Hierarchical Model-Based Imitation Learning for Planning in Autonomous Driving

Eli Bronstein Mark Palatucci[1] Dominik Notz Brandyn White Alex Kuefler Yiren Lu Supratik Paul Payam Nikdel[1] Paul Mougin Hongge Chen Justin Fu Austin Abrams Punit Shah Evan Racah[1] Benjamin Frenkel Shimon Whiteson Dragomir Anguelov

***Abstract*— We demonstrate the first large-scale application of model-based generative adversarial imitation learning (MGAIL) to the task of dense urban self-driving. We augment standard MGAIL using a hierarchical model to enable generalization to arbitrary goal routes, and measure performance using a closed-loop evaluation framework with simulated interactive agents. We train policies from expert trajectories collected from real vehicles driving over 100,000 miles in San Francisco, and demonstrate a steerable policy that can navigate robustly even in a zero-shot setting, generalizing to synthetic scenarios with novel goals that never occurred in real-world driving. We also demonstrate the importance of mixing closed-loop MGAIL losses with open-loop behavior cloning losses, and show our best policy approaches the performance of the expert. We evaluate our imitative model in both average and challenging scenarios, and show how it can serve as a useful prior to plan successful trajectories.**

I. INTRODUCTION

Driving at scale in dense urban environments remains difficult due to the complexity of interactions between large numbers of diverse actors. In these scenarios, it is difficult to apply classic motion planning methods that require defining cost functions such that the emergent behavior fully aligns with human expectations [1]. This motivates imitation learning (IL), which uses expert demonstrations to learn either a cost function or a policy directly over actions [2]–[4].

In practice, any imitation model used for motion planning needs additional safety considerations to enforce hard constraints such as collision avoidance and kinematic feasibility. Nonetheless, studying the driving ability of an imitative model in isolation gives an indication of when and where it produces a feasible prior that could be used in an AV stack to plan successful trajectories. In these situations, motion planning could reduce to verifying trajectories from a model instead of generating them with bespoke solutions.

A common challenge with IL is covariate shift, also known as the "DAgger problem" [5]. This occurs when the policy makes small errors that cause it to visit states outside of its training distribution, resulting in compounding error and divergent behavior. Intuitively, this occurs when the policy encounters unfamiliar states, and is similar to challenges in offline reinforcement learning [6].

State-of-the-art imitation methods like model-based generative adversarial imitation learning (MGAIL) [7] address covariate shift through *closed-loop training*, where dynamics are simulated and losses are backed up over the time horizon. Hence, the value of a decision depends on its long-term consequences, in contrast to open-loop behavior cloning, which treats each timestep independently. While theory predicts the importance of closed-loop training [8], empirical evidence in the self-driving literature is limited [9]. This work affirms the benefit of closed-loop training on a practical, large-scale, and difficult motion planning task.

In autonomous driving, a learned motion policy must not only realistically imitate the expert, but also be goal-directed. Such a policy can be challenging to develop due to the confounding of high-level task planning and low-level motion planning [10]: often a trajectory is observed from the expert, but the high-level intents or goals that affect lane choice, future route, or final destination are hidden, making it difficult to recover the causal factors that led to the observed trajectory. One promising solution is the use of hierarchical methods, which decompose the problem into a high-level goal generation module and a low-level goal-conditioned motion policy [11], [12]. During training, this allows the motion policy to associate the goal with the expert's intent-driven behavior. At inference time, this approach offers the flexibility to specify novel goals and generalize beyond the observed expert trajectories.

Ultimately, we aim to develop a policy that can safely navigate a diversity of driving situations and accomplish novel goals, including those not demonstrated by the expert. While we show that employing closed-loop training with respect to the ego vehicle's dynamics is instrumental in creating such a policy, an important question is how to properly evaluate it. Given a dataset of driving scenes with logged vehicle trajectories, evaluating an imitative policy on the same goals achieved by the expert can lead to an overly optimistic performance estimate due to spurious correlations between input features [13]. For example, other vehicles' logged trajectories can influence the policy to follow the expert's goal, rather than actively interacting with other actors to reach its own goal. For this reason, it is critical to evaluate the policy's ability to follow novel goals. This poses a challenge when simulating driving because, as the autonomous vehicle (AV) diverges from its logged trajectory to achieve a new goal, other actors' logged trajectories may become unrealistic. To address this issue, we introduce the combination of goal generalization with *closed-loop evaluation*, in which the policy attempts to reach novel goals in the presence of realistic actors that react to the AV's new actions [9].

Even with the simulation of other interactive actors, it is also important to measure performance in challenging and rare scenarios. Aggregating over a large dataset can mask the performance on difficult but uncommon situations, misrepresenting the model's ability to handle the "long-tail" [14].

The key ingredients for closed-loop, machine-learned planner

All authors with Waymo Research. Contact: shimonw@waymo.com
[1]Work performed while employed at Waymo.

development are, now for the first time, readily available: closed-loop imitation learning with MGAIL, hierarchical goal-based policies, and realistic interactive agents. In this work, we show how to train and evaluate such a system by demonstrating the first application of MGAIL on a large and practical self-driving task of ego vehicle motion planning for dense urban driving. Our method outperforms prior imitation approaches based on pure open-loop optimization like behavior cloning, and achieves aggregate performance similar to the expert demonstrator. We report several key design choices and experimental contributions:

- We introduce a hierarchical model that combines a high-level graph-based search with a low-level transformer-based MGAIL policy, adding an intermediate set of route features to help the model generalize and follow arbitrary goal routes.
- We evaluate our policy's ability to follow novel goal routes alongside simulated reactive agents in closed-loop in order to obtain more realistic estimates of zero-shot generalization and allow for interaction between the policy and other actors.
- We run experiments on both average driving and challenging scenarios to estimate "long-tail" performance and highlight the best opportunities for hill-climbing.
- We run several ablations and show that augmenting MGAIL's closed-loop adversarial losses with an open-loop behavior cloning loss leads to better performance.

II. Related Work

Imitation learning (IL) has a long history in the robotics and machine learning literature, often appearing under names such as learning from demonstration, apprenticeship learning, inverse reinforcement learning, and inverse optimal control [15]–[20]. For an overview see [21], [22]. IL is also closely related to offline reinforcement learning [6]. Theoretical understanding of IL continues to improve with the seminal work of [5] and more recently [8], [23]. Modern approaches to IL use techniques from generative adversarial networks [7], [24]–[26] and include goal-conditioning [12].

IL has been applied to autonomous driving dating to the early success of ALVINN [2], and more recently [3], [4], [27]–[29]. Combining expert demonstrations and reinforcement learning (RL) offers promising new approaches to scalable self-driving [14], [30]. Despite the excitement of machine learning as a path towards large-scale deployment of AVs, many AV companies still rely heavily on classic search-based planning and trajectory optimization. For a survey of classic approaches, see [31].

While motion forecasting models [32] have had a long history in AV stacks to predict other agent behavior, recent work has applied these models to the ego agent to predict feasible trajectories for direct planning, and can be viewed in the context of open-loop imitation [33]–[39].

Closed-loop simulation continues to advance, both for the purposes of evaluating driving performance through realistic world models [9], [40]–[44], and to train driving policies that could transfer to the real world [45], [46].

The work most similar to ours is [9], which applies model-based imitation and parallel beam search to train simulated agents for testing an AV. By contrast, we focus on the ego AV motion planning problem, where following arbitrary goal routes is critical. We avoid beam search as done by [9], which requires future information from reference trajectories not available in the context of ego agent motion planning [47], [48]. We also focus on dense urban driving.

III. Background

A. Markov Decision Processes

We formulate the planning problem as a Markov Decision Process, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, p_0)$. $\mathcal{S}$ represents the state space, which encapsulates quantities such as the positions and velocities of cars, road features, and dynamic objects within the scenario. $\mathcal{A}$ represents the action space, $\mathcal{R}$ the reward function, γ the discount factor, $\mathcal{T}$ the transition distribution, and p_0 the initial state distribution. A policy, or decision making agent, with parameters θ is denoted by π_θ. A typical goal is to find a policy that maximizes the expected return, $\mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t r(s_t, a_t)\right]$, where $\tau = \{s_0, a_0, \ldots, s_T, a_T\}$ denotes a trajectory.

Many classes of techniques have been proposed to find good policies. Methods such as reinforcement learning and optimal control can be used to directly find a strong policy when the reward function is available [49]. However, it can be difficult to design a suitable reward function in many application areas, including autonomous driving [1]. An alternative approach is imitation learning (IL), in which we assume access to a dataset of trajectories $\mathcal{D} = \{\tau_i\}_{i=0}^{N}$ [21] demonstrated by an expert, denoted as π_E. The objective is then to train a policy to imitate the demonstrations.

B. *Model-based Generative Adversarial Imitation Learning*

The naive approach to imitation learning is to perform supervised learning directly on the expert demonstrations, an approach commonly known as behavior cloning (BC) [2], [50]:

$$\arg\max_\theta \ \mathbb{E}_{s,a\sim\pi_E}[\log \ \pi_\theta(a|s)].$$

However, a policy trained in this way is susceptible to covariate shift at test time, resulting in accumulating error. This means that most BC methods suffer from a quadratic worst-case error scaling with respect to the length of the episode [5].

Generative adversarial imitation learning (GAIL) [24] is an alternative that treats imitation learning as an adversarial game. GAIL samples trajectories using the current policy, trains a discriminator to classify these trajectories against the demonstrations (where the demonstrations are labeled as 1 and sampled trajectories as 0), and then optimizes the policy to make its trajectories indistinguishable from the demonstrations. This can be formalized as finding a Nash equilibrium in the following minimax game between the policy π_θ and the discriminator D_ω (parameterized by ω):

$$\arg\max_\theta \ \arg\min_\omega \ \mathbb{E}_{s,a\sim\pi_\theta}[\log \ D_\omega(s,a)] + \mathbb{E}_{s,a\sim\pi_E}[\log(1 - D_\omega(s,a))].$$

Theoretically, rather than minimizing the error in the conditional distribution $\pi_\theta(a|s)$ as in BC, GAIL minimizes the gap in the occupancy measures (the joint distribution of states and actions $p(s,a)$) between the policy and the expert. This allows GAIL to achieve linear error scaling [8] in the time horizon, greatly reducing the effect of compounding error at the cost of requiring online rollouts through a simulator.

Model-based generative adversarial imitation learning (MGAIL) [7] utilizes a differentiable dynamics model to improve training. GAIL requires the use of high-variance policy gradient estimates or other zero-order optimization methods to optimize the policy because its training objective is not differentiable due to sampling through an unknown transition model. With a differentiable model, gradient estimators such as the reparametrization trick [51] can be used to reduce the variance of the policy update. We leverage the fact that our driving simulator is differentiable, as described in section IV-C, in order to apply MGAIL to our method.

IV. Method

In this section, we first provide an overview of our planning agent's system architecture, including the high-level route generation module and low-level continuous motion policy. We then explain how to evaluate the policy in a data-driven, closed-loop simulation with realistic, interactive agents in order to test generalization to novel goals. Finally, we describe an approach to scalably test the policy's robustness to challenging and rare driving situations.

A. *Route Generation*

In the high-level module, we generate routes across waypoints on a premapped lane-level roadgraph, where nodes are discretized lane waypoints, and edges are maneuvers between them: stay in the lane, take a turn/merge/fork, or execute a lane change (Figure 1).

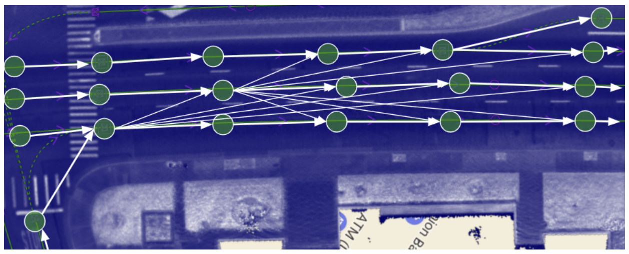

Fig. 1: The routing graph built on a lane-level map. Nodes are fixed lane waypoints, and edges are maneuvers between them (not all lane changes are shown for clarity).

Our primary routing cost is proportional to historical traversal time for each maneuver, so as to minimize ETA. Using a bidirectional A* search with the ALT heuristic [52], we generate one or more routes, depending on the roadgraph. During data collection, the expert is tasked with following the lowest cost route. For training, we condition the low-level motion policy on this same route. However, for evaluation, other routes may be chosen to test the policy's generalization capabilities.

B. *Observation Encoding*

Transformer architectures [53] have been shown to be effective for modeling high-dimensional sequences. We use stacked transformer-based models [47], [54] to encode multimodal observations for the MGAIL policy and discriminator.

We divide the input features into five groups: the AV's own trajectories (e.g., position, heading), other context objects' trajectories (e.g., other vehicles and pedestrians), roadgraph points (positions and types of points sampled from continuous road network features), traffic light signals, and the goal route generated by the high-level module (represented as a sequence of points).

The computational complexity and memory usage of conventional transformer models' cross-attention stages grow quadratically with the features' input dimension. Instead, we use cross-attention with linear computational complexity in terms of the input dimension. Inspired by Set-Transformer [55] and Perceiver [48] approaches, we use learned arrays as queries to fuse the multimodal inputs. This design is flexible since we can fuse the inputs in any order, which is chosen as a hyperparameter based on empirical results. As shown in Figure 2, we first concatenate features in each group and encode them separately using dense layers (size (32, 32), L2 regularization with weight 0.01, and layer normalization [56]). The dimensions of each group's output embeddings are the same. The model is a stack of cross-attention blocks that iteratively attend to each group of features with a fixed-dimensional learnable latent bottleneck (sizes 128 and 64 for the policy and discriminator) . Each cross-attention block has a residual attention layer followed by a dense layer. The output policy and discriminator observation embeddings are then used as inputs to the MGAIL policy and discriminator heads, respectively. For the policy head only, we employ a gated recurrent unit (GRU) [57] to integrate the output policy observation embeddings over time.

C. *Delta Actions Model*

While the trajectories of other vehicles observed during data collection are given, we must choose a dynamics model for the AV. We utilize the delta actions model, which uses offsets to the current state s as actions a to compute the next state $s' = s + a$. Demonstrated actions can be easily obtained by differentiating consecutive demonstrated trajectory states: $a = s' - s$. This model is fully differentiable but is not learned from data. Unlike traditional vehicle dynamics models, this model does not enforce constraints on the behavior space. This characteristic makes it flexible enough to cover a diverse set of driving behaviors, and it could even be used for non-vehicle agents, such as pedestrians. In practice, an AV stack would include a local controller that attempts to realize the next state generated by the dynamics model.

For the policy, we use a Gaussian Mixture Model (GMM) to parameterize the delta actions, where the GMM parameters are predicted by a neural network. The distribution over actions a given a state s is given by:

$$\pi_\theta(a|s) = \sum_{i=1}^{K} \phi_\theta^i(s)\mathcal{N}(a|\mu_\theta^i(s), \Sigma_\theta^i(s)),$$

where $\mathcal{N}(\cdot)$ denotes a Gaussian distribution, K (set to 8) is the number of Gaussians, and ϕ_θ^i, μ_θ^i, and Σ_θ^i for $i \in \{1, \dots, K\}$ are the weights, means, and covariances predicted by the policy network given a state s (see Figure 2). During inference, actions are sampled from the predicted GMM.

A common alternative approach is to directly predict the action values and to minimize the mean squared error (MSE) between the predicted and demonstrated actions. This approach is equivalent to maximizing the log likelihood of the demonstrated action assuming it is sampled from a unimodal Gaussian distribution. However, in our setting multiple actions

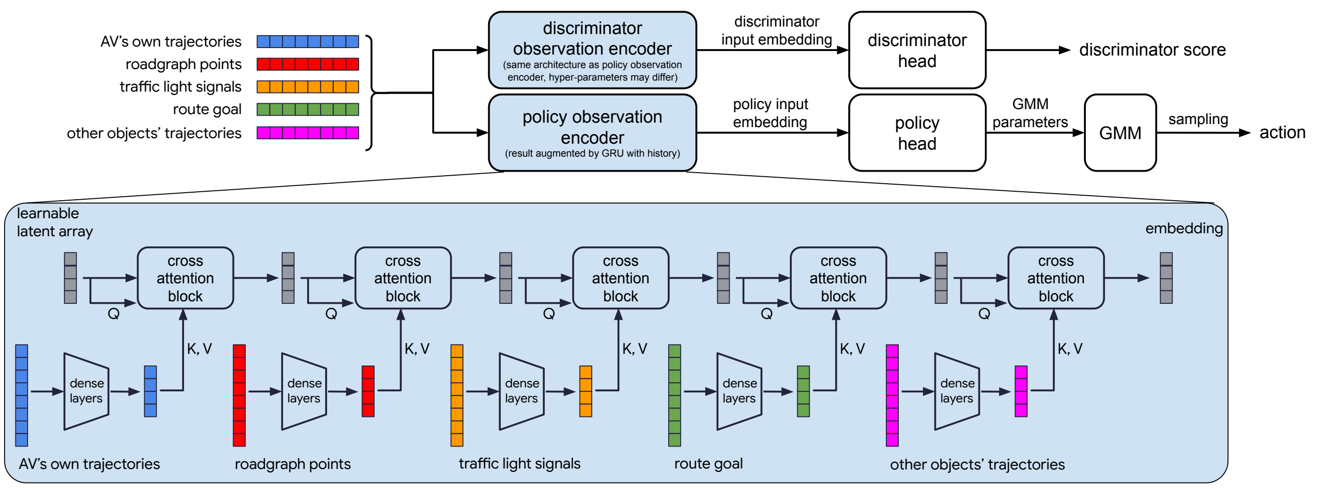

Fig. 2: System diagram of the MGAIL planning agent and its transformer-based observation encoder. "K", "V", and "Q" represent the keys, values, and queries, respectively.

may be appropriate for a given state, which motivates using a GMM to represent different modes.

D. Training Losses

For the low-level continuous motion policy, we combine MGAIL with behavior cloning (BC). The total loss consists of three main components: the discriminator loss $\mathcal{L}_{\mathcal{D}}$, the MGAIL policy loss $\mathcal{L}_{\mathcal{P}}$, and the BC loss $\mathcal{L}_{\mathcal{BC}}$:

$$\mathcal{L} = \lambda_{\mathcal{D}}\mathcal{L}_{\mathcal{D}} + \lambda_{\mathcal{P}}\mathcal{L}_{\mathcal{P}} + \lambda_{\mathcal{BC}}\mathcal{L}_{\mathcal{BC}}, \tag{1}$$

where $\lambda_{\mathcal{D}}$, $\lambda_{\mathcal{P}}$ and $\lambda_{\mathcal{BC}}$ are the coefficients for each loss component. To assess the individual impacts of the MGAIL and BC policy losses, we can adjust the contribution of loss components by reducing or zeroing their coefficients. The BC loss is:

$$\mathcal{L}_{\mathcal{BC}} = -\,\mathbb{E}_{s,a\sim\pi_E}[\log\ \pi_\theta(a|s)]. \tag{2}$$

The MGAIL discriminator loss is:

$$\mathcal{L}_{\mathcal{D}} = \mathbb{E}_{s\sim\pi_\theta}[\log\ D_\omega(s)] + \mathbb{E}_{s\sim\pi_E}[\log(1-D_\omega(s))]. \tag{3}$$

Here we only provide the state s to the discriminator as input similar to [58]. When performing backpropagation, only the discriminator parameters ω are updated. The MGAIL policy loss is:

$$\mathcal{L}_{\mathcal{P}} = -\,\mathbb{E}_{s\sim\pi_\theta}[\log\ D_\omega(s)]. \tag{4}$$

During backpropagation, only the policy parameters θ are updated for this loss term. Since our dynamics model is differentiable, the gradients are propagated back in time and influence the actions of the policy in earlier timesteps [7]. We use the reparametrization trick [51] to differentiate through the stochastic GMM policy. We minimize these losses using Adam [59] (default parameters), a learning rate of 0.0003, and gradient clipping (max L2 norm of 10).

E. Closed-Loop Evaluation with Interactive Agents

We evaluate the policy by having it control the AV in driving situations recorded during data collection. To test the policy's ability to reproduce expert behavior, we can condition it on the expert's goal route, while having other road users (RUs) follow their logged trajectories. However, an ideal AV policy should be able to generalize to novel routes that differ from those followed by the expert. In this situation, if other RUs simply follow their logged trajectories, the simulation is likely to become unrealistic due to the lack of interaction between the AV and other RUs. As the AV diverges from its logged trajectory, RUs should be able to diverge in response. If they do not, following a novel route may be infeasible without causing a traffic conflict.

We therefore study the performance of our policy's zero-shot generalization to novel routes in the presence of an interactive Symphony agent [9] that governs the behavior of RUs near the AV. The Symphony agent is trained separately using a transformer observation encoder and delta actions model using a combination of MGAIL and BC losses, similar to the AV policy. However, it is not goal-conditioned because its objective is simply to drive realistically rather than achieve a specific goal. This allows us to evaluate the goal-directed planning agent in a realistic, closed-loop simulation that more accurately reproduces the complex interactions of real-world driving.

F. Challenge Classifier

While evaluating on randomly sampled segments yields an unbiased estimate of the policy's performance, it is not very informative of how it handles difficult scenarios. Since driving is a safety-critical task, it is crucial to understand our model's failure modes in rare and challenging settings.

With this in mind, we develop a ML classifier to score segments on their potential for producing a realistic collision or "close call" when simulated with a planning agent. The classifier takes as input the segment and outputs a score in $[0, 1]$ with higher values indicating higher likelihood of a simulated collision or close call. We collect training data by simulating a planning agent on logged data and asking human labelers to identify when a realistic collision or close call occurred, if at all. Timesteps at a ± 3 second offset are positive examples (label 1) and the remaining timesteps are negative examples (label 0). Since collisions and close calls are rare, during training we upsample segments that contain positive examples. We use a fully connected multi-layer perceptron with (1024, 1024, 512, 32) hidden layers. To prevent train-test leakage, we train this classifier on a dataset collected at least 3 months prior to those used for the experiments below. This dataset consists of 145k trajectories, 5.6k of which were deemed to have realistic collisions or close calls by human labelers. The remaining negative trajectories do not require human labels. Using the classifier's scores for the entire test set, we can evaluate the policy on subsets of varying challenge levels.

V. Experiments

A. Dataset

We collected a large sample of 10 million expert trajectories, each a segment of 10 seconds, from a fleet of our vehicles operating in San Francisco. During data collection, the high-level module generates one or more routes toward a destination, and the expert attempts to follow the best route. These trajectories represent over 100,000 miles of expert driving, which we split into 80% *train*, 10% *validation*, and 10% *test* sets. Each set contains disjoint *runs*, meaning that trajectories from the same vehicle operating on the same day exist only within one set. The trajectories are sampled at 15 Hz, and represent ego vehicle state (e.g., pose) and exteroceptive state from a robotic perception system (e.g., other actors' bounding boxes, traffic light state). These trajectories are aligned with the coordinate frame of a high-definition map [37].

B. Metrics

We measure driving performance using three key metrics, each of which reports a success/failure indicator for the segment. Specifically, we compute each metric per timestep, and a failure at any timestep indicates a failure for the segment.

- *Road-Route Failure*: If the trajectory deviates from the goal road-route, which is a traversal through a road network that allows the vehicle to be in any valid lane along the route, similar to a Google Maps style route.
- *Collision*: If the bounding box of the ego vehicle intersects with a bounding box of another object.
- *Off-road*: If the bounding box of the ego vehicle deviates from the drivable surface according to the map.

We combine these metrics to generate an overall success metric for a segment (i.e., a segment is marked successful if it follows the road-route without any collisions or off-road driving). We report the average of each metric, and the overall success rate for the given test set. We also report the *Route Progress Ratio*, which is the ratio of the distance traveled along the route by the policy vs. the expert, given that both followed the route. We use this metric to ensure that the policy makes comparable progress to the expert (higher is better). For each experiment, we specify the number of test segments used and report results with 95% confidence sets using the normal approximation. Finally, in the spirit of Goodhart's law, none of these metrics are visible to the model in our experiments, although loss or reward shaping for safety and to enforce constraints is a natural area for future work [28], [60].

We simulate the policy trajectories, play back the expert trajectories, and compute metrics in our simulator using bounding box approximations for vehicle geometry. References to the expert performance are labeled in the results tables as *Playback*.

C. Results

In this section, we present results of several model variants and ablate key modeling and evaluation decisions, using BC and vanilla MGAIL as baselines. We assess the utility of goal route conditioning, compare the performance on logged routes and novel routes to evaluate the policy's generalization capability, and highlight the importance of closed-loop evaluation with interactive agents. We also demonstrate the policy's performance on more challenging test subsets, as well as the best mixture of MGAIL and BC training losses.

1) ***Can the model safely follow logged routes? What is the impact of route conditioning?*** In Table I we compare six model variants to the expert on the *Unbiased Test Set* - an unbiased sample of dense urban driving consisting of 82,198 segments. We initialize the simulator to the initial state of a scene and compute the route once at the beginning of the segment, keeping it fixed for the duration of the rollout. In this setup, we test the model's ability to successfully follow the *logged route*, which is provided as a conditioning feature. In other words, we ask the model to drive the same route that the expert attempted to follow during the test segment, which is the best route produced by the high-level route generation module. In these experiments, the model only controls the AV, while other vehicles follow their logged trajectories.

"BC" indicates behavior cloning loss ($\mathcal{L}_{\mathcal{BC}}$) only, "MGAIL" means adversarial losses ($\mathcal{L}_{\mathcal{D}}$ and $\mathcal{L}_{\mathcal{P}}$) only, and "MGAIL + BC" indicates a mixture of adversarial and BC losses, where the relative weight of each adversarial loss to the BC loss is 2:1. The weights of both MGAIL adversarial losses are equal.

Without the route conditioning features all model variants struggle to follow the route, even if their overall collision and off-road rates are low. The route conditioning features significantly improve performance of all models, with our best MGAIL + BC variant achieving 99.6% of the expert's success rate. The route features also decrease off-road violations for all variants, and they reduce collisions for the MGAIL and MGAIL + BC variants. We also observe that for the segments where both the policy and the expert followed the road-route, each policy variant makes comparable or slightly more progress along the route compared to the expert.

2) ***Can the model generalize to novel routes?*** One danger of conditioning on logged routes is spurious correlation, where the model may not properly learn the causal relationship of the route. In essence, giving the model the same goal route as the expert overestimates its ability to generalize, since other features may correlate with the original route.

To test the model's generalization to arbitrary routes, we created the *Route Generalization Test Set* - a subset of the *Unbiased Test Set* with 18,593 segments where the high-level route generation module produced multiple, distinct routes. For example, one route may go straight through an intersection while another route may be a right turn. This subset is more challenging than the entire test set because segments with multiple routes are likely to have more complex road networks and potentially more

TABLE I: Logged Route on the *Unbiased Test Set*.

Method	Success rate (%)	Route Failure rate (%)	Collision rate (%)	Off-road rate (%)	Route Progress ratio (%)
Playback	98.62±0.08	1.07±0.07	0.05±0.02	0.26±0.03	100.00±0.00
BC	86.07±0.24	9.63±0.20	4.53±0.14	2.21±0.10	105.59±0.40
BC + Route	94.18±0.16	**0.69±0.06**	4.60±0.14	0.75±0.06	98.10±0.33
MGAIL	88.90±0.21	9.73±0.20	1.28±0.08	1.00±0.07	101.22±0.32
MGAIL + Route	97.45±0.11	0.74±0.06	1.20±0.07	0.77±0.06	100.85±0.29
MGAIL + BC	89.84±0.21	8.93±0.19	1.25±0.08	0.73±0.06	105.58±0.36
MGAIL + BC + Route	**98.22±0.09**	**0.69±0.06**	**0.77±0.06**	**0.37±0.04**	105.30±0.32

interactions with other vehicles. In Table II, we show the policy's ability to follow *novel routes*, where we select a goal route that differs from the one the expert attempted to follow (i.e., the *logged route*). In both the logged and novel route sections of the table, the initial conditions of the segments are identical.

As in our previous experiments, the route conditioned features are critical to overall success and route following. For the model variants without these features, performance drops considerably compared to following logged routes. This shows the danger of correlated features (e.g., other vehicles' trajectories), where the model tends to follow the logged trajectory despite not observing it.

We notice a generalization gap for the route conditioned variants, whose overall success rate drops by as much as 12% compared to when following the logged route. However, the performance of the route conditioned MGAIL variants drops less than that of BC, suggesting that MGAIL helps with generalization. We also see that MGAIL + BC continues to perform the best, but on novel routes the value of adding the BC loss is within the margin of error of MGAIL's purely adversarial losses, in terms of the route failure and collision metrics.

As none of our metrics are included in the training losses, these results suggest that purely associative feature conditioning may be insufficient to achieve robust route generalization necessary for driving that is on par with the expert's performance. Constrained optimization and reward shaping may be required, or refinement through reinforcement learning, which we leave for future work.

We also see that on novel routes, the route-conditioned variants have higher collision and off-road rates compared to logged routes. This is unsurprising, since the other vehicles simply follow their logged trajectories and do not react realistically when the policy attempts to follow the novel route, making it challenging or infeasible for it to do so. For the same reason, collision and off-road rates rise when the model is route-conditioned, compared to its unconditioned counterpart. The unconditioned variants do not explicitly attempt to follow the novel route since they do not observe it, so they are more likely to drive in a generally realistic way that does not require other actors to be interactive.

3) ***What is the impact of closed-loop evaluation with interactive agents?*** In order to preserve simulation realism while the AV deviates from its logged trajectory, we introduce interactive agents to represent other RUs (section IV-E). We create the *Interactive Agent Test Set* by assigning the top 8 RUs to the interactive agent at the beginning of each segment based on their proximity to the AV's logged trajectory. We only replace RUs that were fully observed by the AV's perception system during data collection and reject scenarios with fewer than 8 fully observed RUs, yielding 4,406 segments. Due to the constraint that 8 objects must be fully observed, this dataset overrepresents busy or congested scenes. These scenarios may still contain partially observed RUs, but they follow their logged trajectories.

Table III shows the zero-shot performance on novel routes with reactive agents absent and present. Because of the rejection sampling used to obtain denser scenes, this problem setting is more difficult than the *Unbiased Test Set* presented in table II. However, for a fixed AV policy, including reactive agents reduces the collision rate and route failure rate, and increases the success rate. The reduction in collision rate may be explained by reactive agents avoiding conflicts with the AV. However, the fact that the AV policy improves on the route failure metrics indicates that a significant number of failures were due to routes being infeasible without initiating an interaction with another RU. In some cases, the reactive agents may be responsible for collisions with the AV, but it is challenging to determine which agent is at fault. There is still a gap between logged route performance and zero shot generalization to novel routes, even with reactive agents. Performance may be improved by retraining the AV policy alongside reactive agents, but we leave these experiments for future work.

4) ***What is the performance on challenging scenarios?*** To better understand how the model performs in more challenging scenarios, we evaluate it on test subsets of varying difficulty levels. We first compute a score for each segment in the test set using the challenge classifier (section IV-F). We then create four subsets of increasing levels of predicted collisions and close-calls (number of segments for logged and novel routes in parentheses): *Low* (8,436 and 2,442), *All* (82,198 and 18,593), *Medium* (7,148 and 954), and *High* (2,401 and 302), where *All* refers to the entire *Unbiased Test Set* and *Route Generalization Test Set* for logged and novel routes.

We evaluate the model variants on each of these subsets, both for logged routes and novel routes. As shown in Figure 3, the overall success rate of the expert and all model variants is lower on more challenging subsets, and the model performs worse on novel routes than on logged routes, as expected. Variants that use MGAIL outperform BC across all challenge levels, but most acutely on the more difficult subsets, highlighting the importance of MGAIL in providing robustness to harder driving situations. While MGAIL alone performs comparably to, if not better than, MGAIL + BC on the least challenging subset, MGAIL + BC achieves a higher success rate on the average and more difficult subsets for both

TABLE II: Logged Route vs. Novel Route on the *Route Generalization Test Set*.

Method	Logged Route				Novel Route			
	Success rate(%)	Route Failure rate(%)	Collision rate (%)	Off-road rate (%)	Success rate(%)	Route Failure rate(%)	Collision rate (%)	Off-road rate (%)
Playback	95.73±0.29	3.97±0.28	0.01±0.01	0.31±0.08	-	-	-	-
BC	68.97±0.66	26.63±0.64	5.62±0.33	3.51±0.26	20.22±0.58	76.67±0.61	5.72±0.33	3.77±0.27
BC + Route	92.22±0.38	1.90±0.20	4.94±0.31	1.30±0.16	80.34±0.57	7.82±0.39	11.09±0.45	4.87±0.31
MGAIL	69.82±0.66	28.90±0.65	1.69±0.19	1.95±0.20	25.08±0.62	73.78±0.63	**1.63±0.18**	1.97±0.20
MGAIL + Route	95.91±0.28	**1.86±0.19**	1.31±0.16	1.22±0.16	88.11±0.47	5.80±0.34	4.78±0.31	3.16±0.25
MGAIL + BC	71.97±0.65	26.96±0.64	1.72±0.19	1.37±0.17	20.27±0.58	78.52±0.59	1.69±0.19	**1.32±0.16**
MGAIL + BC + Route	**97.05±0.24**	1.90±0.20	**0.67±0.12**	**0.48±0.10**	**89.65±0.44**	**5.67±0.33**	4.41±0.30	1.60±0.18

TABLE III: Novel Route Without vs. With the Interactive Agent on the *Interactive Agent Test Set*.

Method	Without Reactive Agent				With Reactive Agent			
	Success rate (%)	Route Failure rate (%)	Collision rate (%)	Off-road rate (%)	Success rate (%)	Route Failure rate (%)	Collision rate (%)	Off-road rate (%)
BC + Route	59.07±1.45	8.66±0.83	32.29±1.38	6.63±0.73	72.38±1.32	9.02±0.84	17.22±1.11	6.69±0.74
MGAIL + Route	77.77±1.23	**7.24±0.77**	15.70±1.08	3.03±0.51	83.38±1.10	**6.77±0.74**	9.48±0.87	2.71±0.48
MGAIL + BC + Route	**78.05±1.22**	8.95±0.84	**14.09±1.03**	**2.23±0.44**	**85.69±1.03**	7.04±0.75	**7.27±0.77**	**2.20±0.43**

logged and novel routes. We observe the same patterns in the individual metrics that make up the overall success rate.

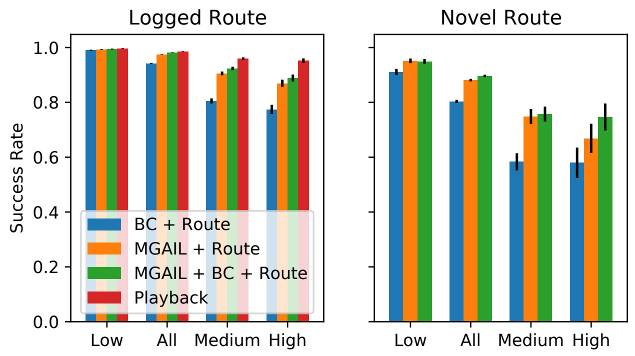

Fig. 3: Logged Route vs. Novel Route success rate for different challenge levels.

*5) **What is the impact of closed-loop training and how should MGAIL be prioritized vs. BC?*** Our results indicate that employing closed-loop training with MGAIL and adding the BC loss produces the highest success rates on both logged and novel routes. This leads to the question of which combination of adversarial and BC losses performs best. To determine this, we evaluate different MGAIL and BC policy loss weight mixtures and compare their success rates. In particular, we show success rates for different relative MGAIL loss weights (Figure 4), which set the ratio of the MGAIL policy loss weight ($\lambda_{\mathcal{P}}$) to the total policy loss weight ($\lambda_{\mathcal{P}} + \lambda_{\mathcal{BC}}$). We obtain the best results by weighting the MGAIL loss more than the BC loss. Furthermore, for novel routes compared to logged routes, a higher relative MGAIL loss weight yields the highest success rate, providing additional evidence that MGAIL is beneficial for generalization.

VI. CONCLUSION AND FUTURE WORK

We demonstrated a hierarchical model-based generative adversarial imitation learning (MGAIL) method that performs similarly to an expert demonstrator on a large unbiased sample of urban driving on key planning metrics. We highlighted the importance of closed-loop training with MGAIL, as well as closed-loop evaluation with interactive agents in order to more accurately assess the planning agent's ability to generalize. Although aggregate performance is similar to the expert, we showed that there still remains a gap on generalization to novel routes, as well as performance on challenging and rare scenarios. This suggests further improvements must focus on this "long-tail" of performance. Refinement through reinforcement learning and reward shaping with safety constraints, or active curriculum learning on specific data subsets are two promising directions for future work. In addition, training the planning agent directly alongside pre-trained interactive agents may improve the policy's ability to interact with and influence other actors, facilitating better generalization to novel goals. Incorporating this policy into a complete AV platform to drive real-world vehicles is a natural next step, which will likely necessitate dynamic rerouting and adapting our action model to include realistic vehicle dynamics.

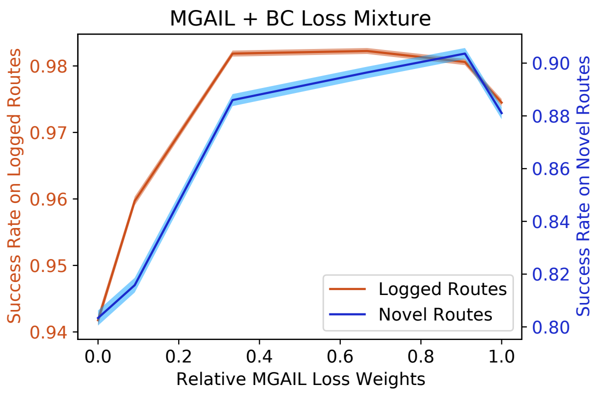

Fig. 4: Success rates of route-conditioned models on logged and novel routes for different relative MGAIL loss weights.

ACKNOWLEDGMENT

We thank Rami Al-Rfou, Ury Zhilinsky, Ben Sapp, James Philbin, and Jonathan Bingham for their helpful comments, and Aleksei Timofeev for his contributions to model-based imitation learning during his employment at Waymo.

REFERENCES

[1] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (mis)design for autonomous driving," *CoRR*, vol. abs/2104.13906, 2021. [Online]. Available: https://arxiv.org/abs/2104.13906

[2] D. A. Pomerleau, "ALVINN: an autonomous land vehicle in a neural network," in *Advances in neural information processing systems 1*, 1989.

[3] M. Bojarski *et al.*, "End to end learning for self-driving cars," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[4] M. Bansal *et al.*, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," 2018. [Online]. Available: http://arxiv.org/abs/1812.03079

[5] S. Ross *et al.*, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AI Stats*, 2011.

[6] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," May 2020.

[7] N. Baram, O. Anschel, I. Caspi, and S. Mannor, "End-to-end differentiable adversarial imitation learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 390–399.

[8] G. Swamy, S. Choudhury, J. A. Bagnell, and Z. S. Wu, "Of moments and matching: A game-theoretic framework for closing the imitation gap," 2021.

[9] M. Igl, D. Kim, A. Kuefler, P. Mougin, P. Shah, K. Shiarlis, D. Anguelov, M. Palatucci, B. White, and S. Whiteson, "Symphony: Learning realistic and diverse agents for autonomous driving simulation," in *Robotics and Automation (ICRA), 2022 IEEE International Conference on*, 2022.

[10] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," 2020.

[11] A. Mandlekar *et al.*, "Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4414–4420.

[12] Y. Ding, C. Florensa, M. Phielipp, and P. Abbeel, "Goal-conditioned imitation learning," June 2019.

[13] P. de Haan, D. Jayaraman, and S. Levine, "Causal confusion in imitation learning," 2019.

[14] A. Jain, L. D. Pero, H. Grimmett, and P. Ondruska, "Autonomy 2.0: Why is self-driving always 5 years away?" 2021.

[15] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *in Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 663–670.

[16] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*, 2004.

[17] F. Behbahani, K. Shiarlis, X. Chen, V. Kurin, S. Kasewa, C. Stirbu, J. Gomes, S. Paul, F. A. Oliehoek, J. Messias, and S. Whiteson, "Learning from demonstration in the wild," Nov. 2018.

[18] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, vol. 8, 2008, pp. 1433–1438.

[19] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning*, 2016, pp. 49–58.

[20] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 729–736.

[21] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Rob. Auton. Syst.*, 2009.

[22] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, apr 2017. [Online]. Available: https://doi.org/10.1145/3054912

[23] P. A. Ortega *et al.*, "Shaking the foundations: delusions in sequence models for interaction and control," 2021.

[24] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[25] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," Oct. 2017.

[26] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," in *Advances in Neural Information Processing Systems*, 2017, pp. 3812–3822.

[27] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," 2018.

[28] M. Vitelli *et al.*, "Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies," 2021.

[29] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," 2019.

[30] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," July 2018.

[31] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," Apr. 2016.

[32] B. Varadarajan *et al.*, "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction," 2021.

[33] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," 2019.

[34] N. Rhinehart, R. McAllister, and S. Levine, "Deep imitative models for flexible inference, planning, and control," 2019.

[35] J. Liu, W. Zeng, R. Urtasun, and E. Yumer, "Deep structured reactive planning," 2021.

[36] S. Casas, A. Sadat, and R. Urtasun, "Mp3: A unified model to map, perceive, predict and plan," 2021.

[37] S. Ettinger *et al.*, "Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset," 2021.

[38] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," 2021.

[39] J. Ngiam *et al.*, "Scene transformer: A unified architecture for predicting multiple agent trajectories," 2021.

[40] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "Trafficsim: Learning to simulate realistic multi-agent behaviors," 2021.

[41] H. Caesar *et al.*, "nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles," 2021. [Online]. Available: https://arxiv.org/abs/2106.11810

[42] M. Zhou *et al.*, "Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving," *arXiv preprint arXiv:2010.09776*, 2020.

[43] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," *arXiv preprint arXiv:1711. 03938*, 2017.

[44] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," 2019.

[45] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," 2017.

[46] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," 2018.

[47] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal motion prediction with stacked transformers," 2021.

[48] A. Jaegle *et al.*, "Perceiver: General perception with iterative attention," in *ICML*, 2021.

[49] M. P. Deisenroth *et al.*, "A survey on policy search for robotics," *Foundations and trends in Robotics*, 2013.

[50] D. Michie, M. Bain, and J. Hayes-Miches, "Cognitive models from subcognitive skills," *IEE control engineering series*, vol. 44, pp. 71–99, 1990.

[51] M. Xu *et al.*, "Variance reduction properties of the reparameterization trick," in *AI Stats*, 2019.

[52] A. Goldberg and C. Harrelson, "Computing the shortest path: A* search meets graph theory," *Symposium on Discrete Algorithms*, 2003.

[53] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, 2017.

[54] J. Mercat *et al.*, "Multi-head attention for multi-modal joint vehicle motion forecasting," in *ICRA*, 2020.

[55] J. Lee *et al.*, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *ICML*, 2019.

[56] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[57] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[58] F. Torabi, G. Warnell, and P. Stone, "Generative adversarial imitation from observation," *arXiv preprint arXiv:1807.06158*, 2018.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[60] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," 2016.