LATENT HIERARCHICAL IMITATION LEARNING FOR STOCHASTIC ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many applications of imitation learning require the agent to avoid mode collapse and mirror the full distribution of observed behaviours. Existing methods that address this *distributional realism* typically rely on hierarchical policies conditioned on sampled *types* that model agent-internal features like persona, goal, or strategy. However, these methods are often inappropriate for stochastic environments, where internal and external factors of influence on the observed agent trajectories have to be disentangled, and only internal factors should be encoded in the agent type to be robust to changing environment conditions. We formalize this challenge as distribution shifts in the *marginal* and *conditional* distributions of agent types under environmental stochasticity, in addition to the familiar covariate shift in state visitations. We propose *Robust Type Conditioning* (`RTC`), which eliminates these shifts with adversarial training under randomly sampled types. Experiments on two domains, including the large-scale *Waymo Open Motion Dataset*, show improved distributional realism while maintaining or improving task performance compared to state-of-the-art baselines.

1 INTRODUCTION

Learning to imitate behaviour is crucial when reward design is infeasible (Amodei et al., 2016; Hadfield-Menell et al., 2017; Fu et al., 2018; Everitt et al., 2021), for overcoming hard exploration problems (Rajeswaran et al., 2017; Zhu et al., 2018), and for realistic modelling of dynamical systems with multiple interacting agents (Farmer and Foley, 2009). Such systems, including games, driving simulations, and agent-based economic models, often have known state transition functions, but require accurate agents to be realistic. For example, for driving simulations, which are crucial for accelerating the development of autonomous vehicles (Suo et al., 2021; Igl et al., 2022), faithful reactions of all road users are paramount. Furthermore, it is not enough to mimic a single mode in the data; instead, agents must reproduce the full distribution of behaviours to avoid sim2real gaps in modelled systems (Grover et al., 2018; Liang et al., 2020), underexplored solutions in complex tasks (Vinyals et al., 2019) and suboptimal policies in games requiring mixed strategies (Nash Jr, 1950).

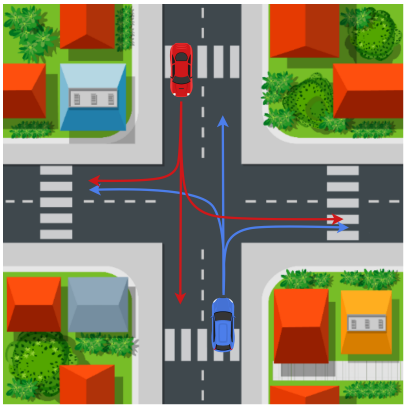


Figure 1: Realised trajectories depend on the *internal* agent *type*, expressing goals and preferences, and *external* factors such as the behaviour of other agents.

Current imitation learning (IL) methods fall short of achieving such *distributional realism* by matching all modes in the data. The required stochastic policy cannot be recovered from a fixed reward function and adversarial methods, while aiming to match the distribution in principle, exhibit mode collapse in practice. Furthermore, progress on distributional realism is hindered by a lack of suitable IL benchmarks, with most relying on unimodal data

and only evaluating per-episode task performance (as measured by rewards), but not mode coverage. By contrast, many applications require distributional realism in addition to good task performance. For example, accurately evaluating the safety of autonomous vehicles in simulation relies on distributionally realistic agents. Consequently, our goal is to improve distributional realism while maintaining strong task performance.

To mitigate mode collapse in complex environments, previous work uses hierarchical policies in an auto-encoder framework (Wang et al., 2017; Suo et al., 2021; Igl et al., 2022). During training, an encoder infers latent variables from observed trajectories and the agent, conditioned on those latent variables, strives to imitate the original trajectory. At test time, a prior distribution proposes distributionally realistic latent values, without requiring access to privileged future information. We refer to this latent vector as an agent's inferred *type* since it expresses intrinsic characteristics of the agent that yield the multimodal behaviour. Depending on the environment, the *type* could, for example, represent the agent's persona, belief, goal, or strategy.

However, these hierarchical methods rely on either manually designed type representations (Igl et al., 2022) or the strong assumption that *all* stochasticity in the environment can be controlled by the agent (Wang et al., 2017; Suo et al., 2021). Unfortunately, this assumption is violated in most realistic scenarios. For example, in the case of driving simulations (fig. 1), trajectories depend not only on the agent's type, expressing its driving style and intent, but also on external factors such as the behaviour of other road users. Crucially, despite being inferred from future trajectories during training, agent types must be independent of these external factors to avoid leaking information about future events outside the agent's control, which in turn can impair generalization at test time under changed, and ex-ante unknown, environmental conditions. In other words, the challenge in learning hierarchical policies using IL in stochastic environments is to disentangle the internal and external factors of influence on the trajectories and only encode the former into the type.

Consider the example of an expert approaching an intersection at the same time as another car. The expert passes if the other car brakes and yields to it otherwise. To reconstruct the scene with ease, a naively trained latent model would not only encode the agent's intended direction (an *internal* decision) but also whether to yield, which depends on the other car (an *external* factor). This is catastrophic at test time when the latent, and hence the yielding decision, is sampled independently of the other car's behaviour. In contrast, if only the expert's intent were encoded in the latent, the policy would learn to react appropriately to external factors.

In this paper, we identify these subtle challenges arising under stochastic environments and formulate them as two new forms of distribution shift for hierarchical policies. Unlike the familiar covariate shift in the state distribution (Ross et al., 2011), these *marginal* and *conditional type shifts* occur in the distribution of the inferred latent type. They greatly reduce performance by yielding causally confused agents that rely on the latent type for information about external factors, instead of inferring them from the latest environment observation. We propose *Robust Type Conditioning* (`RTC`) to eliminate these distribution shifts and avoid causally confused agents through a coupled adversarial training objective under randomly sampled types. We do not require access to an expert, counterfactuals, or manually specified type labels for trajectories.

Experimentally, we show the need for improved distributional realism due to mode collapse in state-of-the-art imitation learning techniques such as GAIL (Ho and Ermon, 2016). Furthermore, we show that naively trained hierarchical models with inferred types improve distributional realism, but exhibit poor task performance in stochastic environments. By contrast, `RTC` can maintain good task performance in stochastic environments while improving distributional realism and mode coverage. We evaluate `RTC` on the illustrative *Double Goal Problem* as well as the large scale *Waymo Open Motion Dataset* (Ettinger et al., 2021) of real driving behaviour.

2 BACKGROUND

We are given a dataset $\mathcal{D} = \{\tau_i\}_{i=1}^N$ of N trajectories $\tau_i = \boldsymbol{s}_0^{(i)}, \boldsymbol{a}_0^{(i)}, \dots \boldsymbol{s}_T^{(i)}$, drawn from $p(\tau)$ of one or more experts interacting with a stochastic environment $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ where $\boldsymbol{s}_t \in \mathcal{S}$ are states and $\boldsymbol{a}_t \in \mathcal{A}$ are actions. Our goal is to learn a policy $\boldsymbol{\pi_\theta}(\boldsymbol{a}_t|\boldsymbol{s}_t)$ to match $p(\tau)$ when replacing the unknown expert and generat-

ing rollouts $\hat{\tau} \sim p(\hat{\tau}) = p(\boldsymbol{s}_0) \prod_{t=0}^{T-1} \boldsymbol{\pi}_{\boldsymbol{\theta}}(\hat{\boldsymbol{a}}_t|\hat{\boldsymbol{s}}_t) p(\hat{\boldsymbol{s}}_{t+1}|\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{a}}_t)$ from the inital states $\boldsymbol{s}_0 \sim p(\boldsymbol{s}_0)$. We simplify notation and write $\hat{\tau} \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}(\hat{\tau})$ and $\tau \sim \mathcal{D}(\tau)$ to indicate rollouts generated by the policy or drawn from the data respectively. Expectations $\mathbb{E}_{\tau \sim \mathcal{D}}$ and $\mathbb{E}_{\hat{\tau} \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}}$ are taken over all pairs $(\boldsymbol{s}_t, \boldsymbol{a}_t) \in \tau$ and $(\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{a}}_t) \in \hat{\tau}$.

Previous work (e.g., Ross et al., 2011; Ho and Ermon, 2016) shows that a core challenge of learning from demonstration is reducing or eliminating the covariate shift in the state-visitation frequencies $p(\boldsymbol{s})$ caused by accumulating errors when using $\boldsymbol{\pi}_{\boldsymbol{\theta}}$. Unfortunately, *Behavioural Cloning* (BC), a simple supervised training objective optimising $\max_{\boldsymbol{\theta}} \mathbb{E}_{\tau \sim \mathcal{D}} \left[\log \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t) \right]$ is not robust to it. To overcome covariate shift, generative adversarial imitation learning (GAIL) (Ho and Ermon, 2016) optimises $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ to fool a learned discriminator $D_{\boldsymbol{\phi}}(\hat{\boldsymbol{a}}_t, \hat{\boldsymbol{s}}_t)$ that is trained to distinguish between trajectories in $\mathcal{D}$ and those generated by $\boldsymbol{\pi}_{\boldsymbol{\theta}}$:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathbb{E}_{\hat{\tau} \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}} \Big[\log(D_{\boldsymbol{\phi}}(\hat{\boldsymbol{a}}_t, \hat{\boldsymbol{s}}_t)) \Big] + \mathbb{E}_{\tau \sim \mathcal{D}} \Big[\log(1 - D_{\boldsymbol{\phi}}(\boldsymbol{a}_t, \boldsymbol{s}_t)) \Big]. \tag{1}$$

The policy can be optimised using reinforcement learning, by treating the log-discriminator scores as costs, $r_t = -\log D_{\boldsymbol{\phi}}(\hat{\boldsymbol{a}}_t, \hat{\boldsymbol{s}}_t)$. Alternatively, if the policy can be reparameterized (Kingma and Welling, 2013) and the environment is differentiable, the sum of log discriminator scores can be optimised directly without relying on high-variance score function estimators by backpropagating through the transition dynamics, $\mathcal{L}_{\text{adv}}(\hat{\tau}) = \mathbb{E}_{\hat{\tau} \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}} \left[\sum_t - \log D_{\boldsymbol{\phi}}(\hat{\boldsymbol{a}}_t, \hat{\boldsymbol{s}}_t) \right]$. We refer to this as *Model-based GAIL* (MGAIL), though in contrast to Baram et al. (2016), we assume a known differentiable environment instead of a learned model.

In this work, we are concerned with multimodal distributions $p(\tau)$ and how mode collapse can be avoided when learning $\boldsymbol{\pi}_{\boldsymbol{\theta}}$. To this end, we assume the dataset is sampled from $p(\tau) = p(\boldsymbol{s}_0) \int p(\boldsymbol{g}) \prod_{t=0}^{T} p(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{g}) p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) d\boldsymbol{g}$, where $\boldsymbol{g}$ is the agent *type*, expressing agent characteristics such as persona, goal, or, strategy. Learned agents matching $p(\tau)$, i.e., with $p(\hat{\tau}) \approx p(\tau)$, are *distributionally realistic*, whereas *realism* describes single trajectories when $\hat{\tau}$ lies in the support of τ, i.e. $p_\tau(\hat{\tau}) > 0$. As we show in section 6, current non-hierarchical adversarial methods (Ho and Ermon, 2016) exhibit mode collapse and are not distributionally realistic.

To combat mode collapse, hierarchical methods (e.g., Wang et al., 2017; Lynch et al., 2020; Suo et al., 2021; Igl et al., 2022) often rely on an encoder to infer latent agent types $\hat{\boldsymbol{g}}_e$ from trajectories during training, $\hat{\boldsymbol{g}}_{\boldsymbol{e}} \sim \boldsymbol{e}_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_e|\tau)$, and optimise the control policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\hat{\boldsymbol{a}}_t|\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{g}}_e)$ to generate trajectories $\hat{\tau}_e$ similar to τ: $\hat{\tau}_e \sim p(\hat{\tau}_e|\hat{\boldsymbol{g}}_e) = p(\boldsymbol{s}_0) \prod_{t=0}^{T-1} \boldsymbol{\pi}_{\boldsymbol{\theta}}(\hat{\boldsymbol{a}}_t|\hat{\boldsymbol{s}}_t, \hat{\boldsymbol{g}}_e) p(\hat{\boldsymbol{s}}_{t+1}|\hat{\boldsymbol{a}}_t, \hat{\boldsymbol{s}}_t)$, with $\hat{\boldsymbol{g}}_e \sim \boldsymbol{e}_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_e|\tau)$. If ground truth trajectories are not accessible during testing, a prior $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)$ can be used to sample distributionally realistic types $\hat{\boldsymbol{g}}_p$. Existing methods use, for example, open loop training (Wang et al., 2017), factored latent types $\hat{\boldsymbol{g}}_e$ (Suo et al., 2021), manually specified encoders (Igl et al., 2022), or combine state-based goals with latent types (Lynch et al., 2020). We indicate by subscript $\hat{\boldsymbol{g}}_p$ or $\hat{\boldsymbol{g}}_e$ whether the inferred type and trajectory are drawn from the prior distribution $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)$ or encoder $\boldsymbol{e}_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_e|\tau)$. Subscripts are omitted for states and actions to simplify notation. Inferred types and predicted trajectories without subscripts indicate that either sampling distribution could be used.

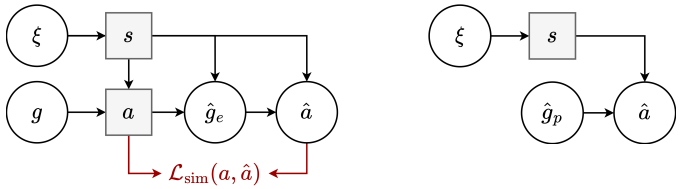
3 Marginal and Conditional Type Shifts

Hierarchical policies are a powerful method for improving distributional realism in imitation learning. However, as we discuss in this section, existing methods either require inflexible, manually designed encoders or require deterministic environments when learning encoders for latent type representations. This assumption is often violated in practice, e.g., in multi-agent settings with stochastic agents, leading to reduced performance. We highlight this restriction by describing how hierarchical policies in stochastic environments suffer from two additional forms of distribution shift. In contrast to the familiar shift in the state visitation distribution $p(\boldsymbol{s})$ (Ross et al., 2011), these shifts are in the marginal and conditional type distributions.

Model The simplified model in fig. 2 has two sources of randomness in the data $\mathcal{D}$: the future environmental noise $\boldsymbol{\xi}$, for example the random actions of other agents, and the multimodal type $\boldsymbol{g}$ of the expert we are

mimicking. $\boldsymbol{\xi}$ constitutes *external* factors while $\boldsymbol{g}$ encodes agent-*internal* decisions influencing the trajectory. Here, the state $\boldsymbol{s}$ is a deterministic function of only $\boldsymbol{\xi}$ as we disregard cross-temporal dependencies. The expert action $\boldsymbol{a}$ depends on type $\boldsymbol{g}$ and state $\boldsymbol{s}$ and we abridge $\tau = (\boldsymbol{s}, \boldsymbol{a})$. To accurately express the multimodality in the expert policy $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{g})$ introduced by the latent type $\boldsymbol{g}$, we learn a hierarchical policy $\boldsymbol{\pi_\theta}(\hat{\boldsymbol{a}}|\boldsymbol{s}, \hat{\boldsymbol{g}})$. During training, when real trajectories τ are available, the inferred type $\hat{\boldsymbol{g}}_e$ is drawn from the encoder $\boldsymbol{e_\theta}(\hat{\boldsymbol{g}}_e|\tau)$. During testing, without access to τ, a prior $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)$ is used. Actions are drawn from the control policy $\boldsymbol{\pi_\theta}(\hat{\boldsymbol{a}}|\boldsymbol{s}, \hat{\boldsymbol{g}})$ and optimisation is performed to minimise a reconstruction loss between ground truth and predicted actions, $\mathcal{L}_{\text{rec}}(\boldsymbol{a}, \hat{\boldsymbol{a}})$. Crucially, for this class of architectures, a distribution shift can occur when sampling the inferred type from the prior $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)$ instead of the encoder $\boldsymbol{e_\theta}(\hat{\boldsymbol{g}}_e|\tau)$. This can come in two forms.

Marginal Type Shift Firstly, the *marginal type shift* is a mismatch between the marginal type distribution under the encoder, i.e., $p(\hat{\boldsymbol{g}}_e) = \mathbb{E}_{p(\tau)}\left[\boldsymbol{e_\theta}(\hat{\boldsymbol{g}}_e|\tau)\right]$ and the prior $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)$, which is trained to minimise $\mathrm{KL}[p(\hat{\boldsymbol{g}}_e) \| p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)]$. Because the prior is mode covering, it might sample inferred types not seen during training. In the context of VAEs (Kingma and Welling, 2013), this is also known as the *prior hole problem* (Rezende and Viola, 2018) and is caused by limited expressiveness of the parametric prior.



(a) Encoder: $\hat{\boldsymbol{g}}_e \sim \boldsymbol{e_\theta}(\hat{\boldsymbol{g}}_e|\tau)$ **(b)** Prior: $\hat{\boldsymbol{g}}_p \sim p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)$.

Figure 2: Simplified, non-temporal setup with environmental noise $p(\boldsymbol{\xi})$ and multi-modality induced by the unobserved agent type $p(\boldsymbol{g})$. We denote $\tau = (\boldsymbol{s}, \boldsymbol{a})$. The inferred type $\hat{\boldsymbol{g}}$ is sampled from $\boldsymbol{e_\theta}(\hat{\boldsymbol{g}}_e|\tau)$ during training (left) and $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)$ otherwise (right). The control policy is $\boldsymbol{\pi_\theta}(\hat{\boldsymbol{a}}|\boldsymbol{s}, \hat{\boldsymbol{g}})$. Circles are random variables and squares deterministic functions. The loss $\mathcal{L}(\boldsymbol{a}, \hat{\boldsymbol{a}})$ penalises differences between $\boldsymbol{a}$ and $\hat{\boldsymbol{a}}$.

Conditional Type Shift The second distribution shift is specific to imitation learning in stochastic environments and has not been, to our knowledge, previously formalised. This *conditional type shift* occurs if the inferred type $\hat{\boldsymbol{g}}_e \sim \boldsymbol{e_\theta}(\hat{\boldsymbol{g}}_e|\tau)$ is not independent of the environmental noise $\boldsymbol{\xi}$ that brought about τ. This induces a shift from the training distribution $p(\hat{\boldsymbol{g}}_e|\boldsymbol{\xi}) = \mathbb{E}_{p(\tau|\boldsymbol{\xi})}\left[\boldsymbol{e_\theta}(\hat{\boldsymbol{g}}_e|\tau)\right]$, which depends on $\boldsymbol{\xi}$, to the prior distribution $p(\hat{\boldsymbol{g}}_p|\boldsymbol{\xi}) = p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)$, which does not. Intuitively, if the encoder captures information about $\boldsymbol{\xi}$ in $\hat{\boldsymbol{g}}_e$ during training when the future is known, then during testing the randomly sampled types $\hat{\boldsymbol{g}}_p$ will not match the now unknown future generated by $\boldsymbol{\xi}$, creating a distribution shift in $p(\hat{\boldsymbol{g}}|\boldsymbol{\xi})$ and hence $p(\hat{\boldsymbol{g}}|\boldsymbol{s})$. Because the control policy $\boldsymbol{\pi_\theta}(\hat{\boldsymbol{a}}|\boldsymbol{s}, \hat{\boldsymbol{g}})$ conditions on *pairs* $(\boldsymbol{s}, \hat{\boldsymbol{g}})$, changes in the conditional distribution $p(\hat{\boldsymbol{g}}|\boldsymbol{s})$ induce a covariate shift in its inputs.

Causal Confusion Training the policy $\boldsymbol{\pi_\theta}(\hat{\boldsymbol{a}}|\boldsymbol{s}, \hat{\boldsymbol{g}})$ under types $\hat{\boldsymbol{g}}_e$ containing information about $\boldsymbol{\xi}$ results in causally confused policies: rather than learning the correct causal dependency of $\hat{\boldsymbol{a}}$ on $\boldsymbol{\xi}$ through the observation $\boldsymbol{s}$, the policy wrongly relies on $\hat{\boldsymbol{g}}_e$, thereby not generalising to new state-type pairs $(\boldsymbol{s}, \hat{\boldsymbol{g}}_p)$ during testing when types $\hat{\boldsymbol{g}}_p$ are sampled independently of $\boldsymbol{\xi}$ from the prior. As an extreme case, $\hat{\boldsymbol{g}}_e$ could encode the exact action $\boldsymbol{a}$ that the policy would learn to decode to minimise the training loss $\mathcal{L}(\boldsymbol{a}, \hat{\boldsymbol{a}}) = 0$, while entirely ignoring observations $\boldsymbol{s}$. This policy would fail under randomly drawn types when appropriate reactions to $\boldsymbol{s}$ are required. If, however, the encoder only captured information about the true type $\boldsymbol{g}$, the policy would learn to correctly infer appropriate actions from $(\boldsymbol{s}, \hat{\boldsymbol{g}})$ and hence generalise during testing.

Recall the pass/yield interaction described in the introduction. In this situation, $\boldsymbol{g}$ expresses the expert's intended direction and $\boldsymbol{\xi}$ the other car's behaviour. If the inferred latent $\hat{\boldsymbol{g}}_e$ prescribes whether to yield, it depends on $\boldsymbol{\xi}$ and hence renders some combinations of $(\boldsymbol{\xi}, \hat{\boldsymbol{g}}_e)$ unseen during training. But these combinations will occur during testing when $\hat{\boldsymbol{g}}_p$ is sampled independently of $\boldsymbol{\xi}$, resulting in out-of-distribution samples and distributional shift. For example, neither agent yielding results in a collision. Furthermore, because during training, $\hat{\boldsymbol{g}}_e$ correctly prescribes the required behaviour, the policy learns to rely on $\hat{\boldsymbol{g}}_e$ instead of inferring the correct information about $\boldsymbol{\xi}$ directly from observations of the other car.

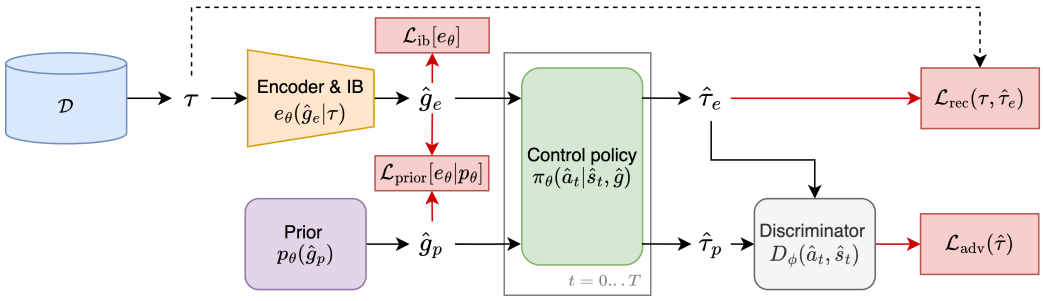


Figure 3: *Robust Type Conditioning* (RTC): The control policy $\boldsymbol{\pi_\theta}(\hat{\boldsymbol{a}}_t|\hat{\boldsymbol{s}}_t,\hat{\boldsymbol{g}})$ is trained under inferred types $\hat{\boldsymbol{g}}$ sampled from both the encoder $\boldsymbol{e_\theta}(\hat{\boldsymbol{g}}_e|\tau)$ and the prior $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)$. The reconstruction loss $\mathcal{L}_{\text{rec}}(\tau,\hat{\tau}_e)$ avoids mode collapse. The adversarial loss $\mathcal{L}_{\text{adv}}(\hat{\tau}_p)$ under prior types prevents causally confused policies and ensures good task performance. $\mathcal{L}_{\text{prior}}$ optimises the prior to sample distributionally realistic types and the information bottleneck loss $\mathcal{L}_{\text{ib}}$ reduces covariate shift.

To summarize, learning hierarchical policies in stochastic environments does not generalise well due to marginal and conditional shifts in the inferred latent type. This translates to reduced task performance through causally confused policies that rely on the inferred type not only to determine the behavioural mode but also for information about the environment that cannot be known in advance during testing.

4 Robust Type Conditioning

We present *Robust Type Conditioning* (RTC), a method for improving distributional realism in imitation learning. Unlike previous hierarchical methods with learned type encoders, RTC does not require the environment to be deterministic for good task performance. RTC follows the auto-encoder framework discussed in sections 2 and 3 but avoids causally confused policies and distribution shifts in state-types pairs $(\boldsymbol{s},\hat{\boldsymbol{g}})$ between training and testing. The two core augmentations of RTC are: i) including prior-sampled types during training to avoid causally confused policies and ii) restricting the information quantity communicated through the type, which reduces conditional type shift by minimising the amount of information the type $\hat{\boldsymbol{g}}_e$ contains about $\boldsymbol{\xi}$, i.e., $I(\hat{\boldsymbol{g}}_e;\boldsymbol{\xi})$, while maintaining high mutual information $I(\hat{\boldsymbol{g}}_e;\boldsymbol{g})$ with the true type $\boldsymbol{g}$.

To learn realistic *and* distributionally realistic behaviour, RTC combines four losses: the reconstruction loss $\mathcal{L}_{\text{rec}}$, the information bottleneck loss $\mathcal{L}_{\text{ib}}$, the adversarial loss $\mathcal{L}_{\text{adv}}$, and the prior loss $\mathcal{L}_{\text{prior}}$ (see fig. 3):

$$\begin{aligned}\mathcal{L}_{\text{RTC}} = \mathbb{E}_{\mathcal{D}(\tau)\boldsymbol{e_\theta}(\hat{\boldsymbol{g}}_e|\tau)\boldsymbol{\pi_\theta}(\hat{\tau}_e|\hat{\boldsymbol{g}}_e)}\big[\mathcal{L}_{\text{rec}}(\tau,\hat{\tau}_e) + \beta\mathcal{L}_{\text{ib}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}}(\hat{\tau}_e) + \mathcal{L}_{\text{prior}}(\tau)\big] \\ + \mathbb{E}_{\mathcal{D}(\tau)p_{\bar{\boldsymbol{\theta}}}(\hat{\boldsymbol{g}}_p)\boldsymbol{\pi_\theta}(\hat{\tau}_p|\hat{\boldsymbol{g}}_p)}\big[\qquad\qquad \lambda_{\text{adv}}\mathcal{L}_{\text{adv}}(\hat{\tau}_p) + \mathcal{L}_{\text{prior}}(\tau)\big],\end{aligned} \tag{2}$$

where $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)$ is a learned prior and $\boldsymbol{\pi_\theta}(\hat{\tau}|\hat{\boldsymbol{g}})$ is shorthand for generating trajectories $\hat{\tau}$ by rolling out the learned control policy $\boldsymbol{\pi_\theta}(\hat{\boldsymbol{a}}|\hat{\boldsymbol{s}},\hat{\boldsymbol{g}})$ in the environment. We denote by $\bar{\boldsymbol{\theta}}$ parameters of sampling distributions that are not updated through backpropagation and λ_{adv} and β are scalar weights.

We now introduce the individual terms. First, $\mathcal{L}_{\text{rec}}$ is a reconstruction loss between τ and $\hat{\tau}_e$ which prevents mode collapse by penalizing the agent for being unable to mimic τ. This also optimises the encoder to capture useful information about the trajectory τ in the inferred type $\hat{\boldsymbol{g}}_e$. The loss $\mathcal{L}_{\text{rec}}$ can take different forms. For example, in section 6.1 we use the BC loss $\mathcal{L}_{\text{rec}}(\tau) = -\log \boldsymbol{\pi_\theta}(\boldsymbol{a}_t|\boldsymbol{s}_t,\hat{\boldsymbol{g}}_e)$ while in section 6.2 we minimise the L_2 distance between agent positions in $\boldsymbol{s}_t$ and $\hat{\boldsymbol{s}}_t$. State-based losses like the L_2 reconstruction loss require access to a training environment able to resimulate the conditions of the original trajectory τ as we assume that τ is still approximately optimal.

The loss $\mathcal{L}_{\text{prior}}(\tau) = \mathbb{E}_{\hat{\boldsymbol{g}}_e\sim\boldsymbol{e}_{\bar{\boldsymbol{\theta}}}(\hat{\boldsymbol{g}}_e|\tau)}\left[\log p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_e)\right]$ optimises the prior to propose *distributionally realistic* types, even if τ is unavailable. Unfortunately, as discussed in section 3, this can induce marginal and conditional type

shifts. To eliminate these, we add the adversarial loss $\mathcal{L}_{\text{adv}}(\hat{\tau}) = \sum_t -\log D_{\boldsymbol{\phi}}(\hat{\boldsymbol{a}}_t, \hat{\boldsymbol{s}}_t)$, where $D_{\boldsymbol{\phi}}(\hat{\boldsymbol{a}}_t, \hat{\boldsymbol{s}}_t)$ is a learned discriminator (see section 2). Crucially, and unlike previous methods, $\mathcal{L}_{\text{adv}}$ is also minimised under inferred types sampled from the prior (second line in eq. (2)), in addition to those sampled from the encoder. If $\hat{\boldsymbol{g}}_e$ and $\hat{\boldsymbol{g}}_p$ maintain the same support, this reduces causal confusion in the policy because any information about $\boldsymbol{\xi}$ contained in $\hat{\boldsymbol{g}}$ is now unreliable since it might have been sampled randomly from the prior.

One can also view sampling from the prior as a causal intervention $do(\hat{\boldsymbol{g}})$ in which $\hat{\boldsymbol{g}}$ is changed independently of the environmental factor $\boldsymbol{\xi}$. De Haan et al. (2019) show that causal confusion can be avoided by applying such interventions and optimising the policy to correctly predict the counterfactual expert trajectory distribution, in our case $p_{\text{expert}}(\tau|\boldsymbol{\xi}, do(\hat{\boldsymbol{g}}))$. Unfortunately, we do not have access to this counterfactual trajectory. Instead, we rely on the generalisation of $\boldsymbol{\pi_\theta}$ to get us 'close' to such a counterfactual trajectory for types $do(\hat{\boldsymbol{g}})$ and then refine the policy locally using the adversarial objective.

So far, we have reduced the causal confusion of learned hierarchical policies and hence assured good task performance. We have not, however, removed the conditional distribution shift in the latent type, which is caused by encoding information about external factors $\boldsymbol{\xi}$, instead of information about the true type $\boldsymbol{g}$, in the latent type $\hat{\boldsymbol{g}}_e$. A possible failure case is that the policy simply learns to ignore the latent type, resulting in good task performance but unimproved distributional realism.

As a solution, we employ an informational bottleneck on types $\hat{\boldsymbol{g}}_e \sim \boldsymbol{e_\theta}(\hat{\boldsymbol{g}}_e|\tau)$ which filters information about $\boldsymbol{\xi}$ while letting information about $\boldsymbol{g}$ pass, thereby preventing this failure case and allowing for strong task performance *and* distributional realism. To see why, note that all information about $\boldsymbol{\xi}$ is also communicated through the visited states $\boldsymbol{s}$ to which the control policy $\boldsymbol{\pi_\theta}$ has access (see fig. 2). By contrast, information about $\boldsymbol{g}$ can *only* be accessed by the policy through $\hat{\boldsymbol{g}}_e$. Consequently, if the information bandwidth of $\hat{\boldsymbol{g}}_e$ is limited or costly, preference is given to information about the true expert type $\boldsymbol{g}$ as information about $\boldsymbol{\xi}$ can also be inferred directly from $\boldsymbol{s}$. Both continuous type representations using a variational information bottleneck with $\mathcal{L}_{\text{ib}} = \text{KL}[p(\hat{\boldsymbol{g}}_e)\|\mathcal{N}(\hat{\boldsymbol{g}}_e; 0, I)]$ (Alemi et al., 2016) and discrete type representations using straight-through gradient estimation work well in practice (see section 6.2).

To accommodate optimisation under inferred types drawn from both the encoder $\boldsymbol{e_\theta}$ and the prior $p_{\boldsymbol{\theta}}$, we split each minibatch $\mathcal{B} = \{\tau^{(b)}\}_b^{N_b}$ of N_b trajectories sampled from $\mathcal{D}$ into two parts. For the fraction f of trajectories in $\mathcal{B}$ the rollouts $\hat{\tau}_e$ are generated from types sampled from the encoder $\hat{\boldsymbol{g}}_e \sim \boldsymbol{e_\theta}(\hat{\boldsymbol{g}}_e|\tau)$ and all four losses are optimised (first line in eq. (2)). For the remaining fraction $(1-f)$ of trajectories types are sampled from the prior $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_p)$ and only $\mathcal{L}_{\text{adv}}$ and $\mathcal{L}_{\text{prior}}$ are optimised (second line in eq. (2)).

Optimisation of $\mathcal{L}_{\text{adv}}$ and $\mathcal{L}_{\text{rec}}$ can either be performed directly, similar to MGAIL (Baram et al., 2016), by using a differentiable environment and reparameterised policies and encoder (Kingma and Welling, 2013) or by treating them as rewards and using RL methods such as TRPO (Schulman et al., 2015; Ho and Ermon, 2016) or PPO (Schulman et al., 2017). The losses $\mathcal{L}_{\text{prior}}$ and $\mathcal{L}_{\text{ib}}$ can always be optimised directly.

5 Related Work

Several previous works combine adversarial training with autoencoder architectures in the image domain. Makhzani et al. (2016) use an adversarial loss on the latent variable in place of the KL-regularization used in VAEs. However, this eliminates the information bandwidth regularization for continuous latents which we show to be important for hierarchical imitation learning. Larsen et al. (2016) aim to learn a similarity metric for visual inputs using latent representations of the discriminator. This is valuable for imitation learning from raw images (Rahmatizadeh et al., 2018), but is not required for our experimental domains. Lastly, Chrysos et al. (2018), similar to our work, use an additional autoencoding loss to better capture the data distribution in the latent space. However, they consider denoising images instead of imitation learning under stochasticity.

Hierarchical policies have been extensively studied in RL (e.g., Sutton et al., 1999; Bacon et al., 2017; Vezhnevets et al., 2017; Nachum et al., 2019; Igl et al., 2020) and IL. In RL, they improve exploration, sample

efficiency and fast adaptation. By contrast, in IL, hierarchies are used to capture multimodal distributions, improve data efficiency (Krishnan et al., 2017; Le et al., 2018), and enable goal conditioning (Shiarlis et al., 2018). Similar to our work, Wang et al. (2017) and Lynch et al. (2020) learn to encode trajectories into latent types that influence a control policy. Crucially, both only consider deterministic environments and hence avoid the distribution shifts and unwanted information leakage we address. They extend prior work in which the type, or context, is provided in the dataset (Merel et al., 2017). Khandelwal et al. (2020) and Igl et al. (2022) use manually designed encoders specific to road users by expressing future goals as sequences of lane segments. This avoids information leakage but cannot express all characteristics of human drivers, such as persona, and cannot transfer to other tasks. Futures states in deterministic environments (Ding et al., 2019), language (Pashevich et al., 2021), and predefined strategy statistics (Vinyals et al., 2019) have also been used as types.

Information theoretic regularization offers an alternative to learning hierarchical policies using the auto-encoder framework (Li et al., 2017; Hausman et al., 2017). However, these methods are less expressive since their prior distribution cannot be learned and only aim to cluster modes already captured by the agent but not penalize dropping modes in the data. This provides a useful inductive bias but often struggles in complex environments with high diversity, requiring manual feature engineering (Eysenbach et al., 2019; Pathak et al., 2019).

Lastly, TrafficSim (Suo et al., 2021) uses IL to model driving agents and controls all stochasticity in the scene but uses independent prior distributions for separate agents. Hence, while no conditional distribution shift in $p(\hat{\boldsymbol{g}}|\boldsymbol{\xi})$ can occur (as $\boldsymbol{\xi}$ is constant), distribution shifts in $p(\hat{\boldsymbol{g}}^{(i)}|\hat{\boldsymbol{g}}^{(j)})$, and hence the joint marginal $p(\hat{\boldsymbol{g}}^{(1:N)})$ can occur for latent types $\hat{\boldsymbol{g}}^{(i)}, \hat{\boldsymbol{g}}^{(j)}$ of agents $i \neq j$ with $i, j \in \{1 \dots N\}$ and $\hat{\boldsymbol{g}}^{(1:N)} = [\hat{\boldsymbol{g}}^{(1)} \dots \hat{\boldsymbol{g}}^{(N)}]$: when drawn from the encoder, goals $\hat{\boldsymbol{g}}^{(i)}$ and $\hat{\boldsymbol{g}}^{(j)}$ are coordinated through conditioning on the joint agent future, while they are independent when drawn from the prior. They use a biased "common sense" collision avoidance loss, motivated by covariate shift in visited states. Our work suggests that marginal type shift might also explain the benefits gained. In contrast, our adversarial objective is unbiased. See appendix A.1 for more related work on *agent modelling* in multi-agent settings, *behavioural prediction* and *causal confusion*.

6 Experiments

We show in two stochastic environments with multimodal expert behaviour that i) existing adversarial methods suffer from insufficient distributional realism, ii) existing hierarchical methods cannot achieve good task performance *and* distributional realism and iii) `RTC` improves distributional realism while maintaining excellent task performance. We discuss differences in *realism*, *coverage* and *distributional realism* in fig. 6.

We compare the following models: `MGAIL` uses a learned discriminator and backpropagates gradients through the differentiable environment. It also optimises a `BC` loss as we found this to improve performance. `Symphony` (Igl et al., 2022), building on `MGAIL`, utilises future lane segments as manually specified types (see appendix A.5). Our implementation of `Symphony` outperforms the results from (Igl et al., 2022) due to the additional use of a value function. `InfoMGAIL` (Li et al., 2017) augments `MGAIL` to elicit distinct trajectories for different types. This introduces an inductive bias but does not directly penalise mode collapse. Our methods, `RTC-C` and `RTC-D`, use a continuous or discrete type respectively. We also perform the ablation `Hierarchy-NoPT` (*No Prior Training*) which only uses the first line in eq. (2), i.e. $f = 1$. `Hierarchy-NoPT` is similar to existing hierarchical methods, such as the proprietory TrafficSim (Suo et al., 2021), in that it learns the prior but does not use it during training, only inference. It thereby does not account for distribution shifts in the latent types, as discussed in section 3.

6.1 Double Goal Problem

In the double goal problem, the expert starts from the origin and creates a multimodal trajectory distribution by randomly choosing and approaching one of two possible, slowly moving goals located on the 2D plane. Stochasticity is introduced through randomized initial goal locations and movement directions. Nevertheless,

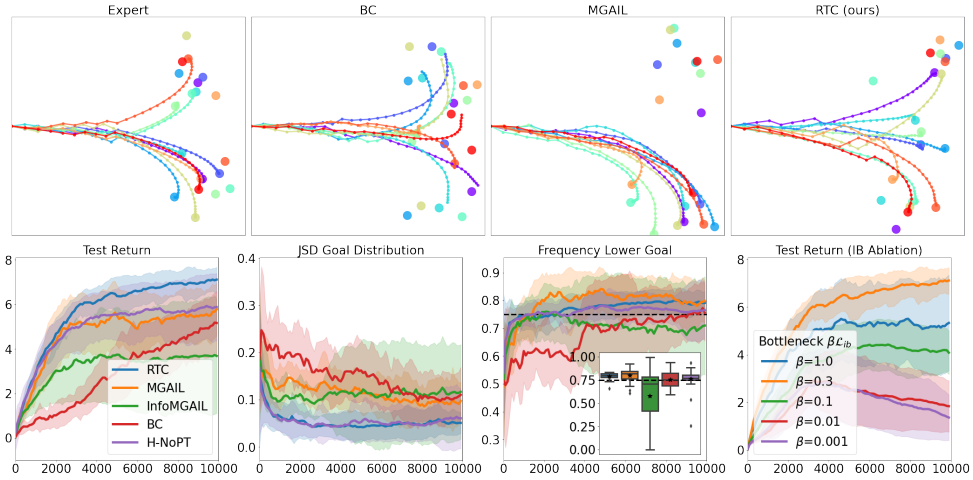


Figure 4: *Top:* Visualization of ten randomly sampled goal pairs and associated trajectories. *Bottom:* Training curves, exponentially smoothed and averaged over 20 seeds. Shading shows the standard deviation. The inset shows the distribution at the last training step. Boxes show quartiles, whiskers extreme values, diamonds outliers, and stars the mean.

the *lower* and *upper* goal $\{\boldsymbol{g}_l, \boldsymbol{g}_u\}$ remain identifiable by their location as $y_l < 0$ for $\boldsymbol{g}_l$ and $y_u > 0$ for $\boldsymbol{g}_u$ (see fig. 4). While both goals are equally easy to reach, the expert has a preference $P(G = \boldsymbol{g}_l) = 0.75$. Sufficiently complex expert trajectories prevent `BC` from achieving optimal performance, requiring more advanced approaches. The expert follows a curved path and randomly resamples the selected goal for the first ten steps to avoid a simple decision boundary along the x-axis in which experts in the lower half-plane always target goal $\boldsymbol{g}_l$. `RTC` uses the BC loss as reconstruction loss $\mathcal{L}_{\text{rec}}(\tau) = -\log \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}_t | \boldsymbol{s}_t, \hat{\boldsymbol{g}}_e)$ and continuous types. All policies use a bimodal Gaussian mixture model as action distribution. Performance is measured as the number of steps for which the agent is within $\delta = 0.1$ distance of one of the goals. We take $h_s = \text{sign}(y_T)$ of the final agent position $[x_T, y_T]$ to indicate the approached goal and measure distributional realism as the divergence between the empirical distributions, JSD $(p_{\text{agent}}(h_s) \| p_{\text{expert}}(h_s))$. Details can be found in appendix A.3.

Figure 4 shows that `MGAIL` improves task performance compared to `BC`. Our method, `RTC`, improves it further, possibly because given a type, the required action distribution is unimodal. Importantly, `RTC` substantially improves distributional realism, achieving lower JSD values. To analyse this result, we show $p_{\text{agent}}(h_s = -1)$, the frequency of targeting the lower goal. Not only is `RTC`'s average value of $p_{\text{RTC}}(h_s = -1)$ closer to the true value of 0.75, it is also more stable across seeds, resulting in a lower JSD. The bias introduced by `InfoMGAIL` reduces task performance without improving distributional realism. As expected, the ablation `Hierarchy-NoPT` achieves excellent distributional realism through the learned hierarchy but suffers reduced task performance due to unaccounted distribution shifts. Lastly, the rightmost plot of fig. 4 shows that the information bottleneck is necessary.

6.2 Waymo Open Motion Dataset (WOMD)

To evaluate `RTC` on a complex environment we use the *Waymo Open Motion Dataset* (Ettinger et al., 2021) consisting of $487K$ segments of real world driving behaviour. Distributionally realistic agents are critical for driving simulations, for example for estimating safety metrics. Diverse intents and driving styles cause the data to be highly multimodal. Stochasticity is induced through the unpredictable behaviour of other cars, cyclists and pedestrians. We use $\mathcal{L}_{\text{rec}}(\tau, \hat{\tau}) = \sum_t^T \mathcal{L}_{\text{Huber}}(\boldsymbol{s}_t, \hat{\boldsymbol{s}}_t)$ where $\mathcal{L}_{\text{Huber}}$ is the average Huber loss of the four vehicle bounding box corners. More details can be found in appendix A.4.

Table 1: Averages and standard deviation over 10 training runs on *WOMD*.

	Collision rate (%) ↓	Off-road time (%) ↓	MinADE (m) ↓	Curvature JSD ($\times 10^{-3}$) ↓	Progress JSD ($\times 10^{-3}$) ↓
Data Distribution	1.16	0.68	-	-	-
MGAIL	5.39 ± 0.68	0.89 ± 0.12	1.34 ± 0.08	1.32 ± 1.48	3.81 ± 1.29
Symphony	6.39 ± 0.95	0.90 ± 0.06	1.40 ± 0.12	0.97 ± 0.62	6.44 ± 5.25
InfoMGAIL - C	5.21 ± 0.37	0.89 ± 0.14	1.29 ± 0.07	1.24 ± 0.93	4.40 ± 1.47
InfoMGAIL - D	4.82 ± 0.29	0.84 ± 0.10	1.35 ± 0.11	0.77 ± 0.44	4.01 ± 1.45
Hierarchy-NoPT	35.08 ± 0.44	1.83 ± 0.42	$\mathbf{1.12 \pm 0.01}$	1.76 ± 2.05	2.54 ± 0.63
RTC - C	4.23 ± 0.16	$\mathbf{0.68 \pm 0.04}$	1.15 ± 0.10	$\mathbf{0.43 \pm 0.06}$	$\mathbf{2.17 \pm 0.65}$
RTC - D	$\mathbf{4.21 \pm 0.24}$	0.74 ± 0.06	$\mathbf{1.12 \pm 0.10}$	0.89 ± 0.66	2.56 ± 0.54

We use the percentage of segments with collisions and time spent off-road as proxy metrics for task performance. Mode coverage is measured by the minimum average displacement error, $\text{minADE} = \mathbb{E}_{\tau\sim\mathcal{D},\{\hat{\tau}_i\}_i^K\sim\boldsymbol{\pi_\theta}}\left[\min_{\hat{\tau}_i}\frac{1}{T}\sum_{t=1}^{T}\delta(\boldsymbol{s}_t,\hat{\boldsymbol{s}}_{i,t})\right]$, where δ is the Euclidean distance between agent positions and we find the minimum over $K = 16$ rollouts (hierarchical methods use K independently sampled types). Lower *minADE* implies better mode coverage, but does not directly measure the relative frequency of modes, e.g., low probability modes may be overrepresented. To measure distribution matching in driving intent, we use the *Curvature JSD* (Igl et al., 2022): in lane branching regions, such as intersections, it maps trajectories to the nearest lane and extracts its curvature as feature h_{cur}. To compute $\text{JSD}\left(p_{\text{agent}}(h_{cur})\|p_{\text{expert}}(h_{cur})\right)$, the value of h_{cur} is discretize into 100 equisized bins. To measure the driving style distribution, we extract the progress feature $h_{style} = \delta(\hat{\boldsymbol{s}}_0, \hat{\boldsymbol{s}}_T)$ and use the same discretization to compute the JSD.

Results are provided in table 1. Both versions of `RTC` improve task performance (collisions and off-road events) and distributional realism metrics (minADE and divergences) compared to the flat `MGAIL` baseline, as well as previous hierarchical approaches (`Symphony`, `InfoMGAIL` and `Hierarchy-NoPT`). Both type representations, `RTC-C` and `RTC-D`, perform similarly, showing robustness of `RTC` to different implementations. The advantage of `RTC` in achieving *both* good task performance and distributional realism becomes clearest by comparing it to `Hierarchy-NoPT`. While `Hierarchy-NoPT` achieves some improvements in distributional realism, is has nearly an order of magnitude more collisions. This is a consequence of the distributional shifts discussed in section 3, which `RTC` is able to avoid. On this task, good task performance of distributionally realistic methods is particularly impressive, as one might expect them to collide more frequently than the flat `MGAIL` baseline: `MGAIL` prioritises easier to learn modes (e.g. driving straight) over difficult ones (e.g. turning left), so higher distributional realism likely implies driving more manoeuvres with higher risk of collision.

7 Conclusions, Limitations, and Future Work

This paper identified new challenges in learning hierarchical policies from demonstration to capture multi-modal trajectory distributions in stochastic environments. We expressed them as *marginal* and *conditional type shifts*, in addition to the covariate shift of state visitations in IL. We proposed *Robust Type Conditioning* (`RTC`) to eliminate these distribution shifts and showed improved distributional realism while maintaining or improving task performance on two stochastic environments, including the Waymo Open Motion Dataset (Ettinger et al., 2021). Future work will address *conditional* distributional realism by not only matching the marginal distribution $p(\tau)$, but the conditional distribution $p(\tau|\boldsymbol{\xi})$ under a specific realization of the environment. For example, drivers might change their intent based on the current traffic situation or players might adapt their strategy as the game unfolds. Achieving such conditional distributional realism will require new models, metrics, and augmentations of `RTC`.

REFERENCES

A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

N. Baram, O. Anschel, and S. Mannor. Model-based adversarial imitation learning. *arXiv preprint arXiv:1612.02179*, 2016.

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *European Conference on Computer Vision*, pages 624–641. Springer, 2020.

Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019.

G. G. Chrysos, J. Kossaifi, and S. Zafeiriou. Robust conditional generative adversarial networks. *arXiv preprint arXiv:1805.08657*, 2018.

H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019.

P. De Haan, D. Jayaraman, and S. Levine. Causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov. Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset. *CoRR*, abs/2104.10133, 2021. URL https://arxiv.org/abs/2104.10133.

T. Everitt, M. Hutter, R. Kumar, and V. Krakovna. Reward tampering problems and solutions in reinforcement learning: A causal influence diagram perspective. *Synthese*, 198(27):6435–6467, 2021.

B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

J. D. Farmer and D. Foley. The economy needs agent-based modelling. *Nature*, 460(7256):685–686, 2009.

J. Fu, K. Luo, and S. Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkHywl-A-.

A. Grover, M. Al-Shedivat, J. Gupta, Y. Burda, and H. Edwards. Learning policy representations in multiagent systems. In *International conference on machine learning*, pages 1802–1811. PMLR, 2018.

D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, and A. Dragan. Inverse reward design. *Advances in neural information processing systems*, 30, 2017.

K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/632cee946db83e7a52ce5e8d6f0fed35-Paper.pdf.

H. He, J. Boyd-Graber, K. Kwok, and H. Daumé III. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pages 1804–1813. PMLR, 2016.

P. Hernandez-Leal, B. Kartal, and M. E. Taylor. Agent modeling as auxiliary task for deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment*, volume 15, pages 31–37, 2019.

J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

A. Hu, Z. Murez, N. Mohan, S. Dudas, J. Hawke, V. Badrinarayanan, R. Cipolla, and A. Kendall. Fiery: Future instance prediction in bird's-eye view from surround monocular cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15273–15282, 2021.

M. Igl, A. Gambardella, J. He, N. Nardelli, N. Siddharth, W. Böhmer, and S. Whiteson. Multitask soft option learning. In *Conference on Uncertainty in Artificial Intelligence*, pages 969–978. PMLR, 2020.

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. *arXiv preprint arXiv:2205.03195*, 05 2022.

B. Ivanovic and M. Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2375–2384, 2019.

S. Khandelwal, W. Qi, J. Singh, A. Hartnett, and D. Ramanan. What-if motion prediction for autonomous driving. *arXiv preprint arXiv:2008.10587*, 2020.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

S. Krishnan, R. Fox, I. Stoica, and K. Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on robot learning*, pages 418–437. PMLR, 2017.

A. M. Lamb, A. G. ALIAS PARTH GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016.

A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

H. Le, N. Jiang, A. Agarwal, M. Dudik, Y. Yue, and H. Daumé III. Hierarchical imitation and reinforcement learning. In *International conference on machine learning*, pages 2917–2926. PMLR, 2018.

Y. Li, J. Song, and S. Ermon. Infogail: Interpretable imitation learning from visual demonstrations. *Advances in Neural Information Processing Systems*, 30, 2017.

Y. Liang, C. Guo, Z. Ding, and H. Hua. Agent-based modeling in electricity market using deep deterministic policy gradient algorithm. *IEEE Transactions on Power Systems*, 35(6):4180–4192, 2020.

Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou. Multimodal motion prediction with stacked transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7577–7586, 2021.

C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *Conference on robot learning*, pages 1113–1132. PMLR, 2020.

A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016. URL http://arxiv.org/abs/1511.05644.

J. Merel, Y. Tassa, D. TB, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

O. Nachum, S. Gu, H. Lee, and S. Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1emus0qF7.

J. F. Nash Jr. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1): 48–49, 1950.

P. A. Ortega, M. Kunesch, G. Delétang, T. Genewein, J. Grau-Moya, J. Veness, J. Buchli, J. Degrave, B. Piot, J. Perolat, et al. Shaking the foundations: delusions in sequence models for interaction and control. *arXiv preprint arXiv:2110.10819*, 2021.

G. Papoudakis and S. V. Albrecht. Variational autoencoders for opponent modeling in multi-agent systems. *arXiv preprint arXiv:2001.10829*, 2020.

J. Park, Y. Seo, C. Liu, L. Zhao, T. Qin, J. Shin, and T.-Y. Liu. Object-aware regularization for addressing causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 34:3029–3042, 2021.

A. Pashevich, C. Schmid, and C. Sun. Episodic transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15942–15952, 2021.

D. Pathak, D. Gandhi, and A. Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pages 5062–5071. PMLR, 2019.

T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020.

R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.

R. Raileanu, E. Denton, A. Szlam, and R. Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*, pages 4257–4266. PMLR, 2018.

A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

D. J. Rezende and F. Viola. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

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. JMLR Workshop and Conference Proceedings, 2011.

T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer, 2020.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, pages 4654–4663. PMLR, 2018.

S. Suo, S. Regalado, S. Casas, and R. Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409, 2021.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

C. Tang and R. R. Salakhutdinov. Multiple futures prediction. *Advances in Neural Information Processing Systems*, 32, 2019.

A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess. Robust imitation of diverse behaviors. *Advances in Neural Information Processing Systems*, 30, 2017.

C. Wen, J. Lin, T. Darrell, D. Jayaraman, and Y. Gao. Fighting copycat agents in behavioral cloning from observation histories. *Advances in Neural Information Processing Systems*, 33:2564–2575, 2020.

A. Xie, D. P. Losey, R. Tolsma, C. Finn, and D. Sadigh. Learning latent representations to influence multi-agent interaction. *arXiv preprint arXiv:2011.06619*, 2020.

Y. Yuan, X. Weng, Y. Ou, and K. M. Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9813–9823, 2021.

Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

A APPENDIX

A.1 ADDITIONAL RELATED WORK

Unlike our work, *agent modelling* (Grover et al., 2018; Papoudakis and Albrecht, 2020) often assumes knowledge of agent identities in multi-agent systems and aims at learning a useful representation for each identity. In contrast, we neither know the true type $\boldsymbol{g}$ of the imitated agent, nor the identity of external stochastic noise source $\boldsymbol{\xi}$. Furthermore, applications of *opponent modelling* in RL settings (e.g., Papoudakis and Albrecht, 2020; He et al., 2016; Raileanu et al., 2018; Hernandez-Leal et al., 2019; Xie et al., 2020) are generally unconcerned about distributional realism and do not consider distribution shifts.

Behaviour Prediction (BP) also forecasts future trajectories. Unless future steps are predicted independently of the evolution of the scene (e.g., not auto-regressively) (Chai et al., 2019; Cui et al., 2019; Phan-Minh et al., 2020; Liu et al., 2021), these methods also suffer from covariate shift in the state visitations (Bengio et al., 2015; Lamb et al., 2016). Furthermore, if hierarchical methods are used to capture the multimodality in the data (Tang and Salakhutdinov, 2019; Casas et al., 2020; Ivanovic and Pavone, 2019; Salzmann et al., 2020; Yuan et al., 2021; Hu et al., 2021), they are vulnerable to the same marginal and conditional type shifts we consider. While none of these works take these challenges into account, they often use small discrete latent spaces (e.g., Tang and Salakhutdinov, 2019; Ivanovic and Pavone, 2019; Salzmann et al., 2020), mitigating the severity of the distribution shifts and future information leakage by limiting the information bandwidth of latent types. Furthermore, prediction quality metrics such as displacement-based metrics or log-likelihood are less sensitive to yield lower performance due to covariate shift, which primarily impacts interactions with the environment, such as collisions.

As discussed in section 3, the conditional type shift is exacerbated by causally confused policies relying on the latent type for information about environmental noise. Unlike in most literature on causal confusion (De Haan et al., 2019), our nuisance variables are hence not part of the current state, but the learned latent state. Distribution shift is induced not through earlier actions but through sampling from the prior instead of the encoder. Prior work on causal confusion typically relies on problem specific regularization (e.g. Wen et al., 2020; Park et al., 2021) or has access to an expert or task rewards (e.g. De Haan et al., 2019; Ortega et al., 2021). Instead, our work relies on generalisation over latent types to generate counterfactual trajectories. This generalisation is enabled by the information bottleneck and results are refined by the adversarial loss.

A.2 LIMITATIONS AND SOCIETAL IMPACT

While `RTC` notably improves distributional realism (see section 6), it does not achieve it perfectly, especially in the long tail of the data distribution. This has implications for its use, for example in economic simulations to evaluate policy proposals or in driving simulations to evaluate autonomous vehicles, where this limitation has to be taken into account and the simulation results should not be trusted unconditionally.

As `RTC` is application agnostic, the societal impact depends on where it is used. Here, we focus on agent-based simulations as we anticipate this to create the highest impact. Examples include better policy decisions through economic simulations, safer autonomous vehicles through driving simulations, better AI in games or improved safety precautions for large crowds of people. For other use-cases, e.g., in armed conflicts, the societal impact will depend on the intention of the simulation. Furthermore, we stress once more, that for many use-cases, precautions have to be taken to account for remaining errors in the learned agents.

Lastly, depending on the use case, algorithmic bias has to be taken into account if mode-collapse might be prevented more effectively by `RTC` for certain strata in the population.

A.3 DETAILS ON DOUBLE GOAL PROBLEM

The agent observation

$$\boldsymbol{s}_t = [\boldsymbol{s}_t, \boldsymbol{g}_{l,t}, \boldsymbol{g}_{u,t}, \boldsymbol{a}_{t-1}] \in \mathbb{R}^8 \text{ with } \boldsymbol{s}_t = [x_t, y_t],\ \boldsymbol{g}_{i,t} = [x_{i,t}, y_{i,t}],\ i \in \{u, l\} \tag{3}$$

contains the 2D position of the agent, $\boldsymbol{s}_t$, as well as two marked locations $\boldsymbol{g}_{l,t}, \boldsymbol{g}_{u,t}$ of the lower and upper goal. Because the current agent position cannot uniquely identify the currently selected goal, the observation also contains the last agent action $\boldsymbol{a}_{t-1}$ with the simple transition function $\boldsymbol{s}_{t+1} = \boldsymbol{s}_t + \boldsymbol{a}_t$. The goal locations are randomly sampled at the beginning of each episode. The lower (upper) goal is always located in the lower (upper) half of the x, y plane. Their horizontal and vertical distances from the initial agent position are uniformly sampled within rectangular bounds $x_i^{(g)} \in [1.8, 2.2]$ and $|y_i^{(g)}| \in [0.3, 0.7]$. Each episode has a fixed horizon of $T = 30$ steps over the course of which each goal moves by $\|\boldsymbol{s}_{i,T}^{(g)} - \boldsymbol{s}_{i,0}^{(g)}\| = 0.15$ in a random direction.

For the first 10 timesteps, the agent randomly resamples the target goal with $P(G = \boldsymbol{g}_l) = 0.75$ to avoid a simple decision boundary along the x-axis in which experts in the lower half-plane always target goal $\boldsymbol{g}_l$. The expert action is

$$\boldsymbol{a}_t = 0.1\sqrt{\|\Delta_t\|}\frac{\boldsymbol{d}_t}{\|\boldsymbol{d}_t\|} \quad \text{where} \quad \boldsymbol{d}_t = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.05 \end{bmatrix} \Delta_t \quad \text{and} \quad \Delta_t = (\boldsymbol{g}_t - \boldsymbol{s}_t). \tag{4}$$

The expert approaches the goal faster along the x-axis, hence creating a curved path. To avoid over-shooting, the step-size reduces by $\sqrt{\|\Delta_t\|}$ as the agent proceeds towards the goal.

All networks use simple MLPs with two latent layers and a latent dimension of 256. To capture their shape, the discriminator acts on entire trajectories, aggregating across time using max-pooling over a 32 dimensional per-timestep embedding. All policies are parameterised as Gaussian mixture models with two modes. `RTC` uses a continuous bottleneck of size 2 with additional regularization term $\mathcal{L}_{\text{ib}}^{\beta}(\tau) = \beta KL\left[\boldsymbol{e}_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}}_e|\tau)\|\mathcal{N}(0, I)\right]$ to regulate the information bandwidth of $\hat{\boldsymbol{g}}_e$. We use the BC loss for $\mathcal{L}_{\text{rec}}(\tau, \hat{\tau}_e) = -\log \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t, \hat{\boldsymbol{g}}_e)$.

Batch size is 1024 for training and $10K$ for evaluation. Results shown in fig. 4 are evaluated every 100 steps and exponentially smoothed with a decay rate of 0.9. The learning rate is 0.01 for BC (lower learning rates performed worse) and 0.004 for both MGAIL and `RTC`, which were tuned independently for values $lr \in [0.02, 0.01, 0.004, 0.002, 0.001]$. $f = 0.5$ was used to split between $\mathcal{B}_{\text{encoder}}$ and $\mathcal{B}_{\text{prior}}$ (no tuning was performed). Lastly, without further tuning, $\lambda_{\text{adv}} = 1$ was used. Training time is about 7h without hardware acceleration.

A.4 DETAILS ON WAYMO OPEN MOTION DATASET

The Waymo Open Motion Dataset (Ettinger et al., 2021) (published under Apache License 2.0) consists of segments of length $9s$ sampled at $10Hz$. The available training and validation splits in the dataset consist of $487K$ and $49K$ segments each, which are used for training and testing the agent respectively. Due to memory constraints, we filter for segments with less than 256 agents and $10K$ points describing the lane geometry - resulting in $428K$ train and $39K$ test segments. 250 segments from the training split are used for validation to select the training checkpoint for evaluation. In each segment, we learn to control two agents at a frequency of $3.33Hz$, repeating actions three times. Similar to (Igl et al., 2022), the actions of other agents are replayed from the logged data. The collision metric measures the number of segments, in percent, for which at least one pair of bounding boxes overlaps for at least one timestep. The off-road metric similarly detects for how much time the agent's bounding box overlaps with off-road areas.

The state $\boldsymbol{s}_t = [\boldsymbol{s}_t^{(a)}, \boldsymbol{s}_t^{(SS)}, \boldsymbol{s}_t^{(DS)}, \boldsymbol{s}^{(RU)}]$ contains the agent's position and heading $\boldsymbol{s}_t^{(a)}$, static features $\boldsymbol{s}_t^{(SS)}$ such as lane boundaries, expressed as a set of points, dynamical features $\boldsymbol{s}_t^{(DS)}$ such as traffic light states, and the positions and headings of all other road users $\boldsymbol{s}_t^{(RU)}$.

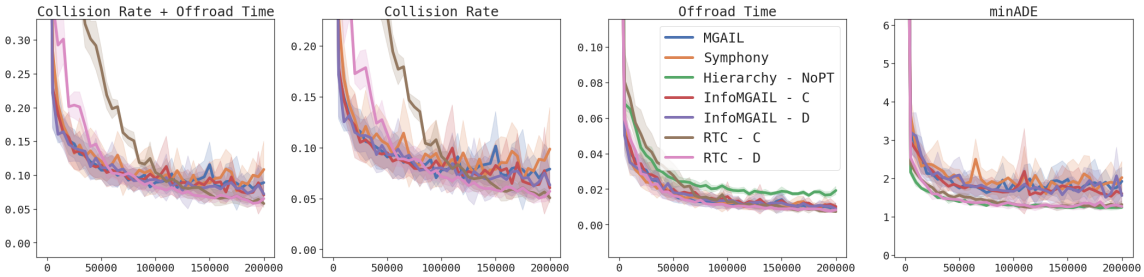


Figure 5: *Waymo Open Motion Dataset*: Performance on the validation set during training. Distributional realism metrics are not shown as their evaluation is high variance on the small validation set.

Similarly to section 6.1, we use an additive transition model in which the policy predicts the change in the agents state $\boldsymbol{s}_{t+1}^{(a)} = \boldsymbol{s}_{t}^{(a)} + \boldsymbol{a}_t$. Dynamic features and roadgraph users are replayed from the logged data, similar to (Igl et al., 2022).

All positions and headings are first normalised to be relative to the observing ego-agent. MLPs are used to encode each object and point individually and per-type max-pooling is used to aggregate over a variable number of inputs. The resulting three embeddings (one for the ego-agent, one for other road-users and one for the scene), each of size 64, are concatenated and passed either to the policy, discriminator or value function, whose encoders are not shared and which consist of MLPs with two latent layers of size 64 for discriminator and value function and 128 for the policy. The inference encoder $\boldsymbol{e_\theta}$ for `RTC` only observes future agent positions $\boldsymbol{s}_{1:T}^{(a)}$ which are each concatenated with an eight-dimensional learned positional embedding and individually encoded to dimension 128 and max-pooled along the time dimension. A Gaussian mixture model with 8 modes was used for all policies, although we find empirically that typically only up to three are used after training.

We train for $200K$ gradient steps and select the model checkpoint for evaluation with the lowest sum of collision rates and off-road time on the validation set. To stabilize training for all methods, we discount gradients through time with $\gamma = 0.9$ and bootstrap from a learned value function every 10 steps. We anneal f from 1 to $f_{\text{min}} = 0.5$ over the course of training. Initially, high values of f encourage meaningful information in $\hat{\boldsymbol{g}}_e$ while lower values address covariate and type shifts and improve performance. A learning rate of 0.0001, which was tuned for MGAIL, was used for all evaluated methods. Each batch contained 24 segments and training was performed on a single V100 (per seed) and required about 4-5 days. We used $\lambda_{\text{adv}} = 4.0$ and $\beta = 0.01$ for $\mathcal{L}_{\text{ib}}(\tau)$ for continuous type representations of size 2. Discrete type representations used three one-hot vectors of size 16, trained using Straight Through gradient estimation. We found performance to be marginally better for three vectors, compared to one, without noticeable performance increases for additional or larger vectors. Smaller vectors with only four values only performed slightly worse. The Huber loss $\mathcal{L}_{\text{rec}}$ uses $\delta = 30$.

A.5 Details on Symphony baseline

`Symphony` implements the hierarchical policy proposed in Igl et al. (2022) (called 'MGAIL+H' in their results). Agent types represent high-level driving intent and are expressed as a sequence of road-segments to be followed. They are encoded into a latent vector by expressing them as a fixed-length sequence of points $\{[x_i, y_i]\}_{i=1}^{N_s}$. Each point is concatenated with a positional embedding, then encoded individually, and subsequently max-pooled along the time-dimension. The pooled embedding is provided as additional inputs to both the discriminator and the policy.

During training, lane sequences are extracted from the given data trajectory τ. The prior $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{g}})$, which is used during testing when no ground truth trajectories τ are available, predicts a categorical distribution over all possible sequences of lane-segments which the agent could follow in a scene. To allow for a variable number of such sequences, the logits are predicted individually per sequence.

A.6 Details on InfoGAIL Baseline

Like `RTC`, `InfoMGAIL` (Li et al., 2017) is a general method for learning a hierarchical agent from demonstrations. What makes it a suitable baseline is that, like in `RTC`, the higher level policy captures a distribution over alternative trajectories that can be taken. It does so in an unsupervised fashion by introducing an additional reward that incentivizes the policy to produce state-action pairs from which an additionally trained discriminator (also called 'posterior') can infer the type on which the policy was conditioned. In other words, it rewards the policy for producing distinct trajectories for different types, where the type is drawn from a fixed prior when generating rollouts.

A crucial difference between `InfoMGAIL` and `RTC` is that `InfoMGAIL`'s goal is to disentangle trajectories, but it does not target distributional realism directly. In particular, because the prior from which the types are sampled is fixed, it might not even be able to properly capture the true distribution of trajectories. This is especially true for the uniform discrete prior used in the original `InfoMGAIL` paper, which assumes a uniform distribution over trajectory modes. Furthermore, the additional posterior reward introduces bias, potentially harming task performance. Lastly, because mode collapse is not directly penalized in the additional loss (only 'non-distinctiveness' of trajectories), it might not improve distributional realism at all.

In our experiments, we augment `InfoMGAIL` in several ways:

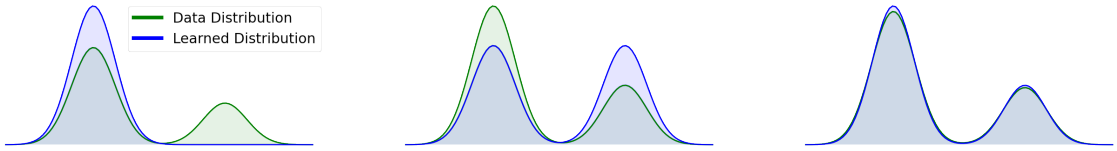
- We not only try discrete latents, but also continuous ones. For continuous priors we use the same GMM as posterior as we use as prior in RTC.
- We additionally provide the posterior with the initial state as input. Unlike in the examples used in the `InfoMGAIL` paper, we believe that for more complex WOMD data, the current state is insufficient to determine modes.
- To make it comparable in our setup, we optimise it using Info(M)GAIL, i.e. the posterior score of the true type is added as differentiable loss term, not as reward for TRPO. The network architecture of the posterior is the same one as we used for our `MGAIL` discriminator.
- We greatly increase the number of latent dimensions. In (Li et al., 2017), 2 and 3 dimension were used for the two experiments. We tried $d \in [3, 10, 30, 100]$. We also tried $\lambda_1 \in [0.01, 0.03, 0.1, 0.3, 1.0]$ as regularization strength for the additional loss term.
- Lastly, we are also adding a `BC` term to `InfoMGAIL` as we found this stabilizes training greatly.
- In contrast to the original implementation, we are not using pre-training and do not make use of additional shaping rewards.

A.7 Coverage and Distributional Realism Metrics

While coverage is easy to achieve on the Double Goal Problem, we measure it on the Waymo Open Motion Dataset (section 6.2) using

$$\text{minADE} = \mathbb{E}_{\tau \sim \mathcal{D}, \{\hat{\tau}_i\}_i^K \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}} \left[\min_{\hat{\tau}_i} \frac{1}{T} \sum_{t=1}^{T} \delta(\boldsymbol{s}_t, \hat{\boldsymbol{s}}_{i,t}) \right]$$

using a fixed number of $K = 16$ rollouts per segment. Intuitively, the more modes are covered by a given agent, the closer one of all K rollouts should be to a given trajectory from the dataset, resulting in a lower *minADE*.



(a) **(b)** **(c)**

Figure 6: Differences between *realism*, *coverage* and *distributional realism*. The data distribution $P(X_D) \in \Delta(\mathcal{X})$ is shown in green, blue denotes a learned distribution $P_\theta(X_L) \in \Delta(\mathcal{X})$. **(a)** Data from learned distribution is *realistic*, i.e. $\text{supp}(X_L) \subseteq \text{supp}(X_D)$, but not *distributionally realistic*. **(b)** The learned distribution achieves *coverage* but not *distributional realism*: the frequencies of modes are not matched. **(c)** The learned distribution is *distributionally realistic*. In practice, the dimensionality of $\mathcal{X}$ is often too high, requiring us to measure distributional realism only in selected features $h(X)$. Consequently, distributional realism in $h(X)$ does not necessarily imply good realism, i.e. task performance.

We want to measure distributional realism as the divergence between the expert distribution $p_{\text{expert}}(\tau)$ and the predicted distribution $p_{\text{agent}}(\hat{\tau})$. However, since the space of possible trajectories is far too large to directly measure JSD $(p_{\text{agent}}(\hat{\tau}) \| p_{\text{expert}}(\tau))$, we extract scalar features h from trajectories and measure the divergence on those features. For the Double Goal Problem, we would like to capture whether the agent is approaching $\boldsymbol{g}_l$ or $\boldsymbol{g}_u$, for which we extract $h_s = \text{sign}(y_T)$, i.e. whether the agent is in the lower or upper half of the plane at the last timestep. In the driving domain, we measure progress as the total distance travelled over the $9s$ segment, i.e., $h_{style} = \delta(\hat{\boldsymbol{s}}_0, \hat{\boldsymbol{s}}_T)$. Measuring this distance as a straight line avoids measurement noise through swerving or jittering of the agent. Lastly, to measure high-level intent, i.e. whether the agent prefers going left, right or straight at branching points such as intersections, we follow Igl et al. (2022) and extract as feature h_{cur}, i.e., the curvature of the lane segments being followed right after possible branching points in the road.