectories with reactive and feasible agent behaviors for autonomous driving. IEEE Transactions on Intelligent Transportation Systems **23**(12), 24474–24487 (2022)
38. Zhang, Z., Liniger, A., Dai, D., Yu, F., Van Gool, L.: Trafficbots: Towards world models for autonomous driving simulation and motion prediction. arXiv preprint arXiv:2303.04116 (2023)
39. Zhong, Z., Rempe, D., Xu, D., Chen, Y., Veer, S., Che, T., Ray, B., Pavone, M.: Guided conditional diffusion for controllable traffic simulation. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 3560–3566. IEEE (2023)
40. Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., Zhang, K., Ji, C., Yan, Q., He, L., et al.: A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. arXiv preprint arXiv:2302.09419 (2023)
41. Zhou, M., Luo, J., Villella, J., Yang, Y., Rusu, D., Miao, J., Zhang, W., Alban, M., Fadakar, I., Chen, Z., Huang, A.C., Wen, Y., Hassanzadeh, K., Graves, D., Chen, D., Zhu, Z., Nguyen, N., Elsayed, M., Shao, K., Ahilan, S., Zhang, B., Wu, J., Fu, Z., Rezaee, K., Yadmellat, P., Rohani, M., Nieves, N.P., Ni, Y., Banijamali, S., Rivers, A.C., Tian, Z., Palenicek, D., bou Ammar, H., Zhang, H., Liu, W., Hao, J., Wang, J.: Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving (2020)
42. Zhou, Y., Chia, M.A., Wagner, S.K., Ayhan, M.S., Williamson, D.J., Struyven, R.R., Liu, T., Xu, M., Lozano, M.G., Woodward-Court, P., et al.: A foundation model for generalizable disease detection from retinal images. Nature pp. 1–8 (2023)
43. Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., et al.: Rt-2: Vision-language-action models transfer web knowledge to robotic control. In: 7th Annual Conference on Robot Learning (2023)