

# Deep Latent Competition: Learning to Race Using Visual Control Policies in Latent Space

Wilko Schwarting<sup>\*,1</sup>, Tim Seyde<sup>\*,1</sup>, Igor Gilitschenski<sup>\*,1</sup>,  
Lucas Liebenwein<sup>1</sup>, Ryan Sander<sup>1</sup>, Sertac Karaman<sup>2</sup>, Daniela Rus<sup>1</sup>  
<sup>1</sup>MIT CSAIL, <sup>2</sup>MIT LIDS, <sup>\*</sup>equal contribution  
{wilkos,tseyde,igilitschenski}@mit.edu

**Abstract:** Learning competitive behaviors in multi-agent settings such as racing requires long-term reasoning about potential adversarial interactions. This paper presents Deep Latent Competition (DLC), a novel reinforcement learning algorithm that learns competitive visual control policies through self-play in imagination. The DLC agent imagines multi-agent interaction sequences in the compact latent space of a learned world model that combines a joint transition function with opponent viewpoint prediction. Imagined self-play reduces costly sample generation in the real world, while the latent representation enables planning to scale gracefully with observation dimensionality. We demonstrate the effectiveness of our algorithm in learning competitive behaviors on a novel multi-agent racing benchmark that requires planning from image observations. Code and videos available at <https://sites.google.com/view/deep-latent-competition>.

**Keywords:** Autonomous Racing, Multi-agent Reinforcement Learning, World Models

## 1 Introduction

Driverless racing is a challenging task promising to push the limits of autonomous navigation as it requires robust operation of the entire driving stack at high speeds. Racing becomes particularly challenging when multiple agents simultaneously compete within the same environment. This requires both, fast processing of observations and reasoning about opponent behavior. By addressing these challenges, racing can provide novel insights for the deployment of autonomous systems. In the context of Multi-Agent Reinforcement Learning (MARL), racing can benchmark competitiveness as it requires reasoning about the interplay between ego actions and opponent behavior. This is particularly challenging when operating under partial observability in high-dimensional input spaces and in the absence of structured priors over environment behavior.

Recent approaches to autonomous racing tend to assume access to a nominal environment model [1, 2, 3]. Some also leverage game-theoretic frameworks such as iterated best response [4, 5, 6]. However, in racing

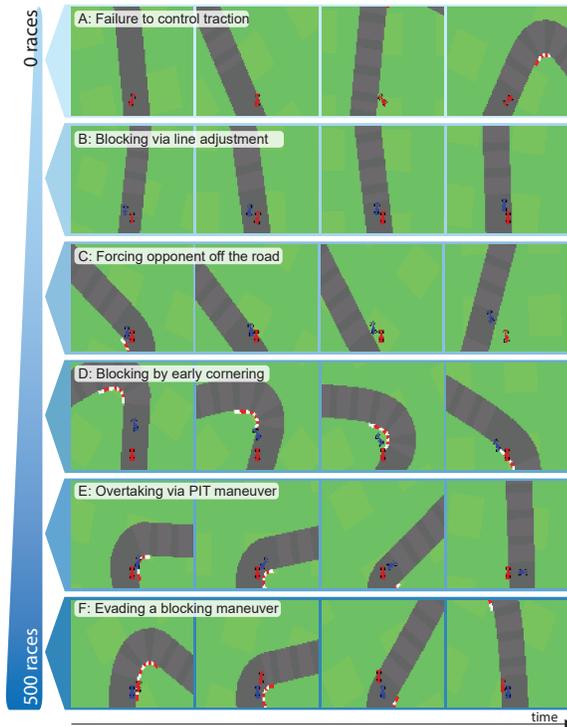


Figure 1: Skill acquisition from experience.

and other real-world multi-agent settings, relevant behavioral and environmental features may be too complex to be captured by MPC or purely game-theoretic approaches that assume perfect knowledge of the state and the system dynamics. MARL provides an alternative framework for modelling rich agent interactions. Impressive successes have been demonstrated for large-scale competition in discrete action spaces [7] with access to privileged ground truth information such as categorized entity lists and multi-layer maps. Continuous control applications of MARL in competitive settings that only provide image observations are less well-studied. In particular, combining high-dimensional observations such as images, partial observability of other agents, multi-agent world model learning, and acquiring complex competitive behaviors through self-play is still a challenge.

We address this research gap by proposing Deep Latent Competition (DLC), a novel model-based MARL method that operates on raw image inputs. While this method can be applied to a range of problems, we focus on demonstrating it in the context of two-player racing. Our approach learns a world model for imagining competitive behavior in latent-space. This allows for training agents via imagined self-play such that they can predict opponent behavior and incorporate the expected outcomes of action sequences in their policy selection. We further learn to predict the belief of other agents purely based on observations from the ego agent’s perspective. We validate this methodology on a novel multi-agent racing benchmark based on OpenAI Gym [8] that requires the application of continuous visual control policies. In summary, this work contains the following contributions:

- A novel model-based reinforcement learning algorithm for **learning competitive control policies** for multi-agent problems from raw image observations through self-play in latent space
- A multi-agent world model structure that allows for **(a) imagination of competing agents’ behavior** in a learned latent space and **(b) estimation of the beliefs of others** from own observations
- **Extensive evaluations** in a **new multi-agent racing benchmark** demonstrating superiority over approaches that do not reason about other agents in imagination

## 2 Related Work

**Autonomous racing and navigation** Most recent approaches considering autonomous racing assume knowledge of the underlying dynamics model and use machine learning based techniques for improving said model. This can be employed in conjunction with a variety of control approaches [1, 2, 3, 9]. Game-theoretic methods additionally explicitly model interactions with other agents for decision making [10, 4, 11, 5, 12, 6, 13]. These approaches usually do not operate on high-dimensional input spaces and often impose assumptions on the type of interactions. On the other hand, learning-based end-to-end navigation approaches [14, 15, 16, 17, 18] can operate directly on high-dimensional sensor data but typically involve no inductive biases for considering interactions.

**Multi-agent RL** Recently, MARL agents have surpassed human-level performance in many multi-agent environments including complex board and card games such as Go [19], chess, shogi [20], and Poker [21, 22]. MARL agents also reached grandmaster-level performance in the real-time strategy game Star Craft II [7, 23], and showed complex emergent tool use in hide-and-seek [24]. We are motivated by the success of recent algorithms for continuous control tasks in cooperative, competitive, and team competition environments [25, 26, 27, 28]. While these agents leveraged privileged information, such as entity lists, states, and maps, we present an agent that learns competitive strategies directly from raw image observations. A common thread in MARL are self-play auto-curricula [29, 30, 31, 32]. In contrast, we gain competitiveness from imagined self-play in a learned multi-agent latent world model.

**Latent imagination in RL** Use of neural networks, particularly recurrent neural networks, for modeling the evolution of the environment allowing for *"mental imagination"* has been proposed as early as 1990 [33] and recently revisited in [34]. In a similar spirit, variational inference approaches have been combined with the linear-quadratic-regulators for learning to control from raw images [35, 36]. Another line of recent algorithms combines latent (multi-step) imagination with video prediction [37, 38, 39], achieving state-of-the-art performance on several standard benchmarks. We draw inspiration from these ideas and generalize the concept of multi-step latent imagination to multiplayer settings.

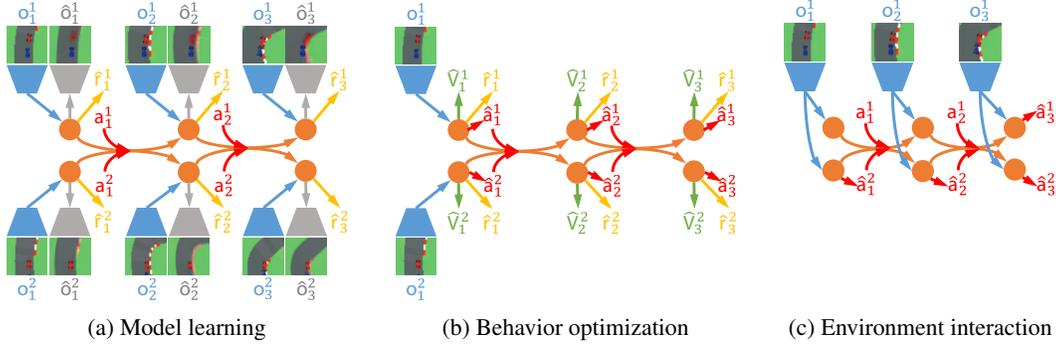


Figure 2: (a) The agent learns to encode observations into separate latent states for each agent based on reconstruction and reward prediction. The learned transition model propagates all agents’ latent states jointly. (b) Self-play in imagination: The agent predicts state values and optimizes actions that maximize future returns by propagating gradients back through imagined game trajectories. The agent’s competitiveness continuously improves through self-play. (c) The agent estimates the own and other agents’ current state based on an encoding of own historic observations only and predicts the actions of the other agent and the own actions to be executed in the environment. True observations and actions of other agents are not available.

### 3 Representing Multi-Agent World Models

We define competitive visual control in our racing domain as a MARL problem. A collection of agents interacts within the environment and learns to optimize their behavior in an effort to maximize individual cumulative reward. MARL is typically modelled as a Markov game [40, 41, 42], in which each agent solves a partially observable Markov decision process (POMDP) [43, 44, 45]. In the following, we first introduce the general problem definition, outline our approach to imagined self-play and introduce our representation learning approach.

**Problem formulation of competitive MARL** We define the POMDP of agent  $i \in \{1, \dots, n\}$  as the tuple  $M^i = \langle S, A^i, T, \Omega^i, O^i, R^i \rangle$ , where  $S$  denotes the set of environment states,  $A^i$  the continuous action set, and  $T : S \times A^1 \times \dots \times A^n \rightarrow \Pi(S)$  the corresponding transition function with associated probability distribution  $\Pi(\cdot)$ . For each agent  $i$ , we furthermore define a reward function  $R^i : S \times A^i \rightarrow \mathbb{R}$ , as well as an observation set  $\Omega^i$  with corresponding observation function  $O^i : S \times A^i \rightarrow \Pi(\Omega^i)$ . In the following, we consider a homogeneous set of agents with identical action and observation spaces as well as reward functions, such that  $X^i = X$  for  $X = \{A, \Omega, O, R\}$ .

We do not assume prior knowledge about the environment. Thus, the nominal reward function  $R$ , state transition function  $T$ , and observation function  $O$  are unknown. Let  $q_\phi(a_t^i | o_{<t}^i, a_{<t}^i)$  denote the policy of agent  $i$ , conditioned only on its own observation-action history, and define the associated expected return over a race of duration  $T$  to be  $\mathbb{E} \sum_{t=1}^T r_t^i$ . The objective is then to develop an agent that maximizes the expected return in the absence of prior knowledge about nominal environment and opponent behavior. This defines an extensive-form game as the agents are competing for reward over multiple timesteps, while only receiving instantaneous environment feedback via high-dimensional observations  $o_t^i$  and scalar rewards  $r_t^i$ .

**Learning through imagined self-play** Model-based reinforcement learning consists of the tasks of (a) model learning, (b) behavior optimization, and (c) environment interaction. Model-based MARL extends model learning to include predictions of other agents’ behavior, while behavior optimization needs to account for competitive fitness. The training proceeds centralized. While the learning algorithm has access to the observation-action histories of all agents, the deployment is decentralized, providing each agent only with their individual observation-action history. As detailed in Figure 2, our algorithm iteratively executes the following:

- Learning a world model consisting of the joint dynamics and reward function based on previous experience of all agents, see Figure 2a. Learning to predict how the world evolves conditioned

on own and expected opponent actions enables each agent to imagine the outcome of games without requiring additional real-world experience.

- Learning action and value function models through policy iteration on imagined model rollouts. The agents interact with their adversaries through imagined self-play and acquire increasingly competitive behaviors without the necessity for execution in the real world, see Figure 2b.
- Competing in the real world to collect novel experience and judge the performance of the current behavior. Each agent only has access to their own observation-action history and performance indirectly depends on how well the states of opponents are being estimated, see Figure 2c.

**Representing multi-agent world models** Leveraging a multi-agent world model accelerates learning through imagined self-play. The agent optimizes its behavior by simulating interactions with its opponents in the environment without execution in the real world. In contrast to single-agent world models [35, 36, 46], this requires explicit representation of the state and action of each agent, as well as a mechanism for an agent to predict behavior of another agent. In the following, we will consider a two-player game ( $n = 2$ ) and extend the representation model formulation provided in [38] to yield

$$\begin{aligned}
\text{Representation model:} & \quad p_\theta (s_t^1, s_t^2 | s_{t-1}^1, s_{t-1}^2, a_{t-1}^1, a_{t-1}^2, o_t^1, o_t^2) \\
\text{Transition model:} & \quad q_\theta (s_t^1, s_t^2 | s_{t-1}^1, s_{t-1}^2, a_{t-1}^1, a_{t-1}^2) \\
\text{Encoder model:} & \quad q_\theta (z_t^i, \tilde{z}_t^{-i} | o_t^i) \\
\text{Observation model:} & \quad q_\theta (o_t^i | s_t^i) \\
\text{Reward model:} & \quad q_\theta (r_t^i | s_t^i)
\end{aligned} \tag{1}$$

where  $p$  and  $q$  denote distributions in latent space, with  $\theta$  as their joint parameterization. The representation model encodes observations  $(o_t^1, o_t^2)$  into Markovian model states  $(s_t^1, s_t^2)$  [38], which are propagated under a joint transition function to predict future model states  $(s_{t+1}^1, s_{t+1}^2)$ . We explicitly provide the underlying encoder model to emphasize that each agent not only learns an embedding corresponding to its own viewpoint,  $z_t^i$ , but additionally learns to predict embeddings of the opponent,  $\tilde{z}_t^{-i}$ . This is crucial during deployment, as ground truth observations and actions of opponents are not available. Instead, we leverage our predicted embeddings  $\tilde{z}_t^{-i}$  in conjunction with an action model  $q_\phi (a_t^{-i} | s_t^{-i})$  and employ a slightly modified representation model  $p_\theta (s_t^i, s_t^{-i} | s_{t-1}^i, s_{t-1}^{-i}, a_{t-1}^i, a_{t-1}^{-i}, o_t^i)$  that is only conditioned on the observation of the ego agent,  $o_t^i$ . The encoder model is part of the representation model and embeds observations into embedding states based on which model states are then generated. For each agent, we furthermore define an individual observation model  $q_\theta (o_t^i | s_t^i)$  and reward model  $q_\theta (r_t^i | s_t^i)$ . The underlying architectures follow [46, 38], where the transition model is represented by a recurrent state space model (RSSM), the encoder and observation models by a convolutional neural network (CNN) and transposed CNN, respectively, and the reward model by a dense neural network. Training of these models then proceeds centralized, such that the learning algorithm is given access to the interaction histories of each agent.

## 4 Learning to Compete by Imagined Self-Play

Our proposed algorithm learns a world model from ground truth data, based on which behavior is refined through imagined self-play. In the following, we introduce the objectives for the two stages.

**Representation learning** The representation learning objective combines image reconstruction with reward prediction in order to discover latent spaces that not only offer compact representations of environment states but further facilitate prediction of associated trajectory performance. Because agents are actively competing for reward, their states are propagated jointly through the transition model with each agent learning to predict relevant opponent states. The observation model then not only provides reconstruction signals for the ego perspective via the true ego state  $s_t^i$ , but also for the opponent perspective via the predicted opponent state  $\tilde{s}_t^{-i}$ . Following [38], the models introduced in Eq. (1) are optimized to maximize a reformulation of their variational lower bound objective:

$$\begin{aligned}
J_{M,\hat{s}} &= \mathbb{E}_{\mathcal{D}} (\sum_t (J_{O,t} + J_{R,t} + J_{D,t})) \\
J_{O,t} &= \ln q(o_t^1 | \hat{s}_t^1) + \ln q(o_t^2 | \hat{s}_t^2) \\
J_{R,t} &= \ln q(r_t^1 | \hat{s}_t^1) + \ln q(r_t^2 | \hat{s}_t^2) \\
J_{D,t} &= -\beta \text{KL}(p(\hat{s}_t^1, \hat{s}_t^2 | \hat{s}_{t-1}^1, \hat{s}_{t-1}^2, a_{t-1}^1, a_{t-1}^2, o_t^1, o_t^2) \parallel q(\hat{s}_t^1, \hat{s}_t^2 | \hat{s}_{t-1}^1, \hat{s}_{t-1}^2, a_{t-1}^1, a_{t-1}^2)),
\end{aligned} \tag{2}$$

where the general model state  $\hat{s}$  may either originate from ground truth embeddings  $s$  or their predictions  $\tilde{s}$ . In practice, we optimize a linear combination of  $J_{M,\hat{s}}$  with  $\hat{s} = \{(s_t^1, s_t^2), (s_t^1, \tilde{s}_t^2), (\tilde{s}_t^1, s_t^2)\}$ . This enables learning of latent representations conducive to solving the task via the ground truth embeddings, while constraining predicted embeddings to sensible representations within that space.

**Behavior learning** The behavior learning objective optimizes for competitive fitness by maximizing the expected return of the action model  $q_\phi(a_t^i | s_t^i)$  over a  $T$ -step race. The agent can imagine outcomes of potential interaction sequences by leveraging the learned transition model, therefore bypassing execution in the real world through imagined self-play. Generating entire race sequences can be computationally prohibitive and we follow [38] in complementing finite horizon model rollouts with predictions from a value model  $v_\psi(s_t^i)$  in order to approximate returns corresponding to an extensive form race. The action and value model are then trained jointly using policy iteration on the objectives

$$\max_{\phi} \mathbb{E}_{q_\theta, q_\phi} \left( \sum_{\tau=t}^{t+H} V_\lambda(s_\tau) \right), \quad \min_{\psi} \mathbb{E}_{q_\theta, q_\phi} \left( \sum_{\tau=t}^{t+H} \|v_\psi(s_\tau) - V_\lambda(s_\tau)\|^2 \right), \quad (3)$$

where  $V_\lambda(s_\tau)$  represents an exponentially recency-weighted average of the  $k$ -step value estimates  $V_N^k(s_\tau)$  to stabilize the learning [45]. We provide the corresponding value function definitions as

$$\begin{aligned} V_\lambda(s_\tau) &= (1 - \lambda) \sum_{n=1}^{H-1} \lambda^{n-1} V_N^n(s_\tau) + \lambda^{H-1} V_N^H(s_\tau), \\ V_N^k(s_\tau) &= \mathbb{E}_{q_\theta, q_\phi} \left( \sum_{n=\tau}^{h-1} \gamma^{n-\tau} r_n + \gamma^{h-\tau} v_\psi(s_h) \right), \end{aligned} \quad (4)$$

where  $h = \min(\tau + k, t + H)$ . This process leverages imagined trajectories  $\{(s_\tau^1, s_\tau^2, a_\tau^1, a_\tau^2)\}_{\tau=t}^{t+H}$  over a horizon of  $H$  starting from each timestep  $t$  within the sampled batch sequence, where opponent behavior is estimated by querying the action model with predicted latent viewpoints as  $q_\phi(a_t^{-i} | \tilde{s}_t^{-i})$ . The resulting algorithm optimizes policies by back-propagating analytic value gradients of imagined self-play trajectories through the learned multi-agent world model.

**Deep latent competition** The resulting algorithm is provided as pseudocode in Algorithm 1. It runs for  $K$  episodes and proceeds in two phases: during the online phase, data is collected from a  $T$ -step race with each agent only having access to their respective observations and relying on predicted opponent behavior. During the offline phase, representation learning and policy iteration propagate information into the world model and the action model based on the interaction histories of all agents. To this end, batch sequences of length  $L$  are sampled from replay memory  $\mathcal{D}$  to serve as targets for representation learning. The behavior is then refined in simulation based on rollout trajectories of length  $H$  starting from the ground truth samples which are used in generating the value estimates according to Eq. 4. Here, we set the underlying parameters to  $T = 1000$ ,  $L = 50$  and  $H = 15$ . All models are then optimized on the previous objectives with the Adam optimizer [47].

---

**Algorithm 1:** Deep Latent Competition (DLC)

---

```

1 Initialize : model parameters  $\{\theta, \psi, \phi\}$  randomly; memory  $\mathcal{D}$  with 5 random episodes
2 for episode  $k \leftarrow 1$  to  $K$  do
3   for timestep  $t \leftarrow 1$  to  $T$  do
4     Observe  $o_t^1$  and predict embeddings  $z_t^1, \tilde{z}_t^2 \sim q_\theta(z_t^1, \tilde{z}_t^2 | o_t^1)$ 
5     Propagate states  $s_t^1, \tilde{s}_t^2 \sim p_\theta(s_t^1, \tilde{s}_t^2 | s_{t-1}^1, \tilde{s}_{t-1}^2, a_{t-1}^1, \tilde{a}_{t-1}^2, z_t^1, \tilde{z}_t^2)$ 
6     Generate action  $a_t^1 \sim q_\phi(a_t^1 | s_t^1)$ , predicted response  $\tilde{a}_t^2 \sim q_\phi(\tilde{a}_t^2 | \tilde{s}_t^2)$ 
7     Execute  $a_t^1$  in the environment
8     Add episode transitions  $\{(o_t^1, o_t^2, a_t^1, a_t^2, r_t^1, r_t^2)\}_{t=1}^T$  to memory  $\mathcal{D}$ 
9     for trainstep  $s \leftarrow 1$  to  $S$  do
10      Sample batch of sequences  $\{(o_t^1, o_t^2, a_t^1, a_t^2, r_t^1, r_t^2)\}_{t=b}^{b+L} \sim \mathcal{D}$ 
11      Use the encoder model to predict embeddings  $z_t^1, \tilde{z}_t^1, z_t^2, \tilde{z}_t^2$ 
12      Use the representation model to predict states  $s_t^1, \tilde{s}_t^1, s_t^2, \tilde{s}_t^2$ 
13      Update  $\theta$  via representation learning on  $\{(s_t^1, s_t^2), (s_t^1, \tilde{s}_t^2), (\tilde{s}_t^1, s_t^2)\}$ 
14      Compute value estimates  $V_\lambda^1(s_\tau^1), V_\lambda^2(s_\tau^2) \leftarrow \text{rollout}(s_t^1, s_t^2, H)$ 
15      Update  $\phi$  and  $\psi$  based on Eq. (3) for all targets  $\{V_\lambda^1(s_\tau^1), V_\lambda^2(s_\tau^2)\}$ 

```

Online

Model Update

Policy Update

Offline

---

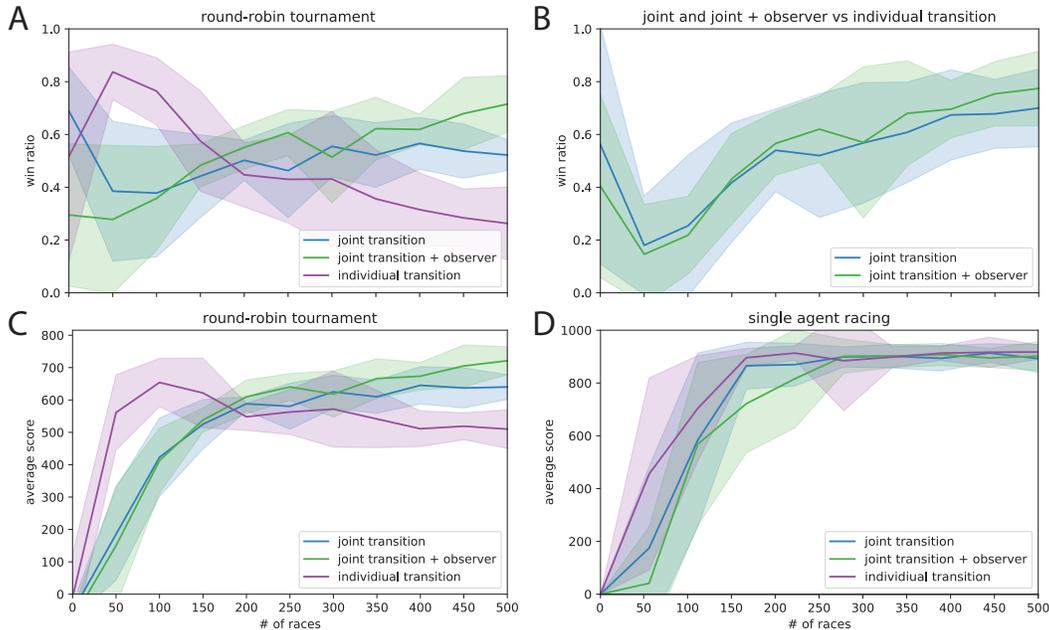


Figure 3: **A**: Win ratio of all agents in a round-robin tournament. **B**: Agents with joint and joint + observer transition compete directly against an agent with individual transition function. **C**: Average score of all agents in a round-robin tournament. **D**: Single agent racing performance disentangles general skill learning from learning to compete through interaction.

## 5 Latent Racing Experiments

We demonstrate the ability of DLC to learn competitive visual control policies in a novel multi-agent racing environment, and compare performance against baselines to highlight the importance of both the joint transition model and the learned observer. We further highlight the learned representation model’s capability of predicting opponent viewpoints from ego observations.

**Racing environment** We propose `MultiCarRacing-v0`, a novel multi-agent racing environment for learning competitive visual control policies<sup>1</sup>. The environment extends the Gym task `CarRacing-v0` [8] and provides each agent with top-down 96x96 pixel image observations from their ego perspective based on which continuous control inputs need to be selected. The viewpoint is motivated by recent results in the context of driving [48, 49, 50] and holds the promise of future deployment on physical platforms. The environment allows for differentiating between skillful and competitive driving. While the former is the basis for high-performance racing, learning to beat a skillful opponent is a far greater challenge. Interactions between agents are sparse but information-rich: only when agents collide, push, or block each other do they directly impact each others state.

**Dynamics and rewards** The vehicles in the environment exhibit slip and collision dynamics. Breaking or accelerating too hard induces skidding and in combination with steering causes substantial understeering or oversteering (drifting). Similarly, moving off the track lowers available friction. Collision dynamics allow for elaborate interaction strategies during the race: pushing other agents off the track, blocking overtaking attempts, or turning an opponent sideways via a PIT maneuver. We show examples in Figure 1. The reward mapping follows `CarRacing-v0`, in that each agent incurs a loss of  $-0.1$  per timestep and receives a reward for each visited tile along the track. We incentivize competition by discounting rewards based on visitation order: the first agent to visit a track tile is rewarded with  $+1000/N$ , while the second agent receives  $+500/N$  (on an  $N$ -tile track).

<sup>1</sup>Code available at [https://github.com/igilitschenski/multi\\_car\\_racing](https://github.com/igilitschenski/multi_car_racing)

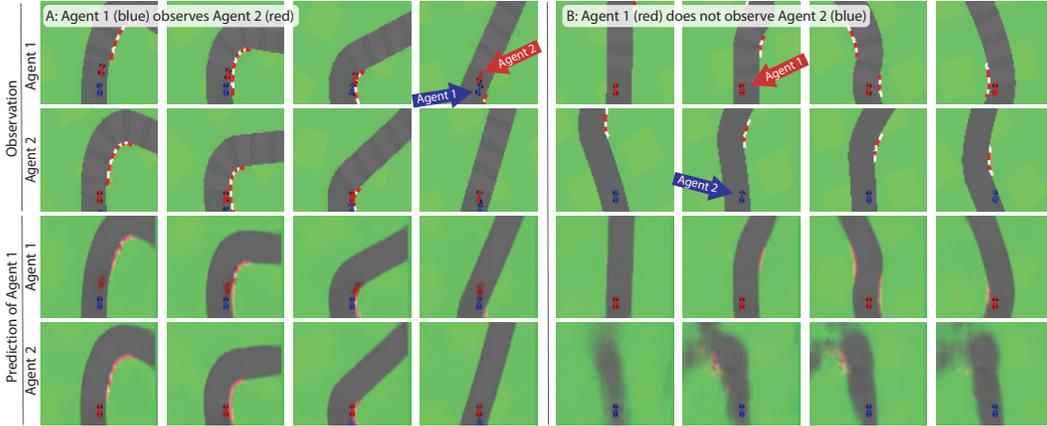


Figure 4: **Closed loop prediction**: **A**: Agent 1 observes the opponent and is therefore able to reconstruct both views. **B**: 1 is unable to observe 2 and reconstructs their view with high uncertainty.

**Benchmarking performance** The algorithm presented in this work, DLC, learns a multi-agent world model that enables imagined self-play by combining the underlying transition function with an observer capable of predicting opponent latent states. We refer to this method as *joint transition + observer* and compare performance against two baselines. The first is *joint transition*, which propagates ground truth latent states of both agents and allows for assessing performance of the observer. The second is *individual transition*, which propagates ground truth latent states individually and highlights the added value of a joint transition model. Each method is trained on 500 races. We evaluate the resulting performance in a round-robin tournament of 100 races for each pairing (300 races per tournament) at multiple stages of training progress. We repeat this for 5 random seeds.

Figure 3A shows the resulting win-ratio and Figure 3B the average score for the round-robin tournament. The representation learning problem for *individual transition* is easier as the learned transition function does not need to disentangle how the states and actions of both agents affect the transition. Therefore, the *individual transition* baseline starts off strong. However, after 200 races both joint transition methods have learned to compete more effectively through imagined self-play. The *individual transition* baseline can not leverage this effect, as agent actions do not affect opponents states during imagined rollouts and self-play may only occur in the real world. As visible in Figure 3C, after 500 races both joint transition approaches win 70-80% of races against the *individual transition* baseline. Furthermore, both joint transition methods perform similarly, suggesting that the learned observer is capable of predicting opponent latent states in a way that is sufficient in order to induce learning of competitive behaviors.

The single agent racing performance, see Figure 3D, further allows us to disentangle general skill learning from learning to compete. It confirms that the *individual transition* baseline is able to acquire general racing skills faster. Similarly, the joint transition methods do not outperform the *individual transition* baseline in the single agent racing case, such that their performance increase for the multi-agent setting in Figure 3A can be explained by their increased ability to compete.

**Predicting the opponents latent state from ego observations** Given an agent’s own history of observations and controls, we can estimate the compact latent state and it’s associated reconstruction to visualize the representation model’s understanding of the world. More interestingly, we can create a reconstruction of the predicted opponent’s latent state to visualize how well the agent can infer the opponent view. Given a predicted opponent state we can predict their actions based on the learned policy. Figure 4 provides two scenarios with reconstructions based only on observations of agent 1.

In Figure 4A, the opponent is within the field of view of agent 1. The reconstructions are focused and close to the ground truth for both the ego and opponent viewpoints, indicating that the latent state is accurate and well understood. In Figure 4B, agent 1 never observes its opponent and is unable to accurately predict their view. However, instead of predicting only noise, agent 2 is imagined to drive on a straight road with the possibility of an upcoming turn. This is a sensible prediction given that agent 2 is currently not observable and in turn will not immediately affect the motion plan of agent 1.

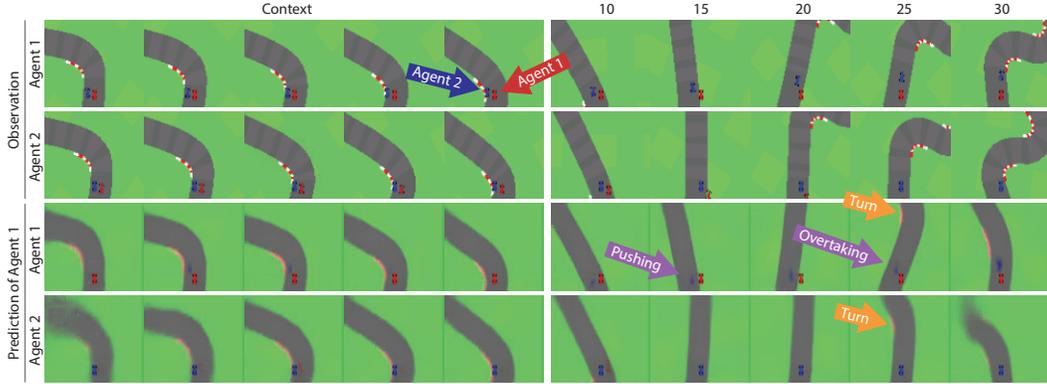


Figure 5: **Open loop prediction:** Agent 1 (red) receives 5 observation frames as context and predicts the own view and the view of agent 2 (blue). We employ the learned dynamics to forward propagate 25 additional timesteps into the future without any further observations.

**Imagining interaction sequences** To facilitate efficient self-play in a learned world model, we require all agent states to be jointly propagated in a consistent manner. In Figure 5, we provide 5 observations of agent 1 as context and investigate the model’s ability to predict forward in time for 25 additional timesteps. Similarly to Figure 4, we exclude ground-truth observations of agent 2. While actions of agent 1 are available for the full horizon, the learned policy predicts the actions of agent 2.

Referring to the context frames, we observe the benefit of recursively estimating states in agent 1’s reconstruction of agent 2’s viewpoint: while the reconstruction has high uncertainty in the first frame, accuracy improves with every new observation. Agent 1’s reconstruction of their ego view is detailed from the start with the blue agent remaining blurry. Moving beyond the context frames, the predicted views quickly diverge from the ground truth. This is expected as the agent is unable to see beyond the first turn. In the following imagination, the blue agent overtakes the red agent by forcing them off the track (Figure 5 prediction of agent 1, timestep 15). This is reasonable, as the blue agent leaves the left turn on the inside with an opportunity to push the red agent off the road. The imagined sequences feature detailed predictions of the track, including markings on an upcoming left turn (timestep 25).

Most importantly, the predictions of both agents remain consistent. The two agents’ relative positions with each other and the track are in correspondence along the full horizon. Likewise, the predicted characteristics of a track including curbs and a left turn are consistent between both agents’ predicted views (timestep 25). Consistency is crucial as it allows to learn from imagined self-play in a world model. If the two predictions were to diverge, the impact of the ego actions on another agent’s behavior could not be estimated. The outcome of the imagined games would be uncertain. The joint transition model helps keeping predictions consistent, as it allows for information exchange between both agents’ latent states during forward propagation.

## 6 Conclusion

We present Deep Latent Competition (DLC), a novel reinforcement learning algorithm that learns competitive visual control policies through self-play in imagination. The DLC agent can imagine interaction sequences in the compact latent space based on a multi-agent world model that combines a joint transition function with opponent viewpoint prediction. The behavior is then optimized by back-propagating the analytic value gradients of these imagined game trajectories through the learned world model. Experiments in a novel continuous visual control racing environment demonstrate that the DLC agent learns to make consistent multi-agent forward predictions. Optimizing competitive behaviors through imagined self-play based on these joint predictions yields an agent that performs superior to an agent that propagates ground truth observations separately. In the future, we aim to deploy the DLC agent on hardware platforms to yield competitive racecar driving in the real world. Extending the framework to include more complex game-theoretic considerations in the forward predictions offers another intriguing avenue for future work.

## Acknowledgments

This work was supported in part by Qualcomm and Toyota Research Institute (TRI). This article solely reflects the opinions and conclusions of its authors and not TRI, Toyota, or any other entity. We thank them for their support. The authors further would like to acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC resources. We would also like to thank Danijar Hafner for open-sourcing the Dreamer agent.

## References

- [1] E. Alcalá, V. Puig, J. Quevedo, and U. Rosolia. Autonomous racing using Linear Parameter Varying-Model Predictive Control (LPV-MPC). *Control Engineering Practice*, 95, 2020. 1, 2
- [2] J. V. Carrau, A. Liniger, X. Zhang, and J. Lygeros. Efficient implementation of Randomized MPC for miniature race cars. In *European Control Conference (ECC)*, 2016. 1, 2
- [3] A. Liniger and J. Lygeros. Real-Time Control for Autonomous Racing Based on Viability Theory. *IEEE Transactions on Control Systems Technology*, 27(2):464–478, 2019. 1, 2
- [4] R. Spica, E. Cristofalo, Z. Wang, E. Montijano, and M. Schwager. A Real-Time Game Theoretic Planner for Autonomous Two-Player Drone Racing. *IEEE Transactions on Robotics*, pages 1–15, 2020. 1, 2
- [5] Z. Wang, T. Taubner, and M. Schwager. Multi-agent sensitivity enhanced iterative best response: A real-time game theoretic planner for drone racing in 3D environments. *Robotics and Autonomous Systems*, 125, 2020. 1, 2
- [6] A. Liniger and J. Lygeros. A Noncooperative Game Approach to Autonomous Racing. *IEEE Transactions on Control Systems Technology*, 28(3):884–897, 2020. 1, 2
- [7] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. 1, 2
- [8] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint: 1606.01540*, 2016. 1, 5
- [9] W. Schwarting, J. Alonso-Mora, and D. Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018. 2
- [10] W. Schwarting, A. Pierson, S. Karaman, and D. Rus. Stochastic Dynamic Games in Belief Space. *arXiv preprint: 1909.06963*, 2019. 2
- [11] Z. Wang, R. Spica, and M. Schwager. Game Theoretic Motion Planning for Multi-robot Racing. In *International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2019. 2
- [12] G. Williams, B. Goldfain, P. Drews, J. M. Rehg, and E. A. Theodorou. Autonomous Racing with AutoRally Vehicles and Differential Games. *arXiv preprint: 1707.04540*, jul 2017. 2
- [13] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus. Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences*, 116(50):24972–24978, 2019. 2
- [14] D. A. Pomerleau. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1989. 2
- [15] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus. Learning Robust Control Policies for End-to-End Autonomous Driving From Data-Driven Simulation. *Robotics and Automation Letters*, 5(2), 2020. 2
- [16] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Proceedings of Robotics: Science and Systems (RSS)*, 2019. 2
- [17] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint:1604.07316*, 2016. 2

- [18] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. Learning to drive in a day. *arXiv preprint 1807.00412*, 2018. 2
- [19] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017. 2
- [20] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018. 2
- [21] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017. 2
- [22] N. Brown and T. Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019. 2
- [23] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, and R. Tsing. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv preprint: 1708.04782*, 2017. 2
- [24] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from multi-agent autotutorials. In *International Conference on Learning Representations*, 2019. 2
- [25] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell. Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019. 2
- [26] S. Iqbal and F. Sha. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2019. 2
- [27] S. Liu, G. Lever, N. Heess, J. Merel, S. Tulyasuvunakool, and T. Graepel. Emergent coordination through competition. In *International Conference on Learning Representations (ICLR)*, 2019. 2
- [28] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations*, 2018. 2
- [29] E. Lockhart, M. Lanctot, J. Pérolat, J.-B. Lespiau, D. Morrill, F. Timbers, and K. Tuyls. Computing approximate equilibria in sequential adversarial games by exploitability descent. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019. 2
- [30] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Perolat, D. Silver, and T. Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2
- [31] J. Heinrich, M. Lanctot, and D. Silver. Fictitious self-play in extensive-form games. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. 2
- [32] J. Heinrich and D. Silver. Deep Reinforcement Learning from Self-Play in Imperfect-Information Games. *arXiv preprint: 1603.01121*, 2016. 2
- [33] J. Schmidhuber. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *International Joint Conference on Neural Networks (IJCNN)*, 1990. 2
- [34] D. Ha and J. Schmidhuber. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2, A, A
- [35] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015. 2, 3
- [36] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine. SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning. In K. Chaudhuri and R. Salakhutdinov, editors, *International Conference on Machine Learning (ICML)*, 2019. 2, 3
- [37] Ł. Kaiser, M. Babaeizadeh, P. Milos, B. Osinowski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, A. Mohiuddin, R. Sepassi, G. Tucker, and H. Michalewski. Model Based Reinforcement Learning for Atari. In *International Conference on Learning Representations (ICLR)*, 2020. 2

- [38] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 2, 3, 3, 4, 4, A, A, A, A
- [39] T. Seyde, W. Schwarting, S. Karaman, and D. Rus. Learning to plan optimistically: Uncertainty-guided deep exploration via latent model ensembles. *arXiv preprint:2010.14641*, 2020. 2
- [40] L. S. Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953. 3
- [41] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 1994. 3
- [42] P. J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005. 3
- [43] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998. 3
- [44] M. Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of artificial intelligence research*, 13:33–94, 2000. 3
- [45] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 3, 4
- [46] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018. 3, 3, A
- [47] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint:1412.6980*, 2014. 4, A
- [48] P. Drews, G. Williams, B. Goldfain, E. A. Theodorou, and J. M. Rehg. Aggressive Deep Driving: Combining Convolutional Neural Networks and Model Predictive Control. In *Conference on Robot Learning (CoRL)*, 2017. 5
- [49] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by Cheating. In *Conference on Robot Learning (CoRL)*, 2020. 5
- [50] K. Mani, S. Daga, S. Garg, S. S. Narasimhan, M. Krishna, and K. M. Jatavallabhula. MonoLayout: Amodal scene layout from a single image. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020. 5
- [51] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018. A

## A Network Architectures

Layer Type	Input (dimensions)	Output (dimensions)	Additional Parameters
<b>Transition model (<i>imagine 1-step</i>)</b>			
Dense	$s_{\tau-1,s}$ (60), $a_{\tau-1}$ (6)	$fc_{t,i}^1$ (600)	a=ELU
GRU	$fc_{t,i}^1$ (600), $s_{\tau-1,d}$ (400)	$rs_{\tau}$ (400), $s_{\tau,d}$ (400)	a=tanh
Dense	$rs_{\tau}$ (400)	$fc_{t,i}^2$ (600)	a=ELU
Dense	$fc_{t,i}^2$ (600)	$\mu_{\tau,s}^{prior}$ (60), $\sigma_{\tau,s}^{prior}$ (60)	a=None
<b>Transition model (<i>observe 1-step</i>)</b>			
Dense	$s_{\tau,d}$ (400), $z_{\tau}$ (2048)	$fc_{t,o}^1$ (600)	a=ELU
Dense	$fc_{t,o}^1$ (600)	$\mu_{\tau,s}^{post}$ (60), $\sigma_{\tau,s}^{post}$ (60)	a=None
<b>Encoder model</b>			
Conv2D	obs (96, 96, 3)	cv1 (31, 31, 32)	a=ReLU, s=3, k=(4,4)
Conv2D	cv1 (31, 31, 32)	cv2 (14, 14, 64)	a=ReLU, s=2, k=(4,4)
Conv2D	cv2 (14, 14, 64)	cv3 (6, 6, 128)	a=ReLU, s=2, k=(4,4)
Conv2D	cv3 (6, 6, 128)	cv4 (2, 2, 256)	a=ReLU, s=2, k=(4,4)
<b>Observation model</b>			
Dense	$s_{\tau,d}^i$ (200), $s_{\tau,s}^i$ (30)	$fc_o^1$ (1, 1, 1024)	a=None
Deconv2D	$fc_o^1$ (1, 1, 1024)	dc1 (5, 5, 128)	a=ReLU, s=2, k=(5,5)
Deconv2D	dc1 (5, 5, 128)	dc2 (13, 13, 64)	a=ReLU, s=2, k=(5,5)
Deconv2D	dc2 (13, 13, 64)	dc3 (31, 31, 32)	a=ReLU, s=2, k=(6,6), p=1
Deconv2D	dc3 (31, 31, 32)	dc4 (96, 96, 3)	a=ReLU, s=3, k=(6,6)
<b>Reward model</b>			
Dense	$s_{\tau,d}^i$ (200), $s_{\tau,s}^i$ (30)	$fc_r^1$ (400)	a=ELU
Dense $\times$ 1	$fc_r^{\{1\}}$ (400)	$fc_r^{\{2\}}$ (400)	a=ELU
Dense	$fc_r^2$ (400)	$fc_r^3$ (1)	a=ELU
<b>Value model</b>			
Dense	$s_{\tau,d}^i$ (200), $s_{\tau,s}^i$ (30)	$fc_v^1$ (400)	a=ELU
Dense $\times$ 2	$fc_v^{\{1,2\}}$ (400)	$fc_v^{\{2,3\}}$ (400)	a=ELU
Dense	$fc_v^3$ (400)	$fc_v^4$ (1)	a=ELU
<b>Action model</b>			
Dense	$s_{\tau,d}^i$ (200), $s_{\tau,s}^i$ (30)	$fc_a^1$ (400)	a=ELU
Dense $\times$ 3	$fc_a^{\{1,2,3\}}$ (400)	$fc_a^{\{2,3,4\}}$ (400)	a=ELU
Dense	$fc_a^4$ (400)	$\mu_a$ (3), $\sigma_a$ (3)	a=ELU

Table 1: General network architectures of the underlying models. The transition model is joint in our implementation and the associated input and output variables correspond to joint representations (e.g. the previous action  $a_{\tau-1}$  (6) can be interpreted as  $a_{\tau-1}$  ( $2 \times 3$ )). We note that repeated layers have been condensed with Dense  $\times i$  referring to application of the same dense layer architecture  $i$  times. The employed parameter abbreviations are referring to: a=activation, k=kernel, p=padding, s=stride.

Based on the general network architectures provided in Table 1, we comment on how the two parts of the transition model integrate with each other and provide further details on each of the models.

**Transition model** The joint transition model follows the recurrent state space model (RSSM) architecture presented in [46, 38]. The RSSM is extended to the multi-agent setting and propagates joint model states consisting of a deterministic and a stochastic component, respectively denoted by  $s_{t,d}$  and  $s_{t,s}$  at time  $t$ . The stochastic component  $s_{t,s}$  is implemented via a diagonal Gaussian distribution and its derivation is provided in the next section. The transition model then predicts priors for the associated mean and standard deviation based on the previous model state and applied action (*imagine 1-step*). In the presence of observations, prior estimates can be updated to posterior estimates (*observe 1-step*). The transition model may then initialize its states by propagating posteriors based on a context sequence (*imagine 1-step and observe 1-step*) from which interactions can be imagined by propagating prior estimates (*imagine 1-step*).

**Encoder model** The encoder parameterization follows the architectural choices presented in [34], where we adapt the convolutional layers to match the dimensionality of our observations. The agent leverages two encoders in parallel, one for generating latent vectors of the ego perspective and one for predicting latent vectors of the opponent perspective. Inputs are  $96 \times 96$  RGB image observations.

**Observation model** The observation model follows the decoder architecture presented in [34], where we adapt the transposed convolutional layers to match the dimensionality of our observations. The image observations of agent  $i$  are reconstructed from the associated model states  $s_{\tau,d}^i$  and  $s_{\tau,s}^i$ .

**Reward and value model** Rewards and values of agent  $i$  are both predicted as scalar values from fully-connected networks that operate on the associated model states  $s_{\tau,d}^i$  and  $s_{\tau,s}^i$ , similar to [38].

**Action model** The action model follows [38], where the predicted mean  $\mu_a$  is rescaled and passed through a tanh to allow for saturated action distributions. It is combined with a softplus standard deviation based on  $\sigma_a$  and the resulting Normal distribution is again squashed using a tanh [38, 51].

**Order independence** We note as an implementation detail that the transition distribution  $q_\theta(s_t^1, s_t^2 | s_{t-1}^1, s_{t-1}^2, a_{t-1}^1, a_{t-1}^2)$  should be independent of the order in which agents are provided. Thus, reversing the ordering and considering  $q_\theta(s_t^2, s_t^1 | s_{t-1}^2, s_{t-1}^1, a_{t-1}^2, a_{t-1}^1)$  should yield the same distribution. We achieve this by two passes through the learned transition model and subsequent averaging. In the second forward pass the agents’ input order to the transition model is flipped.

**Training parameters** Most of the training parameters correspond to the implementation of [38], a state-of-the-art model-based RL algorithm for learning to plan in latent-space from image observations. DLC trains every 1000 environment steps for 200 iterations with the Adam optimizer [47]. The batch size is set to 50. The representation, value and actor model are respectively trained with learning rates  $6e-4$ ,  $6e-4$ , and  $8e-5$ . Gradients over the magnitude of 100 are clipped for all models. The prior  $\sigma_{\tau,s}^{prior}$  and posterior  $\sigma_{\tau,s}^{post}$  variance in the transition model are bounded from below to a minimum value of 0.1. The model loss on true observations  $J_{M,s_t^1,s_t^2}$  is weighted twice as much as the model losses on predicted opponent observations  $J_{M,s_t^1,\bar{s}_t^2}$  and  $J_{M,\bar{s}_t^1,s_t^2}$ . Throughout, we use  $\gamma = 0.99$  and  $\lambda = 0.95$ . The model learning horizon is  $L = 50$  whereas the imagination horizon is  $H = 15$ . Value and action models are trained on the same trajectory rollouts.

**Environment details** To ensure generalization, we randomized both the color and initial position of all vehicles throughout training. We also evaluated (but did not use) penalization for driving in the backward direction which can occur after returning from a spin on the grass.

## B Derivations

Our representation model is a latent variable model. That is, it can be written as

$$\begin{aligned} p_\theta(S_t | S_{t-1}, A_{t-1}, O_t) &= p_\theta(s_t^1, s_t^2 | s_{t-1}^1, s_{t-1}^2, a_{t-1}^1, a_{t-1}^2, o_t^1, o_t^2) \\ &= \int \int p_\theta(s_t^1, s_t^2 | s_{t-1}^1, s_{t-1}^2, a_{t-1}^1, a_{t-1}^2, z_t^1, z_t^2) \\ &\quad \cdot p(z_t^1 | o_t^1) \cdot p(z_t^2 | o_t^2) dz_t^1 dz_t^2, \end{aligned}$$

where  $S_t := \{s_t^1, s_t^2\}$  is the concatenation of both states and  $A_{t-1}, O_{t-1}$  are defined analogously. Each agent  $i$  maintains its own belief of the world, without having access to both observations. Thus, at test time it computes instead

$$p_\theta(S_t|S_{t-1}, A_{t-1}, o_t^i) = \int \int p_\theta(s_t^1, s_t^2 | s_{t-1}^1, s_{t-1}^2, a_{t-1}^1, a_{t-1}^2, z_t^i, \tilde{z}_t^{-i}) \cdot q_\theta(z_t^i, \tilde{z}_t^{-i} | o_t^i) dz_t^i d\tilde{z}_t^{-i} .$$

Therefore, we never need to learn  $p(z_t^i | o_t^i)$ . Instead we directly learn the representation  $q_\theta(z_t^i, \tilde{z}_t^{-i} | o_t^i)$ . This is implemented as a convolutional encoder  $f_E(o^i)$  making  $q_\theta(z_t^i, \tilde{z}_t^{-i} | o_t^i)$  a Dirac distribution

$$q_\theta(z_t^i, \tilde{z}_t^{-i} | o_t^i) = \delta((z_t^i, \tilde{z}_t^{-i}) - f_E(o_t^i)) .$$

## C Additional Visualizations of Common Scenarios

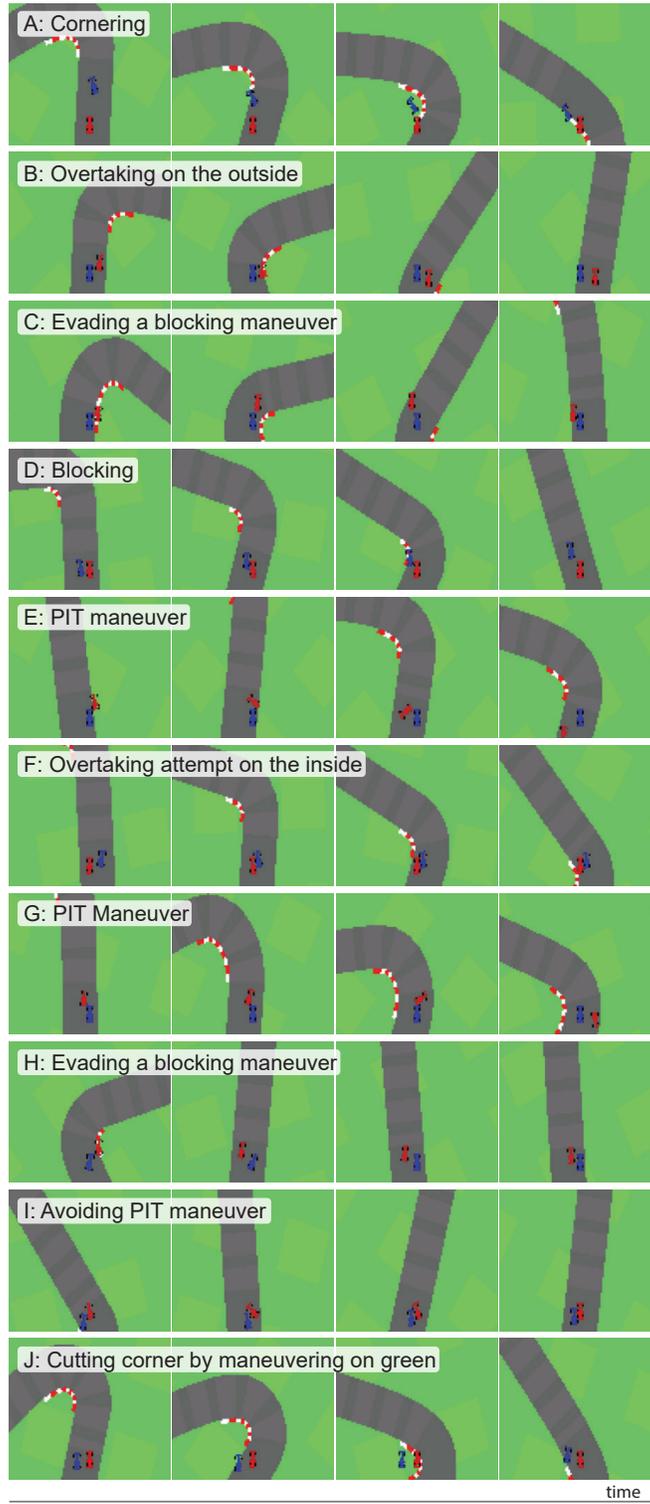


Figure 6: The agent has learned to leverage a large variety of racing skills towards their competitive advantage. The skills include cornering, blocking, overtaking, and forcing others off the road.

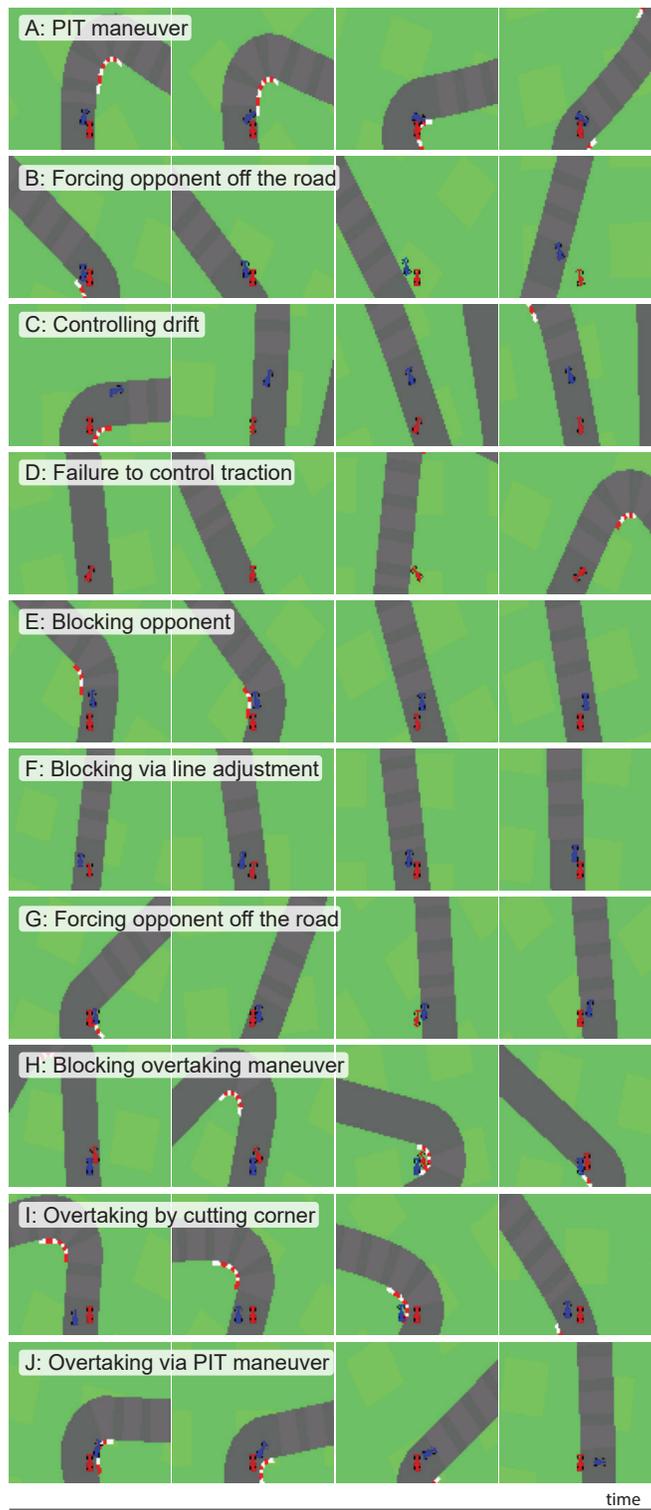


Figure 7: During a race, agents have to drive at the dynamical limits of handling to move as fast as possible along the track. While they don't receive penalties for leaving the road, moving onto the slippery grass increases the risk of spinning out.