IMITATIVE MODELS: PERCEPTION-DRIVEN FORECASTING FOR FLEXIBLE PLANNING AND CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Imitation learning provides an appealing framework for autonomous control: in many tasks, demonstrations can be readily obtained from human experts, and using demonstration data removes the need for costly and potentially dangerous on-policy trials in the real world. On the other hand, model-based reinforcement learning offers considerably more flexibility: a model learned from data can be reused at test-time to achieve a wide variety of goals. In this paper, we aim to combine these benefits to learn imitative models: predictive models that can be used to plan behaviors at test-time that achieve user specified goals, while remaining close to the distribution of behaviors seen in the demonstration data. We find that this method substantially improves on the performance of both direct imitation and model-based RL in a simulated driving task, and can be learned efficiently and safely using only demonstration data without on-policy data collection.

1 INTRODUCTION

Reinforcement learning (RL) algorithms offer the promise of automatically learning controllers and behavioral skills for a range of tasks from raw sensory inputs, with minimal manual engineering. However, reinforcement learning at its core requires iterative rounds of data collection: the agent must repeatedly interact with the world using its latest behavioral strategy, collect more data, update some internal model or policy, and repeat. While this is natural in some settings, safety-critical domains often make this exceptionally difficult. Consider, for example, the domain of autonomous driving, which we will use as a working example in this paper. Deploying a partially trained policy on a real-world vehicle can result in a dangerous crash. On the other hand, prior data of real-world driving is plentiful, especially from human drivers. How can such data be leveraged most effectively to learn a driving policy? Off-policy reinforcement learning algorithms can, in principle, make use of this data to learn a model, while imitation learning methods can use this data to directly learn a policy. As we will discuss below, both options have substantial downsides.

Off-policy reinforcement learning methods, such as value function-based methods (Watkins & Dayan, 1992; Mnih et al., 2013) or model-based reinforcement learning (Kuvayev & Sutton, 1996; Deisenroth & Rasmussen, 2011), can train a model agnostic to the data collection method. Particularly in the case of model-based reinforcement learning, such a model can not only use off-policy data, but does not even require a known reward function at training time, making this option particularly appealing in the real world. Once trained, the model can be used to flexibly achieve a variety of user specified goals: insofar as the model is an accurate model of the world, any feasible goal can in principle be achieved by planning through the model, essentially treating the model as a simulator. However, in practice, both model-based and model-free reinforcement learning algorithm are very vulnerable to distributional shift (Bagnell & Schneider, 2001; Ross et al., 2011): when acting in the real world according to the learned model or policy, the agent will visit states that are different from those seen during training, and in those it will likely be unable to determine an effective course of action. This is especially problematic when the data comes from a curated source, such as demonstration data from human drivers: this data intentionally excludes adverse events such as crashes, which means that the model does not learn that a crash is even possible, and therefore cannot know how to prevent it. Therefore, reinforcement learning algorithms typically require additional on-policy data collection and data aggregation: iterative rounds of additional data collection, where the model is used to act in the world, collect more data, and append this data to the model's training set (Liang et al., 2018). This can be impractical in safety-critical settings such as driving, where

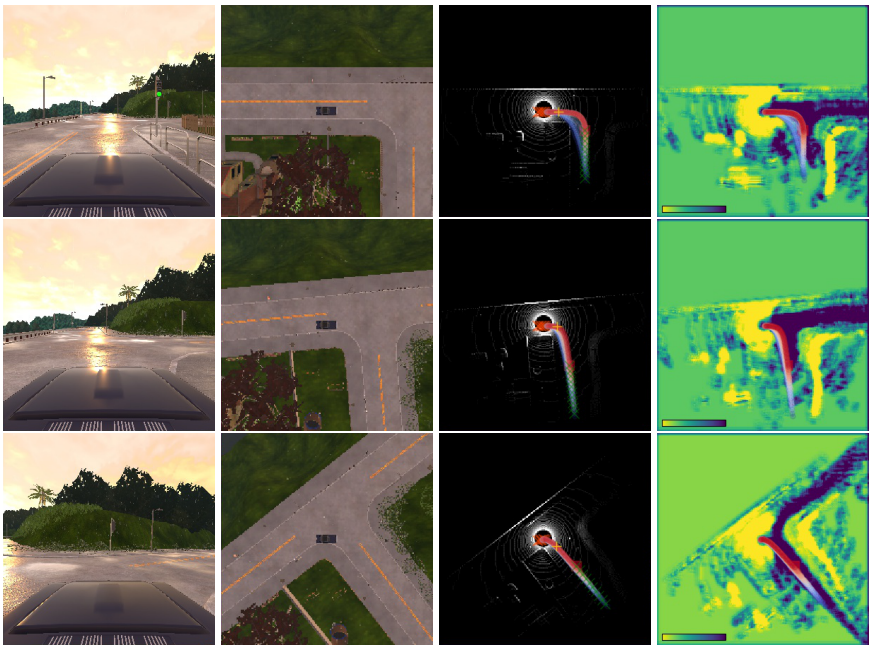


Figure 1: We apply our model to navigating a car in the CARLA simulator. *Columns 1,2:* Images depicting the current scene. *Column 3:* LIDAR input and high-level goals are provided to our learned deep generative trajectory model, and plans to the goals are computed under the model's posterior. Each plan is colored according to its likelihood ranking, with red indicating the best plan. The red square indicates the chosen high-level goal, and the yellow cross indicates a point along our plan used as a set point for a proportional controller. *Column 4:* Our model can incorporate arbitrary test-time costs, and use them to adjust its planning objective and plan ranking.

demonstration data is readily available, but trial-and-error learning in the real world is difficult and dangerous.

Imitation learning algorithms can make use of expert-provided demonstration data and, in principle, can learn effective control strategies without any additional on-policy data collection. This makes them simple and practical to deploy in the real world, especially for safety-critical tasks such as autonomous driving. However, imitation learning offers comparatively little flexibility in terms of the task: standard imitation learning methods simply copy the behavior of the expert, and while several works have proposed to augment imitation learning with goal conditioning (Codevilla et al., 2017), these goals must be specified in advance during training, and are typically comparatively simple (e.g., turning left or right).

In this work, we develop a control algorithm that bridges imitation learning and model-based reinforcement learning. The combination of these methods offers several highly desirable properties. By learning from human-provided data, we can capture the distribution of human-like behaviors without the need for manually specified cost or reward functions that describe general task constraints. For example, in the case of a driving task, our model automatically ensures that the car stays on the road and obeys lane markings. This also enables us to dispense with on-policy data collection and data aggregation, learning entirely from previously provided demonstration data. On the other hand, by incorporating goal-driven planning, as in model-based RL, our model can easily follow user-specified goals at test-time without any additional training.

The main idea behind our method is to reformulate the control problem as one of probabilistic inference: in order to determine a course of action that satisfies some goal, our method infers a sequence of actions and states that maximize the probability of that goal being met, under the prior that the trajectory distribution should match the distribution over trajectories observed in the data. This inference procedure resembles planning or trajectory optimization in model-based reinforcement learning, while the process of learning this model resembles imitation learning. Setting up a

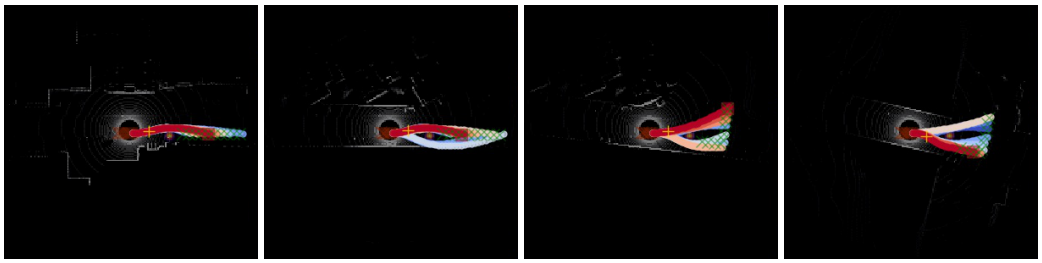


Figure 2: Density Planning to goals subject to an cost at test time. The cost (circular yellow bump) corresponds to an imaginary "pothole", which the planner is tasked with avoiding. The planner prefers routes that curve around the pothole (red=high preference, blue=low preference), stay on the road, and respect intersections. Demonstrations of this behavior were never observed by our model.

model that is amenable to this type of inference-based planning requires a powerful and flexible distribution over sequences, which we construct by building a sequence model with normalizing flows, conditioned on high-bandwidth LIDAR observations of the current environment.

The main contribution of our paper is a model-based reinforcement learning method that hybridizes model-based RL and imitation learning, using a stochastic model of expert behavior that acts as a prior for inferring plans that achieve user-specified goals. We demonstrate our method on a simulated driving task (see Figure 1), and illustrate how it can be used to achieve navigational tasks at test-time that follow waypoints produced by a conventional planner, while at the same time obeying the rules of the road and avoiding collisions. In contrast to standard imitation learning, our method produces an interpretable distribution over trajectories and can follow navigational goals without additional training. Some examples of such goals are illustrated in Figures 2. In contrast to model-based RL, our method does not require on-policy data collection or aggregation. In a comparative evaluation, we find that our method substantially outperforms both model-based RL and direct imitation learning, and we observe that it can efficiently learn near-perfect navigation through the static-world CARLA simulator from just 7000 training examples. Video results of our method are available.[1]

2 METHOD

Intuitively, we plan state trajectories towards a goal by searching over a restricted set of "valid" transitions: transitions seen performed previously by expert drivers. Rather than reasoning strictly about "valid" vs "invalid" transitions, however, we take a probabilistic approach. Our method fits a probabilistic dynamical model, q, to the distribution of expert driving behavior, p, passively observed in a training set. We learn a probabilistic model because expert behavior is inherently multimodal: choosing to turn either left or right at an intersection are both common decisions. At test time, q serves as a learned prior over the space of *undirected* expert trajectories. To execute samples from this distribution, or the maximum a posteriori (MAP) transition, is to imitate an expert driver in an undirected fashion.

Besides simply imitating the demonstrations, we wish to *direct* our agent to desired goals, and have the agent reason automatically about the mid-level details necessary to achieve these goals. High-level goals take the form of *waypoints* in our setting. To direct our car to each waypoint, we propose path planning as an inference procedure: we compute the state trajectory that maximizes the probability an expert would have chosen that trajectory *conditioned on reaching the waypoint*. This corresponds to computing the MAP state trajectory with our probabilistic imitation model as our prior and a waypoint likelihood function. The inference task corresponds to a hidden Markov process with two observed boundary conditions: the current position of the car s_0 and the waypoint position s_{N+1}, which we assume the vehicle can reach. All other states $(s_1, \ldots, s_N)$ are hidden. At test time, our method plans a trajectory $s_{1:N} = (s_1, \ldots, s_N)$ under the learned trajectory distribution q by conditioning q on achieving the goal: $s_{1:N}^{\text{plan}} = \arg\max_{s_{1:N}} q(s_{1:N}|s_{N+1}; \theta)$.

[1] https://sites.google.com/view/imitativeforecastingcontrol

One method for approximating an expert distribution with a deep generative model is the reparameterized pushforward policy (R2P2) approach (Rhinehart et al., 2018). R2P2's use of pushforward distributions, employed in normalizing flow models (Rezende & Mohamed, 2015) and RealNVP (Dinh et al., 2016) allows it to efficiently minimize both type I and II errors. It can compute $q(x)$ for arbitrary $x \in \mathcal{X}$, to optimize $KL(p, q)$, which heavily penalizes mode loss, or type II errors. Reducing type I errors can be achieved by minimizing $KL(q, p)$, which penalizes q heavily for generating bad samples, as judged by p. Because the PDF p is not generally known, R2P2 first approximates p with a model $\tilde{p}$ trained to predict a spatial cost map. Then, this cost map is used in $KL(q, \tilde{p})$ to penalize samples from q that land in high cost regions of $\tilde{p}$. The full objective is $KL(p, q) + \beta KL(q, \tilde{p})$. Because of these useful attributes and its relevance to our domain, we adopt this model for the first stage of our method: imitation learning, and develop useful inference tasks based on this model.

2.1 Preliminaries

We first discuss the R2P2 framework before we introduce our inference tasks. R2P2 learns a conditional distribution $q(s_{1:N}|\phi)$, where ϕ is the contextual information. In our application, ϕ consists of LIDAR features and a small window of previous agent positions. $q(s_{1:N}|\phi)$ is induced through an invertible, differentiable warping function (also known as a diffeomorphism): $f(z; \phi) : \mathbb{R}^{T\times 2} \mapsto \mathbb{R}^{T\times 2}$. f warps latent samples from a base distribution $z \sim q_0$ to the output space over $s_{1:N}$, where q_0 is a multivariate gaussian distribution with identity covariance.

The structure of $f(z; \phi)$ is what makes this framework suitable for our purposes: *f embeds the sequential evolution of learned stochastic dynamics*. At each step, a state in the output trajectory is given by $s_t = \mu_t(\psi_t) + \sigma_t(\psi_t) z_t$. As $z_t \sim \mathcal{N}(0, I)$, the conditional distribution of s_t is $q(s_t|\psi_t) = \mathcal{N}(s_t; \mu(\psi_t), \Sigma = \sigma(\psi_t)\sigma(\psi_t)^T)$. The ψ_t represents the information that is causally available at $t-1$, and includes ϕ as well as the partial trajectory: $\psi_t = [s_{1:t-1}, \phi]$. By learning q that assigns high likelihood to expert behaviors, and low likelihood otherwise, we obtain an estimate of the dynamics model that can be used to score the quality of arbitrary trajectories according to how likely they are to be generated by the expert. This property lets us plan human-like trajectories.

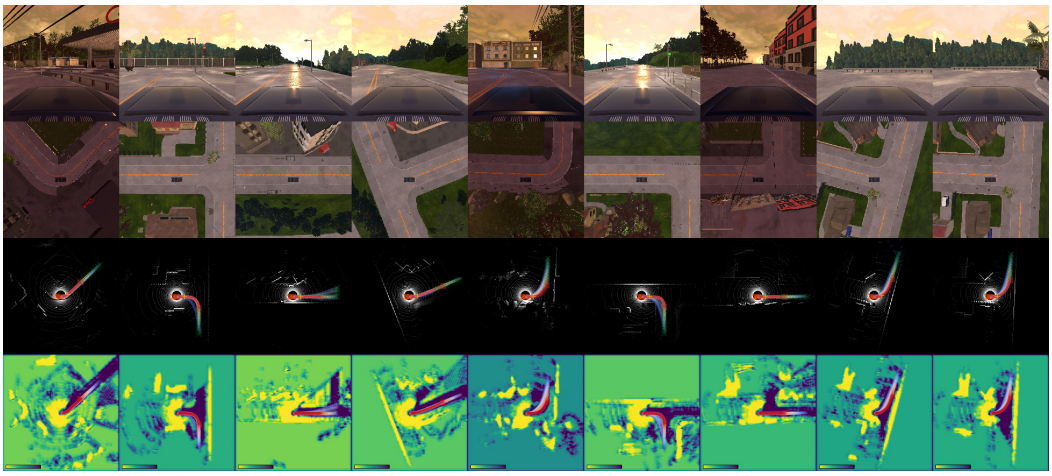


Figure 3: Examples from our approach planning and controlling (Best viewed digitally). *Row 1:* Frontal image as observed by the agent. *Row 2:* Overhead image for visualization purposes. *Row 3:* The LIDAR map overlayed with our method's predicted plans to each waypoint. It is particularly interesting to inspect the behavior of our approach at intersections, as it requires the planner to precisely perceive scene in order to plan a feasible path. *Row 4:* The predicted cost maps for each scene. The planning objective is augmented with trajectory costs, which assists the planning to stay on the road.

2.2 GOAL PLANNING WITH A DIFFERENTIABLE DENSITY

Using the sequence of conditional distributions that comprise q, we can construct the joint distribution over trajectories of arbitrary length. Extending the trajectory horizon corresponds to more steps of stochastic dynamics evolution. The joint distribution over a length-N partial trajectory is given by:

$$q(s_{1:N}) = \prod_{t=1}^{N} q(s_t|\psi_t) = \prod_{t=1}^{N} \mathcal{N}(s_t; \mu(\psi_t), \Sigma(\psi_t)). \tag{1}$$

Now, we consider conditioning the model on some goal g_{N+1}, and optimizing the trajectory $s_{1:N}$ under the likelihood of the learned model q:

$$\begin{aligned}
\max_{s_{1:N}} \mathcal{L}_N^{\text{single}}(s_{1:N}) &= \max_{s_{1:N}} q(s_{1:N}|g_{N+1}) \\
&= \max_{s_{1:N}} \frac{q(g_{N+1}|s_{1:N})q(s_{1:N})}{q(g_{N+1})} \\
&= \max_{s_{1:N}} q(g_{N+1}|s_{1:N})q(s_{1:N}) \\
&= \max_{s_{1:N}} \log q(g_{N+1}|s_{1:N}) + \log q(s_{1:N}) \\
&= \max_{s_{1:N}} \log \mathcal{N}(g_{N+1}; \mu(\psi_{N+1}), \Sigma(\psi_{N+1})) + \sum_{t=1}^{N} \log \mathcal{N}(s_t; \mu(\psi_t), \Sigma(\psi_t)) \\
&= \max_{z_{1:N}} \log \mathcal{N}(g_{N+1}; \mu(\psi_{N+1}), \Sigma(\psi_{N+1})) + \sum_{t=1}^{N} \log \mathcal{N}(s_t; \mu(\psi_t), \Sigma(\psi_t)) \quad (2)
\end{aligned}$$

where the change of optimization variables from states $s_{1:N}$ to latents $z_{1:N}$ follows from the invertibility of f. Planning to the goal g_{N+1} then amounts to optimization of Eq. 2, which we perform with gradients $\nabla_{z_{1:N}} \mathcal{L}_N^{\text{single}}(f(z_{1:N}))$.

2.3 MULTIGOAL PLANNING

In addition to providing a test-time cost function, a user (or program) may wish to be more specific in their desired path to a goal. In particular, by conditioning on two successive goals, a user can effectively communicate their desired final velocity (desired speed and desired vehicle orientation) when arriving at the goal. For instance, when parking, slow speeds and precise vehicle orientation are important. We can implement multigoal planning by extending equation 2 to condition on two final goals:

2.4 TEST-TIME COSTS

As previously discussed, the advantage of conditional imitation learning is that a user or route planning program can communicate *where* they desire the agent to go at a high level, without necessarily knowing the best and safest actions: the planning-as-inference procedure will plan a path that is similar to how an expert would have acted to reach the given goal. If the user desires more control over the plan, our model has the additional flexibility to accept arbitrary user-specified cost maps (or other trajectory metrics) at test time. For example, a user (or program) may have updated knowledge of new hazardous regions at test time which they wish to avoid, such as pot holes. In Figure 2, we demonstrate our model planning around a cost bump corresponding to a pothole. In our model, we leveraged the learned cost map approach of (Rhinehart et al., 2018) to produce cost maps at test time.

$$
\begin{aligned}
\max_{s_{1:N}} \mathcal{L}_N^{\text{double}}(s_{1:N}) &= \max_{s_{1:N}} q(s_{1:N-1}|g_{N+1}, g_N) \\
&= \max_{s_{1:N}} q(g_{N+1}, g_N|s_{1:N-1})q(s_{1:N-1}) \\
&= \max_{s_{1:N}} q(g_{N+1}|g_N, s_{1:N-1})q(g_N|s_{1:N-1})q(s_{1:N-1}) \\
&= \max_{s_{1:N}} \log q(g_{N+1}|g_N, s_{1:N-1}) + \log q(g_N|s_{1:N-1}) + \log q(s_{1:N-1}) \\
&= \max_{s_{1:N}} \log \mathcal{N}(g_{N+1}; \mu(\psi_{N+1}^{g_N}), \Sigma(\psi_{N+1}^{g_n})) + \mathcal{L}_{N-1}^{\text{single}}(s_{1:N-1}), \qquad (3)
\end{aligned}
$$

where $\psi_{N+1}^{g_N} = [s_{1:N-1}, g_N, \phi]$.

3 Related Work

Previous work has explored conditional imitation learning to for autonomous driving. Two model-free approaches were proposed by Codevilla et al. (2017), to map images to safe actions. The first approach uses three network 'heads', each head only trained on an expert's left/straight/right turn maneuvers. At test time, directing the robot corresponds to choosing the network head associated with desired maneuver. Their second method input the goal location into the network, however, this did not perform as well. Whilst model-free conditional imitation learning is effective at directing safe navigation given a discrete set of user directives, our model-based conditional imitation learning has several advantages. First, is flexibility to handle more complex directives post training, perhaps exiting diagonally from an unusual intersection or avoiding hazardous pot holes (Figure 2). Our model is also interpretable since it can visualize all multimodal trajectories it considers in a given situation (Figure 1).

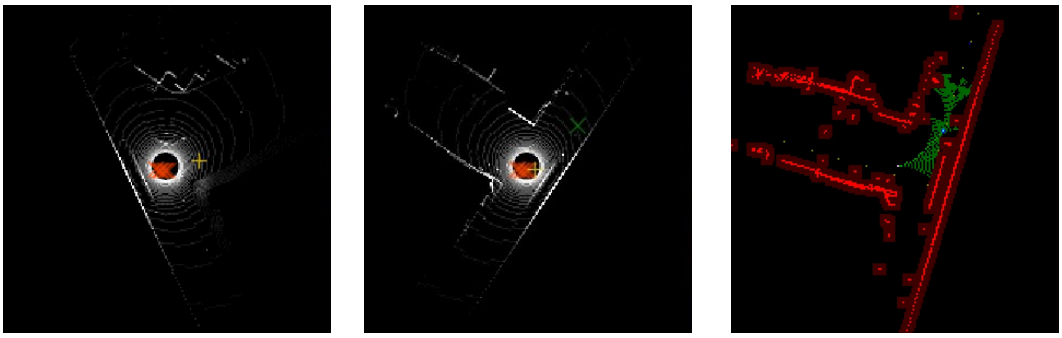
Work by Liang et al. (2018) also makes use of multi-headed model-free conditional imitation learning for initial training to then 'warm start' a DDPG driving algorithm (Lillicrap et al., 2015). Whilst such warm starting speeds up DDPG training, any subsequent DDPG post fine-tuning is inherently trail-and-error based, without guarantees of safety, and may crash during this learning phase. By contrast, our method never executes unlikely transitions w.r.t. expert behavior at training time nor at test time. Whilst our path planning method assumes all waypoints can be reached, we still compute the log likelihood of each planned path, and stop the car if no path reaches a minimum log likelihood, indicating no computed path is safe to execute.

Whilst our work does not consider multiagent environments, several methods predict the behaviour of other vehicles or pedestrians. Typically this involves recurrent neural networks combined with Gaussian density layers or generative models based on some context inputs such as LIDAR, images, or known positions of external agents Lee et al. (2017); Schmerling et al. (2018); Zyner et al. (2018); Gupta et al. (2018); Ma et al. (2017). However, none of these methods can evaluate the likelihood of trajectories or repurpose their model to perform other inference tasks.

4 Experiments

We evaluate our method using the CARLA urban driving simulator (Dosovitskiy et al., 2017). Each episode begins with the vehicle randomly positioned on a road in the `Town01` or `Town02` maps. The task is to drive to a goal location, chosen to be the furthest road location from the vehicle's initial position. We use three layers of spatial abstractions to plan to the goal location, common to autonmous vehicle setups: coarse route planning over a road map, path planning within the observable space, and feedback control to follow the planned path (Paden et al., 2016). First, we compute a route to the goal location using A* given knowledge to the road graph. Second, we set waypoints along the route no closer than 20 meters of the vehicle at any time to direct the vehicle. Finally, we use a p-controller (proportional controller) to compute the vehicle steering value. The p-controller was tuned to steer the vehicle towards a set point (target) 5 meters away along the planned path.

To demonstrate the benefit of our method, we consider four metrics: 1) Success rate in driving to the goal location without any collisions. 2) Proportion of time spent driving in the correct lane.



(a) *Proportional controller* The yellow plus indicates the predicted set point for the controller. The red crosses indicate the past 10 positions of the agent.

(b) *Imitation Learning* baseline. Green cross indicates the provided goal, and the yellow plus indicates the predicted set point for the controller. The red crosses indicate the past 10 positions of the agent.

(c) *Model-based RL baseline*, planning left turn. The green regions indicate the model's predicted reachability, the red regions are post-processed LIDAR used to create an obstacle map.

Figure 4: Baseline methods we compare against.

3) Frequency of crashes into obstacles. 4) Passenger comfort, by comparing the distribution of accelerations (and higher-order terms) between each method.

To contrast the benefits of our method against existing approaches, we compare against several baselines. Since our approach bridges imitation learning and model-based reinforcement learning, we include a purely imitation learning baseline algorithm, and a purely model-based baseline algorithm.

Proportional control (PC): To demonstrate the value *any* learned control method, we test without any learned model, using only a p-controller to follow the high-level waypoints along the route. This corresponds to removing the middle layer of autonomous vehicle decision abstraction. The proportional controller is quite effective when the set point is nearby, but fails badly when the set point is far away, as it attempts to cut corners.

Imitation learning (IL): We designed an imitation-learning based approach to control the vehicle. A common straightforward approach to imitation learning is behavior-cloning: learning to predict the actions taken by a demonstrator Codevilla et al. (2018); Pomerleau (1989); Mahler & Goldberg (2017). Our setting is that of goal-conditioned IL: in order to achieve different behaviors, the imitator is tasked with generating controls after observing a target high-level waypoint. Instead of directly predicting agent controls from the provided scene features and goal, we train a model to predict points from the expert trajectory that are one second ahead along the expert trajectory to the goal, which serves as the set point for the proportional controller. We found this method very effective for generative stable control on straightaways. When the model encounters corners, however, prediction is more difficult, as in order to successfully avoid the curbs, the model must plan implicitly a safe path. We train the model on the same dataset of observed expert trajectories, using a very similar architecture as our primary model.

Model based RL (MB): To compare against a purely model-based reinforcement learning algorithm, we propose a model predictive control baseline. This baseline first builds a forwards dynamics model $f : \{s_{t-3}, s_{t-2}, s_{t-1}, s_t, a_t\} \rightarrow s_{t+1}$ given observed expert data. We use a MLP with two hidden layers, each 100 units. Note that our forwards dynamics model does not imitate the expert preferred actions, but only models what is physically possible. Together with a LIDAR map to locate obstacles, this baseline uses its dynamics model to plan through the free-space to the waypoint whilst avoiding obstacles. We plan forwards over 20 time steps using a breath first search search over CARLA steering angle [-0.3, -0.1, 0., 0.1, 0.3], noting valid steering angles are normalized to [-1,1], with constant throttle at 0.5, noting the valid throttle range is [0,1]. We perform a simple breadth first search expanding each state by the available actions, and only the closest 50 nodes closest to the waypoint. We chose this search since it efficiency reaches the waypoint, but constantly searched around obstacles along the way to avoid getting stuck. The convert the lidar images into and ostacle maps, we simple expanded all obsatcles by the approzimta eraidus of the car, 1.5 meters.

Performance results that compare our methods against baselines according to multiple metrics are includes in Table 1. With the exception of the success rate metric, lower numbers are better. For reference, we additionally include the expert driver we aim to mimic (we use CARLA's inbuilt autopilot as the expert driver). We define success rate as the proportion of episodes where the vehicles navigated across the road map to a goal location on the other side without any collisions. In our experiments we do not include any other drivers or pedestrians, so a collision is w.r.t. a stationary obstacle. Collision impulse are the collision intensities (accumulated during episodes, averaged over episodes) with unit kg·m/s. 'Wrong lane' and 'Off road' percentage of the vehicle invading other lanes or offroad (averaged over time and episodes).

While autonomous vehicle safety metrics are arguably the most important metric, passenger comfort is relevant also. Passenger comfort can be ambiguous to define, so we instead simply record certain driving statistics of each method with which we can compare to the expert driver. In particular, we record the second to sixth derivatives of the position vector with respect to time, respectively termed acceleration, jerk, snap, crackle, and pop. In Table 1 we note the 99th percentile of each statistic given all data collected per path planning method. Generally speaking, lower numbers correspond to a smoother more pleasant driving experience.

The poor performance of the proportional control (PC) baseline indicates that the high-level waypoints do not communicate sufficient information about the correct driving direction, meaning that a local learned policy is indeed required to navigate these environments effectively. Imitation learning (IL) achieves better levels of comfort than model-based RL, but exhibits substantially worse generalization based on the training data, since it does not reason about the sequential structure in the task. Model-based RL (MB) succeeds on most of the trials in the training environment, but exhibits worse generalization. Notably, model-based RL also scores much worse than IL in terms of staying in the right lane and maintaining comfort, which is consistent with our hypothesis: it is able to achieve the desired goals, but does not capture the behaviors in the data. Our method performs the best according to all metrics, exceeding the success rate of model-based RL and the comfort level of imitation learning.

Table 1: Evaluation metrics in the CARLA driving environment: we evaluate different path planning methods based on two CARLA environments: `Town01`, which each method was trained on; an `Town02`: a test environment.

`Town01`	Successes	Collision Impulse	Wrong lane	Off road	Accel	Jerk	Snap	Crackle	Pop
Baselines:									
Proportional Control (PC)	0 / 10	8.92	18.6 %	12.1 %	0.153	0.925	9.19	85.8	785
Imitation Learning (IL)	5 / 10	1.28	0.2 %	0.32 %	0.060	0.313	2.52	17.4	169
Model-Based (MB)	**10 / 10**	**0.00**	9.3 %	0.82 %	0.062	0.353	2.69	26.1	261
Our method	**10 / 10**	**0.00**	**0.0 %**	**0.00 %**	**0.054**	**0.256**	**1.50**	**13.8**	**136**
`Town02`	Successful	Collision Impulse	Wrong lane	Off road	Accel	Jerk	Snap	Crackle	Pop
Baselines:									
Proportional Control (PC)	2 / 10	12.5	5.0 %	4.99 %	0.204	1.040	6.77	59.1	611
Imitation Learning (IL)	2 / 10	8.87	2.2 %	1.03 %	0.319	0.798	3.66	33.3	319
Model Based (MB)	7 / 10	2.56	12.0 %	3.53 %	0.134	0.967	6.06	63.1	575
Our method	**8 / 10**	**0.41**	**0.4 %**	**0.27 %**	**0.054**	**0.613**	**2.64**	**21.4**	**289**

5 Discussion

We proposed a method for model-based control that combines elements of imitation learning and model-based reinforcement learning. Our method estimates a distribution over human behavior from data, and then plans paths that achieve user-specified goals at test time while maintaining high probability under this distribution. We demonstrated several advantages of our algorithm in autonomous driving scenarios. In the context of model-based RL, our method mitigates the distributional shift issue by explicitly optimizing for plans that stay close to the data. This implicit allows our method to enforce basic safety properties: in contrast to model-based RL, which requires negative examples to understand the potential for adverse outcomes (e.g., crashes), our method automatically avoids such outcomes simply because they do not occur in the data. In the context of imitation learning, our method provides substantially more expressivity than the simple directional commands explored in prior conditional imitation learning work, enabling it to achieve arbitrary user-specified goals at test-time.

REFERENCES

J Andrew Bagnell and Jeff G Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 2, pp. 1615–1620. IEEE, 2001.

Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio López, and Vladlen Koltun. End-to-end driving via conditional imitation learning. *arXiv preprint arXiv:1710.02410*, 2017.

Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9. IEEE, 2018.

Marc Deisenroth and Carl E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number CONF, 2018.

Leonid Kuvayev and Richard S. Sutton. Model-based reinforcement learning with an approximate, learned model. In *in Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*, pp. 101–105, 1996.

Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. DESIRE: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 336–345, 2017.

Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. CIRL: Controllable imitative reinforcement learning for vision-based self-driving. *arXiv preprint arXiv:1807.03776*, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Wei-Chiu Ma, De-An Huang, Namhoon Lee, and Kris M Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 4636–4644. IEEE, 2017.

Jeffrey Mahler and Ken Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In *Conference on Robot Learning*, pp. 515–524, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pp. 305–313, 1989.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Nicholas Rhinehart, Kris M. Kitani, and Paul Vernaza. R2P2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 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, 2011.

Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9. IEEE, 2018.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Alex Zyner, Stewart Worrall, and Eduardo Nebot. Naturalistic driver intention and path prediction using recurrent neural networks. *arXiv preprint arXiv:1807.09995*, 2018.