Supplementary Material - Rethinking Closed-loop Training for Autonomous Driving

Chris Zhang*[1,2], Runsheng Guo*†[3], Wenyuan Zeng*[1,2],
Yuwen Xiong[1,2], Binbin Dai[1], Rui Hu[1], Mengye Ren†[4], and Raquel Urtasun[1,2]

[1] Waabi [2] University of Toronto
[3] University of Waterloo [4] New York University
{czhang,wzeng,yxiong,bdai,rhu,urtasun}@waabi.ai
r9guo@uwaterloo.ca@uwaterloo.ca mengye@nyu.edu

In this supplementary material, we provide details about the implementation of our method and benchmark, as well as more experimental analysis. In the following, we introduce the backbone architecture and trajectory sampler in Section 1. We then provide implementation details of learning in Section 2.1 and reward functions in Section 2.2. The proof of our theoretical analysis is presented in Section 3. We also explain our benchmark construction in more details in Section 4. We show some additional quantitative and qualitative analysis of TRAVL in Section 5 and Section 6 respectively. Finally, a high level overview this work can be found in the video `rethinking_clt.mp4`.

1 Technical Details of TRAVL

1.1 Backbone Architecture

Given an input rasterization tensor, our backbone network first performs three layers of 3×3 convolution with 64 channels. It then applies 4 consecutive *ResBlock* units. Each block consists of a 1×1, 3×3 and 1×1 convolutional layer as well as a skip-connection between input and output features. The input channels of these 4 units are $(64, 256, 1024, 4096)$ respectively and the convolution kernels for each layer has the same number of channels as the inputs, except for the last 1×1 layer that upsamples channels for the next stage. Besides, the 3×3 layer in each unit has a stride of 2 to downsample the feature map. Finally, we use two additional 3×3 convolutional layers to reduce the channel number to $C = 512$ without further downsampling. This produces a final backbone feature map $\mathbf{F} \in \frac{H}{8} \times \frac{W}{8} \times 512$.

1.2 Trajectory Sampler

As stated in the main paper, we use a trajectory sampler which samples longitudinal and lateral trajectories with respect to reference lanes. In Figure 1 we show a visualization of a trajectory sample set. As our trajectory sampler

* Denotes equal contribution.
† Work done during affiliation with Waabi.

considers map priors through the Frenet frame, it can produce smooth trajectories compatible with the lane shapes. This introduces inductive biases to driving maneuvers and is expected to ease the learning.

1.3 Planned vs. executed trajectory mismatch during MPC

Because our method plans in an MPC fashion, an entire trajectory is selected as the action but only the initial segment is executed, resulting in a mismatch. One way we have tried to address this mismatch is through executing an entire trajectory during rollout (instead of replanning in an MPC fashion) to collect experience into the replay buffer, yet we didn't notice significant gains over our current approach. One potential reason for this is that, even though using the entire trajectory is less theoretically complex, it also significantly reduces the number of simulated (state, action) pairs since an action now takes longer simulation time to execute. With limited computation resources, such an approach might degrade the performance due to less data.

2 Learning

2.1 Learning Objective

Recall that our learning process alternates between the *policy evaluation* and *policy improvement* step. The policy evaluation step we use is described in Eq. 3 in the main paper, as well as below

$$Q^{k+1} \leftarrow \arg\min_{Q_\theta} \mathbb{E}_{\mathcal{D}} \left[\underbrace{\left(Q_\theta(s,\tau) - \mathcal{B}_\pi^k Q^k(s,\tau)\right)^2}_{\text{Q-learning}} + \alpha_k \underbrace{\mathbb{E}_{\tau' \sim \mu(\tau'|s)} \left(R_\theta(s,\tau') - r'\right)^2}_{\text{Counterfactual Reward Loss}} \right],$$
$$s.t. \quad Q_\theta = R_\theta + V_\theta, \quad V_\theta = \gamma \mathcal{P}_\pi^k Q^k. \tag{1}$$

The policy improvement step is described in Eq. 1 in the main paper, as well as below

$$\pi^{k+1} \leftarrow (1-\epsilon) \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi}[Q^{k+1}(s,a)] + \epsilon U(a), \tag{2}$$

For the *policy evaluation* step, we use SGD to optimize an empirical objective (*i.e.*, a mini-batch estimation) of Eq. 1. Note that Eq. 1 can be converted as a Lagrangian term. Essentially, we take gradient steps of the following loss function

$\mathcal{L}$ over parameter θ to obtain the optimal solution for Eq. 1,

$$\mathcal{L} = \frac{1}{|\mathcal{D}|} \sum_{(s,\tau,r,s')} \Bigg[\underbrace{\left(R_\theta(s,\tau) + V_\theta(s,\tau) - r - \gamma Q^k(s', \pi^k(\tau'|s'))\right)^2}_{Q-learning} + \underbrace{\alpha \frac{1}{|\mathcal{T}|} \sum_{\tau' \neq \tau, \tau' \in \mathcal{T}} (R_\theta(s,\tau') - r')^2}_{Counterfactual-loss} + \underbrace{\lambda \left(V_\theta(s,\tau) - \gamma Q^k(s', \pi^k(\tau'|s'))\right)^2}_{Lagrangian-loss} \Bigg]. \tag{3}$$

However, this involves an expensive double loop optimization, *i.e.*, outer loop for iterating Q^k and inner loop for minimizing $\mathcal{L}$. We hence simply apply one gradient step for the inner loop updating Q^k to Q^{k+1}. In practice, we also found using a cross-entropy loss to minimize the KL-divergence between e^{-R_θ} and $e^{-r'}$ helps stabilize training compared to using ℓ_2 loss for the counterfactual term, possibly because the former one is less prone to outlier estimation of r' which is caused by our approximated world modeling. Our overall learning process is summarized in Algorithm 1.

Implementation Details: We train our method using the Adam optimizer [3]. We use a batch size of 10 and learning rate of 0.0001. To accelerate training, we collect simulation data asynchronously with 9 instances of the simulator and store them in a prioritized replay buffer. We initialize the ϵ as 0.1 and linearly decay it to 0.01 in the first 200k steps, and terminate learning at 1 million steps as the model converges. Besides, we use $\gamma = 0.95$, $\alpha = 1.0$ and $\lambda = 0.01$.

Our imitation learning baselines use the same input representation and network as our model. We replace the learning loss with ℓ_2 loss for the control based model and max-margin for the trajectory sampling based model [7]. We use a well-tuned auto-pilot model as our expert demonstration, which has access to all ground-truth world states in the simulation. Note that we use rasterization of detection boxes as inputs for all approaches. Therefore the baselines are similar to the privileged agent in LBC [1].

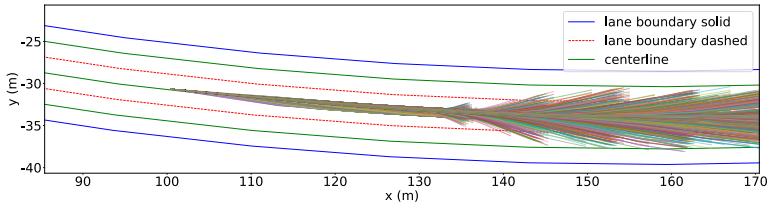

Fig. 1: Example samples from our trajectory sampler which uses map information.

2.2 Reward Function

Our reward function R is a linear combination of *progress*, *collision*, and *lane following* terms:

$$R(s^t, \tau^t, s^{t+1}) = C_p \cdot R_p(s^t, \tau^t, s^{t+1}) + C_c \cdot R_c(s^t, \tau^t, s^{t+1}) + C_l \cdot R_l(s^t, \tau^t, s^{t+1}),$$

where $C_p = 0.6, C_c = 40.0, C_l = 1.0$ are constants designed to balance the relative scale between the reward terms. R_p is the progress term, and rewards the agent for distance traveled along the goal lane. Here, a goal lane is defined by an external router and we assume to have access to it. We use D_{travel} to denote the traveled distance between the projections of s^t and s^{t+1} on the goal lane, and D_{lane} to denote the distance between s^t and its projection. The *progress reward* is defined as $R_p = e^{-0.2 \times D_{lane}} D_{travel}$, where the term $e^{-0.2 \times D_{lane}}$ penalizes the agent for driving further from the goal lane (D_{lane}). R_c is a term penalizing the agent for collisions, and is defined as:

$$R_c(s^t, \tau^t, s^{t+1}) = \begin{cases} -1.0 \text{ if the agent has collided at } s^{t+1}, \\ 0.0 \quad \text{otherwise.} \end{cases}$$

Finally, R_l is a lane following term penalizing the agent for deviating from the goal lane. For an action $\tau = \{(x^0, y^0), (x^1, y^1), \cdots (x^T, y^T)\}$, R_l is defined as the sum of the negative distances between each (x^i, y^i) and its projection on the goal lane.

3 Theoretical Analysis

Lemma 1. *Assuming R is bounded by a constant R_{max} and α_k satisfies*

$$\alpha_k < \left(\frac{1}{\gamma^k C} - 1 \right)^{-1} \left(\frac{\pi^k}{\mu} \right)_{min}, \tag{4}$$

with C an arbitrary constant, iteratively applying Eq. 1 and the policy update step in Eq. 2 converges to a fixed point.

Algorithm 1 TRAVL: TRAjectory Value Learning

Require: Simulator, Training Scenario Set
Initialization: $\mathcal{D} \leftarrow \emptyset$, $\pi(\tau|s) \leftarrow$ Uniform(τ), TRAVL network $\leftarrow$ random weights.
Asynchoronous Experience Collection:
while Learning has not ended **do**
  Sample a scenario variation from the training scenario set.
  Produce $(s^t, \tau^t, r^t, s^{t+1})$ by interacting the policy π and the simulator on the sampled scenario.
  Store $(s^t, \tau^t, r^t, s^{t+1})$ to the replay buffer $\mathcal{D}$.
end while
Learning:
for $k = 0, \cdots,$ max_iter **do**
  Draw (mini-batch) samples $(s^t, \tau^t, r^t, s^{t+1})$ from $\mathcal{D}$.
  Draw a set of trajectory samples $\mathcal{T}$ given s^t.
  Compute $R_\theta(s^t, \tau^t)$, $V_\theta(s^t, \tau^t)$ and $R_\theta(s^t, \tau')$, $V_\theta(s^t, \tau')$ for $\tau' \in \mathcal{T}$ using TRAVL network.
  Evaluate $r' = R(s^t, \tau', s')$ for $\tau' \in \mathcal{T}$ using reward functions.
  Compute $\mathcal{L}$ using Eq. 3.
  Update network parameter θ using gradients of $\mathcal{L}$.
  $Q_\theta \leftarrow R_\theta + V_\theta$.
  $\pi(\tau|s) \leftarrow \begin{cases} \arg\max_\tau Q_\theta(s,\tau), & \text{with probability } 1-\epsilon \\ \text{randomly sample } \tau, & \text{with probability } \epsilon. \end{cases}$
end for

Proof. To prove lemma 1 is correct, it suffices to show that the updating rule in Eq. 1 leads to $\lim_{k=\infty} ||Q^{k+1} - Q^k||_\infty = 0$. To find out the optimal Q_θ at iteration k, we take the derivative of the R.H.S. and set it to 0 as follows

$$\mathbb{E}_{\tau\sim\pi^k}\left[Q_\theta(s,\tau) - \mathcal{B}_\pi^k Q^k(s,\tau)\right] + \alpha_k \mathbb{E}_{\tau'\sim\mu}\left[Q_\theta(s,\tau') - \gamma \mathcal{P}_\pi^k Q^k(s,\tau') - \mathbb{E}_{\tau\sim\pi^k} r'\right] = 0.$$

Now we will interchange τ and τ' in the second term of the equation above ($\alpha_k \mathbb{E}_{\tau'\sim\mu}[\cdots]$) and use the fact that $\mathbb{E}_\mu[\cdots] = \mathbb{E}_\pi[\frac{\mu}{\pi}\cdots]$ to obtain.

$$\mathbb{E}_{\tau\sim\pi^k}\left[Q_\theta(s,\tau) - \mathcal{B}_\pi^k Q^k(s,\tau) + \frac{\alpha_k\mu}{\pi^k} Q_\theta(s,\tau) - \frac{\alpha_k\mu}{\pi^k}\gamma\mathcal{P}_\pi^k Q^k(s,\tau) - \frac{\alpha_k\mu}{\pi^k}\mathbb{E}_{\tau'\sim\pi^k} r'\right] = 0.$$

Thus by definition of Q^{k+1} in Eq. 1, we have

$$\begin{aligned} &\Rightarrow Q^{k+1}(s,\tau) = Q_\theta(s,\tau) \\ &= \frac{\pi^k}{\pi^k + \alpha_k\mu}\mathcal{B}_\pi^k Q^k(s,\tau) + \frac{\alpha_k\mu}{\pi^k + \alpha_k\mu}\gamma\mathcal{P}_\pi^k Q^k(s,\tau) + \frac{\alpha_k\mu}{\pi^k + \alpha_k\mu}\mathbb{E}_{\tau'\sim\pi^k} r'. \end{aligned} \tag{5}$$

Note that $\mathcal{B}_\pi^k = R + \gamma\mathcal{P}_\pi^k$, and we can further simplify Q^{k+1} as

$$Q^{k+1} = \mathcal{B}_\pi^k Q^k(s,\tau) - \frac{\alpha_k\mu}{\pi^k + \alpha_k\mu}\left[R(s,\tau) - \mathbb{E}_{\tau'\sim\pi^k} r'\right]. \tag{6}$$

Now, we only need to show Eq. 6 leads to $||Q^{k+1} - Q^k||_\infty \to 0$. First, it can be shown that $||\mathcal{B}_\pi^k Q^k - \mathcal{B}_\pi^{k-1} Q^{k-1}||_\infty \leq \gamma ||Q^k - Q^{k-1}||_\infty$ following [4]. Therefore we have

$$\begin{aligned} ||Q^{k+1} - Q^k||_\infty \leq & ||\mathcal{B}_\pi^k Q^k - \mathcal{B}_\pi^{k-1} Q^{k-1}||_\infty \\ & + ||\frac{\alpha_k \mu}{\pi^k + \alpha_k \mu}[R - \mathbb{E}_{\pi^k} r'] - \frac{\alpha_{k-1}\mu}{\pi^{k-1} + \alpha_{k-1}\mu}[R - \mathbb{E}_{\pi^{k-1}} r']||_\infty \\ & \leq \gamma ||Q^k - Q^{k-1}||_\infty + 2R_{max} \left(||\frac{\alpha_{k-1}\mu}{\pi^{k-1} + \alpha_{k-1}\mu}||_\infty + ||\frac{\alpha_k \mu}{\pi^k + \alpha_k \mu}||_\infty \right). \end{aligned} \tag{7}$$

Using $\alpha_k < \left(\frac{1}{\gamma^k C} - 1\right)^{-1} \left(\frac{\pi^k}{\mu}\right)_{min}$, we have

$$\begin{aligned} ||Q^{k+1} - Q^k||_\infty & < \gamma ||Q^k - Q^{k-1}||_\infty + 2R_{max} \left(\gamma^{k-1} C + \gamma^k C\right) && (8) \\ & < \gamma \left[\gamma ||Q^{k-1} - Q^{k-2}||_\infty + 2R_{max} \left(\gamma^{k-2} C + \gamma^{k-1} C\right)\right] \\ & \quad + 2R_{max} \left(\gamma^{k-1} C + \gamma^k C\right) && (9) \\ & \cdots \\ & < \gamma^k ||Q^1 - Q^0||_\infty + 2R_{max} C (1+\gamma) k \gamma^{k-1}. && (10) \end{aligned}$$

Therefore, we have

$$\lim_{k \to \infty} ||Q^{k+1} - Q^k||_\infty = 0.$$

□.

Theorem 1. *Under the same conditions as Lemma 1, our learning procedure converges to Q^*.*

Proof. To show the sequence of Q^k converges to Q^*, we first show that Q^{k+1} is sufficiently close to the following value $\hat{Q}^{k+1}$ when k is large,

$$Q^{k+1} \to \hat{Q}^{k+1} := \left(I - \gamma \mathcal{P}_\pi^k\right)^{-1} \left[\frac{\pi^k}{\pi^k + \alpha_k \mu} R + \frac{\alpha_k \mu}{\pi^k + \alpha_k \mu} \mathbb{E}_{\pi^k} r'\right]. \tag{11}$$

To see this, we take a subtraction between $(I - \gamma \mathcal{P}_\pi^k) Q^{k+1}$ and $(I - \gamma \mathcal{P}_\pi^k) \hat{Q}^{k+1}$. We have,

$$\begin{aligned} ||\left(I - \gamma \mathcal{P}_\pi^k\right) \left(Q^{k+1} - \hat{Q}^{k+1}\right)||_\infty &= ||\gamma \mathcal{P}_\pi^k Q^k - \gamma \mathcal{P}_\pi^k Q^{k+1}||_\infty \\ &= \gamma ||\mathcal{P}_\pi^k \left(Q^k - Q^{k+1}\right)||_\infty. \end{aligned} \tag{12}$$

Note that $\mathcal{P}_\pi^k$ is the transition matrix coupled with policy π. This means that for arbitrary matrix A, $||\mathcal{P}_\pi A||_\infty \leq ||A||_\infty$. Therefore, we have

$$||\left(I - \gamma \mathcal{P}_\pi^k\right) \left(Q^{k+1} - \hat{Q}^{k+1}\right)||_\infty \leq \gamma ||\left(Q^k - Q^{k+1}\right)||_\infty \to 0. \tag{13}$$

Besides, it is also easy to see that

$$\begin{aligned} ||\Big(I-\gamma\mathcal{P}_\pi^k\Big)\Big(Q^{k+1}-\hat{Q}^{k+1}\Big)||_\infty &= ||\Big(Q^{k+1}-\hat{Q}^{k+1}\Big)-\gamma\mathcal{P}_\pi^k\Big(Q^{k+1}-\hat{Q}^{k+1}\Big)||_\infty \\ &\geq ||\Big(Q^{k+1}-\hat{Q}^{k+1}\Big)||_\infty - ||\gamma\mathcal{P}_\pi^k\Big(Q^{k+1}-\hat{Q}^{k+1}\Big)||_\infty \\ &\geq \Big(1-||\gamma\mathcal{P}_\pi^k||_\infty\Big)||\Big(Q^{k+1}-\hat{Q}^{k+1}\Big)||_\infty. \end{aligned} \tag{14}$$

Again, since $\mathcal{P}_\pi^k$ is the transition probability matrix, we know $(1-||\gamma\mathcal{P}_\pi^k||_\infty) > 0$. Hence, we have

$$\begin{aligned} &\big(1-||\gamma\mathcal{P}_\pi^k||_\infty\big)||\Big(Q^{k+1}-\hat{Q}^{k+1}\Big)||_\infty \leq \gamma||\big(Q^k-Q^{k+1}\big)||_\infty \to 0. \\ &\Rightarrow Q^{k+1}\to\hat{Q}^{k+1}. \end{aligned} \tag{15}$$

When $k\to\infty$, given this fact and $\alpha_k\to 0$, we have

$$Q^\infty = \left(I-\gamma\mathcal{P}^\infty\right)R.$$

Note that this is exactly the fixed point of the standard Bellman operator, *i.e.*, $Q^* = \mathcal{B}Q^* = R+\gamma\mathcal{P}^*Q^*$. Therefore, we know $Q^\infty = Q^*$. □.

4 Benchmark Dataset

This section provides additional details about the free-flow and targeted scenarios we use for our benchmark datasets, including how we generate and split scenarios into train, validation and test sets.

Free-flow Scenarios: Our free-flow dataset aims to model nominal traffic conditions, and consists of 7 scenario types. Differences in these scenario types include having more or less aggressive actors, actors making fewer lane changes, a larger proportion of large vehicles (e.g., trucks), faster actors, and larger variations in actor speed. Each scenario type is defined by specifying a distribution over the ego-vehicle's initial state (e.g., speed, location), actor speeds, actor classes (e.g., car, bus, truck), actor IDM [6] profile (e.g., aggressive, cautious), and actor MOBIL [2] profile (e.g., selfish, altruistic). Additional parameters configure actor density and the map (e.g., map layout, number of lanes). Sampling a free-flow scenario amounts to first uniformly sampling a scenario type and then sampling the scenario-defining parameters from the aforementioned distributions.

Targeted Scenarios: Our targeted scenario set consists of 24 distinct scenario types covering 3 common ego-routing intentions for highway driving. Scenarios corresponding to different ego intentions have different success criteria:

1. Lane Follow: Ego-vehicle must reach a goal point in the lane without deviating from the lane.

2. Lane Change: Ego-vehicle must make a lane change towards a target lane and then reach a goal point on the target lane.
3. Lane Merge: Ego-vehicle is driving on a lane that is ending and must merge into another lane.

Besides, any collision or speed limit violation happens during the scenario also accounts as a failure. To generate diverse traffic scenarios, the 3 aforementioned scenes can be combined with zero or more actors, where each actor can be scripted with 1 of 5 behavior patterns (braking, accelerating, blocking lane, cut into lane, negotiating lane change). A concrete example of a scenario type is a lane follow scenario where an actor is cutting in front of the ego-vehicle from another lane. Through varying the ego-routing intention, actor behaviors, and actor placements, we designed 24 scenario types for our targeted scenario set, which aim to cover the space of traffic interactions that would be encountered during driving.

Each scenario type is parameterized by a set of behavioral and map parameters, and an endless amount of scenario variations can be generated through varying these parameters. Behavioral parameters control the details of the interaction between the ego-vehicle and other actors, such as initial relative speeds, initial relative distances, and how an actor performs a maneuver (e.g., aggressiveness of cut-in). Map parameters control the layout of the map such as the curvature of the roads, the geometry of a merging lane, and the number of lanes.

Note that while the process manually designing scenarios require human effort (*e.g.* compared to learned or adversarial-based approaches,) we'd like to highlight that such a creation process encodes prior knowledge and makes the created scenarios more semantically meaningful, as each type of scenario targets a specific capability or requirement of autonomous driving. This ensures we have a good coverage of real-world traffic situations. Our scenarios can also adapt to different AV policies since we use intelligent actors and smart triggers which can adjust automatically depending on the AV's maneuvers.

Creating Dataset Splits: As described in the benchmark section of the main text (Section 4), we use the all-pairs methodology to construct our test set for targeted scenarios. While enumerating all possible parameter combinations thoroughly covers the scenario space, it is too expensive as the number of combinations grows exponentially with the number of configurable parameters. All-pairs produces a much smaller set by carefully selecting the combinations of parameter variations [5], *i.e.* a set that ensures all possible combinations of variations for any pair of parameters are presented. The assumption behind this approach is that many interesting behaviors can be triggered by changing a single parameter or a pair of parameters. As a result, a test set with this property provides good coverage of the input space.

However, the standard all-pairs approach assumes that all parameters are discrete, whereas many of our scenario parameters are continuous. To this end, we partition each of our continuous scenario parameters into non-overlapping buckets (a contiguous range of values). For example, the time an actor takes

to cut in front of the ego-vehicle is a continuous parameter. We can bucket the values for this parameter into $[1, 2]$ seconds, $[3, 4]$ seconds and $[5, 6]$ seconds, changing the semantics of the cut-in behavior from aggressive to mild. This essentially discretizes continuous variables into coarse-grained discrete variables, upon which the all-pairs approach can be applied. Once the discrete choice of which bucket to use has been made for a scenario's continuous parameters, we generate the exact value of each such parameter by uniform sampling within the selected bucket.

5 Metrics Breakdown

In this section we show the metrics in Table 1 of the main paper broken down by scenario types to provide more fine-grained analysis. Specifically, we categorize scenarios in our targeted set into normal, negotiating and reacting scenarios. Normal scenarios are those nominal scenarios such as lane following with normal-behaving actors (driving in their lane without any extra maneuvers) in the scene. Negotiating scenarios require negotiations with other actors, such as squeeze-in lane changes and merges. Finally, reacting scenarios are those where the ego-vehicle must react to another actor, *e.g.* an actor cutting in.

From Figure 2, we see that most methods are able to achieve low collision rate and satisfactory progress on the normal scenarios. However, for more complex negotiating and reacting scenarios, baseline methods have difficulty exhibiting safe and efficient driving maneuvers. Specifically, we can see that control signal based methods have very high collision rate on difficult scenarios, possibly due to the lack of long-term reasoning. Second, on-policy RL techniques such as PPO and A3C cannot achieve good performance. Note that although the policy learned by A3C+T has low collision, it is too conservative and does not progress very much compared to other methods. Finally, combining our trajectory based formulation and off-policy learning achieves better performance, *e.g.*, RAINBOW+T, and our TRAVL is even better with the proposed efficient learning paradigm.

6 Qualitative Results

In this section, we show several qualitative results of our learned TRAVL agent navigating in closed loop simulation. The agent is controlling the center pink vehicle. Beside each actor is the velocity in m/s. Below velocity, acceleration in m/s^2 is shown.

In Figure 3, the agent is driving in scenario where it must merge onto the highway while taking into account other actors in the scene. We see that our agent has learned to drive in complex free-flow traffic situations which mimic the real world.

In Figure 4, we see our agent in a targeted scenario which tests the ability to react to actors cutting in. We see our agent performs the correct maneuver by reacting quickly and slowing down.

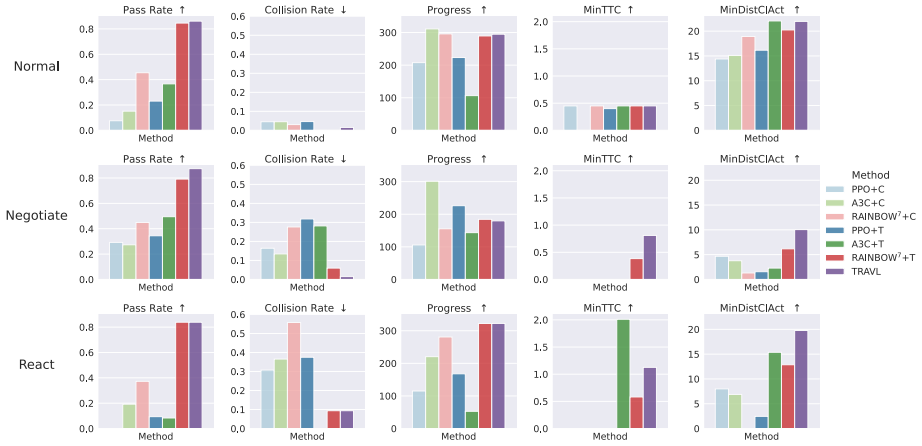

Fig. 2: Metrics broken down by scenario types. Top row shows metrics for Normal scenarios. Middle row shows metrics for Negotiating scenarios. Bottom row shows metrics for Reacting scenarios. We see that while control based methods can avoid collision for Normal scenarios, Reacting scenarios prove more challenging.

In Figure 5, the agent is tasked to squeeze between the two actors. We see the agent has learned to slow down in order to make this lane change.

In Figure 6, this scenario stress tests the agent by initializing it at a very low velocity and requiring it to merge into a crowded lane. We see the agent has learned to speed up to in order to merge into the traffic.

In Figure 7, we see a failure case of our model. In this lane change scenario, we see a fast-travelling actor decelerating. The ego-vehicle mistakenly initiates a lane change in front of that actor when that actor is still going much too fast. Once our agent realizes that the actor cannot slow down in time and that this will cause a collision, it makes a last minute adjustment to avoid collision. While collision is avoided, this is still an unsafe behavior.

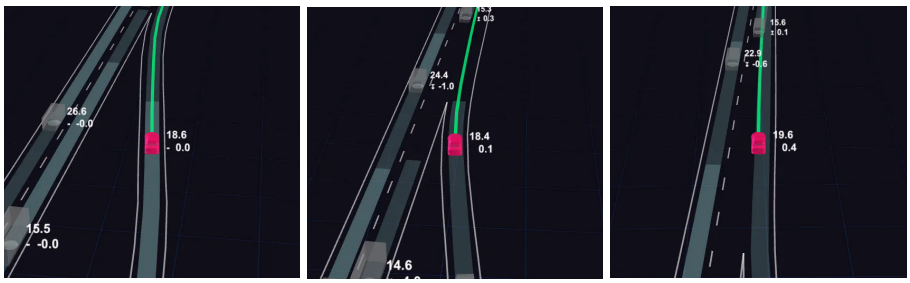

Fig. 3: Our agent successfully navigates a free-flow scenario where it must merge.

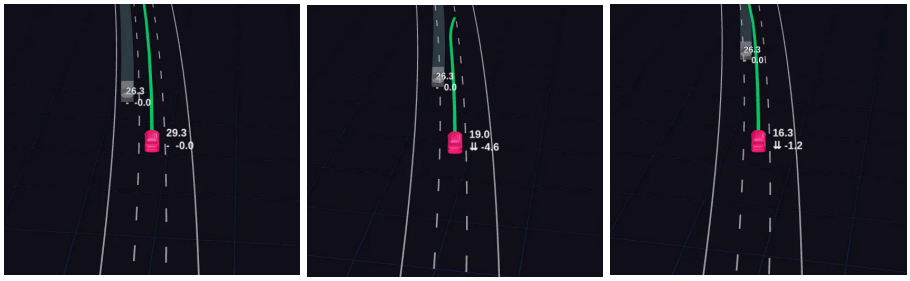

Fig. 4: Our agent reacts to an actor cutting in during a targeted scenario.

Fig. 5: Our agent slows down in order to lane change between two actors.

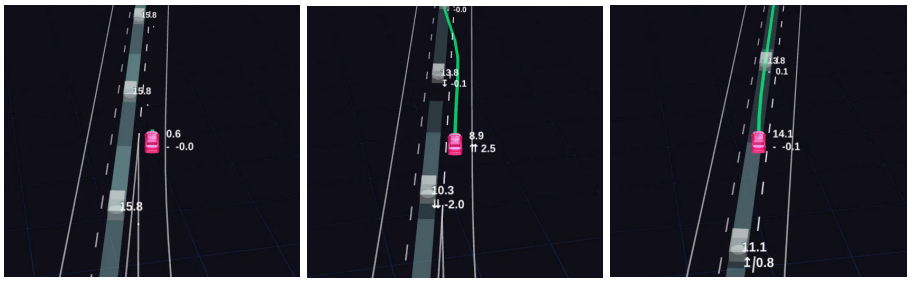

Fig. 6: Our agent is initiated with very low velocity. It has learned that it must speed up in order to merge into traffic.

Fig. 7: Here we see a failure case of our model. The agent makes a bad decision to initiate a lane change before making a last minute adjustment to avoid collision. While collision is avoided, this is still unsafe behavior.

References

1. Chen, D., Zhou, B., Koltun, V., Krähenbühl, P.: Learning by cheating. In: CoRL (2020) 3
2. Kesting, A., Treiber, M., Helbing, D.: General lane-changing model mobil for car-following models. Transportation Research Record (2007) 7
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv (2014) 3
4. Melo, F.S.: Convergence of q-learning: A simple proof. Institute Of Systems and Robotics, Tech. Rep pp. 1–4 (2001) 6
5. Microsoft: Microsoft/pict: Pairwise independent combinatorial tool 8
6. Treiber, M., Hennecke, A., Helbing, D.: Congested traffic states in empirical observations and microscopic simulations. Physical review E (2000) 7
7. Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., Urtasun, R.: End-to-end interpretable neural motion planner. In: CVPR (2019) 3