# Deep Imitative Models for Flexible Inference, Planning, and Control

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Imitation learning provides an appealing framework for autonomous control: in many tasks, demonstrations of preferred behavior can be readily obtained from human experts, removing the need for costly and potentially dangerous online data collection in the real world. A disadvantage of imitation learning is its limited flexibility to reach new goals safely at test time. In contrast, classical model-based reinforcement learning (MBRL) offers considerably more flexibility: a model learned from data can be reused at test-time to achieve a wide variety of goals, yet its dynamics model only captures what is possible, not what is preferred, resulting in potentially dangerous behavior outside the distribution of expert behavior. In this paper, we aim to combine these benefits to learn Imitative Models: probabilistic predictive models able to plan expert-like trajectories to achieve arbitrary goals. We find this method substantially outperforms both direct imitation and classical MBRL in a simulated driving task, and can be learned efficiently from a fixed set of expert demonstrations. We also show our model can flexibly incorporate user-supplied costs as test-time, can plan to sequences of goals, and can even perform well with imprecise goals, including goals on the wrong side of the road.

## 1 Introduction

Reinforcement learning (RL) generally requires *online* learning: the agent must collect more data with its latest strategy, use this data to update itself, and repeat. While this is natural in some settings, deploying a partially trained policy on a real-world robot can be dangerous. In these settings the behavior must be learned *offline*, usually with expert demonstrations. How can we incorporate such demonstrations into a flexible robotic system, like an autonomous car? Imitation learning (IL) can learn policies that stay near the expert's distribution, but does not offer sufficient flexibility, and is difficult to integrate with conventional components like planning algorithms. Model-based RL (MBRL) algorithms can learn flexible dynamics models from demonstrations, but drift from the distribution of expert behavior. Our proposed method offers both flexibility and behaves like an expert by *planning through a model-based distribution of expert behavior*.

In MBRL [Kuvayev and Sutton, 1996], any data collection method can be used to train a dynamics model. 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 be achieved by planning through the model. However, in practice, model-based and model-free RL algorithms are vulnerable to distributional drift [Thrun, 1995, Ross and Bagnell, 2010]: when acting according to the learned model or policy, the agent will visit states that are different from those seen during training, and in those it is unlikely 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. Therefore, RL algorithms typically require additional online data collection [Englert et al., 2013, Liang et al., 2018].

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.


Figure 1: We apply our approach to navigation in CARLA [Dosovitskiy et al., 2017]. *Columns 1,2:* Images depicting the current scene. The overhead image depicts a $50\,\mathrm{m}^2$ area. *Column 3:* LIDAR input and goals are provided to our deep imitative trajectory model, and plans to the goals are computed under the model's likelihood objective, and colored according to their ranking under the objective, 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 setpoint for a proportional controller. The LIDAR map is $100\,\mathrm{m}^2$, and each goal is $\geq 20\,\mathrm{m}$ away from the vehicle. *Column 4:* Our model can incorporate arbitrary test-time costs, and use them to adjust its planning objective and plan ranking.

Imitation learning algorithms use expert-provided demonstration data and, despite similar distributional drift shortcomings [Ross et al., 2011], can sometimes learn effective control strategies without any additional online data collection [Zhang et al., 2018]. This makes them simple and practical to deploy in the real world, especially for safety-critical tasks such as autonomous driving. However, standard imitation learning offers comparatively little task flexibility since it only predicts low-level expert behavior. While several works have proposed to augment imitation learning with goal conditioning [Dosovitskiy and Koltun, 2016, Codevilla et al., 2018], these goals must be specified in advance during training, and are typically comparatively simple (e.g., turning left or right).

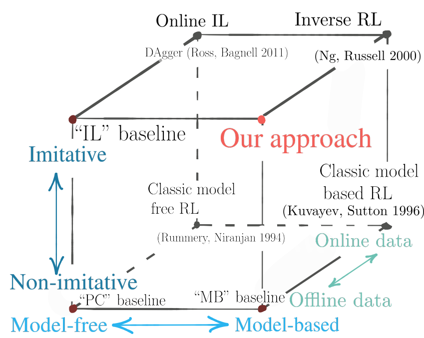


Figure 2: A brief taxonomy of learning-based control methods. In our scenario, we avoid online data collection, specifically from the policy we seek to imitate. We structure our imitation learner with a model to make it flexible to new tasks at test time. We compare against other offline approaches (front face).

In this work, we develop a control algorithm that bridges offline imitation learning and model-based reinforcement learning, whose critical difference to other work is illustrated by the taxonomy cube in Fig. 2. The combination of these methods offers several highly desirable properties. By learning from expert-provided data, we capture the distribution of expert-like behaviors without the need for manually specified cost or reward functions that describe general task constraints. For example, in a driving task, our model automatically ensures the car stays on the road and obeys lane markings (Fig. 1). Conversely, by incorporating model-based goal-driven planning, our model can easily follow user-specified goals at test-time, and can be flexibly repurposed to perform a wide range of tasks at test-time, without any additional training.

Our main contribution is a hybrid offline MBRL and imitation learning method to learn a probabilistic predictive model to plan expert trajectories. We demonstrate our method on a simulated navigated driving task (see Fig. 1), in which it plans to goals produced by a conventional route planner, while obeying the rules of the road and avoiding collisions. In contrast to standard IL, our method produces an interpretable distribution over trajectories and can follow navigational goals without additional training. In contrast to classical MBRL, our method specifically seeks to generate expert-like behaviors without any additional data collection or learning. In a comparative evaluation, we find that our method substantially outperforms both MBRL and model-free imitation learning: it can efficiently learn near-perfect navigation through the static-world CARLA simulator from just 80 episodes of expert driving, equal to 19 hours of driving. We also show that our model can flexibly incorporate and respect costs not seen during training time. Videos of our results are available.[1]

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


Figure 3: Imitative planning to goals subject to a cost at test time. The cost bumps corresponds to simulated “potholes," which the imitative planner is tasked with avoiding. The imitative planner generates and prefers routes that curve around the potholes, stay on the road, and respect intersections. Demonstrations of this behavior were never observed by our model.

## 2 Deep Imitative Models

To understand robot dynamics that are not only possible, but preferred, we first construct a model of expert behaviour. Our method fits a probabilistic model of state trajectories, $q$, to samples of expert trajectories drawn from unknown distribution $p$. A probabilistic model is necessary because expert behavior is multimodal: *e.g.* choosing to turn either left or right at an intersection are both common decisions given identical pasts. Because an expert trajectory drawn from $p$ depends on the expert's observation, we condition $q$ on observations $\phi$ available at prediction time $t = 0$. In our application, $\phi$ includes LIDAR features $M \in \mathbb{R}^{H \times W \times C}$ and a small window of previous agent positions $s_{-\tau:-1} = (s_{-\tau}, \ldots, s_{-1})$: $\phi = [M, s_{-\tau:-1}]$. We define state trajectories as state sequences $s_{1:T}$ whose prior expert probability is a product of conditional distributions: $q_\theta(s_{1:T}|s_0, \phi) = \prod_{t=0}^{T-1} q_\theta(s_{t+1}|s_{0:t}, \phi)$. By learning $q$ that assigns high likelihood to expert trajectories, and low likelihood otherwise, we obtain an estimate of the joint expert-environment 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. At test time, $q(s_{1:T}|s_0, \phi)$ serves as a learned prior over the space of *undirected* expert trajectories. Executing samples from this distribution will 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 application. To direct our agent to each goal, we build a plan from the MAP trajectory: the trajectory maximizing the likelihood of reaching the goal, weighted by the prior probability $q(s_{1:T}|s_0, \phi)$ that an expert would have chosen trajectory $s_{1:T}$. We illustrate our method's application to a navigated driving setting in Fig. 4.

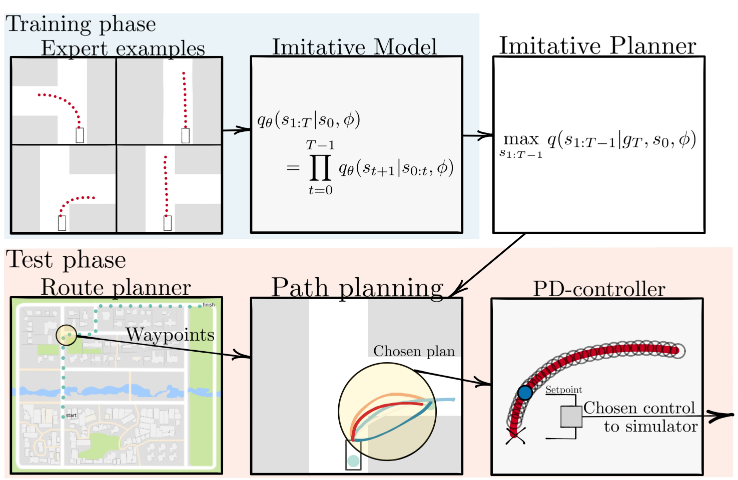


Figure 4: Illustration of our method applied to a navigated driving setting. Our method begins by training an Imitative Model from a dataset of expert examples. After training, the model is repurposed as an *Imitative Planner*. At test time, a route planner provides waypoints to the Imitative Planner, which computes an expert-like path to each goal from the goal and current sensor information $\phi$. The best plan chosen according to the planning objective, and provided to a low-level PID-controller in order to produce steering and throttle actions.

### 2.1 Imitative Planning to Goals

Using the distributions that comprise $q(s_{1:T}|s_0, \phi)$, we construct the joint distribution over trajectories of length $N \leq T-1$. With this, we optimize the state trajectory $s_{1:N}$ conditioned on a sequence of $K$ future goals $g_{N+1:N+K}$, starting after time-step $N$, under an optional trajectory cost $c(s_{1:N}|s_0, \phi)$.

$$\begin{aligned}\max_{s_{1:N}} \mathcal{L}_N &= \max_{s_{1:N}} q(s_{1:N}|g_{N+1:N+K}, s_0, \phi)c(s_{1:N}|s_0, \phi), \\ &= \max_{s_{1:N}} q(s_{1:N}|s_0, \phi)q(g_{N+1:N+K}|s_{0:N}, \phi)c(s_{1:N}|s_0, \phi) \\ &= \max_{s_{1:N}} \log \Big( q(s_{1:N}|s_0, \phi)q(g_{N+1}|s_{0:N})c(s_{1:N}|s_0, \phi) \prod_{t=N+1}^{N+K-1} q(g_{t+1}|g_{N+1:t}, s_{0:N}) \Big) \quad (1)\end{aligned}$$

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 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 costs $c$ at test time. For example, a user (or program) may have updated knowledge of new hazardous regions such as potholes (Fig. 3) or additional cost map.

### 2.2 Model implementation

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 and Mohamed, 2015] and RealNVP [Dinh et al., 2016] allows it to efficiently minimize both type I and II-style errors [Neyman and Pearson, 1933]. 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. Here, $p$ is the sampleable, but unknown, distribution of expert behavior. 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 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 learning procedure to learn $q$, and we also can use $\tilde{p}$ in the final planning objective.

In R2P2, $q(s_{1:T}|\phi)$ is induced through an invertible, differentiable warping function: $f(z; \phi) : \mathbb{R}^{2T} \mapsto \mathbb{R}^{2T}$. $f$ warps samples from a base distribution $z \sim q_0$ to the output space over $s_{1:T}$, where $q_0$ is $\mathcal{N}(0, I_{2T \times 2T})$. The structure of $f(z; \phi)$ makes this framework suitable for our purposes: $f$ embeds the evolution of learned discrete-time stochastic dynamics. Each state in the output trajectory is given by $s_t = \mu_t(s_{1:t-1}, \phi) + \sigma_t(s_{1:t-1}, \phi)z_t = s_{t-1} + (s_{t-1} - s_{t-2}) + m_t(s_{1:t-1}, \phi) + \sigma_t(s_{1:t-1}, \phi)z_t$. The $\mu_t \in \mathbb{R}^2$ and $\sigma_t \in \mathbb{R}^{2\times 2}$ are computed by expressive, nonlinear neural networks. As $z_t \sim \mathcal{N}(0, I_{2\times 2})$, the conditional distribution of $s_t$ is $q(s_t|s_{1:t-1}, \phi) = \mathcal{N}(s_t; \mu(s_{1:t-1}, \phi), \Sigma = \sigma(s_{1:t-1}, \phi)\sigma(s_{1:t-1}, \phi)^T)$. As each one-step distribution is parameterized by the prediction of neural networks that observe previous states (and high-bandwidth LIDAR input), the resulting trajectory distribution is often complex and multimodal. We modified the "RNN" method described in R2P2, used $M = \mathbb{R}^{200\times 200\times 2}$, with $M_{ij}$ representing a 2-bin histogram of points below and above the ground in $0.5\,\mathrm{m}^2$ cells. We used length $T = 40$ trajectories at 5Hz, corresponding to 8 seconds of prediction or planning, and used $\tau = 19$ (2 seconds of past positions $s_{-19:0}$).

## 3 Related Work

Previous work has explored conditional IL for autonomous driving. Two model-free approaches were proposed by Codevilla et al. [2018], to map images to actions. The first uses three network "heads", each head only trained on an expert's left/straight/right turn maneuvers. The robot is directed by a route planner that chooses the desired head. Their second method input the goal location into the network, however, this did not perform as well. While model-free conditional IL can be effective given a discrete set of user directives, our model-based conditional IL has several advantages. Our model has flexibility to handle more complex directives post training, *e.g.* avoiding hazardous potholes (Fig. 3) or other costs, the ability to rank plans and goals by its objective, and interpretability: it can generate entire planned and unplanned (undirected) trajectories. Work by Liang et al. [2018] also uses multi-headed model-free conditional imitation learning to "warm start" a DDPG driving algorithm [Lillicrap et al., 2015]. While warm starting hastens DDPG training, any subsequent DDPG post fine-tuning is inherently trial-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. While our target setting is offline data collection, online imitation learning is an active area of research in the case of hybrid IL-RL [Ross and Bagnell, 2014, Sun et al., 2018] and “safe" IL [Sun et al., 2017, Menda et al., 2017, Zhang and Cho, 2017]. Other methods include inverse reinforcement learning to fit a probabilistic reward model to human demonstrations using the principle of maximum entropy [Ziebart et al., 2008, Sadigh et al., 2016].

# 4 Experiments

We evaluate our method using the CARLA urban driving simulator [Dosovitskiy et al., 2017]. Each test 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. As shown in Fig. 4, we use three layers of spatial abstractions to plan to the goal location, common to model-based (not end-to-end) autonomous 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, Schwarting et al., 2018]. 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 $20\,\mathrm{m}$ of the vehicle at any time to direct the vehicle. Finally, we use a PD-controller (proportional controller) to compute the vehicle steering value. The PD-controller was tuned to steer the vehicle towards a setpoint (target) 5 meters away along the planned path.

We consider four metrics for this task: 1) Success rate in driving to the goal location without any collisions. 2) Proportion of time spent driving in the correct lane. 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 model-free IL and MBRL, we include an IL baseline algorithm, and a MBRL baseline algorithm.

**Proportional control (PC):** The PC baseline uses the PD-controller to follow the high-level waypoints along the route. This corresponds to removing the middle layer of autonomous vehicle decision abstraction, which serves as a baseline for the other methods. The proportional controller is quite effective when the setpoint is nearby, but fails badly when the setpoint is far away (*i.e.* at $20\,\mathrm{m}$).

**Imitation learning (IL):** We designed an IL baseline to control the vehicle. 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, as well as the same $\phi$ observed by our algorithm. Instead of directly predicting agent controls from the provided scene features and goal, we train a model to predict the setpoint for the PD-controller. The model is trained with with the same expert dataset, and predicts setpoints one second in the future. This model must implicitly plan a safe path. We used a network architecture nearly identical to our approach's.

**Model based RL (MB):** To compare against a purely model-based reinforcement learning algorithm, we propose a model-predictive control baseline. This baseline first learns a forwards dynamics model $f : s_{t-3:t}, a_t) \rightarrow s_{t+1}$ given observed expert data. We use an MLP with two hidden layers, each 100 units. Together with a LIDAR map to locate obstacles, this baseline uses its dynamics model to plan through the free-space to the waypoint while avoiding obstacles. We plan forwards over 20 time steps using a breadth-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 $[0, 1]$.

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. 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 (in $\mathrm{N} \cdot \mathrm{s}$) is the average cumulative collision intensities over episodes. “Wrong lane" and “Off road" percentage of the vehicle invading other lanes or offroad (averaged over time and episodes). Passenger comfort is also relevant, but can be ambiguous to define, so we simply 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 driving experience.

Table 1: We evaluate different path planning methods based on two CARLA environments: `Town01`, which each method was trained on; and `Town02`: a test environment.

| `Town01` | Successes | Collision Impulse | Wrong lane | Off road | Accel | Jerk | Snap | Crackle | Pop |
|---|---|---|---|---|---|---|---|---|---|
| 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 RL (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` | Successes | Collision Impulse | Wrong lane | Off road | Accel | Jerk | Snap | Crackle | Pop |
| 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 RL (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** |

Table 2: Incorporating a pothole cost enables our method to avoid potholes

| Approach | Successes | Pothole hits | Wrong lane | Off road |
|---|---|---|---|---|
| Our method without pothole cost, `Town01` | 9 / 10 | 177/230 | 0.06% | 0.00% |
| Our method with pothole cost, `Town01` | 9 / 10 | **10/230** | 1.53% | 0.06% |
| Our method without pothole cost, `Town02` | 8 / 10 | 82/154 | 1.03% | 0.30% |
| Our method with pothole cost, `Town02` | 7 / 10 | **35/154** | 1.53% | 0.11% |

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, MB 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, far exceeding the success and comfort metrics of IL, and far exceeding the lane-obeyance and comfort metrics of MB.

**Avoiding novel obstacles at test-time:** To further illustrate the capability of our method to incorporate test-time costs, we designed a pothole collision experiment. We simulated $2\mathrm{m}$-wide potholes in the environment offset from each waypoint with noise distributed $\mathcal{N}(\mu = [-15\mathrm{m}, 2\mathrm{m}], \Sigma = \mathrm{diag}([1, 0.01]))$. We ran our method that incorporates a test-time cost map of the simulated potholes, and compared to our method that did not incorporate the cost map (and thus had no incentive to avoid potholes). In addition to the other metrics, we recorded the number of collisions with potholes. In Table 2, we see that our method with cost incorporated achieved nearly perfect pothole avoidance, while still avoiding collisions with the environment. To do so, it drove closer to the centerline, and occasionally dipped into the opposite lane. Our model internalized obstacle avoidance by staying on the road, and demonstrated its flexibility to obstacles not observed during training.

## 5 Discussion

We proposed a method for learning behavior 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 to achieve user-specified goals at test time while maintaining high probability under this distribution. We demonstrated several advantages and applications of our algorithm in autonomous driving scenarios. In the context of model-based RL, our method mitigates the distributional drift issue by explicitly optimizing for plans that stay close to the data. This implicitly 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


F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

A. Dosovitskiy and V. Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

P. Englert, A. Paraschos, M. P. Deisenroth, and J. Peters. Probabilistic model-based imitation learning. *Adaptive Behavior*, 21(5):388–403, 2013.

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

X. Liang, T. Wang, L. Yang, and E. Xing. CIRL: Controllable imitative reinforcement learning for vision-based self-driving. *arXiv preprint arXiv:1807.03776*, 2018.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer. Dropoutdagger: A bayesian approach to safe imitation learning. *arXiv preprint arXiv:1709.06166*, 2017.

J. Neyman and E. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London*, A 231:289–337, 1933.

B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

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

S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010.

S. Ross and J. A. Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

S. Ross, G. Gordon, and D. 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*, pages 627–635, 2011.

D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, 2016.

W. Schwarting, J. Alonso-Mora, and D. Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, 2018.

L. Sun, C. Peng, W. Zhan, and M. Tomizuka. A fast integrated planning and control framework for autonomous driving via imitation learning. *arXiv preprint arXiv:1707.02515*, 2017.

W. Sun, J. A. Bagnell, and B. Boots. Truncated horizon policy search: Combining reinforcement learning and imitation learning. In *Proceedings of the Sixth International Conference on Learning Representations (ICLR)*, 2018.

S. Thrun. Learning to play the game of chess. In *Advances in neural information processing systems*, pages 1069–1076, 1995.

J. Zhang and K. Cho. Query-efficient imitation learning for end-to-end simulated driving. In *AAAI*, pages 2891–2897, 2017.

T. Zhang, Z. McCarthy, O. Jowl, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.