Learn Thy Enemy: Online, Task-Aware Opponent Modeling in Autonomous Racing

Letian Chen **Shawn Manuel** **James Delgado** **John Subosits** **Paul Tylkin**
{zac.chen.pi, shawn.manuel, james.delgado.ctr, john.subosits, paul.tylkin}@tri.global

Abstract

Automobile racing provides a unique and challenging environment for studying competitive multi-agent behavior. In creating autonomous racing agents, one consideration is the effect that modeling one's opponents has on finding high-performance policies. In this paper, we study the overall effectiveness of opponent modeling in the context of autonomous racing, as well as the value of different information about one's opponents. We propose a new approach for learning salient characteristics of one's opponent: *Learn Thy Enemy* (LTE), an algorithmic framework that combines reinforcement learning with self-supervised learning about one's opponents. We evaluate LTE against multiple baselines in a CARLA-based simulation of an actual major racetrack. The results demonstrate that LTE substantially outperforms baselines, showing LTE's effectiveness in extracting relevant opponent information automatically during interactions with the aim of better accomplishing the task. Video demonstrations of LTE and baselines are in this Google Drive folder; descriptions of the videos are in the Appendix.

1 Introduction

Deep reinforcement learning (RL) has achieved tremendous success in various problems ranging from games (Silver et al., 2018; Ye et al., 2020) to real-world domains including healthcare (Datta et al., 2021), search and rescue (Yu et al., 2021), and manufacturing (Park et al., 2019; Wang and Gombolay, 2020). RL has also been applied successfully to multi-agent domains (Littman, 1994), where more than one agent operates within a common environment to compete or to cooperate (Buşoniu, Babuška, and De Schutter, 2010). In multi-agent settings, RL agents need to learn not just how to perform a particular task, but also how to work with or compete against others. Despite some very performant demonstrations of agents trained using multi-agent reinforcement learning (MARL) (OpenAI et al., 2019; Wurman et al., 2022), the current state-of-the-art algorithms still lack fast, accurate, and responsive modeling of other agents in the environment. This limits their ability to adapt to unseen adversaries or new partners, thereby restricting the applicability and robustness of learned models. In addition, humans use prior information about their adversaries to develop strategies and gain advantages over opponents (McIlroy-Young et al., 2021).

Among various challenging multi-agent settings, automobile racing is an especially rich domain: it requires not only real-time continuous control that enables sophisticated driving with minimal error tolerance but also strategic play to gain the best advantage over opponents (Daly, 2008). Autonomous racing is a recently-expanding subfield that combines elements of robotics, control theory, and learning for developing performant agents in both simulation and using physical hardware (Betz et al., 2022). Despite prior work on using RL in this context (Jaritz et al., 2018), a crucial aspect of autonomous racing, and automobile racing in general, is the strategic nature of interactions and the

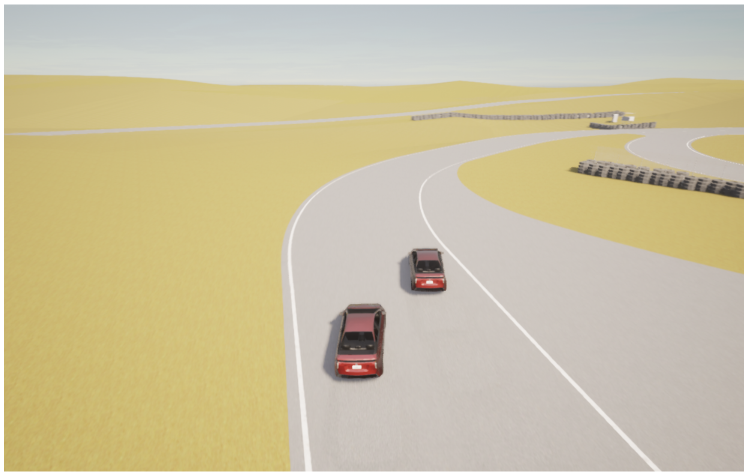

Figure 1: The race track environment in CARLA.

importance of informed opponent models. In our work, we aim to bridge this gap in the literature, using autonomous racing as a testbed for studying the value of opponent modeling.

Nashed and Zilberstein (2022) point out a key limitation in previous literature: most opponent modeling algorithms are evaluated by the accuracy of opponent behavior or strategy prediction and do not conduct experiments that test the entire pipeline (i.e., utilizing the modeled opponent information to improve the performance of the ego agent). As such, it has remained unanswered whether the proposed methods actually improve an agent's capabilities to exploit its opponents' strategies.

In this paper, we propose a novel algorithm, Learn Thy Enemy (LTE), that adaptively infers other agents' strategies online, and learns to leverage the opponent information in maximizing the ego agent's performance. We also propose a testing protocol that directly evaluates the benefit that opponent modeling brings to task performance, instead of only evaluating the accuracy of opponent models. We test our algorithm on a CARLA environment (Dosovitskiy et al., 2017) based on a real, major racetrack and demonstrate the significantly-improved performance of agents trained with Learn Thy Enemy (LTE) over benchmarks that have access to different manually-designed or learned information about opponent strategies.

2 Related Work

There has been abundant work on modeling other agents in the multi-agent literature, in both collaborative and competitive settings (Albrecht and Stone, 2018). In particular, there has been some notable prior work that incorporates models of other agents in studying the emergence of cooperative behaviors (Leibo et al., 2017; Foerster et al., 2018). This work has primarily focused on agents in discrete, stylized domains.

Nashed and Zilberstein (2022) define *opponent modeling* as "the ability to use prior knowledge and observations to predict the behavior of an opponent" and provide a good survey of opponent modeling approaches, in which they categorize these into discriminative classification, generative methods, and policy approximation. In *discriminative classification*, experts design discrete strategies and label data, after which a classification model is obtained from supervised learning (Laviers and Sukthankar, 2011). The discriminative approach relies on the assumption that any expert knowledge about discrete strategies is accurate and helpful in downstream task learning. The *generative methods* model the full generative process of the opponent behavior, often with Bayesian Networks (Wei et al., 2013) or Hidden Markov Models (Sukthankar and Sycara, 2006). However, the generative models rely on even more expert knowledge and data in constructing a high-quality generative process. *Direct policy approximation* relaxes the assumption on expert knowledge by directly learning a mapping from environment states to opponent behaviors (Tang et al., 2020), but still has the tradeoff between the learned accuracy and the amount of data needed for training.

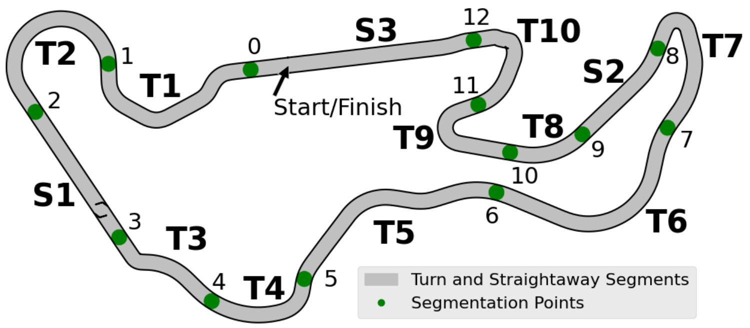

Figure 2: An illustration of the race track used for our experiments. The Ts represent turn segments and the Ss represent straightaway segments of the track. The green dots denote the segmentation points that separate the segments.

Most previous literature focuses solely on evaluating the opponent models. In particular, most previous work in this area underemphasizes or even overlooks the ultimate goal of opponent modeling: opponent exploitation, which measures how the opponent models help an agent achieve larger advantages over its adversaries. As pointed out by Nashed and Zilberstein (2022), the value of the information provided by opponent models cannot be measured by evaluating the model outputs in isolation and has to be evaluated *in situ*. In this work, we evaluate our algorithm and baselines in terms of racing performance to actually test the utility of opponent modeling.

There has also been prior work on autonomous racing, with a recent emphasis on developing super-human agents. GT Sophy (Wurman et al., 2022) has demonstrated a very successful autonomous racing agent against world champion players in the Gran Turismo video game. However, our work focuses on an area overlooked in developing GT Sophy: multi-agent strategic play. Despite the high performance achieved by the agent trained with reinforcement learning, the authors admit GT Sophy lacks the notion of opponents and does not have a strategic plan on how to best gain advantages over its opponents. In our work, we show that our framework can learn to perform the racing task in Carla simulation with a simplified reward structure, and that explicitly modeling opponent information benefits the performance of the ego car in a multi-agent setting. This represents a novel contribution to the autonomous racing literature about what opponent information should be encoded for a policy's use and how an agent can learn the opponent encoding during the training process.

3 Preliminaries

In this work, we use the standard Markov Decision Process (MDP) formalism: $\langle \mathcal{S}, \mathcal{A}, T(s'|s,a), R(s,s'), \gamma, \rho_0(s) \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $T(s'|s,a)$ is the transition function, $R(s,s')$ is the reward function, γ is the discount factor, and $\rho_0(s)$ is the initial state distribution. The goal of a reinforcement learning (RL) algorithm is to find a policy, $\pi(a|s)$, that maximizes the expected return: $\pi^* = \arg\max_\pi \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, s_{t+1})]$, where $\tau \sim \pi$ is short for the episode sampling process: $s_0 \sim \rho_0, a_t \sim \pi(s_t), s_{t+1} \sim T(\cdot|s,a)\ \forall t \geq 0$.

We formalize our problem as a two-player Markov game, a multi-agent extension of an MDP: $\mathcal{M} = (\mathcal{S}, \{\mathcal{A}\}^{1,2}, T, \{R\}^{1,2}, \gamma, \rho_0)$. The Markov game allows each agent to choose its action based on the state, and the transition happens when both agents choose their actions: $T(s'|s,a^1,a^2)$. Each agent i then obtains a reward according to its reward function, $R^i(s,s')$. We use the superscript $-i$ to denote the agent other than the agent i. Each policy's objective is then to maximize its own expected return: $\pi^{i*} = \arg\max_{\pi^i} \mathbb{E}_{\pi^i,\pi^{-i}}[\sum_{t=0}^{\infty} \gamma^t R^i(s_t, s_{t+1})]$. It can be seen that the objective involves the opponent policy, π^{-i}, and our approach aims to model and extract helpful information from the opponent's behavior to increase the ego agent's performance.

4 Environment

We study algorithms in a multi-agent racing environment developed in the CARLA (Dosovitskiy et al., 2017) simulator (see Figure 1 for a view of the environment). Our map is based on a faithful reproduction of a major real-world race track generated from photography, driver interviews, and sensor data collected from on-track experiments. An illustration of the race track is displayed in Figure 2. The vehicles' dynamics implement an approximation to race car physics, with values for car engine output and tire friction profiles corresponding to realistic ranges. The physics profile was tuned using input from professional drivers familiar with the physical course that was virtualized in the simulation and actual vehicles used for the racing task. In our computational experiments, we focus entirely on the two-agent case.

Observations given to the agents consist of a 148-dimensional vector, which includes agent-specific observations (distance traveled, its Cartesian coordinates, velocity, acceleration, tire slip angles, yaw rate, heading, and previous action), track information (heading relative to track direction, proportional distance from track center line to track edges, and 30 forward-looking left/right track edge point pairs spaced proportionally to velocity), and the Cartesian coordinates of the other agent. The action space for each agent is a 2-dimensional vector, consisting of the steering angle and combined throttle-brake (normalized to $[-1, 1]$).

The reward function for each agent consists of three components:

$$R \overset{\text{def}}{=} R_{\text{progress}} + R_{\text{passing}} + R_{\text{collision}} \tag{1}$$

The first component is the progress reward, as shown in Equation 2, where $\mathbb{1}_{\text{on-track}}(\cdot)$ denotes the indicator function to test whether the vehicle is on- or off- track and $f_{\text{progress}}(\cdot)$ is the mapping from vehicle state to the longitudinal position on the track. In other words, the progress reward measures the longitudinal progress on the track, i.e., more progress along the track results in a higher reward, unless the agent goes off-track.

$$R_{\text{progress}}(s_t, s_{t+1}) \overset{\text{def}}{=} \mathbb{1}_{\text{on-track}}(s_{t+1}) \cdot (f_{\text{progress}}(s_{t+1}) - f_{\text{progress}}(s_t)) \tag{2}$$

The second reward component is a passing bonus: each agent receives a reward for passing the other agent, and a symmetric penalty for being passed, following Wurman et al. (2022). The third reward component is a collision penalty: each agent is penalized for collision with other vehicles to discourage aggressive, unrealistic passing. Note that our reward structure is much simpler (three components) than previous literature (Wurman et al. (2022), which consists of eight components) and we demonstrate that such a simple reward function is indeed enough for inducing performant racing behaviors.

We further introduce two episode termination conditions: out-of-boundary termination, which terminate the episode when the vehicle drives significantly off the track, and no-progress termination where the vehicle does not have positive forward-moving speed. The two termination conditions work with the reward function to encourage on-track, canonical driving. If none of the termination conditions are triggered during the race, the episode is truncated at 1600 timesteps.

5 Method

In this section, we start with an analysis of the drawbacks in previous opponent modeling paradigms (depicted in Figure 3 (a) and (b)). We then describe our approach, *Learn Thy Enemy (LTE)*, that overcomes the limitations of previous approaches via a novel mutual-information-maximization objective.

A straightforward approach to modeling opponents is to learn to classify different strategies from labeled behaviors. The opponent encoder can be trained by supervised learning signals with the labeled dataset mapping observation history of opponents, $O_t = \{o_{t'}^{-i}\}_{t'=t-T+1}^{t}$ (e.g., in our racing task, the opponent's position), onto classes or features of opponent strategy. The classified strategy information is then fed into the policy as an auxiliary observation to supplement the environment observation, s_t, as shown in Figure 3(a). This approach relies heavily on expert knowledge about strategy classes and features that define strategies. More crucially, the encoder is agnostic about the reinforcement learning objective, and therefore may not generate information that the policy could utilize to optimize its performance. This is especially important in settings such as automobile racing where expert knowledge may not be sufficient to define the salient characteristics of good strategies – an expert may be able to identify a policy as high-skill but not necessarily quantitatively characterize what makes it so.

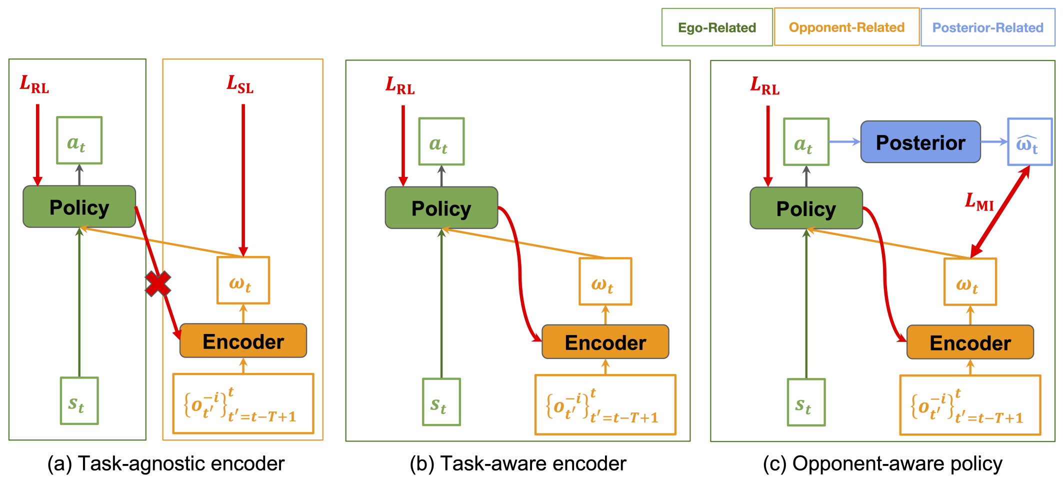

Figure 3: Comparison among three paradigms of opponent modeling. (a) Task-agnostic encoder: the encoder is trained with supervised learning signals and is agnostic about the ego agent's decision-making. (b) Task-aware encoder: the encoder is trained with reinforcement learning (task-aware, i.e., the encoder outputs opponent information that is helpful for the policy to achieve a high reward). (c) Our opponent-aware policy (LTE): in addition to the task-aware encoder, the policy is required to take the opponent's information into consideration by generating actions that can reconstruct the opponent's encoding.

Algorithm 1: Learn Thy Enemy (LTE)

```
Input      : Training iterations $E$, opponent policy set $\Pi^O = \{\pi_i^O\}_{i=1}^N$ where $N$ is the number of
             opponent policies, learning rate $\alpha$, number of training iterations for each environment
             episode $M$, opponent history length $T$, target Q network soft update coefficient $\tau$.
Output     : Learned ego policy, $\pi_\theta$, Q-function, $Q_\xi$, target Q-function, $Q_\Xi^{\text{target}}$, opponent encoder,
             $g_\phi$, and posterior predictor, $q_\psi$.
Initialize : Initialize neural network parameters $\theta, \phi, \psi, \xi$. Copy Q parameters $\xi$ to target Q
             parameters, $\Xi$. Empty the replay buffer $\mathcal{B}$.
1  for $i = 1$ to $E$ do
2      Collect trajectories $\{\tau\}$ with the current policy, $\pi_\theta$, the current opponent encoder, $g_\phi$, and
        sampled opponent policy, $\pi^O \sim \Pi^O$
3      Add trajectories $\{\tau\}$ to replay buffer $\mathcal{B}$
4      for $j = 1$ to $M$ do
5          Sample minibatch $\{(s_t, a_t, O_t, r_t, s_{t+1}, O_{t+1})\} \sim \mathcal{B}$.
6          Calculate opponent encoding, $\omega_t = g_\phi(O_t), \omega_{t+1} = g_\phi(O_{t+1})$.
7          Calculate critic loss,
            $L_{\text{critic}} = \texttt{huber\_loss}(Q_\xi(s_t, \omega_t, a_t), r_t + Q_\Xi^{\text{target}}(s_{t+1}, \omega_{t+1}, \pi_\theta(s_{t+1}, \omega_{t+1}))$ and its
            gradient with respect to $\xi$.
8          Calculate policy loss, $L_{\text{policy}} = -Q_\xi(s_t, \omega_t, \pi_\theta(s_t, \omega_t))$, and its gradient with respect to
            $\theta$ and $\phi$.
9          Calculate mutual information loss, $L_{\text{MI}} = -\log\left[q_\psi(\omega_t | s_t, \pi_\theta(s_t, \omega_t))\right] - H(\omega_t)$, and its
            gradient with respect to $\psi$, $\theta$, and $\phi$.
10         Update $\theta, \phi, \psi, \xi$ based on their accumulated gradients with learning rate $\alpha$.
11         Soft update the target Q network: $\Xi \leftarrow (1-\tau)\Xi + \tau\xi$.
```

In order to make the encoder task-aware and remove the dependency on strong expert knowledge in supervised learning, we propose to allow the reinforcement learning signal, $L_{\text{RL}} \overset{\text{def}}{=} \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, s_{t+1})]$, to update the encoder model, assuming the differentiability of the policy model, as shown in Figure 3 (b).

Although this paradigm allows the encoder to generate helpful encodings for improving the policy's performance, the policy could also largely ignore the output of the encoder, as the output of the encoder during the initial phase of training is random and not useful for learning the task. By the time the encoder learns to extract helpful information, the policy may already learn to ignore the opponent encoding, ω_t (since it was not helpful in earlier stages of training).

As such, we further introduce an auxiliary objective that encourages the learned policy output, a_t, to incorporate the opponent information, ω_t: the mutual information between a_t and ω_t, defined as $I(\omega; a) \stackrel{\text{def}}{=} H(\omega) - H(\omega|a)$, where $H(\bullet)$ is the entropy. Intuitively, the mutual information can be seen as the decrease in entropy (uncertainty about ω's value) once we know the action a. Maximizing the mutual information $I(\omega; a)$ ensures that given ω, the action a will be less stochastic, i.e., a's choice depends on ω.

However, calculating $I(a; \omega)$ directly is intractable due to the integral over all possible ω, and therefore we seek to approximate it with a variational lower-bound, which we denote by L_{MI} (the mutual information objective), as shown in Equation 3, where KL denotes the KL-divergence.

$$\begin{aligned}
I(\omega; a) &\stackrel{\text{def}}{=} H(\omega) - H(\omega|a) \\
&= \mathbb{E}_{(a,\omega)\sim P(a,\omega)}[\log p(\omega|a)] + H(\omega) \\
&= \mathbb{E}_{(a,\omega)\sim P(a,\omega)}[\log \frac{p(\omega|a)q(\omega|a)}{q(\omega|a)}] + H(\omega) \\
&= \mathbb{E}_{a\sim P(a)}[\text{KL}[p(\omega|a)||q(\omega|a)]] + \mathbb{E}_{(a,\omega)\sim P(a,\omega)}[\log q(\omega|a)] + H(\omega) \\
&\geq \mathbb{E}_{(a,\omega)\sim P(a,\omega)}[\log q(\omega|a)] + H(\omega) \\
&\stackrel{\text{def}}{=} L_{\text{MI}}.
\end{aligned} \tag{3}$$

The equality of the $\geq$ holds when $q(\omega|a)$ exactly matches the true posterior, $p(\omega|a)$.

We note that unlike previous mutual information maximization paradigms such as DIAYN (Eysenbach et al., 2019), our latent information, ω, is not freely sampled but is given by the encoder, $g_\phi(O_t)$, dependent on the opponent's behavior history. This has two advantages. First, with a neural network-parameterized ω output distribution, we are readily able to calculate $H(\omega)$ to obtain the value of the lower bound instead of ignoring the constant, $H(\omega)$, as in most previous literature (e.g., Poole et al. (2019)). Second, gradients originating from maximizing L_{MI} update not only the posterior and the policy (the first term of L_{MI}) but also the encoder to extract meaningful, impactful information (both terms of L_{MI}). The task objective, L_{RL}, also updates both the policy and the encoder, aiming to find helpful information from opponent behaviors that could contribute to the high performance of the ego agent.

Combining the idea of task-aware learning and the mutual information maximization for the opponent encoder, we present our algorithm, LTE, in Algorithm 1. For each training iteration (line 1), we first collect rollouts with the current policy and opponent encoder (line 2) by the generative process: $s_0 \sim \rho(\cdot), \omega_t \sim g_\phi(O_t), a_t^1 \sim \pi_\theta(s_t, \omega_t), a_t^2 \sim \pi^O(s_t), s_{t+1} \sim T(s_t, a_t^1, a_t^2)\ \forall t \geq 0$, where π^O denotes an opponent policy. We add the trajectories into a replay buffer, consistent with off-policy reinforcement learning approaches (line 3).

We then run training for M iterations (lines 4-11). In each iteration, we sample a minibatch from the replay buffer (line 5), and calculate the opponent encoding information based on the history of opponent positions (line 6). We then construct the three losses, the RL critic loss L_{critic}, the RL policy loss L_{policy}, and the auxiliary mutual information loss L_{MI} and compute corresponding gradients in lines 7-9, respectively.

It is worth noting the parameters that receive gradient updates from each loss function. The critic loss only updates the Q-function parameters, ξ. Both the policy loss and the mutual information loss update the policy parameters, θ, and the encoder parameters, ϕ, to generate actions that result in high performance in the environment while considering the opponent information and generating helpful opponent information from the encoder. The mutual information loss also updates the posterior network, ψ, to provide correct learning signals for the mutual information loss by lowering the gap between L_{MI} and the true mutual information $I(\omega; a)$.

Once all gradients are calculated, we update learning network parameters in line 10, and update the target Q network with a soft parameter copy from the Q network in line 11, following Silver et al. (2014).

6 Experiments

In the computational experiments described here, we compare the performance of the LTE framework to a number of baselines. The purpose of these experiments is to empirically evaluate whether there are advantages to online, task-aware opponent information extraction as compared to offline, task-unaware opponent modeling approaches.

The first step necessary for evaluating the efficacy of an opponent-modeling framework requires diverse and high-quality policies in the environment for the ego agent to compete against. To do this, we train RL agents with the environment reward and DIAYN intrinsic reward (Eysenbach et al., 2019) which encourages diverse behaviors. We generate five competitive and diverse opponent policies, which are used as the opponents in all of the experiments described next.

6.1 Baseline

We compare LTE against three main groups of benchmarks, as detailed below:

1. Baseline (Position Only): the ego agent only has access to opponent information via the opponent car's location in the states, i.e., a no-explicit-opponent-modeling baseline.
2. Baseline (Opponent Features): the ego agent has access to additional manually-engineered features of each opponent, i.e., an expert-knowledge-infused baseline.
3. Baseline (Future Opponent Positions): the ego agent has access to the predicted future positions of opponents, which are given by a model trained on an offline dataset.

Within the opponent features baseline, we experiment with multiple approaches to insert the expert-specified features:

1. Baseline (Overall Features): six features of the opponents calculated from a dataset consisting of 25 rollout episodes (5 rollouts against each of the five opponent models). The six features include average speed, speed standard deviation, the average distance to track boundary as a proportion to the track width, the percentage of driving on the left side of the track, the average latitudinal position on the track, and the maximum steering angle.
2. Baseline (Turn-By-Turn): as part of the expert knowledge, racing behaviors are generally homogeneous within the scope of one turn and heterogeneous between the turns, as each car needs to decide whether to gain an advantage over or block opponents in each segment of the track. We allow Baseline (Turn-By-Turn) to have access to a set of 13 turn-by-turn based features (six features as in Baseline (Overall Features)), resulting in a total of 78 additional features.
3. Baseline (Current Turn): baseline (Turn-By-Turn) swamps the observation space with all turns' features. However, we could again incorporate the expert knowledge that the ego decision may only focus on the local information about the opponent's strategy. Thus, we propose the Baseline (Current Turn) to take six opponent features corresponding to the turn the ego agent is at as additional information for the ego vehicle.
4. Baseline (One-Hot): to test the efficacy of knowing the identity of the opponent but no other characteristics, we include Baseline (One-Hot) that only provides a one-hot encoding of different opponents to help the ego agent understand that there are different opponents but providing no further information about them.

6.2 Training Setup

We conduct our experiments on Amazon Web Services (AWS) EC2 and each training run of 2,000,000 environment steps takes approximately 14 hours to complete on a single `g4dn.8xlarge` instance (32 2.5Ghz virtual CPUs, 128 GB of RAM, and a NVIDIA T4 GPU with 16 GB VRAM), showing the general feasibility of training the baselines and LTE with a very reasonable amount of computational

Hyperparameter	Value
Hidden dims	[1024, 1024]
Learning rate	0.0001
Batch size	2048
Grad clip norm	10
Nstep	5
Hidden activation	tanh
Discount rate	0.99
Opponent encoder dimension	8
Weight for mutual information loss	1000

Table 1: This table lists the hyperparameters identified by a hyperparameter sweep. All of the values that were evaluated as part of the sweep are in the Supplementary Materials.

	Lap Time (s)	Lead %	Leader Dist. (m)	Opponent Dist. (m)
Position Only	70.94 (2.02)	38.29 (41.44)	32.98 (40.10)	-8.03 (71.39)
Overall Feat.	70.87 (2.09)	58.99 (37.75)	26.94 (36.27)	4.09 (60.60)
Turn-By-Turn	73.21 (2.39)	36.35 (35.18)	62.07 (60.67)	-48.82 (74.99)
Current Turn	71.64 (3.03)	48.01 (40.63)	49.22 (62.96)	-14.12 (102.10)
One-Hot	73.07 (3.29)	39.27 (38.30)	52.58 (56.92)	-32.85 (76.49)
LTE (Ours)	**69.38*** (3.23)**	**73.41*** (37.38)**	**20.65*** (45.73)**	**40.20*** (80.94)**

Table 2: This table summarizes the evaluation results of our algorithm and baselines. The result is an average of training with three different seeds and 100 evaluation rollouts (20 repeats against each opponent model). The numbers in parentheses represent standard deviation. Bold denotes the best performance on the metric. The *** indicates statistical significance of running a Kruskal-Wallis H test where we reject the hypothesis of the median of all experiments' results for a single metric being the same based on a p value threshold of 0.001.

resources. We train LTE and each baseline technique for three seeds (0, 1, and 2), and evaluate the final policy with each of the five opponents for 20 episodes. As such, each metric reported is the result of an average over $3 \times 5 \times 20 = 300$ evaluation rollouts.

We conducted a hyperparameter sweep to determine the best hyperparameter set for LTE and baselines, and concluded the best-performing hyperparameters as shown in Table 1, where `Nstep` denotes the number of steps forward-looking in the critic loss calculation, which modulates variance (large `Nstep` has high Monte Carlo variance) vs. bias (small `Nstep` relies on the biased Q-value estimation).

6.3 Results

We summarize the experimental results in Table 2, where we compare LTE with baselines on four metrics: lap time, lead percentage, distance to leader, and distance to opponent. The lap time is measured as the time spent from first reaching the starting line to arriving at the starting line the second time[1], which is the gold standard in the racing domain for evaluating single-agent racing. As we focus more on how opponent modeling benefits agents in a multi-agent domain, we introduce three additional metrics: the lead percentage which depicts the percentage where the ego car is ahead of the opponent, the distance to leader which is non-negative (and 0 when the ego is the leader), and distance to opponent which is positive when the ego is ahead of the opponent and negative otherwise.

As can be seen in Table 2, LTE outperforms all baselines on all four metrics, showing its strong capability in not only the racing task (minimizing lap time) but also competing with and successfully overtaking its opponent in the multi-agent racing setting (lead percentage, distance to leader, and distance to opponent). Notably, Baseline (Overall Features) outperforms all other expert-knowledge-

[1]Record human lap times on this particular track are in the approximately 80-90 second range. All of the autonomous racing agents that we train, including baselines, are substantially faster. Because our focus in this work is not specifically on comparing human/AI performance, and due to possible differences in the simulation domain relative to actual vehicles on the track, including perception and actuation, we do not discuss this point further.

based baselines, showing the value of adding expert-engineered features compared with just an opponent indicator (one-hot). We also note that Baseline (Future Opponent Positions) completely fails to yield any successful policy and is not able to successfully complete a lap; since we calculated these statistics only for complete laps, we omit it from the table.

These results illustrate LTE's strong capabilities in actually utilizing opponent information in gaining strategic advantages when racing against opponents, as opposed to just demonstrating the quality of opponent modeling with no impact on performance.

7 Conclusion

In this paper, we introduced a new algorithm for opponent modeling and exploitation, Learn Thy Enemy (LTE), and demonstrated its capability to achieve high performance in a rich, dynamic multi-agent racing environment by incorporating information about its opponent. In addition to being conceptually straightforward, training an agent using LTE requires a similar amount of computation as training an agent using other deep RL algorithms (DIAYN and the baselines introduced in our experiments), while exceeding their performance in the environment along both single-agent and multi-agent metrics.

Next, we discuss some limitations of our approach and directions for future work.

7.1 Future Work

The work presented in this paper has a number of potential avenues for future study, along two primary axes: (1) increasing the robustness of the opponent modeling introduced here, and (2) moving beyond simulation to actual autonomous racing agents. In the experiments that we conduct here, we evaluate the baselines with particular pre-trained opponents; we do not evaluate against unseen, or adversarially-generated, opponents. In addition, the opponents are not themselves learning agents, arguably making the task of modeling them simpler. Thus, immediate next steps could include testing the capability of LTE-trained agents to generalize and perform well against unseen opponents, and also to model multiple, heterogeneous opponents at the same time. Possible approaches to this include modifying the neural network architecture of the opponent encoder, g_ϕ, to incorporate multiple opponents, and possibly learning to only pay attention to some subset of them (e.g., agents of similar skill who are likely to have more interactions with the ego agent).

The simulation environment that we conduct our experiments within is based on a high-fidelity reproduction (in terms of terrain and other features) of a real-world major race track, and a possible extension of this work would be to deploy the policies learned to actual physical race cars on this track. Clearly, this direction requires a number of steps not discussed here, including using visual observations instead of agent positions, ensuring safety, dealing with noisy sensor hardware, and actuation delays. Nevertheless, the future vision of this being an initial step towards performant opponent-aware policies that can be used for autonomous racing is promising.

Acknowledgements

The authors are grateful to everyone at the Human Interactive Driving division and at Toyota Research Institute more broadly for your suggestions, feedback, and ideas. Special thanks to Andrew Best, Guy Rosman, Jonathan DeCastro, Tiffany Chen, Hiroshi Yasuda, Jean Costa, Sheryl Chau, Evelyn Dixon, and Avinash Balachandran.

References

Albrecht, S. V.; and Stone, P. 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258: 66–95.

Betz, J.; Zheng, H.; Liniger, A.; Rosolia, U.; Karle, P.; Behl, M.; Krovi, V.; and Mangharam, R. 2022. Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing. *IEEE Open Journal of Intelligent Transportation Systems*, 3: 458–488.

Buşoniu, L.; Babuška, R.; and De Schutter, B. 2010. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, 183–221.

Daly, D. 2008. *Race to Win: How to Become a Complete Champion Driver*. Motorbooks.

Datta, S.; Li, Y.; Ruppert, M. M.; Ren, Y.; Shickel, B.; Ozrazgat-Baslanti, T.; Rashidi, P.; and Bihorac, A. 2021. Reinforcement learning in surgery. *Surgery*, 170(1): 329–332.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, 1–16.

Eysenbach, B.; Gupta, A.; Ibarz, J.; and Levine, S. 2019. Diversity is All You Need: Learning Skills without a Reward Function. *ICLR 2019*.

Foerster, J.; Chen, R. Y.; Al-Shedivat, M.; Whiteson, S.; Abbeel, P.; and Mordatch, I. 2018. Learning with Opponent-Learning Awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 122–130.

Jaritz, M.; de Charette, R.; Toromanoff, M.; Perot, E.; and Nashashibi, F. 2018. End-to-End Race Driving with Deep Reinforcement Learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2070–2075.

Laviers, K.; and Sukthankar, G. 2011. A real-time opponent modeling system for rush football. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Leibo, J. Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; and Graepel, T. 2017. Multi-Agent Reinforcement Learning in Sequential Social Dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 464–473.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, 157–163. Elsevier.

McIlroy-Young, R.; Wang, Y.; Sen, S.; Kleinberg, J.; and Anderson, A. 2021. Detecting Individual Decision-Making Style: Exploring Behavioral Stylometry in Chess. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 24482–24497. Curran Associates, Inc.

Nashed, S.; and Zilberstein, S. 2022. A survey of opponent modeling in adversarial domains. *Journal of Artificial Intelligence Research*, 73: 277–327.

OpenAI; Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; Józefowicz, R.; Gray, S.; Olsson, C.; Pachocki, J.; Petrov, M.; de Oliveira Pinto, H. P.; Raiman, J.; Salimans, T.; Schlatter, J.; Schneider, J.; Sidor, S.; Sutskever, I.; Tang, J.; Wolski, F.; and Zhang, S. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv:1912.06680*.

Park, I.-B.; Huh, J.; Kim, J.; and Park, J. 2019. A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities. *IEEE Transactions on Automation Science and Engineering*, 17(3): 1420–1431.

Poole, B.; Ozair, S.; Van Den Oord, A.; Alemi, A.; and Tucker, G. 2019. On variational bounds of mutual information. In *International Conference on Machine Learning*, 5171–5180. PMLR.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T.; Simonyan, K.; and Hassabis, D. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, 387–395.

Sukthankar, G.; and Sycara, K. 2006. Robust recognition of physical team behaviors using spatio-temporal models. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 638–645.

Tang, Z.; Zhu, Y.; Zhao, D.; and Lucas, S. M. 2020. Enhanced rolling horizon evolution algorithm with opponent model learning: Results for the fighting game AI competition. *arXiv:2003.13949*.

Wang, Z.; and Gombolay, M. 2020. Learning scheduling policies for multi-robot coordination with graph attention networks. *IEEE Robotics and Automation Letters*, 5(3): 4509–4516.

Wei, X.; Lucey, P.; Morgan, S.; and Sridharan, S. 2013. Sweet-spot: Using spatiotemporal data to discover and predict shots in tennis. In *7th Annual MIT Sloan Sports Analytics Conference, Boston, MA*.

Wurman, P. R.; Barrett, S.; Kawamoto, K.; MacGlashan, J.; Subramanian, K.; Walsh, T. J.; Capobianco, R.; Devlic, A.; Eckert, F.; Fuchs, F.; Gilpin, L.; Khandelwal, P.; Kompella, V.; Lin, H.; MacAlpine, P.; Oller, D.; Seno, T.; Sherstan, C.; Thomure, M. D.; Aghabozorgi, H.; Barrett, L.; Douglas, R.; Whitehead, D.; Dürr, P.; Stone, P.; Spranger, M.; and Kitano, H. 2022. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896): 223–228.

Ye, D.; Chen, G.; Zhang, W.; Chen, S.; Yuan, B.; Liu, B.; Chen, J.; Liu, Z.; Qiu, F.; Yu, H.; et al. 2020. Towards Playing Full MOBA Games with Deep Reinforcement Learning. *Advances in Neural Information Processing Systems*, 33: 621–632.

Yu, C.; Liu, J.; Nemati, S.; and Yin, G. 2021. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1): 1–36.

Appendix and Supplementary Material for *Learn Thy Enemy: Online, Task-Aware Opponent Modeling in Autonomous Racing*

A Hyperparameters Used

When training the opponent models using DIAYN, a hyperparameter sweep was conducted to find the final set of hyperparameters, with the models that had the highest total reward at the end of training chosen as the opponent models. The ranges of hyperparameters tested during the sweep were:

Hyperparameter	Value
Hidden dims	[256,256,256,256,256], [256,512,512,256], [512,512,512] [1024,1024], [1024,1024,1024], [2048,2048]
Learning rate	0,001, 0.0005, 0.0001
Batch size	256, 512, 1024, 2048
Replay buffer size	100000, 1000000, 5000000
Grad clip norm	None, 0.5, 1.0, 5.0, 10
Nstep	1,3,5,10
Hidden activation	tanh, relu, leaky relu

Table 3: Hyperparameters tested as part of the hyperparameter sweep.

The final network training hyperparameters for the baseline experiments are shown in Table 4.

Hyperparameter	Value
Hidden dims	[1024, 1024]
Learning rate	0.0001
Batch size	2048
Grad clip norm	10
Nstep	5
Hidden activation	tanh
Discount rate	0.99

Table 4: Final hyperparameter values, as determined by the highest-reward models from the hyperparameter sweep.

In addition to the hyperparameters above, the LTE algorithm uses the following additional hyperparameter values: an encoding dimension size of 8 was used as the size of the encoding network for the opponent policy, and a mutual information weight of 1000 was used.

B Demonstration Videos

Please see this Google Drive folder for some demonstration videos of the trained agents. The red car is the ego agent and the blue car is the opponent. Note that despite the appearance of the vehicles being different, this is purely to make it easier to distinguish the ego and opponent vehicles. The physics profiles of the vehicles in the simulation (i.e., engine power, tire friction values, etc.) are exactly the same. The videos are recorded at 2x real-time.

- `0_baseline_position_only.mp4`: This video demonstrates the initial failure of the ego agent to block an overtake maneuver by the opponent. Subsequently to this, the ego agent is behind.
- `1_baseline_overall_features.mp4`: The ego agent fails to block an overtaking manuever by the opponent.
- `2_baseline_turn_by_turn.mp4`: The ego agent displays reasonable blocking behavior, until an unsuccessful block results in both agents running off the track.
- `3_baseline_current_turn.mp4`: The ego agent fails to block an overtaking maneuver by the opponent.

- `4_baseline_one_hot.mp4`: The ego agent fails to block an overtaking maneuver by the opponent.
- `5_learn_thy_enemy.mp4`: The ego agent outmaneuvers its opponent near the beginning of the track and successfully maintains its positional advantage over the entire lap.