Uncertainty-Aware Decision Transformer for Stochastic Driving Environments

Zenan Li[1,2], **Fan Nie**[2,3], **Qiao Sun**[2], **Fang Da**[4], **and Hang Zhao**[†,1,2]
[1]Tsinghua, [2]Shanghai Qi Zhi, [3]Stanford, [4]QCraft, [†]Corresponding author
Emails: *li-zn23@mails.tsinghua.edu.cn*, *niefan@stanford.edu*, *hangzhao@mail.tsinghua.edu.cn*

Abstract: Offline Reinforcement Learning (RL) enables policy learning without active interactions, making it especially appealing for self-driving tasks. Recent successes of Transformers inspire casting offline RL as sequence modeling, which, however, fails in stochastic environments with incorrect assumptions that identical actions can consistently achieve the same goal. In this paper, we introduce an **UN**certainty-awa**RE** deci**S**ion **T**ransformer (UNREST) for planning in stochastic driving environments without introducing additional transition or complex generative models. Specifically, UNREST estimates uncertainties by conditional mutual information between transitions and returns. Discovering 'uncertainty accumulation' and 'temporal locality' properties of driving environments, we replace the global returns in decision transformers with truncated returns less affected by environments to learn from actual outcomes of actions rather than environment transitions. We also dynamically evaluate uncertainty at inference for cautious planning. Extensive experiments demonstrate UNREST's superior performance in various driving scenarios and the power of our uncertainty estimation strategy.

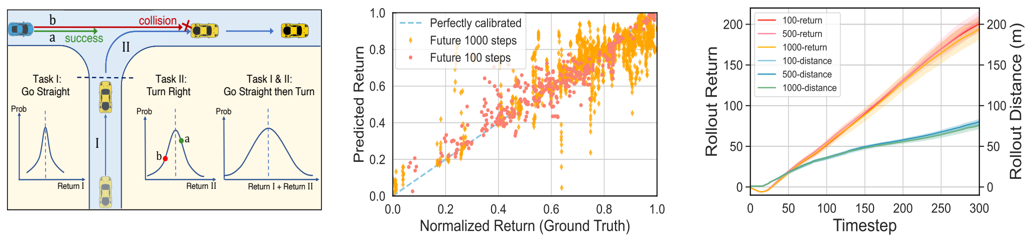
Keywords: Self-Driving, Decision Transformer, Uncertainty-Aware Planning.

1 Introduction

Safe and efficient motion planning has been recognized as a crucial component and the bottleneck in self-driving systems [1]. Nowadays, learning-based planning algorithms like imitation learning (IL) [2] and reinforcement learning (RL) [3] have gained prominence with the advent of intelligent simulators [4] and large-scale datasets [5]. Building on these, offline RL [6, 7] becomes a promising framework for safety-critical driving tasks to learn policies from offline data while retaining the ability to leverage and improve over data of various quality [8, 9]. Nevertheless, the application of offline RL approaches still faces practical challenges: (1) The driving task requires conducting *long-horizon planning* to avoid shortsighted erroneous decisions [10]; (2) The *stochasticity* of environment objects during driving also demands real-time responses to their actions [6, 11].

Recent works propose to leverage the capability of Transformers[12, 13] by reformulating offline RL as a sequence modeling problem [14], which naturally considers outcomes of multi-step decision-making and has demonstrated efficacy in long-term credit assignment [14, 15]. Typically, they train models to predict an action based on the current state and an outcome in hindsight such as a desired future return (i.e., reward-to-go). However, existing works [16, 17, 18] have pointed out that these decision transformers (DTs) tend to be overly optimistic in stochastic environments because they incorrectly assume that actions, which successfully achieve a goal once, can consistently do so in subsequent attempts. This assumption is easily broken in stochastic environments, as the goal can be achieved by accidental environment transitions. In Fig. 1(a), identical turning actions may yield entirely different outcomes w.r.t. the aggressive or cautious behavior of the surrounding blue vehicle.

The key to overcoming the problem is distinguishing between outcomes of decisions and environment transitions, and training models to pursue goals not affected by environmental stochasticity. To the best of our knowledge, only limited works [11, 17, 18] attempt to solve the problem. Generally, they fit



(a) (b) (c)

Figure 1: Motivations of UNREST. (a): Example driving scenario where the variance of return increases when accounting for multiple tasks. (b): Calibration results of return distribution over future 1,000 steps are more uncertain than 100 steps. (c): Rollout returns/distances of sequences maximizing the return of future 100, 500, and 1,000 steps in the dataset are close to each other.

a state transition model, either for sampling pessimistic actions [11] from VAEs [19], or to disentangle conditioning goals from environmental stochasticity [17, 18] by adversarial training [20]. Besides adding complexity, these methods are only applicable when transition functions and generative models can be learned adequately. This is often not the case for driving because of the uncertainty brought by complex interactions and partial observability [21, 22]. Furthermore, driving trajectories encompass stochasticity over an excessive number of timesteps, challenging VAE/adversarial training [23], and diluting the information in hindsight regarding current step decision-making.

In this paper, we initially customize DTs for stochastic driving environments without generative training. Specifically, our insight comes from Fig. 1(a): When going straight, the cumulative rewards from Task I & II (termed as the *global return*) contain too much stochastic influence to provide effective supervision. In contrast, a viable strategy involves conditioning solely on the truncated return from Task I to mitigate the impact of environmental stochasticity (less stochastic timesteps, lower return variance), which still preserves rewards over sufficient timesteps for action optimization. Experiments validate the point, and we summarize the following properties of driving environments:

Property 1 (Uncertainty Accumulation) *The impact of environmental stochasticity on the return distribution accumulates while considering more timesteps, as validated in Fig. 1(b).*

Property 2 (Temporal Locality) *Driving can be divided into independent tasks, where we only need to focus on the current task without considering those much later. Hence, optimizing future returns over a sufficiently long horizon approximates global return optimization, as shown in Fig. 1(c).*

The remaining problem is to specify the span of truncated returns. Specifically, our proposed **UN**certainty-awa**RE** deci**S**ion **T**ransformer (UNREST) quantifies the impacts of uncertainties by the conditional mutual information between transitions and returns, which bypasses the complexity associated with generative modeling. Sequences are then segmented into *certain and uncertain parts* accordingly. In 'certain parts' with minimal impact of uncertainty, we set the conditioning goal as the cumulative rewards to the segmented position (with the number of timesteps), which reflects the actual outcomes of decisions and can be generalized to attain higher rewards. In contrast, in 'uncertain parts' where the environment is highly stochastic, we disregard the erroneous information from returns (with dummy tokens as conditions) and let UNREST imitate expert actions. Dynamic uncertainty evaluation is integrated during inference for cautious planning. **Key contributions are:**

- We present UNREST, an uncertainty-aware decision transformer to apply offline RL in stochastic driving environments. Codes are public at `https://github.com/Emiyalzn/CoRL24-UNREST`.
- Recognizing properties of driving environments, we propose a model-free uncertainty measurement and segment sequences accordingly. Based on these, UNREST replaces global returns in DTs with truncated returns less affected by environments to learn from the actual outcomes of agent actions.
- We extensively evaluate UNREST on CARLA [4], where it consistently outperforms strong baselines (5.2% and 6.5% absolute driving score improvement in seen and unseen scenarios). Additional experiments also prove the efficacy of our uncertainty estimation strategy.

2 Related Works

Offline RL as Sequence Modeling: A recent line of works exploits Transformer's capability into RL by viewing offline RL as a sequence modeling problem. Typically, Decision Transformer

(DT) [14] predicts actions conditioned on target returns and history sequences. DTs naturally consider outcomes of multi-step decision-making and have demonstrated efficacy in long-term credit assignment [14]. Building on this, Trajectory Transformer (TT) [15] further exploits the capacity of Transformers through jointly predicting states, actions, and rewards and planning by beam searching. Generalized DT [24] reveals the theoretical basis of these models: they are trained conditioned on hindsight (e.g., future returns) to match the output trajectories with future information statistics.

However, in stochastic environments like autonomous driving, certain outcomes can be achieved by accidental environment transitions and, thus, cannot provide effective supervision for actions. Notably, this is in fact a more general problem to do with all goal-conditioned learning algorithms [25, 26]. To tackle this issue, ACT [27] and CGDT [28] approximate value functions as conditions without explicitly eliminating environmental stochasticity. ESPER [17] adopts adversarial training [20] to learn returns disentangled from environmental stochasticity as a condition. DoC [18] utilizes variational inference [19] to learn a latent trajectory representation, a new condition that minimizes the mutual information (so disentangled) with environment transitions. Besides, SPLT [11] leverages a conditional VAE [19] to model the stochastic environment transitions. As the driving environment contains various interactions that are difficult to model [21], our study proposes a novel uncertainty estimation strategy to customize DTs without transition or generative models. Besides, different from EDT [29] that dynamically adjusts history length during inference, we focus on segmenting sequences and replacing training conditions to address the overly optimistic problem.

Uncertainty Estimation: One typical method for uncertainty estimation is probabilistic bayesian approximation, either through dropout [30] or VAEs [31], which computes the posterior distribution of model parameters. Besides, the deep deterministic methods propose to estimate uncertainty by exploiting the implicit feature density [32]. In this work, we adopt the ensemble approach [33] widely used in the literature of RL [34], to jointly train K variance networks [35] for estimating the uncertainty of returns. Related works about vehicle planning are discussed in App. A.1.

3 Preliminary

Keeping notations concise, we use subscripts t or numbers for variables at specific timesteps, Greek letter subscripts for parameterized variables, and bold symbols for those spanning multiple timesteps.

3.1 Online and Offline RL

We consider learning in a Markov decision process (MDP) [36] denoted by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and the action space, respectively. Given states $s, s' \in \mathcal{S}$ and action $a \in \mathcal{A}$, $P(s'|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ is the state transition function and $r(s,a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ defines the reward function. Besides, $\gamma \in (0,1]$ is the discount factor. The agent takes action a at state s according to its policy $\pi(a|s) : \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$. At timestep $t \in [1,T]$, the accumulative discounted reward in the future, named return (i.e., reward-to-go), is $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$. The goal of *online RL* is to find a policy π that maximizes the total expected return: $J = \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t,a_t)} \big[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t) \big]$ by learning from the transitions (s, a, r, s') through interacting with the real environment. In contrast, *Offline RL* makes use of a static dataset with N trajectories $\mathcal{D} = \{\boldsymbol{\tau}_i\}_{i=1}^{N}$ collected by certain behavior policy π_b to learn a policy π that maximizes J, thereby avoiding safety issues during online interaction. Here $\boldsymbol{\tau} = \big\{(s_t, a_t, r_t, s'_t)\big\}_{t=1}^{T}$ is a collected interaction trajectory composed of transitions with horizon T.

3.2 Offline RL as Sequence Modeling

Following DTs [14], we pose offline RL as a sequence modeling problem where we model the probability of the sequence token x_t conditioned on all tokens prior to it: $p_\theta(x_t|\boldsymbol{x}_{<t})$, where $\boldsymbol{x}_{<t}$ denotes tokens from step 1 to $(t-1)$. DTs learn policy under a return-conditioned setting where the agent at step t receives an environment state s_t, and chooses an action a_t conditioned on the future return $R_t = \sum_{t'=t}^{T} r_{t'}$. This leads to the following trajectory representation:

$$\boldsymbol{\tau} = (R_1, s_1, a_1, R_2, s_2, a_2, ..., R_T, s_T, a_T), \tag{1}$$

with the objective to minimize the action prediction loss, i.e. maximize the action log-likelihood:

$$\mathcal{L}_{\text{DT}}(\theta) = \mathbb{E}_{\boldsymbol{\tau} \sim \mathcal{D}} \big[-\sum_{t=1}^{T} \log p_\theta(a_t | \boldsymbol{\tau}_{<t}, R_t, s_t) \big]. \tag{2}$$

This objective is the cause for DTs' limitations in stochastic environments, as it over-optimistically assumes actions in the sequence can consistently achieve the corresponding returns. At inference

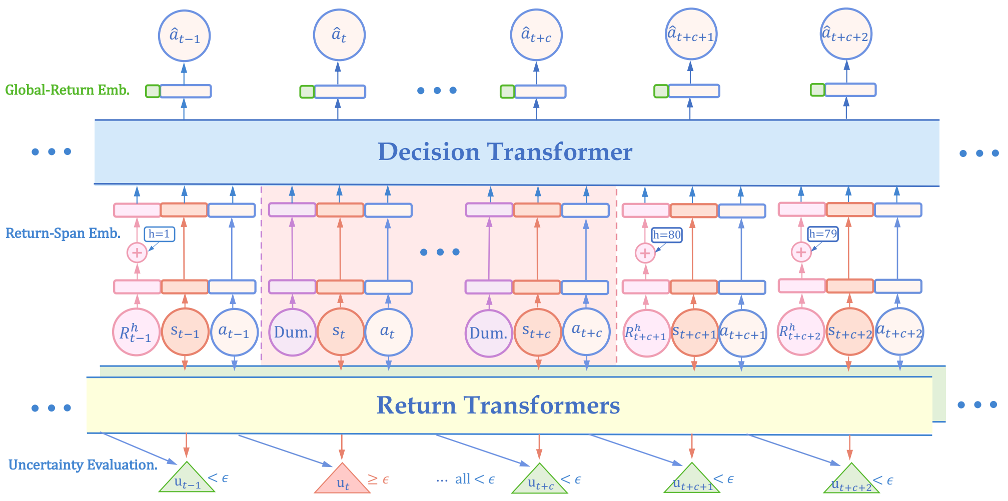


Figure 2: Overview of UNREST. **Lower:** Two return prediction transformers are trained for uncertainty estimation. The sequence is then segmented into certain (no background) and uncertain (orange background) parts w.r.t. estimated uncertainties, with 'certain parts' conditioned on returns to the next segmentation positions, and dummy tokens in 'uncertain parts'. **Upper:** The same architecture as DTs is used for action prediction, except that we add a return-span embedding to the truncated return embedding, and concatenate the global return embedding to the transformer output.

time, given a prescribed high target return, DTs generate actions autoregressively while receiving new states and rewards to update the history trajectory.

4 Approach: UNREST

4.1 Model Overview

An overview of the proposed approach UNREST is illustrated in Fig. 2. To address the overly optimistic issue, our key idea is to quantify the impact of environmental uncertainty, and learn to perform aggressively or cautiously according to different levels of environmental stochasticity.

To achieve this, we train two *return transformers* with different trajectory inputs to identify the impact of environmental uncertainty, obviating the need for complex transition and generative training. The expert sequences are then segmented into certain and uncertain parts w.r.t. estimated uncertainties, each with relabeled conditioning goals to facilitate the *decision transformer* to learn from outcomes of decisions rather than environment transitions. At test time, an *uncertainty predictor* is involved for cautious decision-making at different states. In the following, we introduce each module with details.

4.2 Transformers for Uncertainty Estimation

Instead of conventional uncertainties that reflect variances of distributions [34], in this paper we estimate the impact of environmental stochasticity, what really matters for policy learning, as an indirect measure of environmental uncertainty. In particular, we propose to model the impact of the transition $(s_{t-1}, a_{t-1} \to s_t)$ on return R_t through their conditional mutual information [37].

Specifically, two 'return transformers' are trained to approximate return distributions. Initially, states and actions are embedded by linear layers $f^s_\varphi(\cdot)$ and $f^a_\varphi(\cdot)$. The obtained embeddings are then sequentially processed by two transformers $\mathcal{T}_{\varphi_s}(\cdot)$ and $\mathcal{T}_{\varphi_a}(\cdot)$ separately for return prediction:

$$\begin{aligned} x_{s_t} &= f^s_\varphi(s_t), \quad & x_{a_t} &= f^a_\varphi(a_t), \\ \tilde{x}_{s_t} &\sim \mathcal{T}_{\varphi_s}(\dots, x_{a_{t-1}}, x_{s_t}), \quad & \tilde{x}_{a_t} &\sim \mathcal{T}_{\varphi_a}(\dots, x_{a_{t-1}}, x_{s_t}, x_{a_t}). \end{aligned} \tag{3}$$

Denoting $\tau^{\text{ret}}_{<t} = \{(s_i, a_i)\}_{i=1}^{t-1}$, two models feed their respective outputs into variance networks [35] to predict Gaussian distributions $\mathcal{N}(\cdot)$ of returns, which are optimized by maximizing log-likelihoods:

$$\begin{aligned} p_{\varphi_a}(R_t|\tau^{\text{ret}}_{<t}) &= \mathcal{N}\big(\mu_{\varphi_a}(\tilde{x}_{a_{t-1}}), \sigma_{\varphi_a}(\tilde{x}_{a_{t-1}})\big), \; \mathcal{L}_{\text{return}}(\varphi_a) = \mathbb{E}_{\tau\sim\mathcal{D}}\big[-\textstyle\sum_{t=1}^T \log p_{\varphi_a}(R_t|\tau^{\text{ret}}_{<t})\big], \\ p_{\varphi_s}(R_t|\tau^{\text{ret}}_{<t}, s_t) &= \mathcal{N}\big(\mu_{\varphi_s}(\tilde{x}_{s_t}), \sigma_{\varphi_s}(\tilde{x}_{s_t})\big), \; \mathcal{L}_{\text{return}}(\varphi_s) = \mathbb{E}_{\tau\sim\mathcal{D}}\big[-\textstyle\sum_{t=1}^T \log p_{\varphi_s}(R_t|\tau^{\text{ret}}_{<t}, s_t)\big]. \end{aligned} \tag{4}$$

Practically, the networks are implemented as ensembles, which together form Gaussian Mixture Models (GMM) [38] to better capture the return distributions as in App. A.2. Compared to challenging high-dimensional transition function learning and complex generative training, the return distributions can be better learned with fewer resources. Finally, the impact of the environmental stochasticity, i.e. the conditional mutual information $H(R_t|\boldsymbol{\tau}^{\text{ret}}_{<t}) - H(R_t|\boldsymbol{\tau}^{\text{ret}}_{<t}, s_t)$ is estimated by the KL divergence between these two distributions, as a means to measure the environmental uncertainty at timestep t:

$$u_t = D_{\text{KL}}\big(p_{\varphi_s}(R_t|\boldsymbol{\tau}^{\text{ret}}_{<t}, s_t), p_{\varphi_a}(R_t|\boldsymbol{\tau}^{\text{ret}}_{<t})\big). \tag{5}$$

A larger divergence implies more information brought by s_t for predicting R_t condition on the history, i.e. more influence on the return from the stochastic transition $(s_{t-1}, a_{t-1} \rightarrow s_t)$.

4.3 Transformer for Sequential Decision-Making

In this section, we step to aid DT training in stochastic driving environments. As discussed in Sec. 4.1, we expect UNREST to segment sequences into certain and uncertain parts according to uncertainty estimations and learn to perform aggressively or cautiously within them, respectively. To achieve this, we propose to replace the conditioning global returns with truncated returns in 'certain parts', which are less affected by the environment due to uncertainty accumulation (Prop. 1), thus reliably helping the planner generalize to higher returns after training. In contrast, in 'uncertain parts', the seemingly high return actions may cause safety issues due to environmental uncertainty. Therefore, we ignore the stochastic returns, setting conditions as dummy tokens for behavior cloning.

Segmentation strategy is thus a crucial point to reinvent DTs. To distinguish between different levels of environmental stochasticity, we define an uncertainty threshold ϵ. Next, we estimate uncertainties u_t by Eq. 5 for each timestep and record those larger than ϵ as uncertain. The sequence is then divided into certain and uncertain parts according to these marked uncertain timesteps.

An intuitive example of our segmentation strategy is illustrated in the lower part of Fig. 2. Specifically, the 'certain part' begins at a timestep with uncertainty below the threshold ϵ and persists until segmentation occurs at an uncertain timestep. Subsequently, the 'uncertain part' commences with the newly encountered uncertain timestep and encompasses subsequent ones until the final $c-1$ timesteps are all identified as certain. Since uncertain timesteps may occur intensively and intermittently over a particular driving duration (e.g., at an intersection), the hyperparameter c is introduced to avoid frequent switching between certain and uncertain parts, and ensure a minimum length of segmented sequences. Finally, we represent the segmented sequence as follows:

$$\boldsymbol{\tau}^{\text{seg}} = (h_1, R^h_1, s_1, a_1, h_2, R^h_2, s_2, a_2, ..., h_T, R^h_T, s_T, a_T), \tag{6}$$

where the conditioning return is modified as $R^h_t = \sum_{k=0}^{h_t-1} r_{t+k}$, which only involves rewards in the next h_t steps (called the return-span). In 'uncertain parts', h is always set as zero (i.e., the dummy tokens). In this way, UNREST will learn to ignore the trivial conditioning targets and directly imitate expert actions instead of being misguided by the uncertain return information. In 'certain parts', the return-span h is set to the number of timesteps from the current to the next segmentation step, so the conditioning return R^h incorporates the maximum duration that does not include any uncertain timesteps. With this segmentation and condition design, we derive the following proposition:

Proposition 1 (UNREST Alignment Bound) *Assuming that the rewards obtained are determined by transitions $(s, a \rightarrow s')$ at each timestep and UNREST is perfectly trained to fit the expert demonstrations, then the discrepancy between target truncated returns and URNEST's rollout returns is bounded by a factor of environmental stochasticity and data coverage.*

It reveals that UNREST can generalize to achieve high returns with bounded error under natural assumptions in driving environments. The formal statement and proof are left to App. B.

Notably, due to sequence segmentation, the truncated returns no longer encompass all timesteps from the present to the end of the sequence, which necessitates the return-span as an additional condition to provide information about the count of timesteps involved in returns. This enables UNREST to learn to get return R^h_t over future h_t timesteps. Otherwise, the model may be confused by the substantial differences in the magnitude of return conditions with varying timesteps [39].

Policy formulation: Unlike return transformers, DT takes the segmented sequence $\boldsymbol{\tau}^{\text{seg}}$ as input:

$$\begin{aligned} &\tilde{x}_{s_t} \sim \mathcal{T}_\theta(..., x_{R^h_t}, x_{s_t}, x_{a_t}), \text{ where} \\ &x_{R^h_t} = f^{R^h}_\theta(R^h_t) + f^h_\theta(h_t),\ x_{s_t} = f^s_\theta(s_t),\ x_{a_t} = f^a_\theta(a_t). \end{aligned} \tag{7}$$

Here we use similar notations as return transformers in Sec. 4.2 except that models are parameterized by θ. In the above formulation, a *return-span embedding* $f_\theta^h(h_t)$ is added to the return embedding, which bears the semantic meaning of how many timesteps are involved in the target return. Besides, to get the action distribution, the non-truncated *global-return embedding* $f_\theta^R(R_t)$ is optionally added to the output $\tilde{x}_{s_t}$ to provide additional longer horizon guidance for planning. Using $[\cdot \,||\, \cdot]$ to denote the concatenation of two vectors along the last dimension, the final predicted action distribution is:

$$\pi_\theta(a_t|\boldsymbol{\tau}_{<t}^{\text{seg}}, ..., s_t) = \mathcal{N}\big(\mu_\theta([\tilde{x}_{s_t}||f_\theta^R(R_t)]), \sigma_\theta([\tilde{x}_{s_t}||f_\theta^R(R_t)])\big). \tag{8}$$

The learning objective is modified from Eq. 2 of DT, with detailed training process in App. C:

$$\mathcal{L}_{\text{UNREST}}(\theta) = \mathbb{E}_{\boldsymbol{\tau}^{\text{seg}}\sim\mathcal{D}}\big[-\textstyle\sum_{t=1}^T \log \pi_\theta(a_t|\boldsymbol{\tau}_{<t}^{\text{seg}}, ..., s_t)\big]. \tag{9}$$

4.4 Uncertainty-guided Planning

During inference, we also differentiate between certain and uncertain states to account for environmental stochasticity. To enable real-time uncertainty evaluation, we introduce a lightweight uncertainty prediction model $u_\psi(\cdot)$. We dynamically query the predictor at each timestep to obtain the uncertainty measure. If the current state transition is highly uncertain, we set the conditioning target to a dummy token to facilitate cautious planning, consistent with training. Conversely, the planner acts aggressively at states with certain transitions to attain high target returns. Neural networks or heuristics can be used to implement the predictor, whose results are summarized in App. F.4. Practically, we choose the KD-Tree [40] for its computational efficiency and estimation performance, with states as tree nodes and uncertainties estimated by return transformers as node values.

Algorithm 1: Uncertainty-guided planning

```
Input: History $\boldsymbol{\tau}$, return horizon $H$, state $s_t$,
       policy $\pi_\theta$, uncertainty model $u_\psi$, return
       model $R_\varphi^h$, return percentile $\eta$.
1  # First, update target returns.
2  Update target global return $R_t \leftarrow R_{t-1} - r_{t-1}$;
3  if $h_{t-1} == 1$ or $u_\psi(s_{t-1})$ is True then
4      Reset return span to return horizon $h_t \leftarrow H$;
5      Predict target return $R_t^h \leftarrow R_\varphi^h(\boldsymbol{\tau}, h_t, s_t, \eta)$;
6  else
7      Update return span $h_t \leftarrow h_{t-1} - 1$;
8      Update target return $R_t^h \leftarrow R_{t-1}^h - r_{t-1}$;
9  # Second, evaluate new state is uncertain or not.
10 if $u_\psi(s_t)$ is True then
11     Reset conditioning target $(h_t, R_t^h) \leftarrow \emptyset$;
12 # Finally, select action using trained policies.
13 Sample and take action $a_t \sim \pi_\theta(a_t|\boldsymbol{\tau}, ..., s_t)$;
14 Update history $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau} \cup (h_t, R_t, R_t^h, s_t, a_t)$.
Output: Action $a_t$ and history $\boldsymbol{\tau}$.
```

Different from the conventional planning procedure of DTs, UNREST requires the specification of not only the target global return R_1, but also the initial return-span $h_1 = H$ and the target truncated return R_1^h. After segmentation, the effective planning horizon of the trained sequences is reduced to the return-span h_t. Once h_t reaches 1, we need to reset the target return and the return-span. Practically, we simply reset h_t to a fixed return horizon H. For the truncated return, we train a return prediction model $R_\varphi^h(\cdot)$ similar to that defined in Eq. 4 and take the upper percentile η of the predicted distribution as the new target return to attain. The hyperparameter η can be tuned for a higher target return. We do not need to consider targets at 'uncertain states' since they are just set as dummy tokens. The planning process is summarized in Alg. 1.

5 Experiments

In this section, we conduct extensive experiments to answer the following questions. **Q1:** How does UNREST perform in different driving scenarios? **Q2:** How do components of UNREST influence its overall performance? **Q3:** Does our uncertainty estimation possess interpretability?

5.1 Experiment Setup

Datasets: The offline dataset is collected from CARLA [4] with its built-in Autopilot. Specifically, we collect 30 hours of data from 4 towns under 4 weather conditions, saving tuple (s_t, a_t, r_t) at each timestep. More details about the state, action, and reward definitions are left to App. D.

Metrics: We evaluate models at training and new driving scenarios and report metrics from the CARLA challenge [4, 41] to measure their driving performance: **driving score (the most significant metric** that accounts for various indicators like driving efficiency, safety, and comfort), infraction score, route completion, and success rate. Besides, we also report the normalized rewards (the ratio of total return to the number of timesteps) to reflect driving performance at timestep level [2].

Table 1: Driving performance on train (new) town and weather conditions in CARLA. Mean and standard deviation are computed over 3 seeds. All metrics are recorded in percentages (%) except the normalized rewards. The best results are in bold and our method is colored in gray.

Planner	Driving Score↑	Success Rate↑	Route Completion↑	Infraction Score↑	Normalized Rewards↑
BC	51.9 ± 1.9 (45.6 ± 5.2)	37.9 ± 4.7 (36.3 ± 8.6)	79.7 ± 6.0 (77.1 ± 7.8)	54.5 ± 1.8 (47.0 ± 4.8)	0.63 ± 0.02 (0.61 ± 0.04)
MARWIL [42]	54.3 ± 2.0 (47.8 ± 3.4)	44.8 ± 3.2 (42.2 ± 2.2)	81.4 ± 3.2 (80.2 ± 3.8)	57.9 ± 2.0 (48.4 ± 4.4)	0.65 ± 0.02 (0.63 ± 0.01)
CQL [9]	55.0 ± 2.4 (50.7 ± 2.8)	48.6 ± 4.5 (42.8 ± 4.6)	80.5 ± 3.4 (78.7 ± 5.2)	62.4 ± 3.0 (57.7 ± 5.0)	0.65 ± 0.03 (0.62 ± 0.02)
IQL [43]	55.9 ± 3.3 (52.2 ± 3.3)	50.2 ± 6.2 (44.2 ± 2.2)	77.2 ± 4.6 (68.8 ± 4.2)	68.4 ± 2.4 (62.6 ± 6.2)	0.66 ± 0.03 (0.60 ± 0.01)
DT [14]	55.2 ± 2.0 (47.6 ± 1.2)	48.0 ± 4.7 (46.4 ± 0.3)	82.6 ± 1.0 (81.8 ± 4.9)	57.4 ± 1.3 (47.3 ± 3.4)	0.66 ± 0.01 (0.64 ± 0.02)
TT [15]	58.3 ± 3.3 (54.8 ± 2.2)	45.9 ± 5.2 (52.0 ± 4.4)	78.4 ± 5.8 (77.0 ± 4.6)	63.6 ± 3.6 (56.0 ± 5.0)	**0.74 ± 0.02** (0.64 ± 0.03)
SPLT [11]	57.8 ± 4.9 (56.4 ± 6.1)	30.1 ± 8.1 (39.5 ± 9.5)	36.7 ± 8.7 (48.2 ± 9.7)	**73.9 ± 1.4 (70.7 ± 8.3)**	0.55 ± 0.03 (0.57 ± 0.05)
ESPER [17]	54.8 ± 2.0 (51.2 ± 3.1)	48.5 ± 4.3 (53.3 ± 2.0)	77.3 ± 5.4 (84.7 ± 1.4)	56.2 ± 2.2 (46.9 ± 5.6)	0.64 ± 0.04 (0.63 ± 0.02)
ACT [27]	57.5 ± 2.7 (55.2 ± 4.3)	49.1 ± 4.8 (53.5 ± 3.3)	77.9 ± 4.5 (85.2 ± 3.6)	61.8 ± 2.5 (53.2 ± 4.2)	0.66 ± 0.03 (0.65 ± 0.05)
DoC [18]	56.9 ± 3.1 (54.4 ± 2.3)	49.9 ± 5.2 (54.0 ± 4.2)	79.2 ± 3.6 (84.0 ± 2.8)	60.3 ± 2.4 (51.5 ± 4.8)	0.64 ± 0.03 (0.65 ± 0.04)
UNREST	**63.5 ± 3.2 (62.9 ± 4.0)**	**54.5 ± 7.0 (57.5 ± 5.4)**	**83.8 ± 3.1 (90.0 ± 6.0)**	70.2 ± 2.8 (62.9 ± 3.8)	0.64 ± 0.04 **(0.65 ± 0.03)**
Expert [4]	74.0 ± 6.0 (75.3 ± 1.3)	65.4 ± 8.8 (68.6 ± 5.1)	84.2 ± 4.6 (95.8 ± 1.1)	82.8 ± 3.2 (77.5 ± 1.6)	0.72 ± 0.01 (0.69 ± 0.01)

Baselines: First, we choose two IL baselines: BC and MARWIL [42]. We also include state-of-the-art offline RL baselines: CQL [9] and IQL [43]. Besides, we compare two classic Transformer-based offline RL algorithms: DT [14] and TT [15]. We also select ACT [27] as a baseline, which approximates value functions as the hindsight condition. Finally, we adopt three algorithms as rigorous baselines: SPLT [11], ESPER [17], and DoC [18]. These algorithms fit state transition models and employ generative training to mitigate DTs' limitations in stochastic environments.

5.2 Driving Performance

We evaluate all the models' (baselines inherited from their public implementations) performance at training (Town03) and new (Town05) scenarios (**Q1**). The results are summarized in Tab. 1.

Analyzing the results, we first notice that DT performs worst among all sequence models, with a gap in driving score compared to IQL. We attribute this to the uncertainty of global return it conditions on. ACT gains better performance through conditioning on approximated value functions. However, it does not fundamentally decouple the effects of environmental stochasticity, resulting in suboptimal infraction scores. ESPER and DoC also perform poorly in both scenarios, which may result from ineffective generative training from long-horizon and complex driving demonstrations.

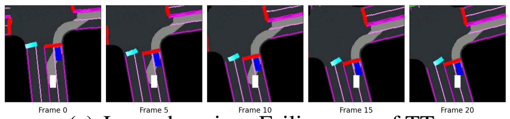
To tackle stochasticity, SPLT learns to predict the worst state transitions and achieves the best infraction score. However, its overly cautious planning process leads it to stand still in many scenarios like Fig. 3(c), resulting in an extremely poor route completion rate and normalized rewards. TT instead learns a transition model regardless of environmental stochasticity and behaves aggressively. It attains the highest normalized rewards at training scenarios since the metric prioritizes planners that rapidly move forward (but with low cumulative rewards because of its short trajectory length caused by frequent infractions). In unseen driving scenarios, TT often misjudges the speed of preceding vehicles, resulting in collisions and lower normalized rewards (than ours) as in Fig. 3(a).

Notably, UNREST achieves the best driving score, route completion, and success rate in both scenarios, and the highest normalized rewards in new scenarios without the need to learn transition or generative models. Its driving score surpasses the strongest baselines, TT over 5% in training scenarios, and SPLT over 6% in new scenarios, achieving a reasonable balance between aggressive and cautious behavior. It indicates that the truncated returns successfully mitigate the impacts of uncertainty and provide effective supervision. In App. F.2 we validate that UNREST runs significantly faster while consuming less memory than TT and SPLT, which additionally fit transition models. More results like uncertainty calibration, hyperparameters (e.g. the uncertainty threshold ϵ), environments beyond driving (e.g. D4RL), and new policies other than BC at uncertain states are studied in App. F.

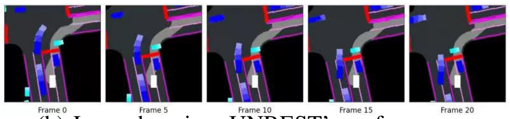
5.3 Ablation Study

In this section, we conduct ablation experiments by removing key components (global return embedding, return-span embedding, ensemble-based uncertainty estimation, and the uncertainty-guided planning process) to explore their impacts on UNREST (**Q2**). Results are shown in Tab. 2 and 3.

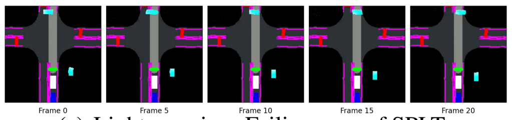
Eliminating any part of the four components harms the driving score in new scenarios. Among them, the global return embedding has the slightest impact, which suggests that the highly uncertain global return does not provide much new information over the truncated return due to temporal locality (Prop. 2). When the return-span embedding is removed, the absolute driving score drops by about 6%.



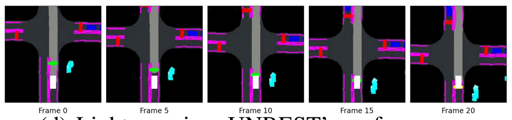
(a) Lane changing: Failing case of TT



(b) Lane changing: UNREST's performance



(c) Light crossing: Failing case of SPLT



(d) Light crossing: UNREST's performance

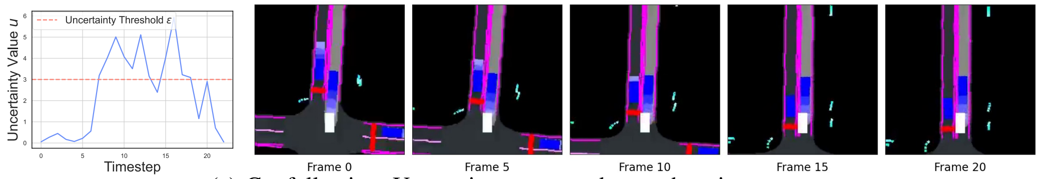
Figure 3: UNREST performs well at failing cases of TT and SPLT. White rectangles are ego-vehicles.

Table 2: Ablation study results for UNREST on train town and train weather conditions.

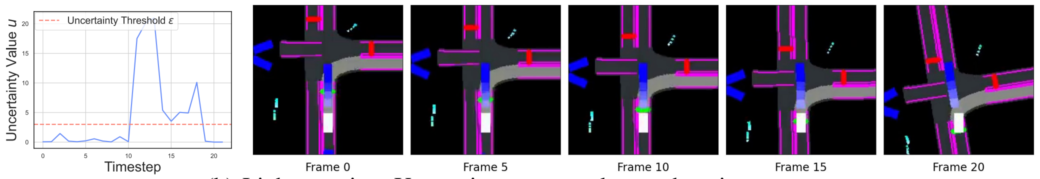
Planner	Driving Score↑	Success Rate↑	Route Co.↑	Infraction Score↑	Norm. Rewards↑
W/o global emb.	**64.5 ± 2.8**	**55.8 ± 6.0**	80.2 ± 4.2	68.0 ± 1.6	0.63 ± 0.03
W/o ret-span emb.	57.0 ± 1.8	33.3 ± 4.5	47.6 ± 3.3	65.6 ± 3.1	0.57 ± 0.01
W/o ensemble	60.4 ± 3.4	51.3 ± 6.3	80.2 ± 4.4	65.5 ± 3.1	0.62 ± 0.03
W/o guided plan	55.1 ± 2.3	42.1 ± 1.5	52.6 ± 6.0	65.2 ± 1.7	0.57 ± 0.02
Full model	63.5 ± 3.2	54.5 ± 7.0	**83.8 ± 3.1**	**70.2 ± 2.8**	**0.64 ± 0.04**

Table 3: Ablation study results for UNREST on new town and new weather conditions.

Planner	Driving Score↑	Success Rate↑	Route Co.↑	Infraction Score↑	Norm. Rewards↑
W/o global emb.	61.8 ± 3.3	55.0 ± 3.8	82.8 ± 3.4	57.8 ± 2.7	0.64 ± 0.03
W/o ret-span emb.	56.2 ± 4.7	47.6 ± 3.7	78.9 ± 5.4	58.2 ± 3.4	0.59 ± 0.02
W/o ensemble	57.4 ± 2.8	48.8 ± 4.1	85.4 ± 3.8	56.7 ± 2.7	0.61 ± 0.04
W/o guided plan	54.0 ± 3.0	54.5 ± 2.7	81.8 ± 3.8	57.3 ± 3.1	0.59 ± 0.03
Full model	**62.9 ± 4.0**	**57.5 ± 5.4**	**90.0 ± 6.0**	**62.9 ± 3.8**	**0.65 ± 0.03**



(a) Car following: Uncertainty curve and state changing process



(b) Light crossing: Uncertainty curve and state changing process

Figure 4: Visualizations of UNREST's uncertainty estimation results.

This implies that the introduction of return-span embedding provides necessary information about the timesteps needed to achieve the target return. Removing the ensemble of return transformers induces a significant performance drop in both scenarios, because a simple Gaussian distribution cannot express the return distribution well. Finally, canceling test time uncertainty estimation causes a substantial decline in driving and infraction scores, which proves the importance of cautious planning.

5.4 Uncertainty Visualization

Finally, we verify the interpretability of our uncertainty estimation through visualizations (Q3). In Fig. 4(a), we observe an increase in uncertainty as the ego-vehicle enters the lane, owing to the lack of knowledge about the other vehicle's behavior. This uncertainty decreases below the threshold when the vehicle stabilizes in the following states. Fig. 4(b) shows a green light crossing scenario. While approaching the light, the unpredictable light state causes the uncertainty value to rise quickly.

6 Conclusion and Limitations

The paper presents UNREST, an uncertainty-aware decision transformer to apply offline RL in stochastic driving environments. Specifically, we propose a novel uncertainty measurement by computing the divergence of return prediction models, bypassing complex transition and generative training. Then, based on properties we discover at driving, we segment sequences w.r.t. estimated uncertainty and adopt truncated returns as conditioning goals. This new condition helps UNREST learn policies that are less affected by stochasticity. Dynamic uncertainty estimation is also integrated at inference for cautious planning. Empirical results demonstrate UNREST's superior performance, lower resource occupation, and effective uncertainty estimation in various driving scenarios.

One limitation of UNREST is that its inference process is somewhat complex with auxiliary models and hyperparameters. A possible improvement direction is directly integrating return and uncertainty predictions into one single model. Besides, though UNREST is evaluated in the CARLA simulator, we believe it can surmount the sim-to-real gap and benefit practical autonomous driving.

Acknowledgments

We thank all the authors for their valuable contributions throughout the paper. This work started when Zenan Li interned at the Shanghai Qi Zhi Institute. Thanks to Shanghai Qi Zhi Institute and QCraft Inc. for their support in computation resources.

References

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[2] A. Hu, G. Corrado, N. Griffiths, Z. Murez, C. Gurau, H. Yeo, A. Kendall, R. Cipolla, and J. Shotton. Model-based imitation learning for urban driving. In *Advances in Neural Information Processing Systems*, 2022.

[3] J. Chen, B. Yuan, and M. Tomizuka. Model-free deep reinforcement learning for urban autonomous driving. In *2019 IEEE intelligent transportation systems conference (ITSC)*, pages 2765–2771. IEEE, 2019.

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[5] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

[6] C. Diehl, T. Sievernich, M. Krüger, F. Hoffmann, and T. Bertran. Umbrella: Uncertainty-aware model-based offline reinforcement learning leveraging planning. *arXiv preprint arXiv:2111.11097*, 2021.

[7] J. Li, C. Tang, M. Tomizuka, and W. Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *arXiv preprint arXiv:2205.11790*, 2022.

[8] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

[9] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[10] C. Zhang, R. Guo, W. Zeng, Y. Xiong, B. Dai, R. Hu, M. Ren, and R. Urtasun. Rethinking closed-loop training for autonomous driving. In *European Conference on Computer Vision*, pages 264–282. Springer, 2022.

[11] A. R. Villaflor, Z. Huang, S. Pande, J. M. Dolan, and J. Schneider. Addressing optimism bias in sequence modeling for reinforcement learning. In *International Conference on Machine Learning*, pages 22270–22283. PMLR, 2022.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[13] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[14] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[15] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[16] D. Brandfonbrener, A. Bietti, J. Buckman, R. Laroche, and J. Bruna. When does return-conditioned supervised learning work for offline reinforcement learning? In *Advances in Neural Information Processing Systems*, 2022.

[17] K. Paster, S. McIlraith, and J. Ba. You can't count on luck: Why decision transformers fail in stochastic environments. *Advances in neural information processing systems*, 2022.

[18] S. Yang, D. Schuurmans, P. Abbeel, and O. Nachum. Dichotomy of control: Separating what you can control from what you cannot. In *The Eleventh International Conference on Learning Representations*, 2022.

[19] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

[20] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32, 2019.

[21] Q. Sun, X. Huang, J. Gu, B. C. Williams, and H. Zhao. M2i: From factored marginal trajectory prediction to interactive prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6543–6552, 2022.

[22] K. P. Murphy. A survey of pomdp solution techniques. *environment*, 2(10), 2000.

[23] Y. Wu, Y. Burda, R. Salakhutdinov, and R. Grosse. On the quantitative analysis of decoder-based generative models. In *International Conference on Learning Representations*, 2017.

[24] H. Furuta, Y. Matsuo, and S. S. Gu. Generalized decision transformer for offline hindsight information matching. In *International Conference on Learning Representations*, 2022.

[25] B. Eysenbach, S. Udatha, R. R. Salakhutdinov, and S. Levine. Imitating past successes can be very suboptimal. *Advances in Neural Information Processing Systems*, 35:6047–6059, 2022.

[26] M. Štrupl, F. Faccio, D. R. Ashley, J. Schmidhuber, and R. K. Srivastava. Upside-down reinforcement learning can diverge in stochastic environments with episodic resets. *arXiv preprint arXiv:2205.06595*, 2022.

[27] C.-X. Gao, C. Wu, M. Cao, R. Kong, Z. Zhang, and Y. Yu. Act: Empowering decision transformer with dynamic programming via advantage conditioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12127–12135, 2024.

[28] Y. Wang, C. Yang, Y. Wen, Y. Liu, and Y. Qiao. Critic-guided decision transformer for offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 15706–15714, 2024.

[29] Y.-H. Wu, X. Wang, and M. Hamaya. Elastic decision transformer. *arXiv preprint arXiv:2307.02484*, 2023.

[30] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[31] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

[32] G. Franchi, X. Yu, A. Bursuc, E. Aldea, S. Dubuisson, and D. Filliat. Latent discriminant deterministic uncertainty. In *European Conference on Computer Vision*, pages 243–260. Springer, 2022.

[33] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[34] Y. Wu, S. Zhai, N. Srivastava, J. M. Susskind, J. Zhang, R. Salakhutdinov, and H. Goh. Uncertainty weighted actor-critic for offline reinforcement learning. In *International Conference on Machine Learning*, pages 11319–11328. PMLR, 2021.

[35] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

[36] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[37] M. Seitzer, B. Schölkopf, and G. Martius. Causal influence detection for improving efficiency in reinforcement learning. *Advances in Neural Information Processing Systems*, 34:22905–22918, 2021.

[38] V. Mai, K. Mani, and L. Paull. Sample efficient deep reinforcement learning via uncertainty estimation. In *International Conference on Learning Representations*, 2022.

[39] K. Hu, R. C. Zheng, Y. Gao, and H. Xu. Decision transformer under random frame dropping. In *The Eleventh International Conference on Learning Representations*, 2023.

[40] S. J. Redmond and C. Heneghan. A method for initialising the k-means clustering algorithm using kd-trees. *Pattern recognition letters*, 28(8):965–973, 2007.

[41] C. team. Carla autonomous driving leaderboard. 2020. URL `https://leaderboard.carla.org/`.

[42] Q. Wang, J. Xiong, L. Han, H. Liu, T. Zhang, et al. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 31, 2018.

[43] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[44] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, et al. Making bertha drive—an autonomous journey on a historic route. *IEEE Intelligent transportation systems magazine*, 6(2):8–20, 2014.

[45] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019.

[46] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[47] V. Kuleshov, N. Fenner, and S. Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International conference on machine learning*, pages 2796–2804. PMLR, 2018.

[48] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15222–15232, 2021.

[49] H. Xu, X. Zhan, and X. Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8753–8760, 2022.

[50] Q. Li, Z. Peng, and B. Zhou. Efficient learning of safe driving policy via human-ai copilot optimization. *arXiv preprint arXiv:2202.10341*, 2022.

Supplementary Materials

A Additional Related Works

A.1 Planning for Self-Driving

Most autonomous driving systems [44] adopt a modular pipeline to break down the massive driving task into a set of submodules, with planning being one of the most fundamental components. The objective of motion planning is to efficiently drive the ego-vehicle to the destination while conforming to safety and comfort constraints, which is essentially a decision problem. Often engineers manually design rules [44] for specific driving scenarios, which struggles to scale for complex tasks. In contrast, IL is the earliest and most widely [2, 45] used learning-based planning algorithm, which learns from offline data for either a cost function [45] or an executable policy [2]. However, they are strictly limited by expert quality [46] and will perform poorly when encountering out-of-distribution (OOD) states [46]. RL algorithms [3] own better performance and generalizability with the downside of trial-and-error learning. To address these shortcomings, we employ offline RL as an alternative to train our planner.

A.2 Uncertainty Estimation

Variance networks estimate uncertainty by loss attenuation [35]. Treating the output as a Gaussian distribution $\mathcal{N}(\cdot)$, given input x and its label y, the network outputs mean $\mu(x)$ and variance $\sigma^2(x)$ at its final layer. The network parameter ϕ is subsequently optimized by maximizing sample log-likelihood:

$$\begin{aligned}\mathcal{L}_{\text{varnet}}(\phi) &= \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[-\log \mathcal{N}_\phi(y|x)\big]\\ &= \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\frac{\left((\mu_\phi(x)-y^2\right)}{\sigma_\phi^2(x)} + \ln\sigma_\phi^2(x)\right].\end{aligned} \tag{10}$$

As the variance predicted by neural networks may be not well-calibrated or overestimated [47], in this paper we practically train an ensemble of K variance networks with data drawn from different subsets of the dataset (implemented by masking) and estimate the variance according to [38]:

$$\sigma^2 = \textstyle\sum_{l=1}^K \sigma_l^2 + \mu_l^2 - \mu^2, \text{ where } \mu = \sum_{l=1}^K \mu_k/K, \tag{11}$$

which has been proven to have strong estimation capability [38].

B Proof for Proposition 1

We next formally state our theorem, followed by detailed proofs of the proposition.

B.1 Theorem Statement

Before stepping into the main theorem, we first introduce the problem setting that we undertake for the sake of concise proof. First, we assume the reward r_t is within $[0,1]$ at each timestep for conveniently bounding the error. Besides, we use $g(s_1, a_{1:h})$ to denote the cumulative rewards by rolling out the open loop sequence of actions $a_{1:h}$ under the deterministic dynamics $P : \mathcal{S}\times\mathcal{A}\to\mathcal{S}$ and reward function $r : \mathcal{S}\times\mathcal{A}\to[0,1]$ (so that the maximum return difference over future h timesteps is h). Moreover, $J_h(\pi)$ represents the rollout rewards of policy π over future h timesteps. Next, we formally introduce the assumption, theorem and its corresponding proof, which is inspired by the reference [16]:

Theorem 1 (UNREST Alignment Bound) *Consider an MDP* $(\mathcal{S},\mathcal{A},P,r,\gamma)$*, expert behavior* $\beta : \mathcal{S}\times\mathcal{A}\to[0,1]$ *and conditioning function* $f : \mathcal{S}\times\mathbb{N}^+\to\mathbb{R}$*. Assume the following:*

1. *Return Coverage:* $p_\beta(g = f(s_1, h)|s_1) \geq \alpha_f$ *for the initial state* s_1 *and return-span* h.
2. *Near Determinism:* $p(r \neq r(s,a)$ *or* $s' \neq P(s,a)|s,a) \leq \delta$ *at all state-action pairs* (s,a) *for dynamics* P *and reward function* r.
3. *Consistency of* f*: f(s,h)=f(s',h-1) for all states s.*

Then (URT shorted for UNREST):

$$\mathbb{E}_{s_1,h}\big[f(s_1,h)\big] - J_h(\pi_f^{URT}) \leq \delta(\frac{1}{\alpha_f} + 2)h^2. \tag{12}$$

As shown by the theorem, the error between the specified target and UNREST's rollout return is bounded by the factor of environment determinism δ, data coverage α, and the horizon h.

Based on the theorem, we aim to demonstrate the generalization ability of UNREST at the identified 'certain states' by our uncertainty estimation stra. Specifically, we have the following lemma:

Lemma 1 (Determinism Equality) *Assuming that the rewards obtained are uniquely determined by transitions* $(s, a \to s')$ *at each timestep, then there exists* $\epsilon > 0$ *and* $\delta > 0$*, such that:*

$$D_{KL}\big[p(R_t|\boldsymbol{\tau}_{<t}) \mid\mid p(R_t|s_t, \boldsymbol{\tau}_{<t})\big] \leq \epsilon \Longleftrightarrow p(r \neq r(s,a) \text{ or } s' \neq P(s,a)|s,a) \leq \delta. \tag{13}$$

While the reverse direction is easy to see (the state transition probability is nearly deterministic so the transitions cannot provide additional information for return prediction), here we focus on proving the forward direction. Let us demonstrate its contrapositive proposition:

$$\text{If } p(r \neq r(s,a) \text{ or } s' \neq P(s,a)|s,a) > \delta, \text{ then } \exists\epsilon, \text{ s.t. } D_{\text{KL}}\big[p(R_t|\boldsymbol{\tau}_{<t}) \mid\mid p(R_t|s_t, \boldsymbol{\tau}_{<t})\big] > \epsilon. \tag{14}$$

First, since r is uniquely (different transitions lead to distinct rewards) determined by $(s, a \to s')$, we can induce that $r \neq r(s,a)$ is equivalent with $s' \neq P(s,a)$ as the only difference can occur in s'. Therefore, we need to prove:

$$\text{If } p(r \neq r(s,a)|s,a) > \delta, \text{ then } \exists\epsilon, \text{ s.t. } D_{\text{KL}}\big[p(R_t|\boldsymbol{\tau}_{<t}) \mid\mid p(R_t|s_t, \boldsymbol{\tau}_{<t})\big] > \epsilon. \tag{15}$$

Considering the extreme case, set $\gamma \to 0$, thus we have $R_t \to r_t$. From the perspective of proof by contradiction, the KL-divergence cannot be smaller than any $\epsilon > 0$, since $p(r_t|\boldsymbol{\tau}_{>t})$ and $p(r_t|s_t, \boldsymbol{\tau}_{>t})$ must differ for that the probability $r_t \neq r(s_{t-1}, a_{t-1})$ is larger than 0 and can be determined by s_t. To this end, we have proved the contrapositive proposition of the forward direction, which is equivalent to the original proposition.

Therefore, we can summarize from the above theorem and lemma that UNREST can generalize to achieve high returns at 'certain states' as long as the corresponding actions are covered by the expert.

B.2 Proof for the Theorem

The next theorem proof is largely built on [16]. Note that we omit the superscript 'URT' to simplify the equations. First, we expand the left term in Eq. 12:

$$\begin{aligned}
\mathbb{E}_{s_1,h}\big[f(s_1,h)\big] - J_h(\pi_f) &= \mathbb{E}_{s_1}\big[\mathbb{E}_{\pi_f|s_1}\big[f(s_1,h) - g_1\big] \\
&= \mathbb{E}_{s_1,h}\Big[\mathbb{E}_{a_{1:h}\sim\pi_f|s_1}\big[f(s_1,h) - g(s_1,a_{1:h})\big]\Big] \\
&\quad + \mathbb{E}_{s_1,h}\Big[\mathbb{E}_{a_{1:h}\sim\pi_f|s_1}\big[g(s_1,a_{1:h}) - g_1\big]\Big] \\
&\leq \mathbb{E}_{s_1,h}\Big[\mathbb{E}_{a_{1:h}\sim\pi_f|s_1}\big[f(s_1,h) - g(s_1,a_{1:h})\big]\Big] + \delta h^2.
\end{aligned} \tag{16}$$

The last step follows by bounding the difference between g_1 and $g(s_1, a_{1:h})$ by the maximum return difference h and a union of probability bound over h timesteps based on the near determinism assumption:

$$h \cdot \sup_{s_1} \bigcup_{t}^{h} p_{a_t\sim\pi_f|s_1}\big(r_t \neq r(s_t,a_t) \text{ or } s_{t+1} \neq P(s_t,a_t)\big) \leq \delta h^2. \tag{17}$$

For the first term in Eq. 16, it can also be expressed and bounded by the maximum return difference:

$$\mathbb{E}_{s_1,h}\Big[\mathbb{E}_{a_{1:h}\sim\pi_f|s_1}\big[f(s_1,h)-g(s_1,a_{1:h})\big]\Big] \leq \mathbb{E}_{s_1,h}\int_{a_{1:h}} p_{\pi_f}(a_{1:h}|s_1)\mathbb{I}\big[g(s_1,a_{1:h}) \neq f(s_1,h)\big]h. \tag{18}$$

To bound this term, we need to further expand the distribution p_{π_f}. First, with the assumption that UNREST is perfectly fitted to the expert dataset, we can deduce the following by the bayesian law:

$$\pi_f(a_1|s_1) = \beta(a_1|s_1)\frac{p_\beta(g_1 = f(s_1,h)|s_1,a_1)}{p_\beta(g_1 = f(s_1,h)|s_1)}. \tag{19}$$

For simplification, we use $\bar{s}_t = P(s_1, a_{1:t-1})$ to denote the state reached by following the deterministic dynamics defined by P till timestep t. Next, based on the near determinism, consistency and coverage assumption, we can expand p_{π_f} to get:

$$\begin{aligned}
&p_{\pi_f}(a_{1:h} \mid s_1) = \pi_f(a_1 \mid s_1)\int_{s_2} p(s_2 \mid s_1, a_1)\, p_{\pi_f}(a_{2:h} \mid s_1, s_2)\\
&\leq \pi_f(a_1 \mid s_1)\, p_{\pi_f}(a_{2:h} \mid s_1, \bar{s}_2) + \delta\\
&= \beta(a_1 \mid s_1)\frac{p_\beta(g_1 = f(s_1,h) \mid s_1, a_1)}{p_\beta(g_1 = f(s_1,h) \mid s_1)} p_{\pi_f}(a_{2:h} \mid s_1, \bar{s}_2) + \delta\\
&\leq \beta(a_1 \mid s_1)\frac{\delta + p_\beta(g_1 - r(s_1,a_1) = f(s_1,h) - r(s_1,a_1) \mid s_1, a_1, \bar{s}_2)}{p_\beta(g_1 = f(s_1,h) \mid s_1)} p_{\pi_f}(a_{2:h} \mid s_1, \bar{s}_2) + \delta\\
&= \beta(a_1 \mid s_1)\frac{\delta + p_\beta(g_2 = f(\bar{s}_2, h-1) \mid \bar{s}_2)}{p_\beta(g_1 = f(s_1,h) \mid s_1)} p_{\pi_f}(a_{2:h} \mid s_1, \bar{s}_2) + \delta\\
&\leq \beta(a_1 \mid s_1)\frac{p_\beta(g_2 = f(\bar{s}_2, h-1) \mid \bar{s}_2)}{p_\beta(g_1 = f(s_1,h) \mid s_1)} p_{\pi_f}(a_{2:h} \mid s_1, \bar{s}_2) + \delta\left(\frac{1}{\alpha_f}+1\right).
\end{aligned} \tag{20}$$

Similarly, we can further expand $p_{\pi_f}(a_{2:h} \mid s_1, \bar{s}_2)$ using the same rule:

$$\begin{aligned}
&p_{\pi_f}(a_{2:h} \mid s_1, \bar{s}_2) = \pi_f(a_2 \mid \bar{s}_2)\int_{s_3} p(s_3 \mid \bar{s}_2, a_2) p_{\pi_f}(a_{3:h} \mid s_1, \bar{s}_2, s_3)\\
&\leq \pi_f(a_2 \mid \bar{s}_2) p_{\pi_f}(a_{3:h}|s_1, \bar{s}_2, \bar{s}_3) + \delta\\
&= \beta(a_2 \mid \bar{s}_2)\frac{p_\beta(g_2 = f(\bar{s}_2, h-1)|\bar{s}_2, a_2)}{p_\beta(g_2 = f(\bar{s}_2, h-1)|\bar{s}_2)} p_{\pi_f}(a_{3:h} \mid s_1, \bar{s}_2, \bar{s}_3) + \delta.
\end{aligned} \tag{21}$$

Substituting this back to Eq. 20, we have:

$$\begin{aligned}
&p_{\pi_f}(a_{1:h} \mid s_1)\\
&\leq \beta(a_1 \mid s_1)\beta(a_2 \mid \bar{s}_2)\frac{p_\beta(g_2 = f(\bar{s}_2, h-1) \mid \bar{s}_2)}{p_\beta(g_1 = f(s_1,h) \mid s_1)} \cdot \frac{p_\beta(g_2 = f(\bar{s}_2, h-1) \mid \bar{s}_2, a_2)}{p_\beta(g_2 = f(\bar{s}_2, h-1) \mid \bar{s}_2)} p_{\pi_f}(a_{3:h} \mid s_1, \bar{s}_2, \bar{s}_3)\\
&\quad + 2\delta\left(\frac{1}{\alpha_f}+1\right)\\
&\ldots \text{ recursively expand } p_{\pi_f}\\
&\leq \prod_{t=1}^{h}\beta(a_t \mid \bar{s}_t)\frac{p_\beta(g_h = f(\bar{s}_h, 1) \mid \bar{s}_h, a_h)}{p_\beta(g_1 = f(s_1,h) \mid s_1)} + h\delta\left(\frac{1}{\alpha_f}+1\right)\\
&= \prod_{t=1}^{h}\beta(a_t \mid \bar{s}_t)\frac{\mathbb{I}[g(s_1, a_{1:h}) = f(s_1,h)]}{p_\beta(g_1 = f(s_1,h) \mid s_1)} + h\delta\left(\frac{1}{\alpha_f}+1\right).
\end{aligned} \tag{22}$$

The last step is deduced by the trajectory determinism and the consistency of the conditioning function f. Multiplying this back to Eq. 18 and noticing that the two indicator functions can never both be 1, we can yield that:

$$\mathbb{E}_{s_1,h}\Big[\mathbb{E}_{a_{1:h}\sim\pi_f|s_1}\big[f(s_1,h)-g(s_1,a_{1:h})\big]\Big] \leq h^2\delta\left(\frac{1}{\alpha_f}+1\right). \tag{23}$$

This can finally yield the bound in Eq. 12 by adding back to Eq. 16.

Algorithm 2: UNREST training procedure

```
Input: Offline dataset $\mathcal{D} = \{\boldsymbol{\tau}_i\}_{i=1}^N$, batch size $B$.
1  # Stage I: Train return transformers.
2  for each iteration do
3      Sample batch $\mathcal{B} = \{\boldsymbol{\tau}_i\}_{i=1}^B$ from $\mathcal{D}$, where $\boldsymbol{\tau}_i = \{(s_t, a_t)\}_{t=1}^T$;
4      Feed sampled sequences into Transformer and get embeddings $\tilde{x}_{s_t}, \tilde{x}_{a_t}$ by Eq. 3;
5      Feed Transformer outputs into variance networks to predict return
        $R_{\varphi_a}(R_t|\boldsymbol{\tau}_{<t}), R_{\varphi_s}(R_t|\boldsymbol{\tau}_{<t}, s_t)$;
6      Update network parameters based on the learning objectives in Eq. 4;
7  # Stage II: Segment sequence w.r.t. estimated uncertainty.
8  Compute uncertainty of each timestep $u_t \leftarrow D_{\mathrm{KL}}(R_{\varphi_a}(R_t|\boldsymbol{\tau}_{<t}), R_{\varphi_s}(R_t|\boldsymbol{\tau}_{<t}, s_t))$;
9  Segment sequence w.r.t. uncertainties $\mathcal{D}^{\mathrm{seg}} \leftarrow \{\boldsymbol{\tau}_i^{\mathrm{seg}}\}_{i=1}^N$, as discussed in Sec. 4.3;
10 # Stage III: Train decision model.
11 for each iteration do
12     Sample batch $\mathcal{B} = \{\boldsymbol{\tau}_i^{\mathrm{seg}}\}_{i=1}^B$ from $\mathcal{D}^{\mathrm{seg}}$, where $\boldsymbol{\tau}_i^{\mathrm{seg}} = \{(h_t, R_t, R_t^h, s_t, a_t)\}_{t=1}^T$;
13     Feed sampled sequences into Transformer and get embeddings $\tilde{x}_{R_t^h}, \tilde{x}_{s_t}, \tilde{x}_{a_t}$ by Eq. 7;
14     if using global return then
15         Concatenate Transformer outputs with global return $\tilde{x}_{s_t} \leftarrow [\tilde{x}_{s_t} \,||\, f^R(R_t)]$;
16     Feed $\tilde{x}_{s_t}$ into variance network to predict action distribution $\pi_\theta(a_t|\boldsymbol{\tau}_{<t}^{\mathrm{seg}}, h_t, R_t, R_t^h, s_t)$;
17     Update network parameters based on learning objective in Eq.9;
Output: Trained return transformers $R_{\varphi_a}(\cdot)$, $R_{\varphi_s}(\cdot)$ and policy $\pi_\theta(\cdot)$.
```

C UNREST Training Procedure

To get a more clear understanding of our training pipeline, we present the training procedure of UNREST in Alg. 2.

D Dataset Information

Our dataset is collected from the CARLA simulator, using its built-in Autopilot, which is a rule-based motion planner. Specifically, CARLA [4] is an open-sourced simulator designed for autonomous driving research. It provides researchers with a realistic environment that faithfully simulates traffic dynamics, weather conditions, and high-fidelity sensor data. The pre-built scenarios in CARLA offer a diverse set of driving scenes, while its high level of customizability and flexibility make it the preferred simulation platform for a majority of autonomous driving researchers.

We collect training data at a frequency of 10Hz, accumulating 30 hours of driving data from four distinct training towns (Town01, Town03, Town04, Town06) under four different weather conditions (ClearNoon, WetNoon, HardRainNoon, ClearSunset). At each timestep, we store the data tuple (s_t, a_t, r_t), whose compositions will be introduced in detail in the next paragraphs.

Reward: The symbol $r_t \in \mathbb{R}$ denotes the reward returned by the environment at the timestep t. Typically, our reward design is revised from Roach [48], encompassing factors such as speed, safety, and comfort.

$$r = r_{\mathrm{speed}} + r_{\mathrm{position}} + r_{\mathrm{rotation}} + r_{\mathrm{action}} + r_{\mathrm{terminal}}. \tag{24}$$

Among them, $r_{\mathrm{speed}} = 1 - \frac{|v - v_{\mathrm{desired}}|}{v_{\mathrm{max}}}$ represents the reward of approaching the target speed; $r_{\mathrm{position}} = -0.5\Delta_p$ represents the reward obtained from driving in the correct position, where Δ_p is the lateral distance between the ego vehicle and the centerline of the target route; $r_{\mathrm{rotation}} = -\Delta_r$ represents the reward obtained from driving in the correct orientation, where Δ_r is the angle difference between the ego vehicle and the centerline of the target route; r_{action} incurs a penalty of -0.1 when the steering in current timestep differs from the previous step by more than 0.01, promoting smooth and comfortable driving; r_{terminal} is 10 when reaching the destination, -10 when encountering a collision or violation of traffic rules, to encourage driving safely.

Safety constraints: In addition to the reward setting, the Constrained Penalized Q-learning (CPQ) [49] algorithm also needs a constraint definition for training. Specifically, we adopt the same setting as in [50], where each time of collision or traffic rule violation incurs a penalty of 1.0. The constraint limit for CPQ is 1.0 (i.e. ensures no safety issue at each ride).

In order to avoid memory overflow in the sequence model, both the action space and state space have been specially designed. Specifically, we adopt a similar setting to SPLT [11].

Action: The action $a \in \mathbb{R}^2$ consists of two values, representing the target angle and target velocity that will be fed into two separate PID controllers.

State: The observed state $s \in \mathbb{R}^{37}$ includes the following components: (i) ego vehicle velocity (ii) relative distance and velocity to the leading vehicle, which will be set to the default maximum value if no leading vehicle is perceived in the horizon (iii) relative distance and velocity to the pedestrian ahead, which will be set to default maximum values if no pedestrian is perceived in the horizon (iv) relative velocities and distances to vehicles in four cardinal directions (front, rear, left, right), also set to default maximum values if no vehicles are perceived in the horizon (v) distance to the traffic light ahead, set to default maximum value if no red light ahead (vi) distance to the stop sign ahead, set to default maximum value if no stop sign ahead (vii) relative angle difference w.r.t. the centerline of the target route (viii) distance to the centerline of the target route (ix) relative positions of the next 10 target waypoints.

As for stochasticity, we introduce stochastic speed ranges in $[20, 40]$m/s and vehicle-specific random sizes of visibility areas in $[10, 30]$m. This results in aggressive vehicles with higher speeds and smaller visibility ranges, as well as cautious vehicles with lower speeds and larger visibility ranges. These characteristics cannot be inferred from the observable state of the ego vehicle, leading to greater uncertainty during its interactions with the environment. Additionally, we also add Gaussian noise to the control signals of the ego vehicle to incorporate stochasticity into its behavior. These special designs, integrating with the complex scenarios and interactions from our collected large dataset, ensure that our training data contains sufficient stochasticity.

E Implementation Details

In this section, we introduce the implementation details, including our training process, evaluation process, and hyperparameters. Specifically, UNREST and all the baselines are implemented with Python 3.7 and PyTorch 1.13.1. Besides, all training processes are run on a NVIDIA A10 while inference is conducted on one NVIDIA 3090 GPU for fair comparison.

E.1 Training Details

Training Process: For all training processes, We employ an AdamW optimizer with a learning rate of 10^{-4}, and a consistent batch size of 256. For the training of sequence models, a history length of 10 is carefully selected to be sampled each time, which corresponds to a real-world duration of 1 second. To determine the number of epochs for training, a dynamic value is assigned based on the dataset's size, defined by the following formula: $n_{\text{epochs}} = \text{int}\left(\frac{1e6}{\text{len(dataset)}} \times n_{\text{ref}}\right)$, where n_{ref} represents a reference epoch number that can be tuned.

Details for segmentation: As discussed in Sec. 4.3, we train two return transformers for sequence segmentation. In practical implementation, to accurately quantify environmental uncertainty, we employ the ensemble method to train each return transformer. Using Eq. 11, we recalculate the variance and expectation of the returns. Then the sequences are segmented into certain and uncertain parts w.r.t. estimated uncertainties according to the principles we introduce in Sec. 4.3.

In our experiments, we discover that 'uncertain states' tend to occur consecutively. To ensure meaningful segmentation, we enforce a minimum 'uncertain part' length of $c = 20$ frames and select the uncertainty threshold $\epsilon = 3$ according to Fig. 5(a). Under these two hyperparameter settings, we find the segmented sequences often correspond to the completion of a significant driving task.

Details for return-span embedding: We introduce the return-span embedding $f_\theta^h(h_t)$ to enhance the interpretability of truncated returns as conditions. Specifically, we explore two variants to incorporate this embedding. The first variant is to add the return-span embedding to the return embedding, as stated in the main text. The second variant considers treating $f_\theta^h(h_t)$ as a distinct token input to the sequence model. However, we observe that the latter approach results in increased memory consumption without yielding improvements in performance (Tab. 8 and Tab. 9).

Details for global return embedding: The approach for obtaining the global return embedding differs from other embeddings due to its high uncertainty. Typically, instead of utilizing parameterized networks, we opt for a coarse uniform discretization approach, resulting in a corresponding one-hot vector (specifically, a 50-dimensional vector in our implementation). This strategy enables us to provide global guidance while mitigating the influence of its uncertainty on the training process.

Details for decision models: In the practical implementation, following DT [14], we directly predict the value of the action instead of its distribution. Specifically, we only retain the mean μ_θ and assume unit variance. Consequently, the loss in Eq. 9 can be reformulated as the mean square loss, resembling the learning objective in DT.

E.2 Evaluation Details

Main setting: To comprehensively evaluate our models' planning capability, we select two distinct driving scenarios from CARLA: the training town under training weather conditions and the new town under new weather conditions. Typically, the training town is chosen as Town03, the most complex town among the training dataset, and the training weathers are 'WetNoon' and 'ClearSunset'. The new town is Town05, the most complex town in the rest of the training set, and the new weathers are 'SoftRainSunset' and 'WetSunset'. For each tested scenario, we carefully set up the vehicles and driving routes w.r.t. the specifications of CARLA Leaderboard [41]. Moreover, we conduct experiments with three different seeds (2022, 2023, 2024) to facilitate the calculation of mean values and variances.

Details for dynamic uncertainty estimation: As we have stated in the main text, we employ dynamic uncertainty estimation at inference time to enable cautious planning. Typically, we implement more variants (results are shown in Tab. 8 and Tab. 9) other than KD-Tree [40] for test-time uncertainty estimation:

- **Network prediction:** The first approach entails training a neural network to forecast environmental uncertainty. Leveraging the uncertainty values computed by the aforementioned return transformers and employing the provided threshold, we assign binary class labels (*certain* and *uncertain*) to each state in the training dataset. Subsequently, these labels are utilized to train an uncertainty network.
- **Heuristic:** The heuristic method turns out to be a straightforward and intuitive alternative. Specifically, we observe that specific dimensions within received states have a strong correlation with uncertainties (e.g. distance to traffic light ahead). Thus, we directly compare the values of these particular dimensions to the preset uncertainty threshold to determine whether the state is uncertain.
- **KD-tree:** Our chosen methodology employs a KD-tree for uncertainty estimation. Specifically, we use all state vectors to initialize the KD-tree. Subsequently, the tree will select a dimension at each level and perform a binary split, ultimately placing all inputs in the leaf nodes (implemented by calling the Sklearn package). At test time, we query the nearest neighbors of the current state from the tree and compute the average uncertainty of its nearby 5 states as the estimated uncertainty.

Table 4: Hyperparameters for training return transformers for Sequence Segmentation.

Hyperparameters	**Value**	**Hyperparameters**	**Value**
# Transformer layers	4	# Transformer heads	8
Embedding dimension	128	Batch size B	256
Sampled Sequence length	10	Discount γ	0.95
Learning rate	1e-4	Dropout	0.1
Optimizer	AdamW	Weight decay	False
Ensemble size K	5	Data mask probability	0.6
# Reference epoch n_{ref}	50	Transformer Activation	GELU

Table 5: Hyperparameters for training UNREST decision models.

Hyperparameters	**Value**	**Hyperparameters**	**Value**
Uncertainty threshold ϵ	3.0	Min 'uncertain part' length c	20
# Transformer layers	4	# Transformer heads	8
Embedding dimension	128	Batch size B	256
Sampled Sequence length	10	Discount γ	1.0
Optimizer	AdamW	Weight decay	False
Learning rate	1e-4	Dropout	0.1
# Reference epoch n_{ref}	200	Transformer Activation	GELU
Global return dimension	50	Action Activation	Tanh

Table 6: Hyperparameters for UNREST's inference process.

Hyperparameters	**Value**	**Hyperparameters**	**Value**
Max history length	5	Uncertainty threshold ϵ	3.0
Return Horizon H	100	Upper Percentile η	0.7
Deterministic sample	True	# KD-Tree neighbor	5

E.3 Hyperparameters

We select several well-established approaches to serve as our comparative baselines. Specifically, we include Behavior Cloning (BC), Implicit Q-Learning (IQL) [43], as well as prominent sequence models such as Decision Transformer (DT) [14], Trajectory Transformer (TT) [15], SPLT [11], and ESPER [17]. To ensure a fair comparison, for all these baselines, we adopt the default hyper-parameter settings from the repository https://github.com/avillaflor/SPLT-transformer. For Monotonic Advantage Re-Weighted Imitation Learning (MARWIL) [42] and Conservative Q-Learning (CQL) [9], we adopt the default hyperparameter from the Ray RLlib https://docs.ray.io/en/latest/rllib/index.html. Finally for the safe offline RL baseline Constraints-Penalized Q-learning (CPQ) [49], we employ the default hyperparameters from the repository https://github.com/liuzuxin/OSRL.

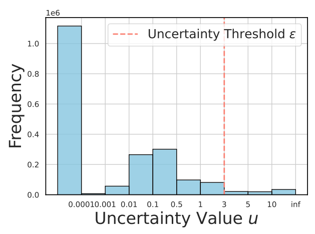
The full list of UNREST's hyperparameters can be found in Tab. 4, Tab. 5, and Tab. 6.

F More Experimental Results

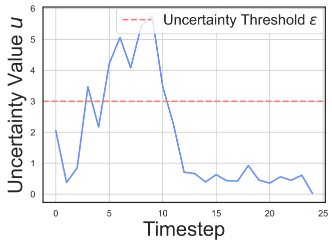
In this section, we supplement more experimental results, including the visualizations, complexity comparisons, sensitivity analysis of hyperparameters, the results for more UNREST variants, and results on environments other than driving (D4RL).

F.1 Visualizations

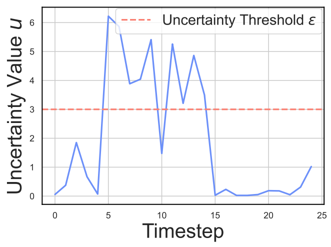
Uncertainty visualizations: We present more visualizations of UNREST's uncertainty estimation in Fig. 5 to validate its interpretability. We first visualize the uncertainty distribution in Fig. 5(a). As we can see, the majority of uncertainty values associated with state transitions fall within the range of 0 to 10^{-4}, aligning with intuitive expectations. In autonomous driving environments with sparse



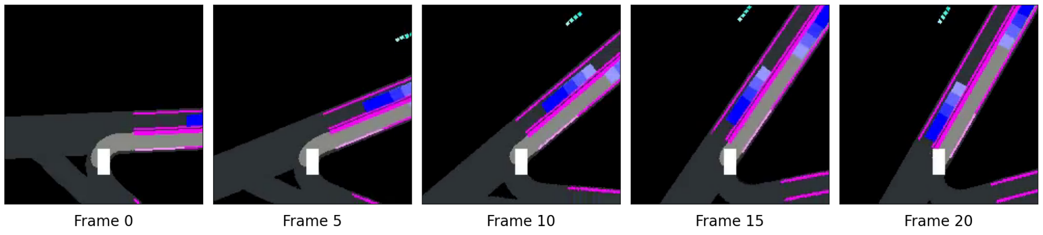
(a) Uncertainty distribution



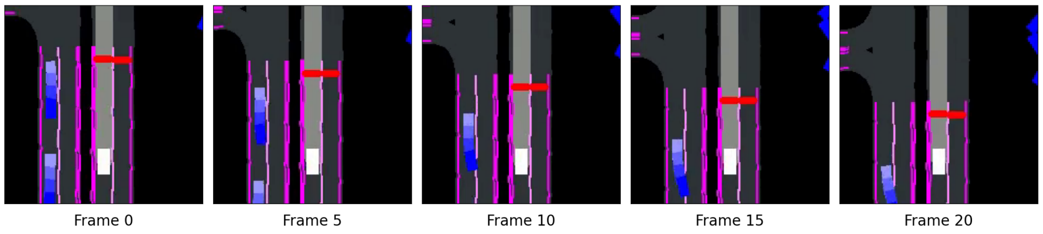
(b) Right turning: Uncertainty curve



(c) Light crossing: Uncertainty curve



(d) Right turning: State changing process



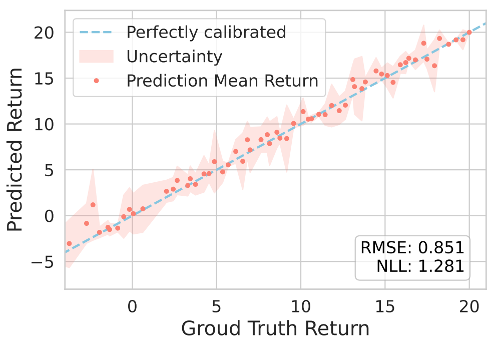
(e) Light crossing: State changing process

Figure 5: More visualizations of UNREST's uncertainty estimation results. The black background indicates non-drivable areas, the dark gray areas represent drivable regions, the white rectangle denotes the ego vehicle, the dark blue rectangles signify surrounding vehicles, the sky blue rectangles indicate pedestrians, the purple lines represent lane boundaries, and the red, green, and yellow markers indicate traffic lights and stop signs. Finally, the light gray path extending from the ego vehicle represents its intended route.

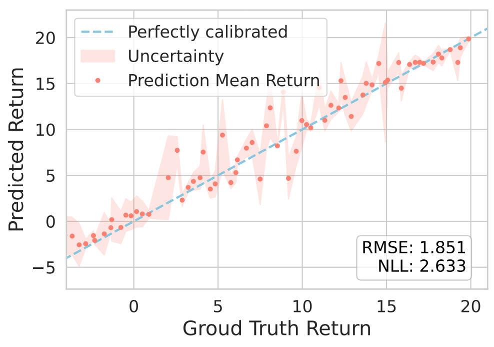
vehicle presence and a limited number of traffic signals, the stochasticity of environment transitions tends to be relatively low. The uncertainty threshold ϵ is accordingly chosen as 3.0 to cover corner cases with large uncertainties.

Then, we seek uncertain transition states within the training dataset and visualize two typical scenarios. Fig. 5(b) and Fig. 5(d) correspond to a turning scenario, while Fig. 5(c) and Fig. 5(e) depict a scenario of passing through a red light. By associating the temporal steps with the uncertainty curves, it can be observed that as the ego-vehicle approaches the turning point, the uncertainty gradually increases since the ego-vehicle cannot forecast the traffic conditions after turning. After the turning behavior is almost completed, the uncertainty decreases below the threshold again. In the scenario of waiting at a red light, the uncertainty rapidly increases as the ego-vehicle approaches the red light due to the uncertain color changes of the traffic light. However, after the ego-vehicle decides to stop in front of the red light at approximately the 15th timestep, the uncertainty decreases sharply below the threshold. These visualization results effectively demonstrate the interpretability of our uncertainty measurement approach.

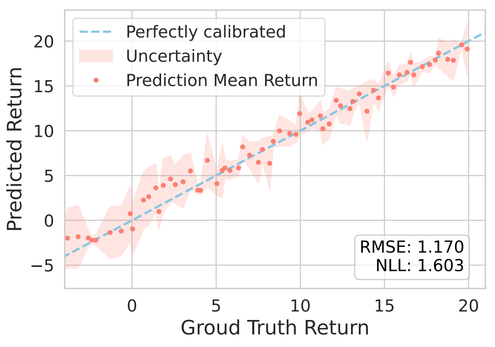
Uncertainty Calibration: To provide a more intuitive impression of UNREST's effectiveness, we present the return distribution calibration results in Fig. 6. Notably, the uncertainty in the figure



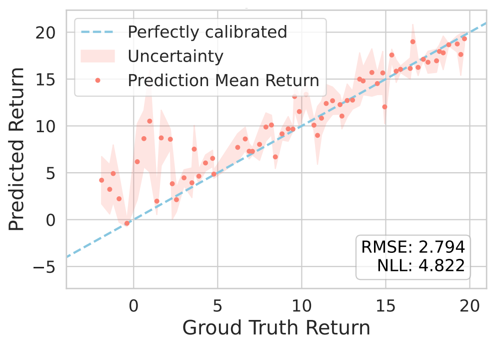
(a) Ensemble $p(R_t|\tau^{\text{ret}}_{<t})$ Calibration



(b) Single $p(R_t|\tau^{\text{ret}}_{<t})$ Calibration



(c) Ensemble $p(R_t|\tau^{\text{ret}}_{<t}, s_t)$ Calibration



(d) Single $p(R_t|\tau^{\text{ret}}_{<t}, s_t)$ Calibration

Figure 6: Calibration results of return distribution using ensemble models are obviously better than that using a single model. We use the standard variance of the networks' predictions as an approximate indicator of uncertainty. The blue line signifies the ground truth, while the red dots denote the predicted mean returns. The areas shaded in orange represent the predicted mean coupled with their respective standard deviations.

denotes the variance of the distribution, *different from the so-called environmental uncertainty we used in the main text* that reflects the impact of environment transitions (through conditional mutual information). The impact of the environment has no ground truth value and can only be interpreted through visualizations like Fig. 5, while the predicted return distribution has corresponding ground truth and thus can be directly calibrated like what we do in Fig. 6. Typically, the figure illustrates that using an ensemble of return transformers can significantly better predict the return distribution than using a single model (closer to the ground truth, with tight uncertainty bands), where it achieves smaller (better) results on two widely adopted calibration metrics: RMSE and NLL than the single return transformer variant.

Sequence length distributions: We present different segmented sequence length distributions in Fig. 7. Analyzing the figures, it is evident that most uncertain sequences tend to possess lengths close to the state transition timestep c (translate from uncertain to certain states if the last $c-1$ steps are all identified as certain), while certain sequences are almost uniformly distributed w.r.t. their lengths.

F.2 Complexity Comparisons

We provide empirical results about space/time usage as shown in Tab. 7. Regarding GPU usage, UNREST demonstrates comparable levels to DT, while significantly less utilization compared to SPLT and TT. Regarding training and inference time, UNREST consumes slightly more time than DT, yet notably faster than TT and SPLT. These results signify that UNREST achieves superior performance while requiring relatively modest computational resources.

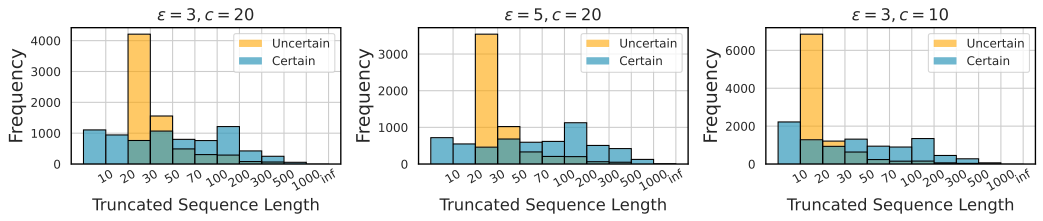


Figure 7: Segmented sequence length distributions with different threshold ϵ and state transition timestep c.

Table 7: Comparison results of training/inference time (10^{-2}s) and training GPU memory (GB) between different sequence models. The training time is calculated w.r.t. multiple iterations, while the inference time is calculated w.r.t. multiple rollout steps. The GPU usage is a fixed value for one training model.

Metric	BC	DT	TT	SPLT	UNREST
GPU usage	1.00	1.46	23.3	1.96	1.48
Training time	0.83 ± 0.13	0.93 ± 0.13	32.7 ± 0.2	4.20 ± 0.1	1.47 ± 0.12
Inference time	0.15 ± 0.01	0.16 ± 0.02	25.6 ± 0.7	0.89 ± 0.02	0.24 ± 0.03

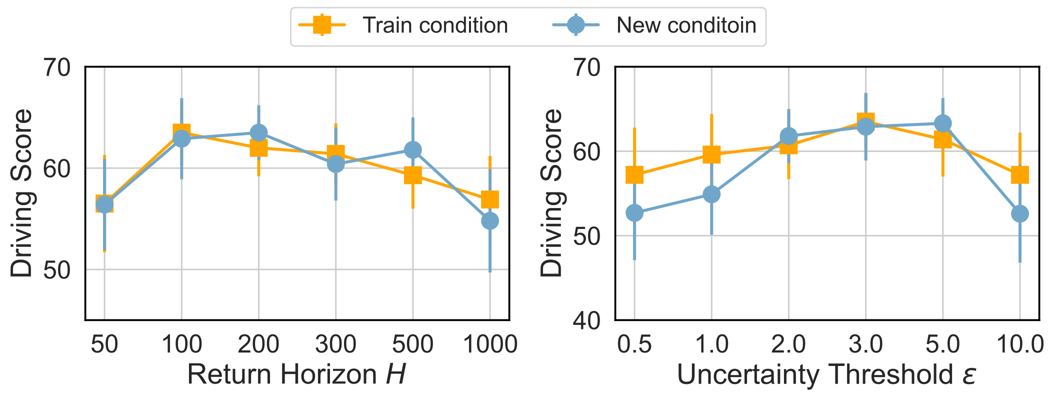


Figure 8: The sensitivity analysis of hyper-parameters H and ϵ.

F.3 Sensitivity Analysis

Impact of return horizon H: We investigate the influence of the return horizon length, denoted as H, on the performance of UNREST. The experimental results are depicted in Fig. 8, where the return horizon varies from 50 to 1000. It is evident that a relatively modest value of H (e.g., H=100) yields the best performance, while minimal and large values of H exhibit noticeable performance deterioration. These findings align with our underlying assumption that shorter sequences can alleviate uncertainty, while excessively short sequences may give rise to shortsightedness, thereby compromising performance.

Impact of uncertainty threshold ϵ: We next investigate the impact of the uncertainty threshold ϵ and present the results in Fig. 8 where we increase ϵ from 0.5 to 10. As shown in the figure, a moderate value of ϵ, namely $\epsilon = 3$ gives the best performance, while smaller and larger values of ϵ exhibit obvious performance degradation. This behavior can be attributed to the fact that the majority of uncertainty values fall within the range of 0 to 10^{-4}. Consequently, excessively small values may incorrectly identify 'certain states' as uncertain, while excessively large values may overlook important 'uncertain states'.

Table 8: Driving performance of more UNREST variants on train town and train weather condition.

Planner	Driving Score↑	Success Rate↑	Route Comp.↑	Infrac. Score↑	Norm. Rewards↑
Tokened ret-span emb.	62.8 ± 3.6	56.2 ± 4.8	81.1 ± 4.7	66.5 ± 2.5	0.62 ± 0.03
Fixed horizon seg.	59.2 ± 2.6	44.7 ± 3.2	78.3 ± 4.8	64.5 ± 3.0	0.66 ± 0.02
Separate BC model	61.0 ± 4.1	51.7 ± 5.3	82.8 ± 3.7	64.6 ± 2.4	0.68 ± 0.04
Heuristic uncertainty	62.1 ± 2.4	28.8 ± 7.4	43.8 ± 6.9	72.6 ± 4.3	0.47 ± 0.03
Predicted uncertainty	60.7 ± 2.7	50.0 ± 4.4	63.6 ± 5.2	65.8 ± 3.4	0.61 ± 0.02
Reweighted BC	**64.2 ± 2.4**	57.5 ± 3.3	85.5 ± 2.7	71.4 ± 2.5	0.68 ± 0.03
Original model	63.5 ± 3.2	54.5 ± 7.0	83.8 ± 3.1	70.2 ± 2.8	0.64 ± 0.04

Table 9: Driving performance of more UNREST variants on new town and new weather conditions.

Planner	Driving Score↑	Success Rate↑	Route Comp.↑	Infrac. Score↑	Norm. Rewards↑
Tokened ret-span emb.	62.0 ± 4.9	56.8 ± 3.0	92.0 ± 5.4	60.3 ± 3.3	0.64 ± 0.04
Fixed horizon seg.	58.5 ± 3.6	46.7 ± 6.2	80.3 ± 2.8	64.7 ± 2.8	0.64 ± 0.02
Separate BC model	59.3 ± 3.4	63.8 ± 4.3	90.0 ± 5.5	59.3 ± 4.4	0.68 ± 0.05
Heuristic uncertainty	56.7 ± 2.6	40.0 ± 7.3	70.0 ± 6.2	72.5 ± 3.0	0.52 ± 0.02
Predicted uncertainty	59.6 ± 3.4	55.6 ± 5.5	88.0 ± 3.7	58.6 ± 4.1	0.64 ± 0.04
Reweighted BC	**63.8 ± 2.3**	59.3 ± 4.7	91.0 ± 5.0	65.1 ± 3.7	0.67 ± 0.02
Original model	62.9 ± 4.0	57.5 ± 5.4	90.0 ± 6.0	62.9 ± 3.8	0.65 ± 0.03

Table 10: Comparison to SOTA baselines on the standard D4RL Mujoco locomotion-v2 domain. For BC, MBOP, CQL, DT, TT, IQL, and SPLT we use the results reported from the SPLT paper [11]. For ESPER, we report the results from their paper [17]. We report the mean and std for our method over 3 seeds with 10 trajectories for each seed. 'Stochastic' denotes customized stochastic environments according to [18]. The symbol '-' is used to indicate the omission of measurement results that are deemed unimportant.

Dataset	Environment	BC	MBOP	CQL	DT	TT	IQL	SPLT	ESPER	DoC	UNREST(ours)
Med-Expert	HalfCheetah	59.9	105.9	91.6	86.8	95.0±0.2	86.7	91.8±0.5	66.95±11.13	85.6±4.3	91.9±0.8
Med-Expert	Hopper	79.6	55.1	105.4	107.6	110.0±2.7	91.5	104.8±2.6	89.95±13.91	91.2±3.3	93.5±5.4
Med-Expert	Walker2d	36.6	70.2	108.8	108.1	101.9±6.8	109.6	108.6±1.1	106.87±1.26	107.3±2.1	105.7±4.4
Medium	HalfCheetah	43.1	44.6	44.0	42.6	46.9±0.4	47.4	44.3±0.7	42.31±0.08	43.2±0.5	44.5±0.9
Medium	Hopper	63.9	48.8	58.5	67.6	61.1±3.6	66.3	53.4±6.5	50.57±3.43	65.4±2.5	79.8±4.2
Medium	Walker2d	77.3	41.0	72.5	74.0	79.0±2.8	78.3	77.9±0.3	69.8±1.2	72.2±2.2	70.6±3.9
Med-Replay	HalfCheetah	4.3	42.3	45.5	36.6	41.9±2.5	44.2	42.7±0.3	35.9±2.0	38.7±0.7	39.4±0.5
Med-Replay	Hopper	27.6	12.4	95.0	82.7	91.5±3.6	94.7	75.0±23.8	50.2±16.1	57.3±9.6	65.7±21.9
Med-Replay	Walker2d	36.9	9.7	77.2	66.6	82.6±6.9	73.9	63.7±4.7	65.5±8.1	66.8±5.5	66.3±6.4
Average		47.7	47.8	77.6	74.7	78.9	76.9	72.9	64.2	70.4	73.0
Med-Stochastic	HalfCheetah	-	-	-	75.8±1.2	83.7±2.8	-	88.6±2.2	77.4±5.6	84.4±4.1	**90.5 ± 3.2**
Med-Stochastic	Hopper	-	-	-	89.4±3.3	93.4±3.6	-	94.8±2.7	92.5±6.9	95.2±5.3	**97.4 ± 4.4**
Med-Stochastic	Walker2d	-	-	-	86.3±2.4	91.7±2.8	-	95.7±3.4	96.4±2.3	98.8±3.2	**100.2 ± 4.1**

F.4 More UNREST Variants

Tab. 8 and Tab. 9 record the detailed performance of more UNREST variants in both training and new scenarios.

As shown in the results, introducing return-span embedding as a separate token yields little improvement but will increase memory usage. Fixed horizon segmentation means segmenting the sequences at a fixed timestep 100. Although simple, it ignores environmental uncertainties and may lead to shortsighted problems. Therefore, it's reasonable that it performs notably worse than our original model. The Separate BC model means separately training an offline RL model in 'certain parts' and a BC model in 'uncertain parts'. While achieving the highest normalized rewards, there is still a significant gap in its driving score compared with the original model, which shows that the joint training of two models can benefit the planning process. Utilizing a heuristic method to estimate uncertainties at inference time obtains the best infraction score, which is apparent since it is designed to promote cautious behaviors in every possible uncertain scenario. However, this method is overly pessimistic and overlooks certain corner cases, ultimately leading to the lowest driving score. Adopting a network to predict uncertainty at inference time yields better performance than the heuristic method, but overall, due to the relatively low dimensionality of the inputs, the original model performs better than it with a faster inference speed. Finally, we also study a Reweighted BC variant, which additionally fits state value functions like that in MARWIL [42] and conducts advantage reweighted behavior cloning at uncertain parts. Since the offline data collected by AutoPilot can be suboptimal, we find reweighted behavior cloning can effectively down-weight those suboptimal actions and improve UNREST's driving scores at both training and new scenarios. However, since it significantly increases model complexity, we have refrained from incorporating this technique into the final model.

F.5 Results on D4RL

For completeness, we evaluate our method on D4RL Mujoco tasks, though they are mainly deterministic tasks. Specifically, our UNREST is generally competitive with SOTA baselines on these tasks, as shown in Tab. 10. The reason for its overall underperformance compared to DT lies in the unnecessarily forced sequence segmentation in such deterministic tasks, where it is more appropriate to retain global returns in order to extend the horizon length considered during planning. **We also test UNREST in three customized stochastic D4RL environments from [18].** Compared to complex generative training (which also harms their performance in complex environments) of DoC [18] and ESPER [17], UNREST achieves the highest rewards with lightweight models.