CarPlanner: Consistent Auto-regressive Trajectory Planning for Large-scale Reinforcement Learning in Autonomous Driving

Supplementary Material

A. Training Procedure

Algorithm 1 outlines the training process for the CarPlanner framework. Notably, during the training of the trajectory generator, we have the flexibility to employ either RL or IL, but, in this work, we do not combine RL and IL simultaneously, opting instead to explore their distinct characteristics separately. The definitions of the loss functions are given in the following.

Loss of non-reactive transition model. The non-reactive transition model β is trained to simulate agent trajectories based on the initial state $\boldsymbol{s}_0$. For each data sample $(\boldsymbol{s}_0, s_{1:T}^{1:N,\text{gt}}) \in \mathcal{D}$, the model predicts trajectories $s_{1:T}^{1:N} = \beta(\boldsymbol{s}_0)$, and the training objective minimizes the L1 loss:

$$L_{\text{tm}} = \frac{1}{T}\sum_{t=1}^{T}\sum_{n=1}^{N}\left\|s_t^n - s_t^{n,\text{gt}}\right\|_1. \tag{5}$$

Mode selector loss. This contains two parts: cross-entropy and side-task loss. The cross-entropy loss is defined as:

$$\text{CrossEntropyLoss}(\boldsymbol{\sigma}, c^*) = -\sum_{i=1}^{N_{\text{mode}}}\mathbb{I}(c_i = c^*)\log\sigma_i, \tag{6}$$

where σ_i is the assigned score for mode c_i, N_{mode} is the number of candidate modes, and $\mathbb{I}$ is the indicator function. The side-task loss is defined as:

$$\text{SideTaskLoss}(\bar{s}_{1:T}^0, s_{1:T}^{0,\text{gt}}) = \frac{1}{T}\sum_{t=1}^{T}\left\|\bar{s}_t^0 - s_t^{0,\text{gt}}\right\|_1, \tag{7}$$

where $\bar{s}_t^0$ is the output ego future trajectory.

Generator loss with RL. The PPO [31] loss consists of three parts: policy, value, and entropy loss. The policy loss is defined as:

$$\begin{aligned}&\text{PolicyLoss}(a_{0:T-1}, d_{0:T-1,\text{new}}, d_{0:T-1}, A_{0:T-1})\\ &= -\frac{1}{T}\sum_{t=0}^{T-1}\min\left(r_t A_t, \text{clip}(r_t, 1-\epsilon, 1+\epsilon)A_t\right),\end{aligned} \tag{8}$$

Algorithm 1 Training Procedure of CarPlanner

Input: Dataset $\mathcal{D}$ containing initial states $\boldsymbol{s}_0$ and ground-truth trajectories $s_{1:T}^{0:N,\text{gt}}$, longitudinal modes $\boldsymbol{c}_{\text{lon}}$, discount factor γ, GAE parameter λ, update interval I.
Require: Non-reactive transition model β, mode selector f_{selector}, policy π, policy old π_{old}.
Step 1: Training Transition Model
for $(\boldsymbol{s}_0, s_{1:T}^{1:N,\text{gt}}) \in \mathcal{D}$ **do**
  Simulate agent trajectories $s_{1:T}^{1:N} \leftarrow \beta(\boldsymbol{s}_0)$
  Calculate loss $L_{\text{tm}} \leftarrow \text{L1Loss}(s_{1:T}^{1:N}, s_{1:T}^{1:N,\text{gt}})$
  Backpropagate and update β using L_{tm}
end for
Step 2: Training Selector and Generator
Initialize training_step $\leftarrow 0$
Initialize policy old $\pi_{\text{old}} \leftarrow \pi$
for $(\boldsymbol{s}_0, s_{1:T}^{0,\text{gt}}) \in \mathcal{D}$ **do**
  Non-Reactive Transition Model:
  Simulate agent trajectories $s_{1:T}^{1:N} \leftarrow \beta(\boldsymbol{s}_0)$
  Mode Assignment:
  Determine $\boldsymbol{c}_{\text{lat}}$ based on $\boldsymbol{s}_0$
  Concatenate $\boldsymbol{c}_{\text{lat}}$ and $\boldsymbol{c}_{\text{lon}}$ to get $\boldsymbol{c}$
  Determine positive mode c^* based on $s_{1:T}^{0,\text{gt}}$ and $\boldsymbol{c}$
  Mode Selector Loss:
  Compute scores $\boldsymbol{\sigma}, \bar{s}_{1:T}^0 \leftarrow f_{\text{selector}}(\boldsymbol{s}_0, \boldsymbol{c})$
  $L_{\text{selector}} \leftarrow \text{CrossEntropyLoss}(\boldsymbol{\sigma}, c^*) + \text{SideTaskLoss}(\bar{s}_{1:T}^0, s_{1:T}^{0,\text{gt}})$
  Generator Loss:
  if Reinforcement Learning (RL) Training **then**
    Use π_{old}, $\boldsymbol{s}_0$, c^*, and $s_{1:T}^{1:N}$ to collect rollout data $(\boldsymbol{s}_{0:T-1}, a_{0:T-1}, d_{0:T-1}, V_{0:T-1}, R_{0:T-1})$
    Compute advantage $A_{0:T-1}$ and return $\hat{R}_{0:T-1}$ using GAE [30]: $A_{0:T-1}, \hat{R}_{0:T-1} \leftarrow \text{GAE}(R_{0:T-1}, V_{0:T-1}, \gamma, \lambda)$
    Compute policy distribution and value estimates: $(d_{0:T-1,\text{new}}, V_{0:T-1,\text{new}}) \leftarrow \pi(\boldsymbol{s}_{0:T-1}, a_{0:T-1}, c^*)$
    $L_{\text{generator}} \leftarrow \text{ValueLoss}(V_{0:T-1,\text{new}}, \hat{R}_{0:T-1}) + \text{PolicyLoss}(d_{0:T-1,\text{new}}, d_{0:T-1}, A_{0:T-1}) - \text{Entropy}(d_{0:T-1,\text{new}})$
  else if Imitation Learning (IL) Training **then**
    Use π, $\boldsymbol{s}_0$, c^*, and $s_{1:T}^{1:N}$ to collect action sequence $a_{0:T-1}$
    Stack action sequence as ego-planned trajectory $s_{1:T}^0 \leftarrow \text{Stack}(a_{0:T-1})$
    $L_{\text{generator}} \leftarrow \text{L1Loss}(s_{1:T}^0, s_{1:T}^{0,\text{gt}})$
  end if
  Overall Loss:
  $L \leftarrow L_{\text{selector}} + L_{\text{generator}}$
  Backpropagate and update f_{selector}, π using L
  Policy Update:
  Increment training_step $\leftarrow$ training_step $+1$
  if training_step $\%\ I == 0$ **then**
    Update $\pi_{\text{old}} \leftarrow \pi$
  end if
end for

where the ratio r_t is given by $r_t = \frac{\text{Prob}(a_t, d_{t,\text{new}})}{\text{Prob}(a_t, d_t)}$, $d_{t,\text{new}}$ and d_t are the policy distributions (mean and standard deviation of Gaussian distribution) at time step t induced by π and π_{old} respectively, the function $\text{Prob}(a, d)$ calculates the probability of a given action a under a distribution d, and A_t is the advantage estimated using GAE [30]. The value and entropy loss are defined as:

$$\text{ValueLoss}(V_{0:T-1,\text{new}}, \hat{R}_{0:T-1}) = \frac{1}{T}\sum_{t=0}^{T-1}\left\|V_{t,\text{new}} - \hat{R}_t\right\|_2^2, \tag{9}$$

$$\text{Entropy}(d_{0:T-1,\text{new}}) = \frac{1}{T}\sum_{t=0}^{T-1}\mathcal{H}(d_{t,\text{new}}), \tag{10}$$

where $V_{t,\text{new}}$ and $\hat{R}_t$ are the predicted and actual returns, and $\mathcal{H}$ represents the entropy of the policy distribution d.

Generator loss with IL. In IL, the generator minimizes the trajectory error between the ego-planned trajectory $s^0_{1:T}$ and the ground-truth trajectory $s^{0,\text{gt}}_{1:T}$. The loss is defined as:

$$L_{\text{generator}} = \frac{1}{T}\sum_{t=1}^{T}\left\|s_t^0 - s_t^{0,\text{gt}}\right\|_1. \tag{11}$$

B. Implementation Details

The hyperparameters of model architecture, PPO-related parameters, and loss weights are summarized in Tab. 5. The magnitudes of value, policy, and entropy loss are 10^3, 10^0, and 10^{-3}, respectively. The trajectory generator generates trajectories with a time horizon of 8 seconds at 1-second intervals, corresponding to time horizon $T = 8$. During testing, these trajectories are interpolated to 0.1-second intervals. The weight of scores generated by the rule and mode selectors is set to a ratio of $1 : 0.3$. In cases where no ego candidate trajectory satisfies the safety criteria evaluated by the rule selector, an emergency stop is triggered. For the Test14-Random benchmark, a replanning frequency of 10Hz is employed, adhering to the official nuPlan simulation configuration. In contrast, for the Reduced-Val14 benchmark, a replanning frequency of 1Hz is used to ensure a fair comparison with Gen-Drive [18].

C. Ablation Study on RL Training

In this part, we examine the training efficiency of CarPlanner, performance of vanilla and consistent auto-regressive frameworks, the use of reactive and non-reactive model in RL training, and the impact of varying the time horizon.

Training efficiency. We compare the efficiency of our model-based framework with that of ScenarioNet [24], which is an open-source platform for model-free RL training in real-world datasets [2, 9]. As shown in Tab. 6, CarPlanner achieves a remarkable improvement in sampling efficiency, outperforming ScenarioNet by two orders of magnitude. Furthermore, CarPlanner not only excels in ef-

Parameter	Value
Feature dimension D	256
Static point dimension D_m	9
Agent pose dimension D_a	10
Activation	ReLU
Number of layers	3
Number of attention heads	8
Dropout	0.1
discount factor γ	0.1
GAE parameter λ	0.9
Clip range ϵ	0.2
Update interval I	8
Weight of selector loss	1
Weight of value loss	3
Weight of policy loss	100
Weight of entropy loss	0.001
Weight of IL loss	1

Table 5. Hyperparameters of model architecture, PPO-related parameters, and loss weights.

Planner	CLS-NR (↑)	Efficiency (samples/sec, ↑)	Num. Samples	Train Time
ScenarioNet [24]	55.60	25.72	7,798,472	3d12h11m38s
CarPlanner-IL	93.41	1,181.46	70,487,200	16h34m12s
CarPlanner	**94.07**	**1,632.25**	70,487,200	11h59m44s

Table 6. Comparison of training efficiency with model-free settings. Experimental results are based on the Test14-Random non-reactive benchmark.

	Design Choices		Closed-loop metrics (↑)				
Model Type	Random Sample	Guide Reward	CLS-NR	S-CR	S-Area	S-PR	S-Comfort
Vanilla	✓	Progress	67.56 ± 0.38	90.97 ± 0.78	94.64 ± 1.72	72.17 ± 0.21	64.21 ± 1.29
	✓	DE	86.89 ± 0.28	97.34 ± 0.37	96.36 ± 0.18	89.90 ± 0.11	**94.03 ± 0.65**
Consistent	✗	FE	88.14	96.86	98.43	91.39	73.73
	✗	DE	**94.07**	**99.22**	**99.22**	**95.06**	91.09

Table 7. Comparison of vanilla and consistent auto-regressive frameworks with different guide reward design. Experimental results are based on the Test14-Random non-reactive benchmark.

Model	Loss	CLS-NR (↑)	Consistent Ratio Lat (↑)	Consistent Ratio Lon (↑)
Vanilla	RL	86.89 ± 0.28	20.00 ± 0.10	8.33 ± 0.00
PLUTO [3]	IL	91.92	62.45	41.80
Consistent	IL	93.41	68.26	43.01
Consistent	RL	**94.07**	**79.58**	**43.03**

Table 8. Comparison for consistency. Experimental results are based on the Test14-Random non-reactive benchmark.

	Closed-loop metrics (↑)				
Transition Model	CLS-NR	S-CR	S-Area	S-PR	S-Comfort
Reactive	91.03	96.92	**99.23**	91.28	90.00
Non-reactive	**94.07**	**99.22**	99.22	**95.06**	**91.09**

Table 9. Comparison of the usage of reactive and non-reactive transition models. Experimental results are based on the Test14-Random non-reactive benchmark.

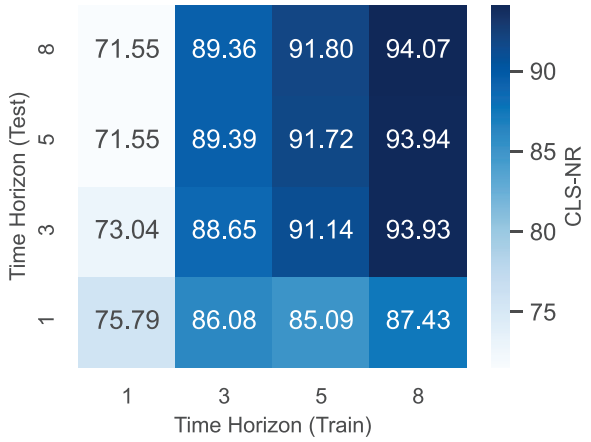


Figure 4. Performance of different training time horizons under different testing time horizons. The value in each cell is the CLS-NR metric on the Test14-Random non-reactive benchmark.

ficiency but also achieves SOTA performance, surpassing ScenarioNet by a wide margin.

Vanilla vs. consistent auto-regressive framework. The results are shown in Tabs. 7 and 8. The consistent auto-regressive framework generates multi-modal trajectories by conditioning on mode representations. In contrast, the vanilla framework relies on random sampling from the action Gaussian distribution to produce multi-modal trajectories. To ensure comparability in the number of modes generated by both frameworks, we sample 60 trajectories in parallel for the vanilla framework. Given that random sampling introduces variability, we average the results across 3 random seeds. For the consistent framework, we use displacement error (DE) and final error (FE) as guide functions to assist the policy in generating mode-aligned trajectories. For the vanilla framework, DE is compared against a progress reward, which encourages longitudinal movement along the route while discouraging excessive lateral deviations that move the vehicle too far from any possible route. The consistent ratio computes the ratio of generated trajectories that fall in their corresponding modes in longitudinal and lateral directions separately.

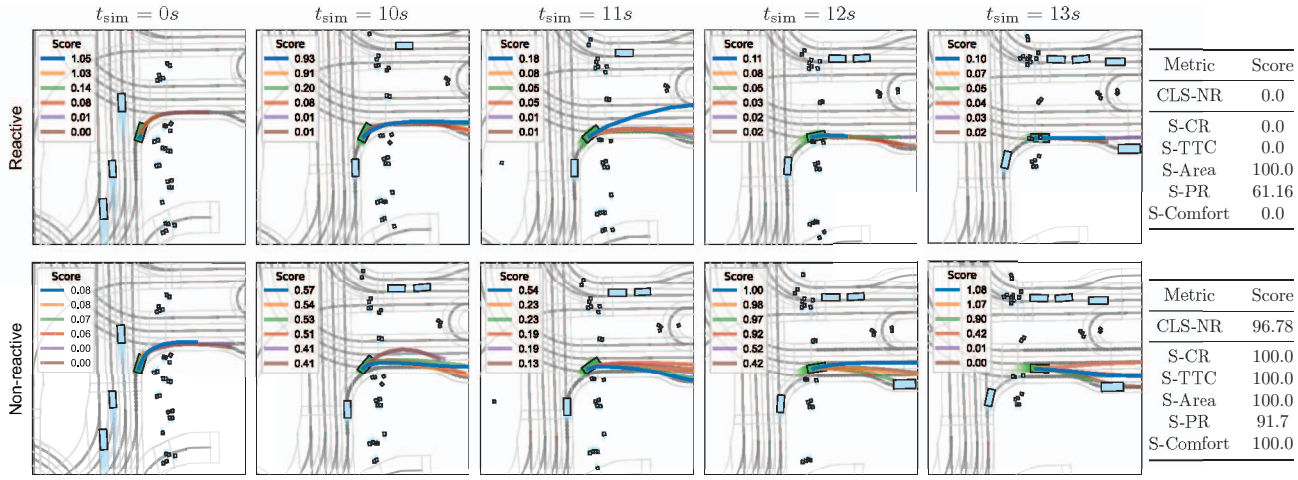
Overall, our proposed consistent framework outperforms the vanilla framework in terms of closed-loop performance, highlighting the benefits of incorporating consistency. Furthermore, RL provides more consistant trajectories than the vanilla framework and IL-based methods. Additionally, we find that DE serves as an effective guide function for policy training, further enhancing closed-loop performance.

Reactive vs. non-reactive transition model. We compare the performance of the CarPlanner framework when trained with reactive and non-reactive transition models. The reactive transition model shares a similar architecture with the auto-regressive planner for the ego vehicle, utilizing relative pose encoding [43] as the backbone network to extract features of traffic agents and predict their subsequent poses. The training loss and hyperparameters are consistent with those used for the non-reactive transition model. As shown in Tab. 9, except for the S-Area metric, using non-reactive transition model outperforms the reactive transition model in our current implementation. The primary difference lies in the assumptions about traffic agents: the reactive transition model assumes that the ego vehicle can negotiate with traffic agents and share the same priority, whereas in the non-reactive model, traffic agents do not respond to the ego vehicle, effectively assigning them higher priority. A representative example is presented in Fig. 5. When trained with the reactive transition model, the planner assumes pedestrians will yield to the vehicle, leading it to attempt to move forward. However, at $t_{\text{sim}} = 12s$, the planner collides with pedestrians, triggering an emergency brake, which negatively impacts safety, progress, and comfort metrics. Although the performance of using reactive transition model is not satisfied currently, it is a more realistic assumption and we will further investigate this in future work.

Time horizon. We evaluate the CarPlanner framework by training it with different time horizons, including 1, 3, 5, and 8 seconds, and testing the planners in each time horizon. The results in Fig. 4 confirm that increasing the time horizon has a positive effect on the performance for both training and testing. A special case is when the training time horizon is set to 1, all tested time horizons exhibit poor performance, highlighting the importance of multi-step learning in RL. Additionally, the observation that increasing the training time horizon enhances closed-loop performance suggests the potential for further improvements by extending the time horizon beyond 8 seconds. However, due to current limitations in data preparation, which is designed for horizons up to 8 seconds, expanding the time horizon would not provide map information or ground-truth trajectories, hindering further analysis. Consequently, we leave this exploration for future work.

D. Comparison with Differentiable Loss

In typical IL setting, the supervision signal provided to the trajectory generator is the displacement error (DE) between the ego-planned trajectory and the ground-truth trajectory. Several works [3, 17, 35] propose to convert non-



Metric	Score
CLS-NR	0.0
S-CR	0.0
S-TTC	0.0
S-Area	100.0
S-PR	61.16
S-Comfort	0.0

Metric	Score
CLS-NR	96.78
S-CR	100.0
S-TTC	100.0
S-Area	100.0
S-PR	91.7
S-Comfort	100.0

Figure 5. Qualitative comparison of using reactive and non-reactive transition model in non-reactive environments. The scenario is annotated as `waiting_for_pedestrian_to_cross`. In each frame shot, ego vehicle is marked as green. Traffic agents are marked as sky blue. Lineplot with blue is the ego planned trajectory.

	Supervision Signals			Closed-loop metrics (↑)					Open-loop metrics (↓)	
Loss Type	DE	Col	Area	CLS-NR	S-CR	S-Area	S-PR	S-Comfort	Col Mean [Min, Max]	Area Mean [Min, Max]
IL	✓	✗	✗	93.41	98.85	98.85	93.87	**96.15**	0.17 [**0.00**, 0.47]	0.09 [**0.00**, 0.40]
	✓	✓	✗	93.67	**99.23**	98.85	94.63	94.23	0.16 [**0.00**, 0.43]	0.07 [**0.00**, 0.27]
	✓	✗	✓	93.12	98.46	98.84	92.88	94.21	0.15 [**0.00**, 0.44]	0.08 [**0.00**, 0.30]
	✓	✓	✓	93.32	98.46	98.46	94.05	95.77	0.15 [**0.00**, 0.43]	0.09 [**0.00**, 0.39]
RL	✓	✗	✗	90.44	97.49	96.91	93.33	90.73	0.17 [**0.00**, 0.49]	0.14 [**0.00**, 0.51]
	✓	✓	✓	**94.07**	99.22	**99.22**	**95.06**	91.09	**0.12** [**0.00**, **0.39**]	**0.05** [**0.00**, **0.22**]

Table 10. Comparison with different loss types and supervision signals. Closed-loop results are based on the Test14-Random non-reactive benchmark. Open-loop results are on validation set.

differentiable metrics, such as avoiding collision (Col) and adherence to drivable area (Area), into differentiable loss functions that can directly backpropagate to the generator. In contrast, CarPlanner leverages an RL framework, which introduces surrogate objectives to indirectly optimize these non-differentiable metrics.

In this part, we compare these two approaches which provide rich supervision signals to the trajectory generator. The results are summarized in Tab. 10. In IL training, the Col and Area metrics are converted into differentiable loss functions, whereas in RL training, Col and Area are treated as reward functions, contributing to the quality reward as described in the main paper. It is important to note that the implementations for differentiable loss functions and reward functions are identical, except that gradient flow is enabled for differentiable loss functions. The open-loop metrics compute the Col and Area values across all candidate multi-modal trajectories, with the Mean, Min, and Max referring to the mean, minimum, and maximum values of the Col and Area metrics within the candidate trajectory set.

Our findings suggest that incorporating Col loss benefits the open-loop Col metric and improves the closed-loop S-CR metrics, thereby enhancing closed-loop performance. However, incorporating Area loss results in better open-loop Area metrics but deteriorates closed-loop performance. Compared to differentiable loss functions, RL with Col and Area as quality rewards yields the trajectory set with the highest overall quality, as evidenced by smaller Mean and Max metrics in open-loop metrics. This improvement can be attributed to RL's ability to optimize the reward-to-go using surrogate objectives that account for future rewards, while differentiable loss functions are limited to timewise-aligned optimization in our current implementation. This distinction is illustrated in Fig. 6: in (a), the loss at time step t is directly computed from s_t^0, meaning that during backward propagation, the loss at time step t cannot influence the optimization of prior time steps. In (b), however, the non-differentiable reward is aggregated into a return (reward-to-go), which serves as a reference for computing the loss at time step t. Through this process, the reward at

Loss Type	Model Type	Mode Type	Mode Dropout	Scorer Side Task	Ego-history Dropout	Backbone Sharing	CLS-NR	S-CR	S-Area	S-PR	S-Comfort
		Design Choices					Closed-loop metrics (↑)				
IL	Vanilla	-	-	-	✓	-	86.48	97.09	97.29	88.05	94.19
	Consistent	Lon	✗	✓	✓	✓	88.79	96.67	96.08	89.63	94.90
	Consistent	Lon-Lat	✓	✓	✓	✓	93.41	98.85	98.85	93.87	**96.15**
RL	Vanilla	-	-	-	✗	-	85.56	97.27	95.70	89.17	93.36
	Consistent	Lon	✗	✓	✗	✗	90.57	97.30	97.68	92.20	94.59
	Consistent	Lon-Lat	✓	✓	✗	✗	**94.07**	**99.22**	**99.22**	**95.06**	91.09

Table 11. Effect of different mode representations. Experimental results are based on the Test14-Random non-reactive benchmark.

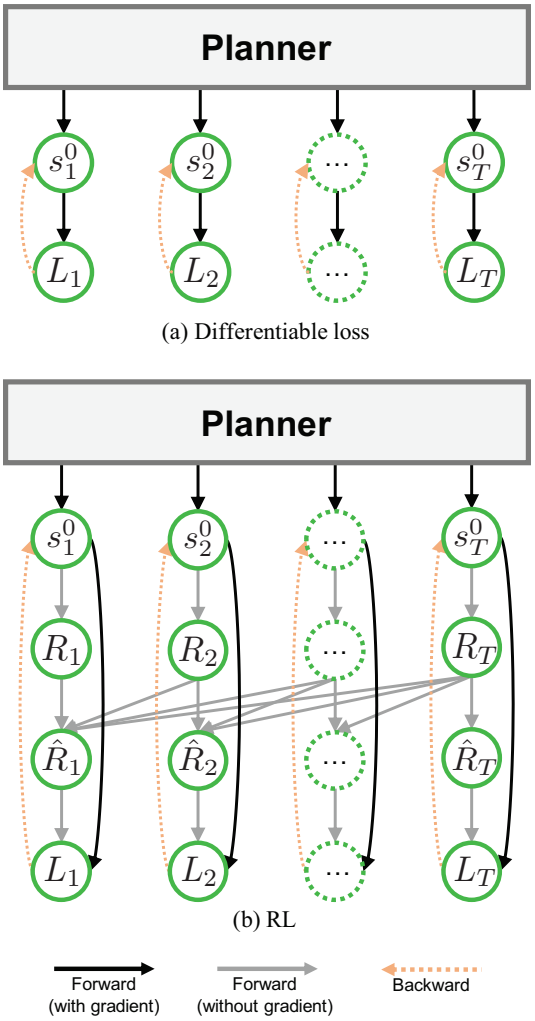


Figure 6. The computational graph of differentiable loss (a) and RL (b) framework for optimizing same metrics such as displacement errors, collision avoidance, and adherence to drivable area.

time step t can influence the trajectory at earlier time steps t' ($t' < t$). In the future, we aim to combine the advantages of differentiable loss which can provide low-variance gradients, and RL which can provide long-term foresight, by model-based RL optimization techniques [6, 13].

E. Effect of Mode Representation

In this part, we examine the impact of mode representations on performance. The results are presented in Tab. 11. For both the vanilla and consistent frameworks, we disable the use of random sampling to focus solely on mode-aligned trajectories. As a result, the vanilla framework can only generate single-modal trajectories, leading to the lowest performance. In the consistent framework, we explore two types of mode representations: Lon and Lon-Lat. The Lon representation assigns modes based on longitudinal movements along the route, whereas the Lon-Lat representation decomposes modes by both longitudinal and lateral movements. Aligned with the main paper, we use ego-history dropout and backbone sharing only for IL training. For the Lon representation, we close mode dropout since it does not rely on any map or agent representation in initial state. The results indicate that introducing consistency provides greater benefits to RL training, with the Lon-Lat representation proving to be more effective than the Lon representation. This suggests that decomposing mode representations into both longitudinal and lateral components enhances the model's ability by providing more explicit mode information.