CRC: CREDIBLE RESIDUAL CALIBRATION FOR TRAFFIC SIGNAL CONTROL UNDER TEST-TIME DRIFT

Anonymous authors
Paper under double-blind review

ABSTRACT

Traffic signal control (TSC) plays a critical role in improving urban traffic efficiency. Although reinforcement learning (RL) has achieved remarkable progress in TSC, RL-based controllers are typically trained under specific traffic conditions and remain frozen during testing. In practice, environmental feedback under dynamic traffic conditions may change after a signal action is applied, leading to performance degradation. Existing solutions mainly rely on adaptation capabilities acquired before testing and remain limited in exploiting feedback changes that emerge at test time. In this paper, we introduce *Action-Conditioned Feedback Distribution (ACFD) drift* to characterize changes in the feedback distribution over next states and rewards conditioned on the current state and action, and propose *Credible Residual Calibration* (***CRC***), a plug-in test-time adaptation framework for TSC. CRC freezes the pretrained base controller and attaches a residual calibrator to its outputs, enabling adaptation through test-time feedback. CRC uses the test-time Bellman inconsistency of the frozen base controller as an observable value-relevant signal of ACFD drift and maintains calibration-state statistics from stepwise feedback. Based on the calibration state, the residual calibrator learns action-wise candidate residual corrections, while action-wise credibility controls their release for decision making. CRC further uses calibration-state-guided action routing to adjust the decision mode and dynamic replay to update the residual calibrator from informative test-time transitions. Extensive experiments on real-world datasets and multiple base TSC controllers demonstrate that CRC consistently improves test-time control performance.

1 INTRODUCTION

Traffic signal control (TSC) is a foundational component of intelligent transportation systems for improving urban traffic efficiency (Zhang et al., 2011). Reinforcement learning (RL) (Sutton & Barto, 1998; Arulkumaran et al., 2017) has become a mainstream technique for TSC (Wei et al., 2021; Chen et al., 2024). However, adapting RL-based TSC methods to changing traffic conditions at test time remains challenging. RL controllers are typically trained under specific traffic conditions and remain frozen during testing. In practice, traffic demand may change unexpectedly, such as intensified peak demand or sudden demand increases during normally low-demand periods, substantially degrading test-time control performance. To alleviate this problem, existing TSC studies have mainly explored rapid adaptation and generalization from two directions. Meta-reinforcement learning-based TSC methods (Zhu et al., 2023; Lou et al., 2022) transfer knowledge across related traffic tasks to enable rapid adaptation to unseen scenarios with limited interaction data or gradient updates. Large language model (LLM)-based TSC methods (Yuan et al., 2026; Zou et al., 2026) leverage the pretrained knowledge and reasoning capabilities of LLMs to improve generalization across traffic scenarios. However, both directions mainly rely on adaptation capabilities acquired before testing or general knowledge encoded in pretrained models, rather than explicitly characterizing and continuously exploiting environmental feedback changes that emerge during test time.

Recently, test-time reinforcement learning (Zuo et al., 2025) has provided a promising direction by using test-time observations and interaction feedback to improve model adaptability during testing. In TSC, each executed signal action naturally produces new feedback for subsequent adaptation. *Nevertheless, before designing test-time adaptation for TSC, a more fundamental question needs to be clarified: what kinds of test-time changes actually lead to the degradation of a trained controller*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

when it remains frozen at test time? A straightforward intuition is to attribute such degradation to changes in the traffic-state distribution, i.e., $p_{\text{test}}(s_t) \neq p_{\text{train}}(s_t)$. However, state changes alone cannot determine whether the learned control policy remains effective, because control performance also depends on how the environment responds to an action. Rather than ignoring state changes, the conditional perspective further examines whether the resulting feedback distribution changes given the current state and action. Our key insight is that *Action-Conditioned Feedback Distribution Drift (ACFD drift)* constitutes a value-relevant source of test-time degradation, as it reflects changes in the feedback associated with state–action pairs and thus whether the action–feedback relationship learned during training remains consistent with test-time traffic dynamics. This relationship is characterized by $p(s_{t+1}, r_t \mid s_t, a_t)$, which describes how the environment responds after a specific control action is applied under a given traffic state. Formally, ACFD drift occurs when $p_{\text{test}}(s_{t+1}, r_t \mid s_t, a_t) \neq p_{\text{train}}(s_{t+1}, r_t \mid s_t, a_t)$.

While test-time drift is defined over the action-conditioned feedback distribution of next states and rewards, its full temporal evolution cannot be directly observed from stepwise test-time interactions. At each step, the controller can only receive post-action feedback after executing a signal action. *Therefore, a key challenge is how to sense drift based on the executed action and its corresponding post-action feedback at each step.* Beyond sensing drift, another critical issue is how to adapt the trained controller at test time. A straightforward strategy is to update all model parameters whenever new environmental feedback is received. However, directly updating the trained controller may disturb its original control capability. Moreover, newly learned adaptations are often based on limited and evolving test-time feedback and may not be sufficiently validated through subsequent interactions, making their direct use potentially unreliable. *Thus, another challenge is to optimize the test-time behavior of the trained controller without disrupting its original control capability, while evaluating the credibility of the learned adaptation.*

To address these challenges, we propose Credible Residual Calibration (***CRC***), a plug-in test-time adaptation framework for TSC. CRC freezes the pretrained base controller and attaches a residual calibrator to its outputs, enabling test-time adaptation through feedback without full-parameter policy updates. Since the action-conditioned feedback distribution over the entire test-time process cannot be directly observed, CRC uses the Bellman inconsistency of the frozen base controller under stepwise test-time feedback as an observable value-relevant signal of ACFD drift. Based on this signal, CRC maintains calibration-state statistics, including recent drift, residual calibration error, and action-wise credibility. The residual calibrator generates action-wise candidate residual corrections, while the credibility coefficient controls their release strength for action selection. CRC further designs calibration-state-guided action routing to adjust the decision mode according to the calibration state, and updates the residual calibrator through dynamic replay.

In summary, the main contributions are as follows:

- We introduce ACFD drift for test-time TSC to characterize changes in next-state and reward feedback conditioned on state-action pairs, and use the test-time Bellman inconsistency of the frozen base controller as an observable value-relevant signal for drift sensing.
- We propose CRC, a plug-in test-time adaptation framework that freezes the base controller and learns action-wise residual corrections from feedback to adapt its test-time behavior.
- We design calibration-aware credible residual release and calibration-state-guided action routing, which jointly determine how residual corrections influence action selection under different calibration states.
- We develop a dynamic replay update strategy that prioritizes test-time transitions with strong drift and large remaining calibration error. Extensive experiments on real-world traffic datasets and multiple base controllers demonstrate the effectiveness of CRC.

2 Related Work

This section first reviews existing TSC studies from two perspectives: RL-based TSC and LLM-based TSC. We then introduce Test-Time Reinforcement Learning (TTRL) to position our work within the broader context of test-time adaptation. More detailed discussions of related studies are provided in Appendix A.

RL-based TSC methods (Wei et al., 2019b; Zheng et al., 2019a; Chen et al., 2020; Wei et al., 2019a) learn signal control policies or action-value functions from traffic states and have achieved strong performance in training scenarios. However, these controllers typically remain frozen at test time, making them vulnerable when test-time traffic dynamics differ from those encountered during training. To improve adaptation, existing RL-based TSC solutions employ meta-reinforcement learning (Zang et al., 2020), treating different intersections and traffic-flow patterns as related tasks and learning transferable initializations or task representations. As a result, the controller can adapt to unseen scenarios with limited interaction data or gradient updates. *However, such adaptation ability is mainly acquired before testing and remains limited in explicitly sensing and continuously exploiting feedback changes that emerge during test-time control.*

LLM-based TSC methods (Lai et al., 2025; Yuan et al., 2026; Zou et al., 2026) convert real-time traffic states into natural-language descriptions and leverage the commonsense knowledge and contextual reasoning capabilities of LLMs to improve zero-shot generalization. In this way, they can alleviate performance degradation when test-time traffic conditions differ from those encountered during training. *However, their adaptability mainly relies on the pretrained knowledge and reasoning capabilities of LLMs rather than explicitly exploiting changes in test-time feedback, while LLM inference may also introduce substantial computational overhead.*

TTRL (Zuo et al., 2025) adapts model parameters directly on unlabeled test data by repeatedly sampling multiple outputs for each test input, estimating pseudo-labels through majority voting, and converting the resulting consensus into reward signals for online policy optimization. *Despite its promising adaptability, existing TTRL research primarily focuses on text-based question-answering or reasoning tasks for LLMs, while its application to TSC remains unexplored.* Directly transferring existing TTRL mechanisms to TSC is also non-trivial. Full-parameter test-time updates may disturb the original control capability of the pretrained controller, while adaptations learned from limited test-time feedback may not be sufficiently credible for immediate decision making. Therefore, TSC requires test-time adaptation that exploits feedback without updating the entire controller. *Unlike existing TTRL methods, this work freezes the base traffic controller and adapts its test-time behavior through credible residual calibration and dynamic replay update.*

3 PRELIMINARIES

This section introduces the definitions and problem formulation, with more details in Appendix B.

Definition 1 (Signalized Intersection) *A signalized intersection consists of multiple incoming and outgoing lanes, where vehicle movements are regulated by traffic signals. Each incoming lane is associated with a permitted traffic movement, including left-turn, through, and right-turn movements.*

Definition 2 (Signal Phase) *A signal phase is a set of mutually non-conflicting traffic movements that are granted the right-of-way simultaneously. Let $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ denote the set of candidate signal phases, where each phase corresponds to a feasible signal control action.*

Problem 1 (RL-based Test-Time Calibration for TSC) *We formulate TSC as a Markov decision process, where the controller observes a traffic state $s_t \in \mathcal{S}$, selects a signal phase $a_t \in \mathcal{A}$, receives a reward r_t, and observes the next state s_{t+1}. Let $Q_\theta(s, a)$ denote the action value of a pretrained RL-based TSC controller, whose parameters θ remain frozen at test time. Test-time calibration aims to exploit test-time interaction feedback (s_t, a_t, r_t, s_{t+1}) to adjust the base action values and improve test-time control performance.*

4 METHODOLOGY

As shown in Fig. 1, our ***Credible Residual Calibration*** for test-time traffic signal control (***CRC***) consists of four components. ***(1) Calibration-Aware Credible Residual Release*** maintains a set of calibration-state statistics, including the **ACFD drift** trend, recent **residual calibration error**, and **action-wise credibility**. These quantities support subsequent residual computation, action selection, and test-time updates. ***(2) Residual Calibrator*** generates residual corrections $\Delta Q_{\phi_t}(s_t, a)$ based on calibration-state statistics, the current state s_t, and the $Q_\theta(s_t, a)$ produced by the frozen pretrained controller. Then, the action-wise credibility coefficient $\rho_t(a) \in [0, 1]$ determines the released residual correction $\rho_t(a)\Delta Q_{\phi_t}(s_t, a)$ used for action selection. Accordingly, the executable

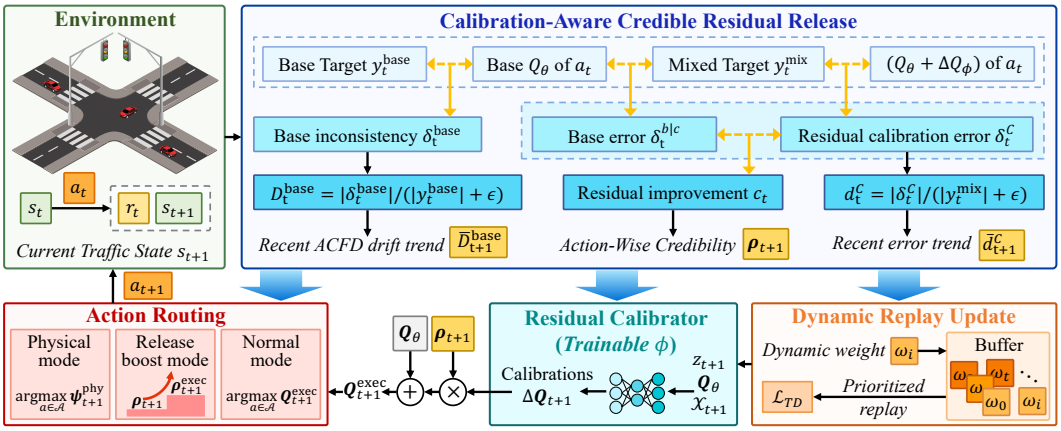


Figure 1: Overview of the CRC framework.

action value is constructed as $Q_t^{\text{exec}}(s_t, a) = Q_\theta(s_t, a) + \rho_t(a)\Delta Q_{\phi_t}(s_t, a)$. ***(3) Calibration-State-Guided Action Routing*** selects an appropriate decision mode under different test-time calibration states, thereby determining how residual corrections are used for action selection. ***(4) Dynamic Replay Update*** prioritizes test-time transitions with strong drift and large remaining calibration error, enabling the calibrator to focus on more informative feedback during test-time updates.

4.1 Calibration-Aware Credible Residual Release

Definition 3 (ACFD drift) *Let $p_{\text{train}}(s_{t+1}, r_t \mid s_t, a_t)$ denote the action-conditioned feedback distribution during training, and let $p_t(s_{t+1}, r_t \mid s_t, a_t)$ denote that encountered at test time. We define ACFD drift as $p_t(s_{t+1}, r_t \mid s_t, a_t) \neq p_{\text{train}}(s_{t+1}, r_t \mid s_t, a_t)$, for test-time state–action pairs (s_t, a_t). This drift characterizes changes in how the traffic environment responds to an executed signal action through the resulting next state and reward.*

Since the action-conditioned feedback distribution evolves throughout the test-time process and cannot be fully observed at any single step, CRC uses stepwise test-time feedback to construct an observable value-relevant signal of drift. Specifically, after executing a_t and receiving the feedback (r_t, s_{t+1}), CRC constructs the pure base Bellman target $y_t^{\text{base}} = r_t + \gamma \max_{a'} Q_\theta(s_{t+1}, a')$ and computes the base Bellman inconsistency as $\delta_t^{\text{base}} = y_t^{\text{base}} - Q_\theta(s_t, a_t)$.

The target y_t^{base} represents the value induced by the current test-time feedback when only the frozen base controller is used for bootstrapping (Sutton & Barto, 1998). If the action-feedback relationship learned during training remains consistent at test time, the base value $Q_\theta(s_t, a_t)$ should be close to this target. Therefore, a large $|\delta_t^{\text{base}}|$ indicates that, after executing a_t under s_t, the actually observed feedback deviates from what the frozen base controller expects. To measure the current-step drift, CRC normalizes δ_t^{base} as $D_t^{\text{base}} = |\delta_t^{\text{base}}|/(|y_t^{\text{base}}| + \epsilon)$. It then maintains the moving average $\bar{D}_{t+1}^{\text{base}} = (1 - \beta_D)\bar{D}_t^{\text{base}} + \beta_D D_t^{\text{base}}$ to capture the recent drift trend. A more detailed explanation of why Bellman inconsistency can reflect ACFD drift is provided in Appendix C.1.

Residual Calibration Error. After sensing drift, CRC further evaluates whether the residual calibrator can reduce the remaining inconsistency under test-time feedback. To this end, CRC constructs a mixed Bellman target $y_t^{\text{mix}} = r_t + \gamma \max_{a'} \left[Q_\theta(s_{t+1}, a') + \Delta Q_{\phi_t^-}(s_{t+1}, a')\right]$, where ϕ_t^- denotes the target calibrator parameters. Compared with y_t^{base}, y_t^{mix} incorporates the target residual calibrator during bootstrapping, and thus provides a calibration-aware reference for measuring the residual calibration error $\delta_t^C = y_t^{\text{mix}} - [Q_\theta(s_t, a_t) + \Delta Q_{\phi_t}(s_t, a_t)]$.

The error δ_t^C measures how much Bellman inconsistency remains after applying the candidate residual correction to the executed action value. A smaller $|\delta_t^C|$ indicates that the residual-calibrated value is more consistent with the current test-time feedback. CRC further normalizes it as $d_t^C = |\delta_t^C|/(|y_t^{\text{mix}}| + \epsilon)$ and maintains the moving average $\bar{d}_{t+1}^C = (1 - \beta_C)\bar{d}_t^C + \beta_C d_t^C$ to capture the recent trend of residual calibration error.

Action-Wise Credibility. CRC defines action-wise credibility to quantify whether the learned residual correction truly improves the value estimate of the executed action under test-time feedback. Specifically, CRC compares two errors under the same mixed target: (1) the base error $\delta_t^{b|c} = y_t^{\text{mix}} - Q_\theta(s_t, a_t)$, which measures the discrepancy between the frozen base value and the calibration-aware target; and (2) the residual calibration error δ_t^C, which measures the remaining discrepancy after applying $\Delta Q_{\phi_t}(s_t, a_t)$. CRC then measures the residual improvement as $\iota_t = \Big((\delta_t^{b|c})^2 - (\delta_t^C)^2\Big) \,/\, \Big((\delta_t^{b|c})^2 + (\delta_t^C)^2 + \epsilon\Big)$. A positive ι_t indicates that $\Delta Q_{\phi_t}(s_t, a_t)$ reduces the value error compared with the frozen base value, while a non-positive value indicates that the residual correction does not improve the executed action under the current feedback. The instantaneous credibility signal is obtained as $c_t = \text{clip}(\iota_t, 0, 1)$. Since each test-time transition only validates the actually executed action, CRC updates only the credibility coefficient of that action through a moving average:

$$\rho_{t+1}(a) = \begin{cases} (1-\beta_\rho)\rho_t(a) + \beta_\rho c_t, & a = a_t, \\ \rho_t(a), & a \neq a_t. \end{cases} \tag{1}$$

Further analysis and discussion of the calibration-state statistics are provided in Appendix C.2.

4.2 Residual Calibrator

For each intersection, the calibrator input is constructed as $x_t = \text{Concat}\,[z_t, \mathbf{Q}_\theta(s_t), \mathbf{A}_\theta(s_t), \chi_t]$. Here, z_t denotes intersection statistical features, such as the current phase, phase unserved time, and queue statistics, whose detailed construction is provided in Appendix C.3. The vector $\mathbf{Q}_\theta(s_t)$ provides the original action-value estimates of the pretrained controller. The base advantage vector $\mathbf{A}_\theta(s_t)$ further highlights the relative preference among actions by removing the mean value, where $A_\theta(s_t, a) = Q_\theta(s_t, a) - \frac{1}{|\mathcal{A}|}\sum_{a'} Q_\theta(s_t, a')$. In addition, $\chi_t = \text{Concat}\left[\bar{D}_t^{\text{base}}, \bar{d}_t^C, \boldsymbol{\rho}_t\right]$ contains recent calibration information, where $\boldsymbol{\rho}_t = [\rho_t(a_1), \ldots, \rho_t(a_{|\mathcal{A}|})]$. These inputs allow the calibrator to consider the current traffic condition, the base preference, and the reliability of previous corrections.

Given x_t, the calibrator computes a hidden representation $h_t = \text{MLP}(x_t)$ and produces two action-wise outputs. The action-level gate $g_t(a) = g_{\min} + (1 - g_{\min})\sigma(f_g(h_t)_a)$ controls the activation strength of the residual pathway for each action, while the residual head $\omega_t(a) = f_\Delta(h_t)_a$ estimates the direction and magnitude of the value correction. They are combined as $\eta_t(a) = g_t(a)\omega_t(a)$ to obtain the gated residual response. Finally, the calibrator applies advantage-style centering to obtain the candidate residual correction $\Delta Q_{\phi_t}(s_t, a) = \eta_t(a) - \frac{1}{|\mathcal{A}|}\sum_{a' \in \mathcal{A}} \eta_t(a')$. This centering encourages the calibrator to adjust relative action preferences rather than uniformly shifting all action values. The computational complexity of the residual calibrator is further analyzed in Appendix C.5.

4.3 Calibration-State-Guided Action Routing

At test time, random trial-and-error exploration used during training is undesirable. CRC therefore adopts rule-based stable exploration when the credibility of calibrated values is low. Accordingly, *Calibration-State-Guided Action Routing* selects among three decision modes based on the current calibration state, and transitions generated by all three modes are used to update the residual calibrator and calibration-state statistics. Specifically, the routing process jointly considers: (1) whether the frozen base controller shows significant drift, measured by the recent drift statistic $\bar{D}_t^{\text{base}}$; (2) whether the residual calibrator can already reduce such drift, measured by the recent residual calibration error $\bar{d}_t^C$; and (3) whether the learned residual correction is sufficiently credible to influence the current decision, measured by the average action-wise credibility $\bar{\rho}_t = \frac{1}{|\mathcal{A}|}\sum_{a \in \mathcal{A}} \rho_t(a)$. Based on these quantities, the action routing forms three decision modes:

- ***Normal calibrated decision mode.*** The action is selected using the released residual correction $\rho_t(a)\Delta Q_{\phi_t}(s_t, a)$. This mode is used when the base controller does not show significant drift, or when the residual calibrator can already handle the drift with sufficient credibility.
- ***Release-boost decision mode.*** When the residual calibration error is already small but the action-wise credibility coefficients remain conservative due to their executed-action-based updates, the routing temporarily increases the release strength by $\rho_t^{\text{exec}}(a) = \min(1, \max[\rho_t(a), \rho_{\min}^{\text{exec}}])$ for the current decision.

- ***Physical decision mode.*** When both the drift and the residual calibration error are high, the routing uses a maximum released-lane queue-length score $\psi_t^{\text{phy}}(a) = \sum_{l \in \mathcal{L}(a)} q_t^l$, where $\mathcal{L}(a)$ is the set of incoming lanes released by action a and q_t^l is the queue length of lane l.

Further explanations are provided in Appendix C.4. The final routed action is:

$$a_t = \begin{cases} \arg\max_{a \in \mathcal{A}} \psi_t^{\text{phy}}(a), & \bar{D}_t^{\text{base}} > \tau_D \text{ and } \bar{d}_t^C > \tau_C, \\ \arg\max_{a \in \mathcal{A}} \left[Q_\theta(s_t, a) + \rho_t^{\text{exec}}(a) \Delta Q_{\phi_t}(s_t, a) \right], & \bar{D}_t^{\text{base}} > \tau_D \text{ and } \bar{d}_t^C \le \tau_C \text{ and } \bar{\rho}_t < \tau_\rho, \\ \arg\max_{a \in \mathcal{A}} \left[Q_\theta(s_t, a) + \rho_t(a) \Delta Q_{\phi_t}(s_t, a) \right], & \text{otherwise.} \end{cases} \tag{2}$$

4.4 Dynamic Replay Update

During training, abundant samples can be collected through exploratory interaction. At test time, however, repeated trial-and-error interaction is impractical, making the available interaction feedback more limited. CRC therefore prioritizes more informative transitions. Specifically, the importance of a transition for updating depends on both the degree of drift and the remaining residual calibration error. To this end, CRC adopts a *Dynamic Replay Update* strategy that prioritizes test-time transitions for residual calibrator optimization according to time-varying replay weights.

Dynamic replay weight. Each transition is stored as $e_t = (s_t, a_t, r_t, s_{t+1}, D_t^{\text{base}}, d_t^C, t, n_t)$ in the replay buffer $\mathcal{B}_t$, where $n_t = 0$ denotes its replay usage count. A sliding-window buffer can be used to keep recent samples, i.e., $\mathcal{B}_t = \{e_i \mid t - W < i \le t\}$, where each step contains transitions from all N intersections. For each replay sample e_i, its current calibration error d_i^C is recomputed using the calibrator with its current parameters, and the dynamic replay weight is assigned as:

$$w_i^{\text{replay}} = \underbrace{\text{clip}\left(\frac{D_i^{\text{base}}}{\tau_D}, 0, 1\right)}_{w_i^{\text{drift}}} \cdot \underbrace{\text{clip}\left(\frac{d_i^C}{\tau_C}, 0, 1\right)}_{u_i^{\text{cal}}} \cdot \underbrace{\frac{1}{\sqrt{n_i + 1}}}_{u_i^{\text{use}}}. \tag{3}$$

Here, w_i^{drift} emphasizes samples collected under stronger drift, u_i^{cal} emphasizes samples with larger remaining calibration error, and u_i^{use} downweights frequently replayed samples.

Prioritized replay update. Based on w_i^{replay}, mini-batches are sampled according to $p_i = (w_i^{\text{replay}} + \epsilon_p)^\alpha / \sum_{j \in \mathcal{B}_t} (w_j^{\text{replay}} + \epsilon_p)^\alpha$, where p_i denotes the sampling probability of e_i. For each sampled mini-batch $\mathcal{M}_g$, the residual calibrator is updated by minimizing the weighted TD loss:

$$\mathcal{L}_{\text{cal}} = \frac{1}{|\mathcal{M}_g|} \sum_{e_i \in \mathcal{M}_g} w_i^{\text{replay}} (\delta_i^C)^2. \tag{4}$$

After each mini-batch update, the usage count of each sampled transition is increased as $n_i \leftarrow n_i + 1$ for $e_i \in \mathcal{M}_g$. The target calibrator is synchronized with the trainable residual calibrator at a fixed frequency. The overall CRC procedure and main notations are summarized in Algorithm 1 and Table 3, respectively, in Appendix C.6 and Appendix C.7.

5 Experiments

In this section, we conduct extensive experiments to answer five research questions (RQs): **RQ1 (*Effectiveness*)**, whether CRC can consistently improve the test-time control performance of different base TSC controllers and how it compares with existing baselines; **RQ2 (*Adaptation Behavior*)**, how CRC adapts during test time in terms of calibration-state evolution, action-routing behavior, action-value calibration, and changes in action decisions; **RQ3 (*Mechanism*)**, how individual CRC components contribute to performance gains and how sensitive CRC is to its key hyperparameters; **RQ4 (*Adaptability*)**, whether CRC remains effective under diverse drift scenarios, including varying drift intensities, multiple drift periods, and unseen intersections; and **RQ5 (*Efficiency*)**, what computational overhead CRC introduces during test-time control.

5.1 Experimental Setup

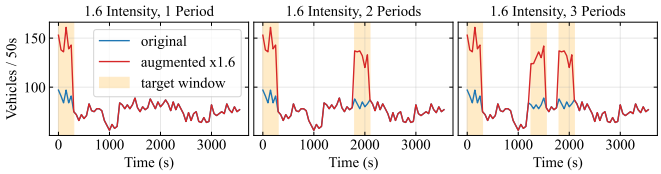


Figure 2: Example construction of drift scenario on JN3.

Datasets and Evaluation Metrics. We evaluate CRC on five public TSC benchmarks (Wei et al., 2019b): HZ1, HZ2, JN1, JN2, and JN3, and JN3, using the CityFlow simulator (Zhang et al., 2019). Each base controller is trained on the original scenario and frozen at test time. We construct diverse drift scenarios by varying valley/peak demand, drift intensity, drift periods, and intersections, as shown in Fig. 2, and compare each base controller with its CRC-enhanced counterpart under these scenarios. We use Average Travel Time (ATT) and Average Waiting Time (AWT) as metrics, where lower values indicate better performance (Zheng et al., 2019b). Details of scenario construction and metric definitions are provided in Appendices D.1 and D.2.

Baselines. We use four representative RL-based TSC methods as base controllers, including CoLight, FRAP, MPLight, and PressLight. We further compare CRC with three meta-RL-based TSC methods, including MetaLight (Zang et al., 2020), MetaGAT (Lou et al., 2022), and MetaVIM (Zhu et al., 2023), and three LLM-based TSC methods, including LLMLight (Lai et al., 2025), CoLLMLight (Yuan et al., 2026), and Traffic-R1 (Zou et al., 2026). In addition, we construct a test-time RL baseline (Zuo et al., 2025) that directly adapts the base controller using test-time interaction feedback. Detailed settings of the baselines are provided in Appendix D.3.

Implementation Details. Experiments are conducted on a server with an Intel Xeon Platinum 8358P CPU and an NVIDIA A40 GPU. CRC uses a three-layer Linear-ReLU residual calibrator with hidden dimension $d_h = 64$ and gate lower bound $g_{\min} = 0.5$. The calibrator is optimized by Adam with a learning rate of 1×10^{-3} and mini-batch size $B = 64$, while dynamic replay uses $G = 32$ updates per trigger. The target calibrator is synchronized every 64 steps, with a sliding-window size of 40 and $\rho_{\min}^{\text{exec}} = 0.9$. Each experiment is repeated five times, and the results are reported as mean $\pm$ standard deviation. Detailed settings are provided in Table 4 in Appendix D.4.

5.2 Effectiveness (RQ1)

Consistent Effectiveness. Table 1 reports the average performance across the valley- and peak-boost scenarios with different drift intensities constructed from the same original benchmark dataset. Our observations are as follows: (1) CRC consistently improves all four base controllers across all five benchmarks and both metrics, with average reductions of 6.88% in ATT and 25.53% in AWT. (2) Moreover, CRC narrows the performance gaps among different base controllers, especially on the JN benchmarks, suggesting that it can partially compensate for differences in their original control capability. (3) Since each comparison uses the same base controller and test scenario, these gains can be attributed to the test-time calibration introduced by CRC. More results are reported in Table 5 in Appendix D.5.

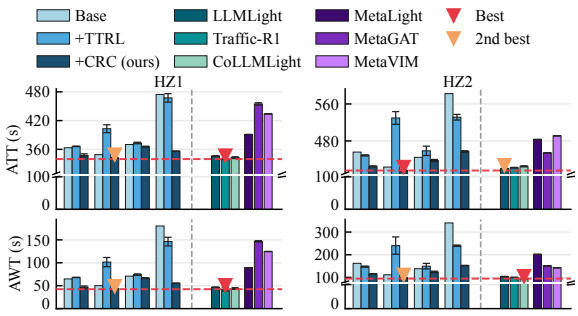


Figure 3: Performance on HZ1 and HZ2. The leftmost four groups in each panel correspond to FRAP, CoLight, MPLight, and PressLight.

Competitive Effectiveness. Fig. 3 presents the comparative results. Our observations are as follows: (1) LLM-based methods show strong test-time performance, while CRC-enhanced RL controllers remain competitive and achieve the best results in several cases. (2) Meta-RL-based methods generally perform worse, possibly because their adaptation relies on meta-learned priors and limited test-time updates, which may be insufficient for continuously evolving traffic conditions. (3) The TTRL-style direct-update baseline does not yield consistent gains and can even degrade the frozen base controller, highlighting the benefit of frozen-controller residual calibration. More comprehensive results are reported in Tables 6–10 in Appendix D.5.

Table 1: Performance comparison of different base controllers with and without CRC.

Benchmark	Metric	*FRAP*			*CoLight*			*MPLight*			*PressLight*		
w/ CRC		×	✓	Δ(%)	×	✓	Δ(%)	×	✓	Δ(%)	×	✓	Δ(%)
HZ1	ATT	363.38	347.47±3.25	↓ **4.38**	348.96	342.34±0.56	↓ **1.90**	370.10	365.72±0.82	↓ **1.18**	474.49	356.12±0.10	↓ **24.95**
	AWT	64.81	46.98±2.39	↓ **27.51**	50.10	43.65±0.54	↓ **12.87**	70.88	66.55±0.82	↓ **6.10**	180.22	55.28±0.16	↓ **69.32**
HZ2	ATT	455.93	425.10±1.06	↓ **6.76**	423.57	416.07±1.14	↓ **1.77**	444.60	437.62±2.39	↓ **1.57**	582.31	457.12±1.74	↓ **21.50**
	AWT	162.23	115.83±0.25	↓ **28.60**	111.58	98.62±2.76	↓ **11.62**	138.37	124.16±3.48	↓ **10.27**	340.28	152.35±0.91	↓ **55.23**
JN1	ATT	360.96	328.14±0.04	↓ **9.09**	323.55	321.41±0.45	↓ **0.66**	351.21	333.46±0.11	↓ **5.05**	354.97	332.23±0.01	↓ **6.41**
	AWT	121.06	85.75±0.22	↓ **29.17**	83.74	79.39±0.50	↓ **5.20**	109.65	89.50±0.08	↓ **18.38**	119.16	88.16±0.20	↓ **26.02**
JN2	ATT	326.45	304.51±0.02	↓ **6.72**	305.59	299.14±0.23	↓ **2.11**	322.07	303.95±0.02	↓ **5.63**	310.30	298.97±0.04	↓ **3.65**
	AWT	79.32	57.16±0.03	↓ **27.93**	59.16	52.25±0.17	↓ **11.69**	74.97	55.87±0.05	↓ **25.47**	63.12	50.90±0.08	↓ **19.36**
JN3	ATT	332.91	297.09±1.05	↓ **10.76**	299.24	293.28±0.71	↓ **1.99**	327.02	297.42±0.30	↓ **9.05**	337.29	295.08±0.07	↓ **12.51**
	AWT	94.72	58.76±0.79	↓ **37.97**	62.05	55.65±0.73	↓ **10.31**	89.73	57.67±0.33	↓ **35.73**	95.21	55.43±0.05	↓ **41.79**

Blue denotes the best result without CRC, and **magenta** denotes the best result with CRC for each benchmark–metric pair. Δ reports the relative reduction achieved by CRC over the corresponding base controller, where a larger value indicates a greater improvement.

5.3 Adaptation Behavior (RQ2)

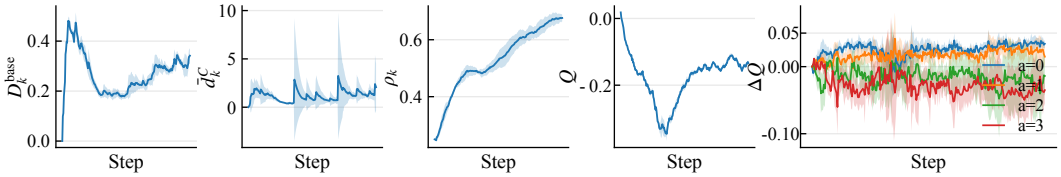


Figure 4: Test-time adaptation behavior of CRC on FRAP under the HZ1 drift scenarios.

Calibration-State Statistics and Action-Value Calibration. Fig. 4 shows the test-time evolution of CRC on FRAP under HZ1 drift. The base inconsistency and calibration error change with test-time feedback, while credibility gradually increases. Meanwhile, CRC produces action-dependent residuals that continuously reshape the relative action values. More comprehensive adaptation statistics are provided in Figs. 14–34 in Appendix D.6.

Action Selection. The normal calibrated, physical, and release-boost modes account for 53.89%, 25.62%, and 20.49% of the decisions, respectively, indicating dynamic routing across different test-time conditions. CRC also changes 34.67% of the actions selected by the base controller, showing substantial but selective decision adjustment.

5.4 Mechanism Analysis (RQ3)

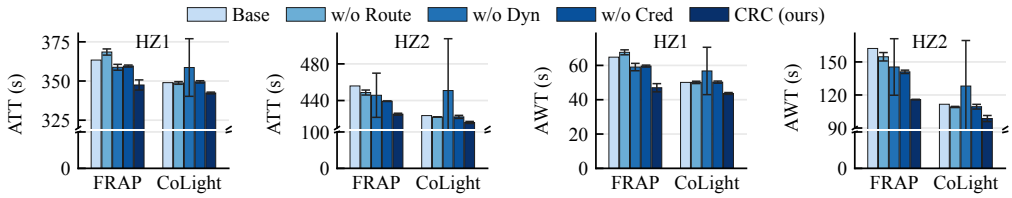


Figure 5: Ablation results of CRC on HZ1 and HZ2, averaged over valley- and peak-boost scenarios with different drift intensities.

Ablation Study. From Fig. 5, we consider four variants: (1) *w/o Cred*, setting action-wise credibility to 1; (2) *w/o Dyn*, updating the calibrator with all collected test-time samples; (3) *w/o Route*, always selecting $\arg\max_a Q_t^{\text{exec}}(s_t, a)$; and (4) *Base*, using the frozen base controller only. The complete CRC achieves the best performance across controllers, datasets, and metrics, while removing any component degrades performance, confirming the complementary roles of credible residual release, dynamic replay, and action routing. Detailed results are provided in Table 11 in Appendix D.7.

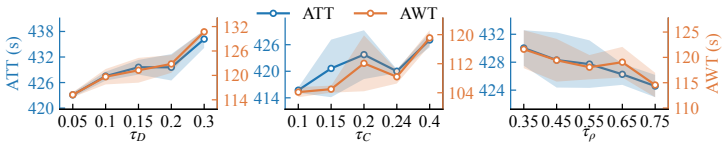
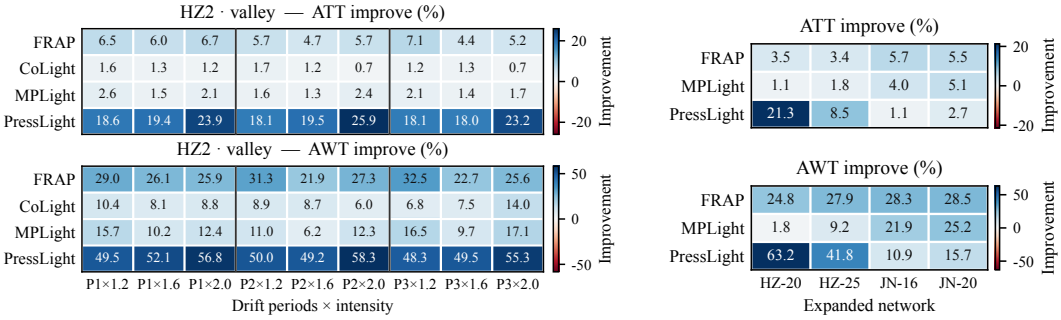


Figure 6: Sensitivity of CRC to the action-routing thresholds.

Sensitivity Analysis. Fig. 6 reports the sensitivity of the routing thresholds for FRAP on HZ2. Within the tested ranges, increasing τ_D or τ_C generally degrades performance, suggesting that earlier drift response and stricter calibration-error screening are beneficial. In contrast, larger τ_ρ generally improves performance, indicating that a stricter credibility criterion before normal calibrated decision making is preferable. Detailed results are reported in Table 12 in Appendix D.7.

5.5 Adaptability (RQ4)



(a) HZ2 valley-boost drift.

(b) Network expansion.

Figure 7: Relative ATT/AWT improvement of CRC under (a) varying drift intensities and periods and (b) network expansion with unseen intersections.

Drift Intensities and Periods. As shown in Fig. 7a, CRC consistently improves both metrics across all base controllers under diverse drift scenarios, indicating stable adaptability to varying drift intensities and temporal patterns. More results are provided in Figs. 35–41 in Appendix D.8.

Unseen Intersections. We further evaluate CRC on expanded networks with different numbers of intersections unseen during training. Since CoLight explicitly depends on the road-network topology, it is not directly applicable to scenarios with newly added intersections. As shown in Fig. 7b, CRC reduces both metrics across different expanded networks and base controllers, demonstrating its adaptability to previously unseen intersections.

5.6 Efficiency (RQ5)

Using FRAP as the base controller, we evaluate CRC in terms of average per-step time and memory usage over multiple augmented HZ1 scenarios and compare it with representative baselines. As shown in Fig. 8, CRC requires only 2.48 s per step and 17.9 MB of memory. CRC incurs a computational overhead comparable to TTRL. Compared with the test-time computational cost reported for Traffic-R1, CRC requires substantially less per-step computation time and memory. These results demonstrate that CRC achieves test-time adaptation with substantially lower computational overhead, indicating favorable efficiency for test-time signal control.

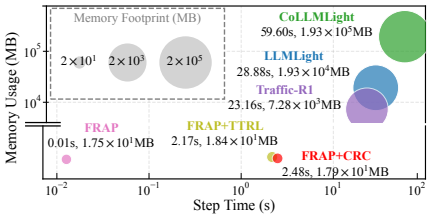


Figure 8: Efficiency comparison.

6 Conclusion

This paper studies test-time adaptation for RL-based TSC under evolving traffic dynamics. We introduce ACFD drift and propose CRC, which adapts a frozen base controller through credible residual calibration, action routing, and dynamic replay. Limitations are discussed in Appendix C.8.

AI USE STATEMENT

Generative AI tools were used for translation, writing polish, and limited code-related assistance. The authors used AI-assisted code to implement the specified data augmentation procedures for constructing test-time traffic scenarios from existing benchmark datasets. The AI tools did not independently determine the experimental settings or dataset construction procedures. All AI-assisted outputs, including text, code, and constructed datasets, were reviewed and validated by the authors. The authors made all final methodological decisions, analyses, interpretations, and conclusions and take full responsibility for this work.

REPRODUCIBILITY STATEMENT

We provide the methodological details necessary to reproduce our approach in the main paper, including the problem formulation, model architecture, and test-time adaptation procedures. Detailed experimental settings, dataset descriptions, and implementation details are provided in the Appendix. Anonymous source code, configuration files, and scripts for reproducing the experiments are provided in the supplementary material.

REFERENCES

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE signal processing magazine*, 34(6):26–38, 2017.

Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3414–3421, 2020.

Hanyang Chen, Yang Jiang, Shengnan Guo, Xiaowei Mao, Youfang Lin, and Huaiyu Wan. Difflight: a partial rewards conditioned diffusion model for traffic signal control with missing data. *Advances in Neural Information Processing Systems*, 37:123353–123378, 2024.

Yang Jiang, Shengnan Guo, Hanyang Chen, Xiaowei Mao, Youfang Lin, and Huaiyu Wan. Proactive-xlight: Proactive traffic signal control with pluggable and reliable traffic prediction. *IEEE Transactions on Mobile Computing*, 24(11):11465–11479, 2025.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

Siqi Lai, Zhao Xu, Weijia Zhang, Hao Liu, and Hui Xiong. Llmlight: Large language models as traffic signal control agents. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 2335–2346, 2025.

Yican Lou, Jia Wu, and Yunchuan Ran. Meta-reinforcement learning for multiple traffic signals control. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 4264–4268, 2022.

Afshin Oroojlooy, Mohammadreza Nazari, Davood Hajinezhad, and Jorge Silva. Attendlight: Universal attention-based reinforcement learning model for traffic signal control. *Advances in Neural Information Processing Systems*, 33:4079–4090, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1290–1298, 2019a.

Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1913–1922, 2019b.

Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation. *ACM SIGKDD explorations newsletter*, 22(2):12–18, 2021.

Zirui Yuan, Siqi Lai, and Hao Liu. Collmlight: Cooperative large language model agents for network-wide traffic signal control. In *International Conference on Learning Representations*, volume 2026, pp. 66687–66715, 2026.

Xinshi Zang, Huaxiu Yao, Guanjie Zheng, Nan Xu, Kai Xu, and Zhenhui Li. Metalight: Value-based meta-reinforcement learning for traffic signal control. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 1153–1160, 2020.

Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The world wide web conference*, pp. 3620–3624, 2019.

Junping Zhang, Fei-Yue Wang, Kunfeng Wang, Wei-Hua Lin, Xin Xu, and Cheng Chen. Data-driven intelligent transportation systems: A survey. *IEEE transactions on intelligent transportation systems*, 12(4):1624–1639, 2011.

Dongbin Zhao, Yujie Dai, and Zhen Zhang. Computational intelligence in urban traffic signal control: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):485–494, 2011.

Guanjie Zheng, Yuanhao Xiong, Xinshi Zang, Jie Feng, Hua Wei, Huichu Zhang, Yong Li, Kai Xu, and Zhenhui Li. Learning phase competition for traffic signal control. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1963–1972, 2019a.

Guanjie Zheng, Xinshi Zang, Nan Xu, Hua Wei, Zhengyao Yu, Vikash Gayah, Kai Xu, and Zhenhui Li. Diagnosing reinforcement learning for traffic signal control. *arXiv preprint arXiv:1905.04716*, 2019b.

Liwen Zhu, Peixi Peng, Zongqing Lu, and Yonghong Tian. Metavim: Meta variationally intrinsic motivated reinforcement learning for decentralized traffic signal control. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11570–11584, 2023.

Xingchen Zou, Yuhao Yang, Zheng Chen, Xixuan Hao, Yiqi Chen, Chao Huang, and Yuxuan Liang. Traffic-r1: Reinforced llms bring human-like reasoning to traffic signal control systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21823–21838, 2026.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Xinwei Long, Ermo Hua, Biqing Qi, et al. Ttrl: Test-time reinforcement learning. *Advances in Neural Information Processing Systems*, 38:131459–131483, 2025.

APPENDIX

TABLE OF CONTENTS

A More Related Work **12**

A.1 RL-Based TSC Methods 12

A.1.1 Representative Base Controllers 13

A.1.2 Meta-RL for Rapid Adaptation 13

A.2 LLM-Based TSC Methods 13

B Detailed Preliminaries **14**

C Method Details **15**

C.1 Why Bellman Inconsistency Can Reflect ACFD drift 15

C.2 Discussion on Calibration-State Statistics 16

C.3 Construction of Intersection Statistical Features 18

C.4 Further Explanation of Action Routing Modes 19

C.5 Computational Complexity of the Residual Calibrator 19

C.6 Complete CRC Algorithm 20

C.7 Notation Summary 20

C.8 Limitations 20

D Experimental Details **20**

D.1 Construction of Test-Time Drift Scenarios 20

D.2 Evaluation Metrics 23

D.3 Baseline Details 24

D.4 Hyperparameter Settings 24

D.5 Additional Results for RQ1: Effectiveness 27

D.6 Additional Results for RQ2: Adaptation Behavior 27

D.7 Additional Results for RQ3: Mechanism Analysis 27

D.8 Additional Results for RQ4: Adaptability 28

A MORE RELATED WORK

A.1 RL-BASED TSC METHODS

We first introduce four representative RL-based TSC methods that serve as the base controllers in our experiments, on which the proposed CRC is further applied. Beyond these base controllers, we summarize RL-based studies that aim to improve rapid adaptation. A representative line of work is meta-reinforcement learning-based TSC, which treats different intersections, traffic-flow patterns, or road-network environments as related tasks and learns transferable parameter initializations or task

representations through meta-training. With such prior knowledge, the learned controller can adapt to unseen traffic scenarios using limited online interactions or a small number of gradient updates.

A.1.1 REPRESENTATIVE BASE CONTROLLERS

CoLight (Wei et al., 2019b). CoLight achieves cooperative control of multiple intersections through graph attention mechanism.

FRAP (Zheng et al., 2019a). FRAP explicitly models the rotational invariance of traffic states and the competition among candidate signal phases, thereby learning a signal control policy with generalization.

PressLight (Wei et al., 2019a). PressLight formulates traffic pressure according to the imbalance of vehicle accumulation between connected road segments. It further incorporates this pressure into the state representation and reward design to improve vehicle discharge efficiency and network throughput.

MPLight (Chen et al., 2020). MPLight adopts FRAP as its base control network and uses traffic pressure to construct both state features and reward signals. It further shares model parameters across intersections to improve scalability.

A.1.2 META-RL FOR RAPID ADAPTATION

MetaLight (Zang et al., 2020). MetaLight learns a transferable parameter initialization through meta-reinforcement learning. When adapting to a new traffic condition, the controller can quickly adapt using only a small number of samples and several gradient updates.

MetaGAT (Lou et al., 2022). MetaGAT constructs task context from historical states, actions, rewards, and state transitions. A task encoder is then used to infer task characteristics online, while a graph attention network supports coordination among multiple intersections.

MetaVIM (Zhu et al., 2023). MetaVIM infers task-related latent variables from historical trajectories to implicitly represent traffic patterns and neighboring influences. These latent representations improve the adaptability of a shared policy.

However, these methods mainly acquire their adaptation capability before testing and are usually designed for adapting to unseen but relatively stable tasks. They remain limited in explicitly sensing feedback changes that emerge during the test-time control process and continuously adapting the controller according to evolving online interactions.

A.2 LLM-BASED TSC METHODS

Recent studies have also explored LLMs for TSC by converting numerical traffic states into language-based descriptions and exploiting the commonsense knowledge and reasoning capabilities of LLMs.

LLMLight (Lai et al., 2025). LLMLight assigns an LLM agent to each intersection and converts numerical traffic observations into language prompts. The LLM directly analyzes traffic conditions and selects signal actions, while domain-specific fine-tuning further improves its traffic control capability. This work establishes the basic paradigm of directly employing LLMs as traffic signal control agents.

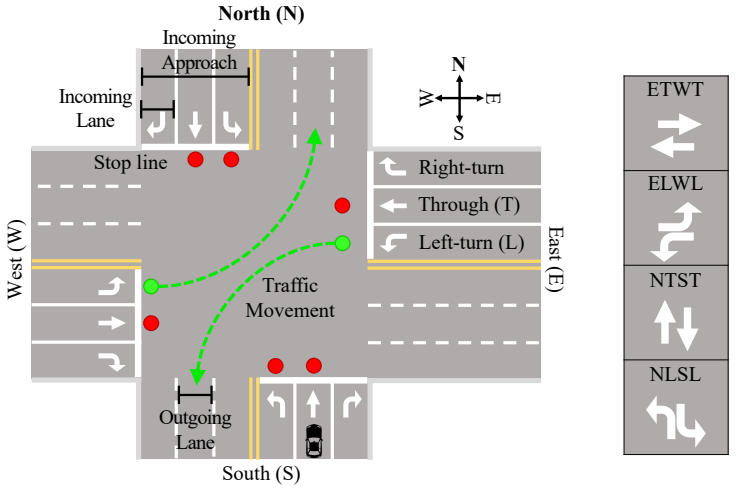
CoLLMLight (Yuan et al., 2026). CoLLMLight addresses the lack of multi-intersection coordination in LLMLight by introducing cooperative reasoning and control among multiple LLM agents. It further extends LLM-based TSC from isolated intersection control to network-level spatiotemporal coordination.

Traffic-R1 (Zou et al., 2026). Traffic-R1 combines human expert knowledge with environment-based self-exploration to develop internal traffic-control reasoning capabilities. It constructs a traffic foundation model designed for generalization to open-world traffic scenarios.

Overall, LLM-based methods enhance zero-shot generalization through pretrained knowledge, contextual reasoning, and cross-task transferability.

B Detailed Preliminaries

Figure 9 illustrates the basic structure of a signalized intersection. Based on this illustration, we provide detailed definitions of the traffic elements, observations, and control actions used in this work. We further present the candidate signal phases.



(a) Signalized intersection (b) Four phases

Figure 9: An illustration of a signalized intersection and phases.

Definition B.1 (Signalized Intersection) *A signalized intersection is an intersection where vehicle movements are regulated by traffic signals. It generally consists of multiple incoming approaches, outgoing approaches, lanes, and traffic signals.*

Definition B.2 (Incoming Lane and Incoming Approach) *An incoming lane is a lane through which vehicles enter an intersection. A group of incoming lanes forms an incoming approach.*

Definition B.3 (Outgoing Lane) *An outgoing lane is a lane through which vehicles leave an intersection after completing a permitted traffic movement.*

Definition B.4 (Traffic Movement) *A traffic movement describes the direction in which vehicles travel through an intersection. Traffic movements are divided into three types: left-turn, through, and right-turn movements.*

Definition B.5 (Lane-Level Vehicle Count) *The lane-level vehicle count denotes the number of vehicles currently located on an incoming lane. It reflects the traffic demand and vehicle accumulation of the corresponding lane.*

Definition B.6 (Traffic Signal Indication) *A traffic signal indication determines whether a traffic movement is permitted to enter the intersection. A green indication grants the corresponding movement the right-of-way, whereas a red indication prohibits the movement from entering the intersection. A yellow interval is inserted during signal transitions to allow vehicles that have already entered the intersection to clear safely. In our experimental setting, the yellow interval lasts for five seconds.*

Definition B.7 (Signal Phase) *A signal phase is a combination of mutually non-conflicting traffic movements that are simultaneously assigned green indications. Right-turn movements are excluded from the phase definition because they are assumed to be continuously permitted in our experimental setting. Figure 9(b) illustrates the four candidate phases.*

Definition B.8 (Traffic State) *At decision step t, the traffic state s_t is constructed from observable traffic information at the intersection.*

Definition B.9 (Traffic Signal Control Action) *At each decision step t, the controller selects an action $a_t \in \mathcal{A}$, where each action corresponds to activating one candidate signal phase. The selected phase remains active for the fixed control duration t_{duration} before the next decision is made. In our experimental setup, t_{duration} is set to 15 seconds. When the selected phase differs from the currently active phase, the first 5 s of the t_{duration} are used for the yellow/all-red transition, and the remaining time is used as the effective green period of the newly selected phase. For example, the first 5 s are used for signal transition and the remaining 10 s correspond to the effective green time. When the selected phase remains unchanged, no transition interval is required, and the entire t_{duration} is used as green time.*

Definition B.10 (Traffic Interaction) *After executing action a_t under state s_t, the traffic environment returns an immediate reward r_t and transitions to the next state s_{t+1}. A traffic interaction sample is represented as (s_t, a_t, r_t, s_{t+1}), which provides the test-time information used to update the residual calibrator. The definitions of state, action, and reward follow the original design of each base controller.*

C Method Details

C.1 Why Bellman Inconsistency Can Reflect ACFD drift

In this section, we provide a more detailed explanation of why the Bellman inconsistency of the frozen base controller can serve as an observable value-relevant signal of ACFD drift.

For a given state-action pair (s, a), let $p_{\text{train}}(s', r \mid s, a)$ and $p_{\text{test}}(s', r \mid s, a)$ denote the action-conditioned feedback distributions in the training and test environments, respectively. ACFD drift occurs when these two conditional distributions are different:

$$p_{\text{test}}(s', r \mid s, a) \neq p_{\text{train}}(s', r \mid s, a). \tag{5}$$

For the frozen base controller Q_θ, the Bellman backup (Kumar et al., 2019) under the training environment can be written as

$$\mathcal{T}_{\text{train}} Q_\theta(s, a) = \mathbb{E}_{(s', r) \sim p_{\text{train}}(\cdot \mid s, a)} \left[r + \gamma \max_{a'} Q_\theta(s', a') \right]. \tag{6}$$

Similarly, the corresponding Bellman backup under the test environment is

$$\mathcal{T}_{\text{test}} Q_\theta(s, a) = \mathbb{E}_{(s', r) \sim p_{\text{test}}(\cdot \mid s, a)} \left[r + \gamma \max_{a'} Q_\theta(s', a') \right]. \tag{7}$$

Since Q_θ is pretrained in the training environment, it is expected to approximately satisfy the Bellman consistency condition under the training distribution:

$$Q_\theta(s, a) \approx \mathcal{T}_{\text{train}} Q_\theta(s, a). \tag{8}$$

At test time, after executing action a_t and observing feedback (r_t, s_{t+1}), CRC computes the base Bellman inconsistency as

$$\delta_t^{\text{base}} = r_t + \gamma \max_{a'} Q_\theta(s_{t+1}, a') - Q_\theta(s_t, a_t). \tag{9}$$

Taking expectation over the test-time feedback distribution conditioned on $(s_t = s, a_t = a)$ gives

$$\mathbb{E}_{\text{test}} \left[\delta_t^{\text{base}} \mid s_t = s, a_t = a \right] = \mathcal{T}_{\text{test}} Q_\theta(s, a) - Q_\theta(s, a). \tag{10}$$

By substituting Eq. 8 into Eq. 10, we have

$$\mathbb{E}_{\text{test}} \left[\delta_t^{\text{base}} \mid s, a \right] \approx \mathcal{T}_{\text{test}} Q_\theta(s, a) - \mathcal{T}_{\text{train}} Q_\theta(s, a). \tag{11}$$

Expanding the two Bellman backups yields

$$\mathbb{E}_{\text{test}} \left[\delta_t^{\text{base}} \mid s, a \right] \approx \int \left[r + \gamma \max_{a'} Q_\theta(s', a') \right] \left[p_{\text{test}}(s', r \mid s, a) - p_{\text{train}}(s', r \mid s, a) \right] ds' dr. \tag{12}$$

Eq. 12 shows that the expected Bellman inconsistency of the frozen base controller is determined by the difference between the training and test action-conditioned feedback distributions, weighted by the Bellman target function. Therefore, when the test-time feedback distribution changes in a way that affects the immediate reward or the next-state value estimate, the frozen base controller will produce a larger Bellman inconsistency. In this sense, δ_t^{base} can be viewed as a value-level projection of ACFD drift.

It should be noted that Bellman inconsistency does not directly characterize the complete temporal evolution of the distributional divergence between $p_{\text{test}}(s', r \mid s, a)$ and $p_{\text{train}}(s', r \mid s, a)$ throughout the test-time process. Instead, it captures the part of ACFD drift that is relevant to the action-value estimate of the frozen base controller. This property is desirable for test-time TSC, because the goal is not to reconstruct the complete feedback distribution, but to identify whether the current feedback makes the frozen action-value estimate unreliable for test-time control.

In practice, only one feedback sample (r_t, s_{t+1}) is observed after executing a_t at each test-time step. Therefore, CRC uses the sample-level Bellman inconsistency in Eq. 9 as an instantaneous drift signal, and normalizes it as

$$D_t^{\text{base}} = \frac{|\delta_t^{\text{base}}|}{|y_t^{\text{base}}| + \epsilon}, \quad y_t^{\text{base}} = r_t + \gamma \max_{a'} Q_\theta(s_{t+1}, a'). \tag{13}$$

To reduce the variance caused by single-step feedback, CRC further maintains a moving average:

$$\bar{D}_{t+1}^{\text{base}} = (1 - \beta_D)\bar{D}_t^{\text{base}} + \beta_D D_t^{\text{base}}. \tag{14}$$

The resulting statistic $\bar{D}_t^{\text{base}}$ provides a recent value-relevant estimate of ACFD drift and is used for credible residual release, action routing, and dynamic replay update.

Overall, Bellman inconsistency provides an observable and control-oriented signal of ACFD drift: it becomes large when test-time action-conditioned feedback changes in a way that alters the Bellman target of the frozen base controller.

C.2 Discussion on Calibration-State Statistics

Comparison of Calibration-State Statistics. Although the three calibration-state statistics are all derived from test-time Bellman-style feedback, they answer different questions in CRC:

- $\bar{D}_t^{\text{base}}$ reflects whether the frozen base controller encounters ACFD drift.
- $\bar{d}_t^C$ reflects how much inconsistency remains after residual calibration.
- $\rho_t(a)$ reflects whether the learned residual correction for action a is credible enough to be released for action selection.

Table 2 summarizes the comparison of these statistics in terms of their computation and interpretation.

Relationship among Calibration-State Statistics. The three calibration-state statistics are complementary rather than mutually determined. In particular, a larger base drift does not necessarily imply lower credibility, and a smaller calibration error does not necessarily lead to higher credibility. To see this, the instantaneous credibility is computed as

$$\iota_t = \frac{(\delta_t^{b|c})^2 - (\delta_t^C)^2}{(\delta_t^{b|c})^2 + (\delta_t^C)^2 + \epsilon}, \qquad c_t = \text{clip}(\iota_t, 0, 1), \tag{15}$$

where $\delta_t^{b|c}$ denotes the Bellman error of the frozen base controller under the mixed target, while δ_t^C denotes the remaining error after residual calibration.

Let $x_t = (\delta_t^{b|c})^2$ and $z_t = (\delta_t^C)^2$. Before clipping, the monotonicity of ι_t is given by

$$\frac{\partial \iota_t}{\partial x_t} = \frac{2z_t + \epsilon}{(x_t + z_t + \epsilon)^2} > 0, \qquad \frac{\partial \iota_t}{\partial z_t} = -\frac{2x_t + \epsilon}{(x_t + z_t + \epsilon)^2} < 0. \tag{16}$$

Table 2: Summary of calibration-state statistics in CRC.

Statistic	Computation	Interpretation
$\bar{D}_t^{\text{base}}$	$\delta_t^{\text{base}} = y_t^{\text{base}} - Q_\theta(s_t, a_t),$ $D_t^{\text{base}} = \frac{\lvert\delta_t^{\text{base}}\rvert}{\lvert y_t^{\text{base}}\rvert + \epsilon},$ $\bar{D}_{t+1}^{\text{base}} = (1-\beta_D)\bar{D}_t^{\text{base}} + \beta_D D_t^{\text{base}}.$	Recent value-level inconsistency of the frozen base controller under test-time feedback, serving as an observable signal of ACFD drift.
$\bar{d}_t^C$	$\delta_t^C = y_t^{\text{mix}} - [Q_\theta(s_t, a_t) + \Delta Q_{\phi_t}(s_t, a_t)],$ $d_t^C = \frac{\lvert\delta_t^C\rvert}{\lvert y_t^{\text{mix}}\rvert + \epsilon},$ $\bar{d}_{t+1}^C = (1-\beta_C)\bar{d}_t^C + \beta_C d_t^C.$	Recent remaining Bellman inconsistency after applying the residual correction, reflecting the current residual calibration error.
$\rho_t(a)$	$\delta_t^{b\lvert c} = y_t^{\text{mix}} - Q_\theta(s_t, a_t),$ $\iota_t = \frac{(\delta_t^{b\lvert c})^2 - (\delta_t^C)^2}{(\delta_t^{b\lvert c})^2 + (\delta_t^C)^2 + \epsilon},$ $c_t = \text{clip}(\iota_t, 0, 1),$ $\rho_{t+1}(a) = \begin{cases} (1-\beta_\rho)\rho_t(a) + \beta_\rho c_t, & a = a_t, \\ \rho_t(a), & a \neq a_t. \end{cases}$	Action-wise historical credibility of the residual correction, measuring whether the learned correction reduces the value error compared with the frozen base value.

Therefore, for a fixed calibration error, credibility increases when the base error is larger; for a fixed base error, credibility decreases as the remaining calibration error increases. More importantly, credibility depends on their *relative difference*, rather than either quantity alone.

A large $\bar{D}_t^{\text{base}}$ therefore does not necessarily imply high credibility. First, $\bar{D}_t^{\text{base}}$ is derived from the pure base target, whereas credibility compares $\delta_t^{b|c}$ and δ_t^C under the mixed target, so they are not directly coupled by a monotonic relationship. Second, even when the base controller exhibits substantial inconsistency, the residual correction is not credible unless it effectively reduces that inconsistency.

Similarly, a small $\bar{d}_t^C$ does not necessarily imply high credibility. Although d_t^C decreases with $|\delta_t^C|$ for a fixed target magnitude, credibility measures whether the residual correction improves upon the corresponding base error. If both $\delta_t^{b|c}$ and δ_t^C are small but close to each other, the residual provides little relative improvement and can still receive low credibility.

Furthermore, $\rho_t(a)$ is not determined by the current transition alone. For the executed action,

$$\rho_{t+1}(a_t) = (1-\beta_\rho)\rho_t(a_t) + \beta_\rho c_t, \tag{17}$$

so $\rho_t(a)$ accumulates credibility evidence over previously executed transitions. For a fixed historical value $\rho_t(a_t)$, $\rho_{t+1}(a_t)$ increases monotonically with c_t. However, because of this temporal smoothing and action-wise updating, the current credibility can lag behind changes in either base drift or calibration error. Hence, $\rho_t(a)$ should be interpreted as accumulated evidence that the residual correction for action a reliably improves the base value estimate, rather than as a direct function of the current drift or calibration error.

Numerical Illustration. Consider several illustrative transitions and omit the small ϵ for clarity. If $|\delta_t^{b|c}| = 1.0$ and $|\delta_t^C| = 0.9$, then

$$c_t \approx \frac{1.0^2 - 0.9^2}{1.0^2 + 0.9^2} = 0.105. \tag{18}$$

Although the base error is large, the residual correction only slightly reduces it, resulting in low credibility.

In contrast, if $|\delta_t^{b|c}| = 0.5$ and $|\delta_t^C| = 0.1$, then

$$c_t \approx \frac{0.5^2 - 0.1^2}{0.5^2 + 0.1^2} = 0.923, \tag{19}$$

indicating strong evidence that the residual correction is effective.

Finally, even a small calibration error does not guarantee high credibility. For example, with $|\delta_t^{b|c}| = 0.05$ and $|\delta_t^C| = 0.04$,

$$c_t \approx \frac{0.05^2 - 0.04^2}{0.05^2 + 0.04^2} = 0.220. \tag{20}$$

Although the absolute calibration error is small, the residual provides only limited improvement over the already small base error.

Overall, base drift characterizes the need for adaptation, calibration error measures the remaining inconsistency after adaptation, and credibility evaluates whether the learned residual correction provides reliable improvement over the base estimate.

C.3 Construction of Intersection Statistical Features

In CRC, the residual calibrator uses an intersection-level statistical feature vector z_t to describe the local traffic condition and signal-service status at each decision step. These features are constructed from directly observable traffic states and signal records, including the current phase, traffic demand intensity, queue congestion level, phase pressure, directional imbalance, and signal-service imbalance.

Let L denote the number of lanes at the intersection and K denote the number of candidate signal phases. At decision step t, the current phase is represented by a one-hot vector:

$$\mathbf{p}_t^{\text{phase}} = [p_t^1, p_t^2, \ldots, p_t^K], \tag{21}$$

where $p_t^k = 1$ if phase k is currently active, and $p_t^k = 0$ otherwise.

The traffic demand intensity is defined as the average number of vehicles over all lanes:

$$D_t = \frac{1}{L} \sum_{l=1}^{L} n_t(l), \tag{22}$$

where $n_t(l)$ denotes the number of vehicles on lane l. This feature reflects the overall traffic demand level at the intersection.

The queue congestion level is defined as:

$$Q_t^{\text{queue}} = \frac{1}{L} \sum_{l=1}^{L} q_t(l), \tag{23}$$

where $q_t(l)$ denotes the number of queued vehicles on lane l. Compared with D_t, this feature focuses more on the congestion level caused by vehicle waiting.

For each phase p, we compute its phase pressure as:

$$P_t(p) = \sum_{l \in \mathcal{L}^{\text{in}}(p)} q_t(l) - \sum_{l \in \mathcal{L}^{\text{out}}(p)} q_t(l), \tag{24}$$

where $\mathcal{L}^{\text{in}}(p)$ and $\mathcal{L}^{\text{out}}(p)$ denote the incoming and outgoing lanes associated with phase p, respectively. The phase pressure vector is then given by:

$$\mathbf{P}_t = [P_t(1), P_t(2), \ldots, P_t(K)]. \tag{25}$$

This vector describes the relative discharge demand of different candidate phases.

To characterize directional imbalance, we first compute the demand intensity D_t^m for each incoming direction m. The directional imbalance is then defined as:

$$I_t = \frac{\max_m D_t^m - \min_m D_t^m}{\sum_m D_t^m + \epsilon}, \tag{26}$$

where ϵ is a small constant for numerical stability. A larger I_t indicates that traffic demand is more unevenly distributed across incoming directions.

In addition, CRC records the unserved time of each phase. Let w_t^k denote the elapsed time since phase k was last served, and define:

$$\mathbf{w}_t = [w_t^1, w_t^2, \ldots, w_t^K]. \tag{27}$$

Based on this vector, the signal-service imbalance is computed as:

$$S_t = \frac{\mathrm{Std}(\mathbf{w}_t)}{\mathrm{Mean}(\mathbf{w}_t) + \epsilon}. \tag{28}$$

A larger S_t means that the service intervals among phases are more imbalanced, indicating that some phases may have been unserved for a relatively long time.

Finally, the intersection statistical feature vector is constructed by concatenating the above features:

$$z_t = \mathrm{Concat}\left[\mathbf{p}_t^{\mathrm{phase}}, D_t, Q_t^{\mathrm{queue}}, \mathbf{P}_t, I_t, \mathbf{w}_t, S_t\right]. \tag{29}$$

C.4 Further Explanation of Action Routing Modes

This section further explains the three decision modes in calibration-state-guided action routing.

Normal calibrated decision mode. This mode is used when the frozen base controller does not show significant drift, or when the residual calibrator has already handled the drift with sufficient credibility. In this case, the action is selected using the normally released residual correction $Q_\theta(s_t, a) + \rho_t(a)\Delta Q_{\phi_t}(s_t, a)$.

Release-boost decision mode. This mode is used when the base controller shows significant drift and the recent residual calibration error is already small, but the average action-wise credibility remains low. This condition indicates that the residual calibrator can already provide useful corrections, while the credibility-based release may still be conservative due to delayed credibility accumulation. Therefore, CRC temporarily increases the release strength by $\rho_t^{\mathrm{exec}}(a) = \min(1, \max[\rho_t(a), \rho_{\min}^{\mathrm{exec}}])$ for the current decision. Note that $\rho_t^{\mathrm{exec}}(a)$ is only used for current action selection and does not overwrite the historical credibility coefficient $\rho_t(a)$, which is still updated by the real feedback after action execution.

Physical decision mode. This mode is used when both the drift and the residual calibration error are high. In this case, the base value estimate is unreliable, and the residual calibrator has not yet sufficiently explained the current test-time feedback. CRC therefore selects the action with the highest physical score $\psi_t^{\mathrm{phy}}(a) = \sum_{l \in \mathcal{L}(a)} q_t^l$. The feedback obtained after executing this physically grounded action is still used to update the residual calibrator and the corresponding action-wise credibility coefficient.

C.5 Computational Complexity of the Residual Calibrator

We analyze the computational complexity of the residual calibrator for each intersection and each decision step. Let $K = |\mathcal{A}|$ denote the number of candidate signal phases, d_z denote the dimension of the intersection statistical feature z_t, d_h denote the hidden dimension of the calibrator, and L_c denote the number of MLP layers.

The calibrator input is constructed as $x_t = \mathrm{Concat}[z_t, \mathbf{Q}_\theta(s_t), \mathbf{A}_\theta(s_t), \chi_t]$, where $\chi_t = \mathrm{Concat}[\bar{D}_t^{\mathrm{base}}, \bar{d}_t^C, \boldsymbol{\rho}_t]$. Therefore, the input dimension is $d_x = d_z + 3K + 2$. Under the statistical feature construction introduced in Appendix C.3, z_t contains the current-phase vector, traffic demand intensity, queue congestion level, phase-pressure vector, directional imbalance, phase unserved-time vector, and signal-service imbalance. Thus, $d_z = 3K + 4$, giving $d_x = 6K + 6$.

For an MLP with hidden dimension d_h, the first layer requires $\mathcal{O}(d_x d_h)$ operations, while the remaining hidden layers require $\mathcal{O}((L_c - 1)d_h^2)$. The two action-wise output heads f_g and f_Δ each

generate a K-dimensional vector from h_t, resulting in a total complexity of $\mathcal{O}(Kd_h)$. The subsequent gate computation, element-wise combination $\eta_t(a) = g_t(a)\omega_t(a)$, and advantage-style centering over all candidate actions require only $\mathcal{O}(K)$ operations.

Therefore, the overall forward complexity of the residual calibrator at each decision step is $\mathcal{O}\left(d_x d_h + (L_c - 1)d_h^2 + Kd_h + K\right)$. Substituting $d_x = 6K + 6$, this becomes $\mathcal{O}\left((K+1)d_h + (L_c - 1)d_h^2 + K\right)$, where constant factors are omitted.

Since the pretrained base controller remains frozen, CRC does not introduce gradient computation or parameter updates for the base controller during test time.

C.6 Complete CRC Algorithm

Algorithm 1 summarizes the overall CRC procedure.

C.7 Notation Summary

A summary of the main notations used throughout this paper is provided in Table 3.

C.8 Limitations

At each test-time interaction step, only one action can be selected and executed from the action set, and the environment returns feedback only for this executed action. Therefore, CRC receives direct TD supervision only from the observed feedback (r_t, s_{t+1}) associated with the executed action, while the counterfactual feedback for the remaining unexecuted actions is unavailable. Consequently, the residual corrections for these unexecuted actions cannot be directly supervised by their own action-conditioned environmental feedback. Future work may explore counterfactual estimation to infer the outcomes of actions that are not selected and executed at each test-time step, thereby providing richer supervision for residual calibration.

D Experimental Details

D.1 Construction of Test-Time Drift Scenarios

Starting from the original JN1, JN2, JN3, HZ1, and HZ2 traffic flows, we construct test-time drift scenarios by modifying only selected temporal regions of the traffic demand while keeping the remaining traffic unchanged. Two basic drift patterns, namely *valley boost* and *peak boost*, are considered.

Valley boost. The original one-hour traffic flow is first divided into 50-second bins. For each consecutive 300-second window W, we count the total number of entering vehicles, denoted by $N(W)$. The valley window is selected as

$$W_{\text{valley}} = \arg\min_{W} N(W). \tag{30}$$

Only vehicles within this window are augmented. Specifically, vehicles in W_{valley} are sampled with replacement until the total number of vehicles in the window reaches approximately $\alpha N(W_{\text{valley}})$, while vehicles outside the selected window remain unchanged. This construction simulates an increase in traffic demand during an originally low-demand period.

Peak boost. Peak boost follows the same procedure, except that the 300-second window with the largest traffic demand is selected:

$$W_{\text{peak}} = \arg\max_{W} N(W). \tag{31}$$

Vehicles within W_{peak} are augmented to approximately $\alpha N(W_{\text{peak}})$, while the remaining traffic is unchanged. This construction simulates further intensification of an existing traffic peak.

For both valley boost and peak boost, we consider three augmentation factors:

$$\alpha \in \{1.2, 1.6, 2.0\}, \tag{32}$$

Algorithm 1 Credible Residual Calibration (CRC)

Initialize frozen base controller Q_θ
Initialize residual calibrator ΔQ_ϕ and target residual calibrator ΔQ_{ϕ^-}
Initialize replay buffer $\mathcal{B}_0 = \emptyset$
Initialize action-wise credibility $\rho_0(a) = c_{\text{prior}}$ for all $a \in \mathcal{A}$
Initialize moving statistics $\bar{D}_0^{\text{base}} = 0$ and $\bar{d}_0^C = 0$
for each test-time step t **do**
  Observe traffic state s_t
  Compute base Q-values $\mathbf{Q}_\theta(s_t)$ and base advantages $\mathbf{A}_\theta(s_t)$
  Construct calibration-state context $\chi_t = \text{Concat}\left[\bar{D}_t^{\text{base}}, \bar{d}_t^C, \boldsymbol{\rho}_t\right]$
  Compute candidate residual corrections $\Delta Q_{\phi_t}(s_t, a)$ for all $a \in \mathcal{A}$
  Compute physical score $\psi_t^{\text{phy}}(a) = \sum_{l \in \mathcal{L}(a)} q_t^l$ for all $a \in \mathcal{A}$
  Select action a_t by Calibration-State-Guided Action Routing in Eq. 2
  Execute a_t and observe reward r_t and next state s_{t+1}
  Compute pure base Bellman target $y_t^{\text{base}} = r_t + \gamma \max_{a'} Q_\theta(s_{t+1}, a')$
  Compute base Bellman inconsistency $\delta_t^{\text{base}} = y_t^{\text{base}} - Q_\theta(s_t, a_t)$
  Compute current-step normalized value-relevant drift signal $D_t^{\text{base}} = \frac{|\delta_t^{\text{base}}|}{|y_t^{\text{base}}| + \epsilon}$
  Update recent drift $\bar{D}_{t+1}^{\text{base}} = (1 - \beta_D)\bar{D}_t^{\text{base}} + \beta_D D_t^{\text{base}}$
  Compute mixed Bellman target $y_t^{\text{mix}} = r_t + \gamma \max_{a'} \left[Q_\theta(s_{t+1}, a') + \Delta Q_{\phi_t^-}(s_{t+1}, a')\right]$
  Compute residual calibration error $\delta_t^C = y_t^{\text{mix}} - [Q_\theta(s_t, a_t) + \Delta Q_{\phi_t}(s_t, a_t)]$
  Compute normalized residual calibration error $d_t^C = \frac{|\delta_t^C|}{|y_t^{\text{mix}}| + \epsilon}$
  Update recent residual calibration error $\bar{d}_{t+1}^C = (1 - \beta_C)\bar{d}_t^C + \beta_C d_t^C$
  Compute mixed-target base error $\delta_t^{b|c} = y_t^{\text{mix}} - Q_\theta(s_t, a_t)$
  Compute residual improvement $\iota_t = \frac{(\delta_t^{b|c})^2 - (\delta_t^C)^2}{(\delta_t^{b|c})^2 + (\delta_t^C)^2 + \epsilon}$
  Compute instantaneous credibility signal $c_t = \text{clip}(\iota_t, 0, 1)$
  Update action-wise credibility $\rho_{t+1}(a) = \begin{cases} (1 - \beta_\rho)\rho_t(a) + \beta_\rho c_t, & a = a_t, \\ \rho_t(a), & a \neq a_t \end{cases}$
  Store transition $e_t = (s_t, a_t, r_t, s_{t+1}, D_t^{\text{base}}, d_t^C, t, n_t = 0)$ into $\mathcal{B}_t$
  Remove outdated transitions from $\mathcal{B}_t$ if a sliding-window buffer is used
  if $|\mathcal{B}_t| \geq B$ **then**
    for $g = 1$ to G **do**
      Recompute y_i^{mix}, δ_i^C, and d_i^C using the current residual calibrator
      Compute dynamic replay weight w_i^{replay} by Eq. 3
      Compute sampling probability $p_i = \frac{(w_i^{\text{replay}} + \epsilon_p)^\alpha}{\sum_{j \in \mathcal{B}_t} (w_j^{\text{replay}} + \epsilon_p)^\alpha}$
      Sample mini-batch $\mathcal{M}_g$ from $\mathcal{B}_t$
      Update residual calibrator parameters ϕ by minimizing the weighted TD loss in Eq. 4
      Increase $n_i \leftarrow n_i + 1$ for each $e_i \in \mathcal{M}_g$
    end for
  end if
  Synchronize target residual calibrator $\phi^- \leftarrow \phi$ at a fixed frequency
end for

which are used to construct different drift intensities.

Fig. 10 and Fig. 11 visualize representative traffic-flow profiles before and after valley and peak boost under different augmentation factors.

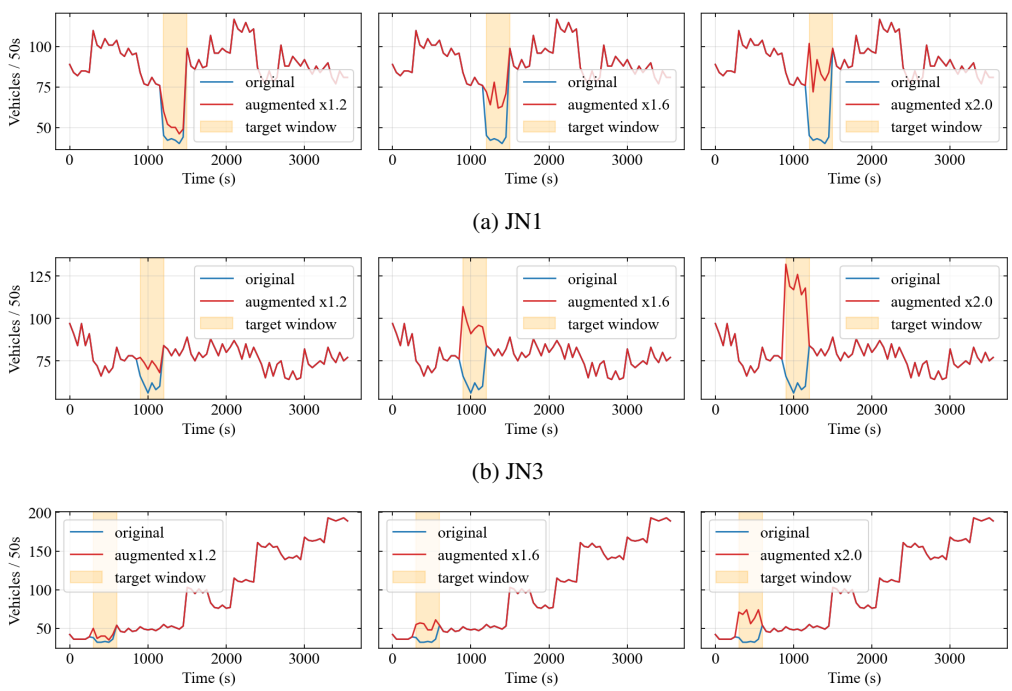
Multi-period drift. To evaluate CRC under repeated traffic changes within the same test process, we further construct multi-period drift scenarios. Instead of modifying only one temporal window, we greedily select two or three non-overlapping extreme 300-second windows from the same traffic flow. Each selected window is augmented independently using the same procedure as valley boost or

Table 3: Summary of main notations used in CRC.

Notation	Description
$\mathcal{S}, \mathcal{A}$	State space and signal-action space.
s_t, a_t, r_t	Traffic state, executed signal action, and received reward at decision step t.
$Q_\theta(s_t, a)$	Action value produced by the pretrained and frozen base controller.
$\mathbf{Q}_\theta(s_t)$	Base action-value vector over all candidate signal actions.
$\mathbf{A}_\theta(s_t)$	Base advantage vector obtained by centering $\mathbf{Q}_\theta(s_t)$.
$\Delta Q_{\phi_t}(s_t, a)$	Candidate residual correction produced by the residual calibrator.
ϕ_t, ϕ_t^-	Parameters of the trainable residual calibrator and target residual calibrator, respectively.
z_t	Intersection statistical feature vector describing the current traffic and signal-service conditions.
χ_t	Calibration-state context, constructed from $\bar{D}_t^{\text{base}}$, $\bar{d}_t^C$, and $\boldsymbol{\rho}_t$.
$g_t(a)$	Action-level gate controlling the activation strength of the residual pathway.
$\omega_t(a)$	Action-wise residual response produced by the residual head.
$\eta_t(a)$	Gated residual response, defined as $g_t(a)\omega_t(a)$.
y_t^{base}	Pure base Bellman target constructed using the frozen base controller.
δ_t^{base}	Base Bellman inconsistency under the current test-time feedback.
D_t^{base}	Normalized current-step ACFD drift signal.
$\bar{D}_t^{\text{base}}$	Moving statistic describing the recent ACFD drift signal.
y_t^{mix}	Mixed Bellman target constructed using the frozen base controller and target residual calibrator.
δ_t^C	Residual calibration error after applying the candidate residual correction.
d_t^C	Normalized current-step residual calibration error.
$\bar{d}_t^C$	Moving statistic describing the recent residual calibration error.
$\delta_t^{b\|c}$	Mixed-target base error used as the reference for evaluating residual improvement.
ι_t	Relative improvement brought by the candidate residual correction.
c_t	Instantaneous credibility signal obtained from ι_t.
$\rho_t(a)$	Action-wise credibility coefficient controlling the released residual correction.
$\boldsymbol{\rho}_t$	Credibility vector over all candidate signal actions.
$\bar{\rho}_t$	Average action-wise credibility used for action routing.
$\rho_t^{\text{exec}}(a)$	Temporarily boosted credibility coefficient used in the release-boost decision mode.
$Q_t^{\text{exec}}(s_t, a)$	Executable action value after credibility-controlled residual release.
$\psi_t^{\text{phy}}(a)$	Physical score of action a based on the queued vehicles in its released lanes.
$\tau_D, \tau_C, \tau_\rho$	Thresholds for ACFD drift, residual calibration error, and average credibility in action routing.
$\mathcal{B}_t$	Test-time replay buffer containing recent interaction transitions.
n_i	Replay usage count of transition e_i.
w_i^{replay}	Dynamic replay weight of transition e_i.
p_i	Sampling probability of transition e_i during prioritized replay.
$\mathcal{M}_g$	Mini-batch sampled for the g-th residual-calibrator update.

peak boost. Specifically, we evaluate one-, two-, and three-window drift settings. For each selected window, the original vehicles are retained, and additional vehicles are generated through sampling with replacement until the number of vehicles reaches approximately α times the original window size. Traffic outside these windows remains unchanged.

Representative examples of the constructed two-window and three-window drift scenarios are shown in Fig. 12 and Fig. 13, respectively.



(a) JN1

(b) JN3

(c) HZ2

Figure 10: Examples of valley-boost traffic-flow construction on different datasets. The blue curves denote the original traffic flows, the red curves denote the augmented traffic flows under $\alpha \in \{1.2, 1.6, 2.0\}$, and the shaded regions indicate the selected target windows.

Newly added intersections. We further construct expanded-network scenarios to evaluate whether CRC can adapt when new intersections appear at test time. The original JN networks contain 12 intersections, for which we construct expanded scenarios containing 16 and 20 intersections. The original HZ networks contain 16 intersections, and the corresponding expanded scenarios contain 20 and 25 intersections. The base controllers are trained using the original networks and remain frozen when evaluated on the expanded networks.

D.2 Evaluation Metrics

We evaluate traffic control performance using Average Travel Time (ATT) and Average Waiting Time (AWT), two widely adopted metrics in existing TSC studies (Zhao et al., 2011; Jiang et al., 2025; Oroojlooy et al., 2020), where lower values indicate better performance.

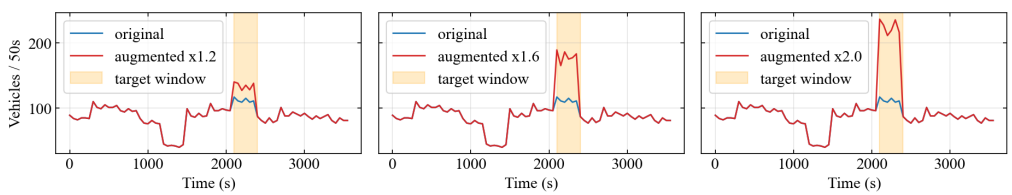
Average Travel Time. Let N denote the number of vehicles that complete their trips during evaluation. For vehicle i, let t_i^{in} and t_i^{out} denote the time when it enters and leaves the road network, respectively. The ATT is defined as

$$\text{ATT} = \frac{1}{N} \sum_{i=1}^{N} \left(t_i^{\text{out}} - t_i^{\text{in}} \right). \tag{33}$$

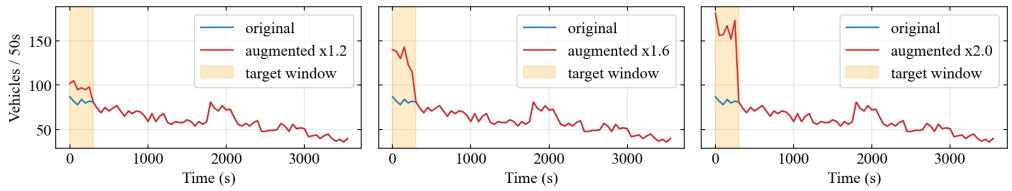
Average Waiting Time. Let w_i denote the accumulated waiting time of vehicle i. The AWT is defined as

$$\text{AWT} = \frac{1}{N} \sum_{i=1}^{N} w_i. \tag{34}$$

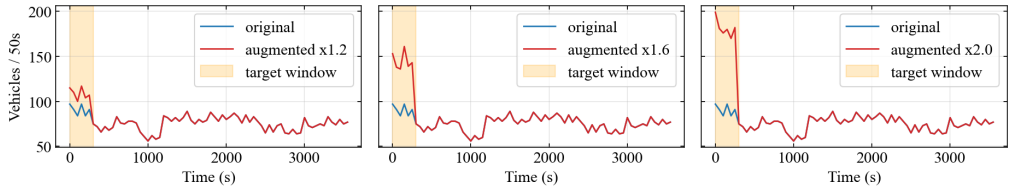
A vehicle is considered to be waiting when its speed falls below the predefined threshold $v_{\text{th}} = 0.1\,\text{m/s}$.



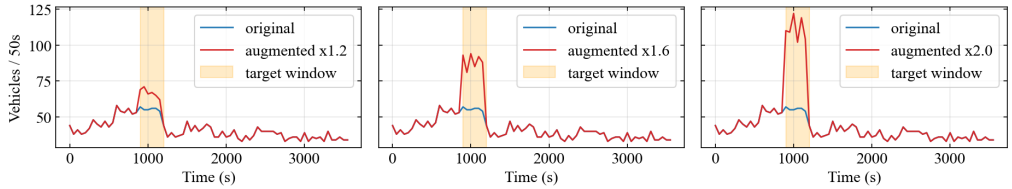
(a) JN1



(b) JN2



(c) JN3



(d) HZ1

Figure 11: Examples of peak-boost traffic-flow construction on different datasets. The blue curves denote the original traffic flows, the red curves denote the augmented traffic flows under $\alpha \in \{1.2, 1.6, 2.0\}$, and the shaded regions indicate the selected target windows.

D.3 Baseline Details

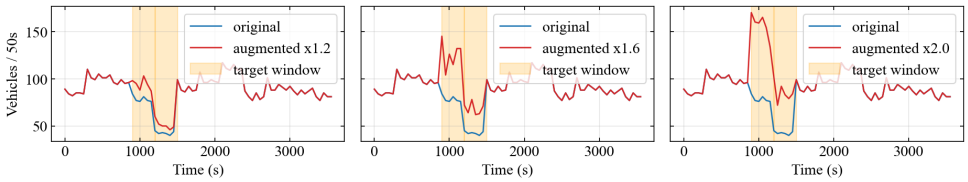
Detailed introductions to the RL-based, meta-RL-based, and LLM-based TSC baselines are provided in Appendix A. All baselines are implemented following the original open-source code and the corresponding experimental settings reported by their authors.

Test-time RL baseline. We construct a test-time RL baseline that directly updates the pretrained RL controller using interaction feedback collected during testing, rather than freezing the base controller and learning a separate residual calibrator. The controller is updated through TD learning on the collected test-time transitions. For a fair comparison, the learning rate, batch size, and number of update iterations are set to the same values as those used for CRC. This baseline compares direct test-time controller adaptation with the residual calibration strategy adopted by CRC.

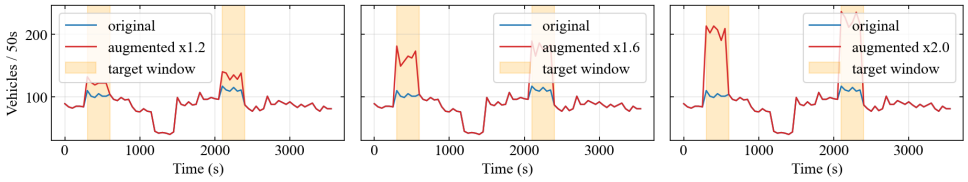
D.4 Hyperparameter Settings

Since each intersection contributes one transition per timestep, a sliding window of 40 timesteps contains up to $40 \times N$ transitions, where N is the number of intersections.

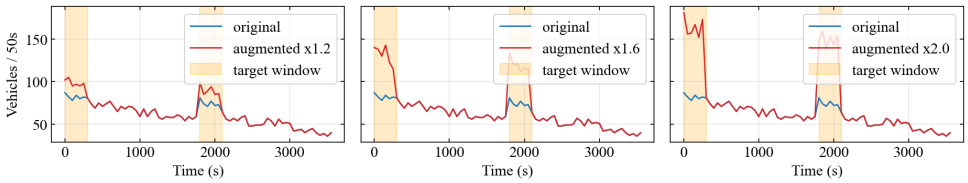
Table 4 summarizes the main hyperparameter settings used in CRC. All CRC hyperparameters are fixed across base controllers and test-time drift scenarios.



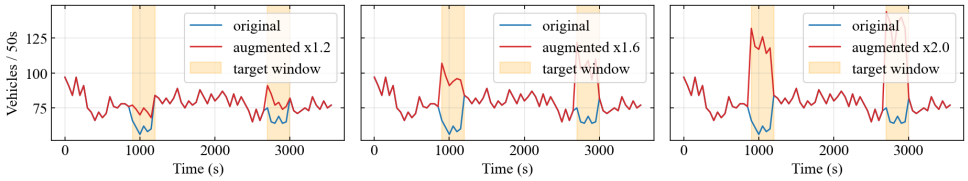
(a) JN1-Valley



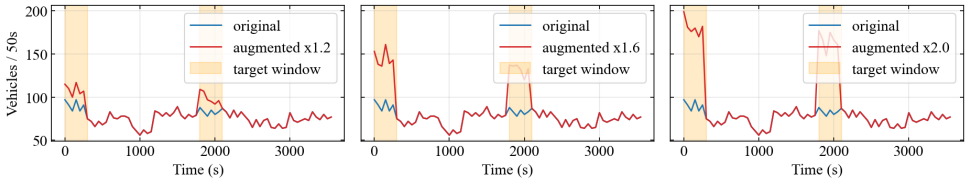
(b) JN1-Peak



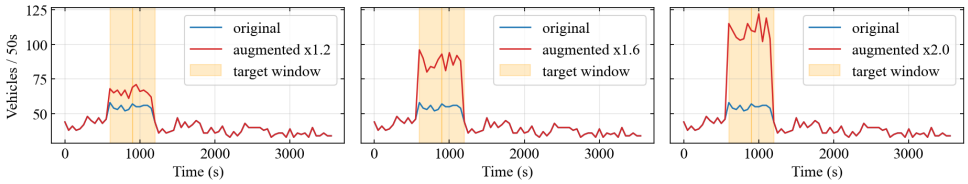
(c) JN2-Peak



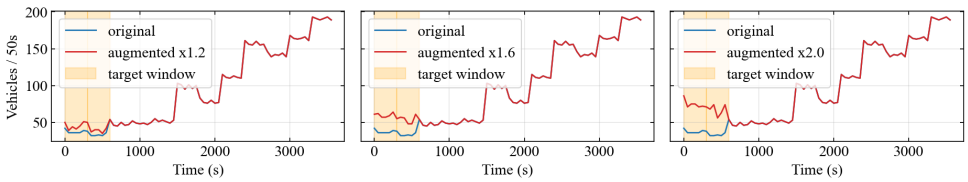
(d) JN3-Valley



(e) JN3-Peak

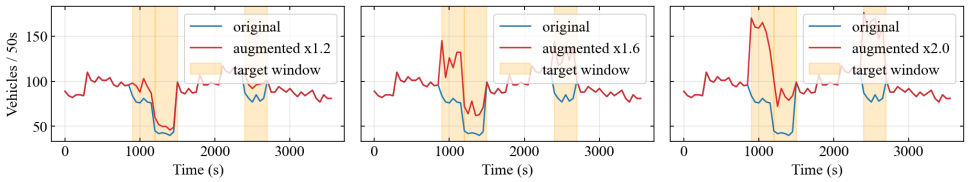


(f) HZ1-Peak

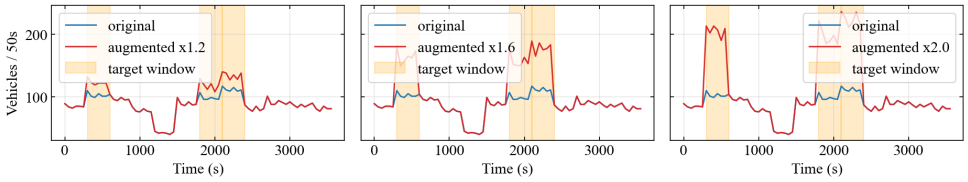


(g) HZ2-Valley

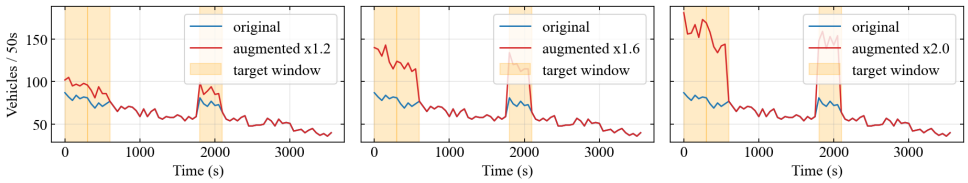
Figure 12: Examples of two-window drift construction on different datasets. The blue curves denote the original traffic flows, the red curves denote the augmented traffic flows under $\alpha \in \{1.2, 1.6, 2.0\}$, and the shaded regions indicate the two selected non-overlapping target windows.



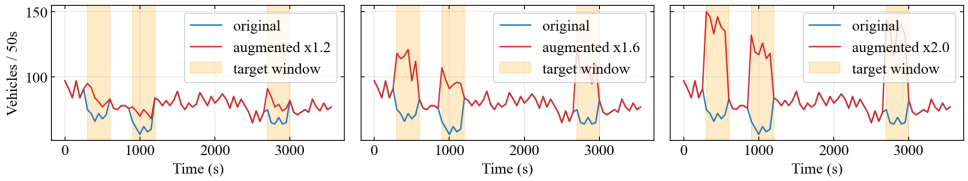
(a) JN1-Valley



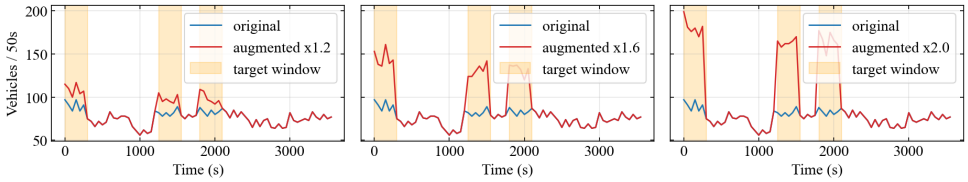
(b) JN1-Peak



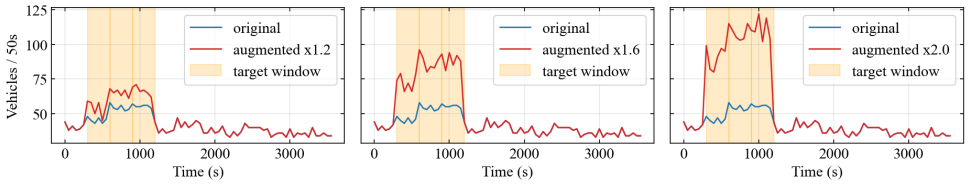
(c) JN2-Peak



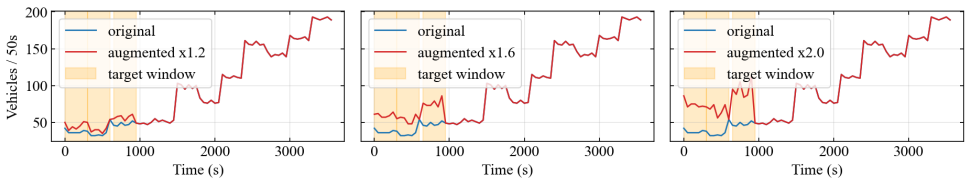
(d) JN3-Valley



(e) JN3-Peak



(f) HZ1-Peak



(g) HZ2-Valley

Figure 13: Examples of three-window drift construction on different datasets. The blue curves denote the original traffic flows, the red curves denote the augmented traffic flows under $\alpha \in \{1.2, 1.6, 2.0\}$, and the shaded regions indicate the three selected non-overlapping target windows.

Table 4: Hyperparameter settings of CRC.

Parameter	Value	Description	Used in
$g_{\min}$	0.5	Lower bound of the action-level gate	Residual Calibrator
d_h	64	Hidden dimension of each MLP layer	Residual Calibrator
L_c	3	Number of Linear-ReLU hidden layers	Residual Calibrator
lr	1×10^{-3}	Adam learning rate	Test-time calibration update
B	64	Mini-batch size for each replay update	Dynamic Replay Update
G	32	Number of replay updates per trigger	Dynamic Replay Update
α	1.0	Prioritization strength for replay sampling	Dynamic Replay Update
β_D	0.1	EMA coefficient for base Bellman inconsistency	Calibration-state statistics
β_C	0.1	EMA coefficient for residual calibration error	Calibration-state statistics
β_ρ	0.1	EMA coefficient for action-wise credibility	Credible Residual Release
$\rho_0(a)$	0.25	Initial credibility for each action	Credible Residual Release
τ_D	0.10	Threshold for base Bellman inconsistency	Action Routing
τ_C	0.40	Threshold for residual calibration error	Action Routing
τ_ρ	0.55	Threshold for average action-wise credibility	Action Routing
T_{sync}	64	Target calibrator synchronization interval	Residual Calibrator
W	40	Sliding-window size	Calibration-state statistics
$\rho_{\min}^{\text{exec}}$	0.9	Minimum release credibility	Action Routing

D.5 Additional Results for RQ1: Effectiveness

This subsection provides more comprehensive results for RQ1. Table 5 reports the fine-grained performance of different base controllers with and without CRC on individual augmented evaluation scenarios, while Table 6 provides the complete comparison with LLM-based, meta-RL-based, and TTRL-based baselines. Tables 7–10 further report detailed results across all individual augmented scenarios.

D.6 Additional Results for RQ2: Adaptation Behavior

This subsection provides additional test-time adaptation statistics of FRAP+CRC across different benchmarks, drift types, and drift intensities, as shown in Figs. 14–34.

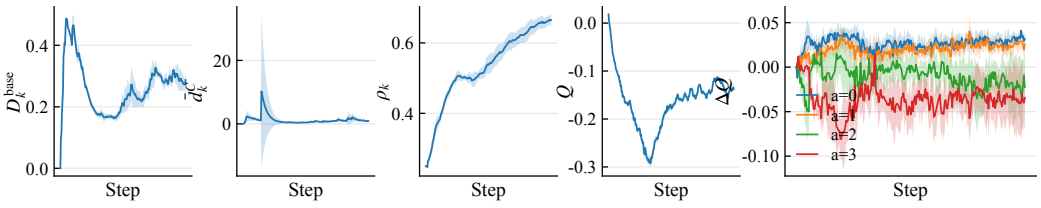


Figure 14: Test-time adaptation behavior of FRAP+CRC on HZ1 under peak boost ($\times 1.2$).

D.7 Additional Results for RQ3: Mechanism Analysis

This subsection provides detailed ablation results for RQ3. Table 11 reports the performance of CRC and its variants without action-wise credibility, dynamic replay, action routing, or the entire calibration module, further examining the contribution of each component. Additionally, Table 12 presents the sensitivity analysis of the threshold hyperparameters in CRC.

Table 5: Detailed performance of different base controllers with and without CRC on individual augmented evaluation scenarios under valley- and peak-boost traffic-demand drift with different drift intensities. This table provides the fine-grained results underlying Table 1. Results are reported as mean ± standard deviation over five independent runs, and Δ denotes the relative reduction achieved by CRC over the corresponding base controller. Lower values are better for both ATT and AWT.

Scenario	Metric	*FRAP*			*CoLight*			*MPLight*			*PressLight*		
w/ CRC		×	✓	Δ(%)	×	✓	Δ(%)	×	✓	Δ(%)	×	✓	Δ(%)
HZ1-Peak ×1.2	ATT	352.78	338.32±0.95	↓ **4.10**	339.44	334.47±0.55	↓ **1.46**	359.21	356.47±1.89	↓ **0.76**	427.36	346.58±1.02	↓ **18.90**
	AWT	57.11	41.21±0.99	↓ **27.84**	43.31	37.91±0.52	↓ **12.47**	62.71	60.76±2.36	↓ **3.11**	134.45	49.48±0.99	↓ **63.20**
HZ1-Peak ×1.6	ATT	362.00	348.03±2.39	↓ **3.86**	346.39	342.49±1.22	↓ **1.13**	369.70	364.05±2.11	↓ **1.53**	481.98	355.33±1.56	↓ **26.28**
	AWT	64.16	47.62±2.19	↓ **25.77**	48.21	43.72±1.25	↓ **9.32**	70.94	65.14±1.86	↓ **8.17**	188.27	54.88±1.43	↓ **70.85**
HZ1-Peak ×2.0	ATT	375.35	365.89±1.68	↓ **2.52**	361.06	356.12±0.76	↓ **1.37**	381.38	378.46±1.49	↓ **0.77**	514.12	368.56±2.87	↓ **28.31**
	AWT	73.15	60.56±1.65	↓ **17.21**	58.78	53.45±0.62	↓ **9.06**	78.98	75.20±2.26	↓ **4.79**	217.95	63.80±3.02	↓ **70.73**
HZ2-Valley ×1.2	ATT	455.58	425.85±1.47	↓ **6.52**	424.40	417.55±1.84	↓ **1.61**	443.21	431.84±4.10	↓ **2.56**	562.37	457.82±6.89	↓ **18.59**
	AWT	165.53	117.46±3.15	↓ **29.04**	113.21	101.43±3.27	↓ **10.41**	137.68	116.03±1.37	↓ **15.73**	307.61	155.47±16.08	↓ **49.46**
HZ2-Valley ×1.6	ATT	453.32	426.27±2.19	↓ **5.97**	422.57	417.17±1.59	↓ **1.28**	443.23	436.49±2.79	↓ **1.52**	569.36	458.67±5.94	↓ **19.44**
	AWT	160.18	118.44±2.66	↓ **26.06**	110.49	101.52±3.20	↓ **8.12**	137.41	123.33±4.17	↓ **10.24**	323.36	154.98±8.11	↓ **52.07**
HZ2-Valley ×2.0	ATT	458.88	427.93±2.51	↓ **6.75**	423.76	418.88±1.59	↓ **1.15**	447.35	438.16±5.61	↓ **2.05**	615.20	468.09±14.55	↓ **23.91**
	AWT	160.99	119.33±6.69	↓ **25.88**	111.04	101.32±1.88	↓ **8.76**	140.01	122.69±5.33	↓ **12.37**	389.88	168.24±21.48	↓ **56.85**
JN1-Peak ×1.2	ATT	351.92	320.09±1.17	↓ **9.04**	316.11	314.82±0.49	↓ **0.41**	340.76	324.61±1.79	↓ **4.74**	350.29	324.86±1.58	↓ **7.26**
	AWT	112.66	78.72±1.19	↓ **30.12**	77.05	73.65±0.34	↓ **4.42**	99.80	81.59±1.64	↓ **18.25**	116.53	81.82±1.56	↓ **29.79**
JN1-Peak ×1.6	ATT	371.73	339.87±1.63	↓ **8.57**	335.81	332.96±0.77	↓ **0.85**	362.28	348.23±0.85	↓ **3.88**	388.98	346.30±0.89	↓ **10.97**
	AWT	131.10	96.05±1.53	↓ **26.73**	94.68	89.47±0.68	↓ **5.50**	119.65	102.75±0.90	↓ **14.13**	161.14	100.74±0.98	↓ **37.48**
JN1-Peak ×2.0	ATT	388.43	355.86±1.21	↓ **8.39**	349.29	347.18±0.60	↓ **0.60**	383.33	367.24±0.94	↓ **4.20**	408.45	366.82±1.49	↓ **10.19**
	AWT	146.75	110.07±1.30	↓ **24.99**	107.32	102.51±0.74	↓ **4.47**	140.31	120.80±0.92	↓ **13.90**	181.59	120.32±1.38	↓ **33.74**
JN1-Valley ×1.2	ATT	346.49	313.72±2.00	↓ **9.46**	310.15	308.12±0.00	↓ **0.66**	335.39	316.10±1.05	↓ **5.75**	322.22	315.44±0.48	↓ **2.10**
	AWT	107.98	73.09±2.11	↓ **32.31**	71.80	67.75±0.00	↓ **5.63**	95.72	74.07±0.82	↓ **22.62**	80.81	73.27±0.46	↓ **9.32**
JN1-Valley ×1.6	ATT	350.05	318.34±1.27	↓ **9.06**	313.48	310.12±0.63	↓ **1.07**	339.76	319.44±0.98	↓ **5.98**	325.76	317.60±0.77	↓ **2.50**
	AWT	110.77	77.14±1.31	↓ **30.36**	74.64	69.45±0.55	↓ **6.95**	98.98	76.60±0.98	↓ **22.61**	84.03	74.77±0.68	↓ **11.02**
JN1-Valley ×2.0	ATT	357.11	323.69±1.02	↓ **9.36**	316.45	314.59±1.26	↓ **0.59**	345.76	328.63±1.50	↓ **4.96**	334.12	326.00±0.81	↓ **2.43**
	AWT	117.10	81.90±1.03	↓ **30.06**	76.95	73.05±1.12	↓ **5.07**	103.41	84.27±1.35	↓ **18.51**	90.88	81.87±0.70	↓ **9.91**
JN2-Peak ×1.2	ATT	317.41	296.02±0.55	↓ **6.74**	297.47	293.79±1.11	↓ **1.24**	314.91	297.99±2.97	↓ **5.37**	301.40	291.88±1.39	↓ **3.16**
	AWT	72.06	50.02±0.61	↓ **30.58**	52.50	48.21±1.07	↓ **8.18**	69.53	51.71±2.78	↓ **25.64**	56.20	45.65±1.46	↓ **18.76**
JN2-Peak ×1.6	ATT	323.88	303.62±2.31	↓ **6.25**	303.28	299.27±0.90	↓ **1.32**	320.23	302.03±1.59	↓ **5.69**	307.91	297.98±1.21	↓ **3.23**
	AWT	77.28	56.34±2.29	↓ **27.09**	57.28	52.45±0.87	↓ **8.43**	73.24	54.31±1.60	↓ **25.85**	61.08	50.27±1.19	↓ **17.71**
JN2-Peak ×2.0	ATT	338.07	318.10±1.32	↓ **5.91**	316.03	312.98±0.73	↓ **0.97**	331.06	316.10±0.99	↓ **4.52**	321.60	309.37±1.11	↓ **3.80**
	AWT	88.63	68.25±1.26	↓ **22.99**	67.71	64.37±0.64	↓ **4.93**	82.12	65.73±1.01	↓ **19.96**	72.08	59.11±0.94	↓ **17.99**
JN3-Peak ×1.2	ATT	324.27	290.84±0.77	↓ **10.31**	293.17	288.43±0.84	↓ **1.62**	317.55	290.14±1.04	↓ **8.63**	298.32	286.85±0.28	↓ **3.85**
	AWT	88.29	53.66±0.70	↓ **39.22**	57.51	51.85±0.80	↓ **9.85**	82.19	52.31±0.98	↓ **36.35**	62.26	49.28±0.27	↓ **20.84**
JN3-Peak ×1.6	ATT	330.89	299.08±2.41	↓ **9.62**	297.44	294.43±1.22	↓ **1.01**	326.11	296.17±1.34	↓ **9.18**	342.49	293.90±0.89	↓ **14.19**
	AWT	93.14	60.73±2.38	↓ **34.80**	60.66	56.61±1.10	↓ **6.67**	89.76	56.94±1.24	↓ **36.56**	101.65	54.63±0.79	↓ **46.25**
JN3-Peak ×2.0	ATT	344.67	318.57±2.39	↓ **7.57**	311.16	306.97±0.58	↓ **1.35**	338.92	311.58±2.28	↓ **8.07**	408.48	310.49±1.33	↓ **23.99**
	AWT	103.01	76.70±2.33	↓ **25.54**	71.24	66.29±0.48	↓ **6.95**	98.49	68.56±2.11	↓ **30.39**	145.23	67.18±1.18	↓ **53.74**
JN3-Valley ×1.2	ATT	324.89	291.90±0.88	↓ **10.16**	293.57	288.90±0.90	↓ **1.59**	320.13	290.07±1.66	↓ **9.39**	301.31	287.38±0.54	↓ **4.62**
	AWT	88.11	54.37±0.74	↓ **38.29**	57.72	52.07±1.02	↓ **9.78**	84.54	52.09±1.43	↓ **38.39**	64.90	49.55±0.50	↓ **23.66**
JN3-Valley ×1.6	ATT	330.98	296.25±1.50	↓ **10.49**	297.07	292.68±0.64	↓ **1.48**	322.55	294.46±1.43	↓ **8.71**	332.86	291.87±0.78	↓ **12.32**
	AWT	93.58	57.83±1.50	↓ **38.21**	60.29	55.15±0.68	↓ **8.52**	85.85	55.44±1.34	↓ **35.42**	95.53	53.02±0.62	↓ **44.49**
JN3-Valley ×2.0	ATT	341.74	308.17±2.50	↓ **9.82**	303.05	300.29±0.63	↓ **0.91**	336.85	304.39±0.79	↓ **9.64**	340.30	302.10±1.01	↓ **11.23**
	AWT	102.22	67.31±2.45	↓ **34.15**	64.86	60.81±0.59	↓ **6.24**	97.56	62.88±0.75	↓ **35.54**	101.72	60.71±0.83	↓ **40.32**

× and ✓ denote the base controller without and with CRC, respectively. **Blue** denotes the best result without CRC, while **magenta** denotes the best result with CRC for each traffic scenario and metric. Δ reports the relative reduction achieved by CRC over the corresponding base controller, where a larger value indicates a greater improvement.

D.8 Additional Results for RQ4: Adaptability

This subsection provides extended results for RQ4 under different drift intensities and numbers of drift periods. Figs. 35–41 report the corresponding performance improvements across HZ1, HZ2, JN1, JN2, and JN3 under peak- and valley-boost scenarios, complementing the representative results presented in the main text.

Table 6: Performance comparison of CRC with LLM-based, meta-RL-based, and TTRL-based TSC methods under test-time drift. For each run, results are first averaged across the augmented evaluation scenarios of the same original dataset, followed by computing mean ± standard deviation over five independent runs. Lower values are better for both ATT and AWT. The best result is shown in **bold**, and the second-best result is underlined.

Category	Method	Metric	HZ1	HZ2	JN1	JN2	JN3
LLM-based Methods	LLMLight	ATT	345.85±0.77	420.42±0.70	332.47±0.84	301.89±0.92	295.94±0.37
		AWT	46.56±0.33	103.38±1.19	88.95±0.73	53.11±0.86	56.50±0.26
	Traffic-R1	ATT	**339.72±1.13**	422.18±0.68	333.55±0.37	**296.71±0.01**	294.64±0.19
		AWT	**42.39±3.84**	100.07±0.32	89.54±0.34	**49.32±0.46**	55.79±0.45
	CoLLMLight	ATT	342.74±1.78	425.16±1.12	329.57±0.03	299.77±0.23	295.79±1.07
		AWT	43.90±1.74	**94.71±0.97**	85.17±0.05	52.67±0.16	57.06±0.86
Meta-RL-based Methods	MetaLight	ATT	391.04±0.01	483.22±0.01	417.05±0.02	377.29±0.01	386.34±0.32
		AWT	89.44±0.01	201.66±0.01	168.63±0.03	123.22±0.00	147.05±0.31
	MetaGAT	ATT	454.79±2.51	454.35±0.00	369.31±0.00	325.32±0.01	368.88±0.16
		AWT	146.83±1.34	151.43±0.00	122.83±0.00	75.75±0.00	126.17±0.16
	MetaVIM	ATT	434.12±0.00	491.19±0.00	471.22±0.00	433.56±0.00	558.39±139.18
		AWT	124.60±0.00	142.79±0.00	230.74±0.00	169.06±0.00	317.22±142.34
TTRL-based Methods	FRAP+TTRL	ATT	366.21±0.53	448.82±1.20	361.86±1.03	326.27±0.81	333.61±0.55
		AWT	68.00±0.30	147.53±2.58	121.86±0.97	79.20±0.86	95.29±0.40
	CoLight+TTRL	ATT	403.06±8.25	520.78±5.09	379.13±6.81	346.01±6.00	352.71±3.76
		AWT	101.55±10.01	218.14±0.88	138.74±3.28	98.83±4.73	117.53±4.29
	MPLight+TTRL	ATT	373.21±1.60	458.65±10.09	357.42±1.31	331.71±2.69	334.51±1.88
		AWT	74.21±1.53	149.64±12.16	115.83±1.20	83.81±2.46	96.70±1.87
	PressLight+TTRL	ATT	467.38±8.89	531.26±6.12	368.71±5.65	322.45±3.86	325.17±3.91
		AWT	146.19±9.44	239.74±3.06	125.85±6.26	74.84±4.06	85.64±3.89
CRC (Ours)	FRAP+CRC	ATT	347.47±3.25	425.10±1.06	328.14±0.04	304.51±0.02	297.09±1.05
		AWT	46.98±2.39	115.83±0.25	85.75±0.22	57.16±0.03	58.76±0.79
	CoLight+CRC	ATT	342.34±0.56	**416.07±1.14**	**321.41±0.45**	299.14±0.23	**293.28±0.71**
		AWT	43.65±0.54	98.62±2.76	**79.39±0.50**	52.25±0.17	55.65±0.73
	MPLight+CRC	ATT	365.72±0.82	437.62±2.39	333.46±0.11	303.95±0.02	297.42±0.30
		AWT	66.55±0.82	124.16±3.48	89.50±0.08	55.87±0.05	57.67±0.33
	PressLight+CRC	ATT	356.12±0.10	457.12±1.74	332.23±0.01	298.97±0.04	295.08±0.07
		AWT	55.28±0.16	152.35±0.91	88.16±0.20	50.90±0.08	**55.43±0.05**

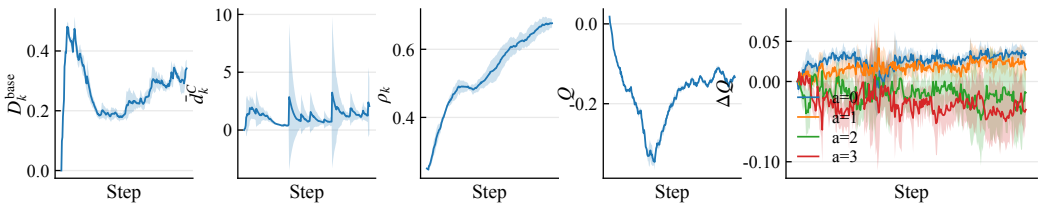


Figure 15: Test-time adaptation behavior of FRAP+CRC on HZ1 under peak boost (×1.6).

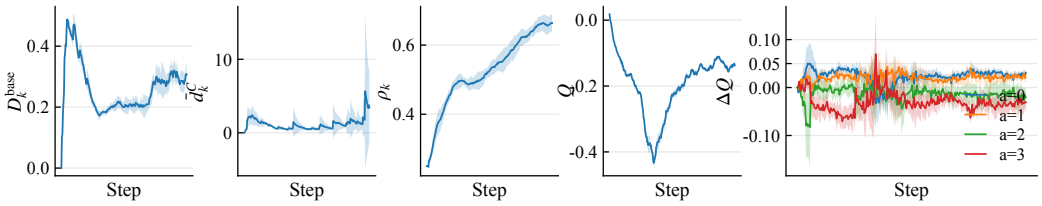


Figure 16: Test-time adaptation behavior of FRAP+CRC on HZ1 under peak boost (×2.0).

Table 7: Detailed performance comparison on individual valley- and peak-boost evaluation scenarios with different drift intensities (Part 1/4). Results are reported as mean ± standard deviation over five runs. This table provides the fine-grained results underlying Table 6. Lower values are better for both ATT and AWT. The best result is shown in **bold**, and the second-best result is underlined.

Category	Method	Metric	HZ1-Peak ×1.2	HZ1-Peak ×1.6	HZ1-Peak ×2.0	HZ2-Valley ×1.2	HZ2-Valley ×1.6
LLM-based Methods	LLMLight	ATT ↓	344.00±0.50	338.56±0.06	354.99±2.86	420.89±0.00	419.96±0.34
		AWT ↓	47.51±0.81	**40.57±1.06**	51.59±0.75	102.46±0.63	103.71±1.72
	Traffic-R1	ATT ↓	**333.82±1.08**	**337.53±2.05**	**347.79±0.25**	422.29±0.11	421.72±0.61
		AWT ↓	**35.93±2.71**	40.78±3.51	**50.47±5.30**	97.69±0.54	99.94±0.02
	CoLLMLight	ATT ↓	333.92±1.73	340.89±2.38	353.41±1.41	424.22±0.79	423.62±0.47
		AWT ↓	38.33±1.85	42.43±1.70	50.93±2.18	**93.35±0.41**	**94.18±1.96**
Meta-RL-based Methods	MetaLight	ATT ↓	372.89±0.00	386.53±0.00	413.70±0.04	480.84±0.02	486.76±0.00
		AWT ↓	75.51±0.00	85.56±0.00	107.25±0.03	198.84±0.03	206.41±0.00
	MetaGAT	ATT ↓	346.85±0.00	363.46±0.00	654.05±7.53	446.39±0.00	450.66±0.00
		AWT ↓	49.33±0.00	61.56±0.00	329.59±4.02	145.76±0.00	146.95±0.00
	MetaVIM	ATT ↓	392.78±0.00	406.58±0.00	503.00±0.00	473.17±0.00	475.47±0.00
		AWT ↓	91.03±0.00	101.55±0.00	181.21±0.00	137.88±0.00	131.36±0.00
TTRL-based Methods	FRAP+TTRL	ATT ↓	356.21±0.27	363.35±1.56	379.07±0.51	448.42±3.96	448.24±1.18
		AWT ↓	60.98±0.38	65.54±1.05	77.49±0.77	145.99±5.13	147.06±2.56
	CoLight+TTRL	ATT ↓	393.39±25.74	393.60±3.54	422.18±6.45	573.51±1.85	490.16±9.85
		AWT ↓	98.95±30.04	92.77±3.28	112.94±3.98	349.10±3.08	150.03±3.58
	MPLight+TTRL	ATT ↓	362.88±1.91	367.22±2.11	389.54±3.35	448.19±3.67	455.65±9.69
		AWT ↓	67.35±1.72	68.66±1.97	86.63±3.38	142.57±1.49	144.92±11.14
	PressLight+TTRL	ATT ↓	442.82±19.18	449.16±6.75	510.17±16.68	513.28±17.53	532.63±6.89
		AWT ↓	134.85±16.31	136.68±5.51	167.04±18.41	223.37±25.37	244.07±4.28
CRC (Ours)	FRAP+CRC	ATT ↓	338.32±0.95	348.03±2.39	365.89±1.68	425.85±1.47	426.27±2.19
		AWT ↓	41.21±0.99	47.62±2.19	60.56±1.65	117.46±3.15	118.44±2.66
	CoLight+CRC	ATT ↓	334.47±0.55	342.49±1.22	356.12±0.76	**417.55±1.84**	**417.17±1.59**
		AWT ↓	37.91±0.52	43.72±1.25	53.45±0.62	101.43±3.27	101.52±3.20
	MPLight+CRC	ATT ↓	356.47±1.89	364.05±2.11	378.46±1.49	431.84±4.10	436.49±2.79
		AWT ↓	60.76±2.36	65.14±1.86	75.20±2.26	116.03±1.37	123.33±4.17
	PressLight+CRC	ATT ↓	346.58±1.02	355.33±1.56	368.56±2.87	457.82±6.89	458.67±5.94
		AWT ↓	49.48±0.99	54.88±1.43	63.80±3.02	155.47±16.08	154.98±8.11

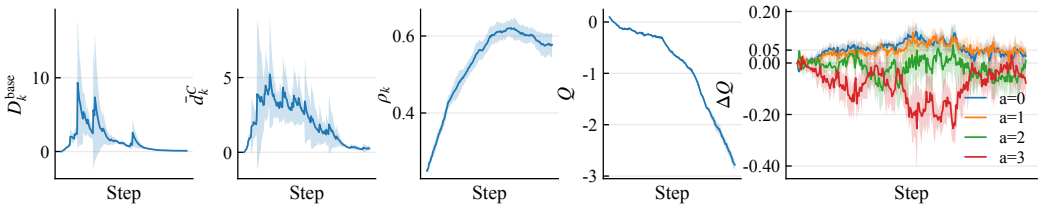


Figure 17: Test-time adaptation behavior of FRAP+CRC on HZ2 under valley boost (×1.2).

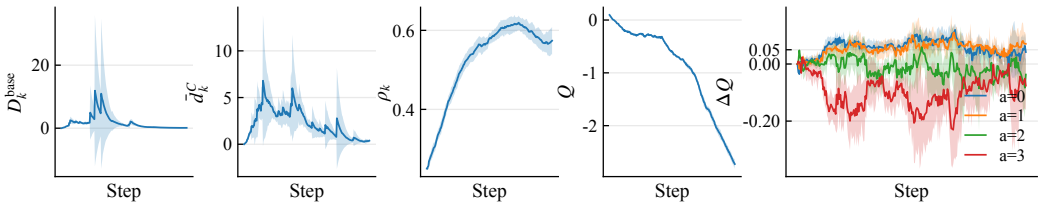


Figure 18: Test-time adaptation behavior of FRAP+CRC on HZ2 under valley boost (×1.6).

Table 8: Detailed performance comparison on individual valley- and peak-boost evaluation scenarios with different drift intensities (Part 2/4). Results are reported as mean ± standard deviation over five runs. This table provides the fine-grained results underlying Table 6. Lower values are better for both ATT and AWT. The best result is shown in **bold**, and the second-best result is underlined.

Category	Method	Metric	HZ2-Valley ×2.0	JN1-Peak ×1.2	JN1-Peak ×1.6	JN1-Peak ×2.0	JN1-Valley ×1.2
LLM-based Methods	LLMLight	ATT ↓	420.42±1.77	323.86±1.03	347.40±0.62	364.11±0.55	315.69±0.20
		AWT ↓	103.97±2.49	81.27±1.11	102.27±0.10	118.41±0.54	73.75±0.49
	Traffic-R1	ATT ↓	422.52±1.34	324.07±1.11	347.03±1.64	367.27±2.30	315.77±0.91
		AWT ↓	102.57±0.43	80.86±1.08	101.51±1.53	120.98±2.15	73.67±1.11
	CoLLMLight	ATT ↓	427.64±2.42	321.71±0.68	342.90±1.47	360.73±0.63	314.87±1.24
		AWT ↓	**96.60±0.64**	78.56±0.34	96.94±1.15	112.09±0.48	72.84±0.88
Meta-RL-based Methods	MetaLight	ATT ↓	482.06±0.00	409.26±0.00	423.63±0.00	433.11±0.00	399.12±0.00
		AWT ↓	199.73±0.00	160.60±0.00	169.57±0.00	177.87±0.00	157.33±0.00
	MetaGAT	ATT ↓	465.99±0.00	354.20±0.00	378.68±0.00	419.11±0.00	344.45±0.00
		AWT ↓	161.59±0.00	110.73±0.00	129.18±0.00	162.54±0.00	102.74±0.00
	MetaVIM	ATT ↓	524.93±0.00	454.84±0.00	479.17±0.00	493.77±0.00	452.30±0.00
		AWT ↓	159.12±0.00	214.08±0.00	239.93±0.00	255.84±0.00	211.61±0.00
TTRL-based Methods	FRAP+TTRL	ATT ↓	449.79±1.53	353.20±1.94	374.31±2.91	388.87±0.87	346.71±1.08
		AWT ↓	149.53±3.57	113.89±1.96	133.27±2.82	147.18±1.17	108.24±1.29
	CoLight+TTRL	ATT ↓	498.68±9.70	361.08±8.58	385.10±15.00	403.34±20.05	370.65±5.84
		AWT ↓	155.30±3.96	129.56±2.50	142.08±11.46	159.09±8.94	129.50±4.59
	MPLight+TTRL	ATT ↓	472.12±27.98	347.79±2.17	366.93±0.89	390.18±5.52	343.91±1.27
		AWT ↓	161.44±29.23	106.58±1.68	124.99±1.40	147.37±4.35	103.32±1.80
	PressLight+TTRL	ATT ↓	547.88±34.24	364.19±8.43	390.21±9.05	406.22±3.60	345.03±15.96
		AWT ↓	251.78±37.70	122.65±5.79	146.89±7.92	158.70±8.24	103.47±14.85
CRC (Ours)	FRAP+CRC	ATT ↓	427.93±2.51	320.09±1.17	339.87±1.63	355.86±1.21	313.72±2.00
		AWT ↓	119.33±6.69	78.72±1.19	96.05±1.53	110.07±1.30	73.09±2.11
	CoLight+CRC	ATT ↓	**418.88±1.59**	**314.82±0.49**	**332.96±0.77**	**347.18±0.60**	**308.12±0.00**
		AWT ↓	101.32±1.88	**73.65±0.34**	**89.47±0.68**	**102.51±0.74**	**67.75±0.00**
	MPLight+CRC	ATT ↓	438.16±5.61	324.61±1.79	348.23±0.85	367.24±0.94	316.10±1.05
		AWT ↓	122.69±5.33	81.59±1.64	102.75±0.90	120.80±0.92	74.07±0.82
	PressLight+CRC	ATT ↓	468.09±14.55	324.86±1.58	346.30±0.89	366.82±1.49	315.44±0.48
		AWT ↓	168.24±21.48	81.82±1.56	100.74±0.98	120.32±1.38	73.27±0.46

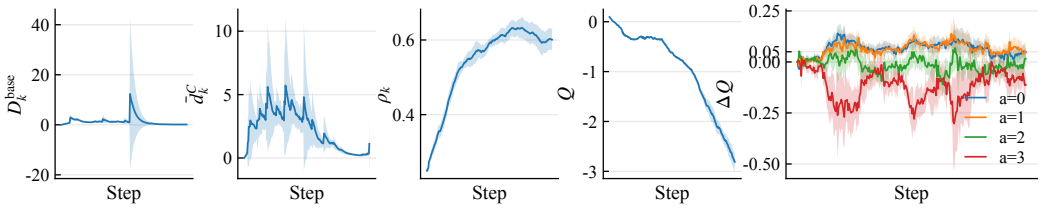


Figure 19: Test-time adaptation behavior of FRAP+CRC on HZ2 under valley boost (×2.0).

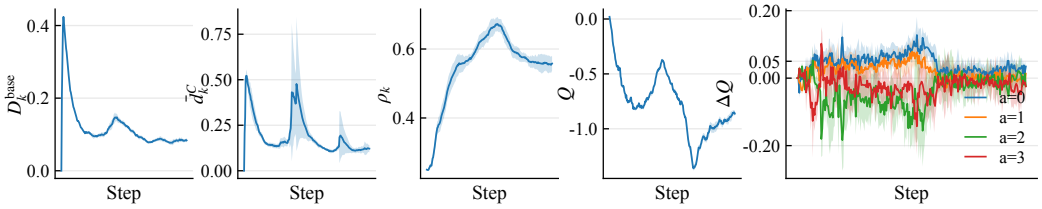


Figure 20: Test-time adaptation behavior of FRAP+CRC on JN1 under peak boost (×1.2).

Table 9: Detailed performance comparison on individual valley- and peak-boost evaluation scenarios with different drift intensities (Part 3/4). Results are reported as mean ± standard deviation over five runs. This table provides the fine-grained results underlying Table 6. Lower values are better for both ATT and AWT. The best result is shown in **bold**, and the second-best result is underlined.

Category	Method	Metric	JN1-Valley ×1.6	JN1-Valley ×2.0	JN2-Peak ×1.2	JN2-Peak ×1.6	JN2-Peak ×2.0
LLM-based Methods	LLMLight	ATT ↓	318.59±2.83	325.19±0.90	295.01±0.13	299.15±0.79	311.53±1.85
		AWT ↓	76.45±2.61	81.56±0.61	47.05±1.44	52.72±0.74	59.57±0.40
	Traffic-R1	ATT ↓	319.92±2.10	327.23±0.31	**288.18±0.63**	**295.28±0.96**	**306.68±0.34**
		AWT ↓	77.12±1.82	83.12±0.46	**44.04±0.86**	**49.91±1.32**	**54.02±1.84**
	CoLLMLight	ATT ↓	314.83±0.29	322.37±0.22	292.27±0.06	298.02±0.23	309.03±0.53
		AWT ↓	72.42±0.39	78.19±0.02	46.87±0.04	51.29±0.18	59.84±0.26
Meta-RL-based Methods	MetaLight	ATT ↓	413.03±0.09	424.15±0.11	352.42±0.03	375.20±0.02	404.26±0.03
		AWT ↓	168.82±0.00	177.57±0.19	102.53±0.00	121.54±0.00	145.58±0.00
	MetaGAT	ATT ↓	350.43±0.00	368.97±0.00	308.67±0.01	320.40±0.01	346.88±0.02
		AWT ↓	107.65±0.00	124.15±0.00	61.33±0.00	71.04±0.01	94.86±0.00
	MetaVIM	ATT ↓	463.14±0.00	484.09±0.00	378.92±0.00	428.02±0.00	493.76±0.00
		AWT ↓	221.44±0.00	241.56±0.00	120.82±0.00	163.60±0.00	222.77±0.00
TTRL-based Methods	FRAP+TTRL	ATT ↓	349.38±1.86	358.67±1.57	316.21±1.16	323.52±0.13	339.07±1.41
		AWT ↓	110.25±1.92	118.34±1.30	70.99±1.10	76.80±0.12	89.79±1.58
	CoLight+TTRL	ATT ↓	373.07±9.13	381.56±4.00	319.86±11.40	351.24±4.63	366.92±2.81
		AWT ↓	133.00±9.03	139.20±2.08	77.84±8.22	102.94±4.50	115.70±2.88
	MPLight+TTRL	ATT ↓	346.22±3.98	349.51±4.10	321.74±4.00	332.51±2.12	340.88±2.29
		AWT ↓	104.98±3.50	107.77±3.83	76.21±3.60	84.52±1.63	90.70±2.42
	PressLight+TTRL	ATT ↓	349.33±6.92	357.29±14.39	307.53±4.35	317.33±1.40	342.48±11.01
		AWT ↓	110.15±4.41	113.22±14.19	62.07±4.33	70.29±1.88	92.15±10.19
CRC (Ours)	FRAP+CRC	ATT ↓	318.34±1.27	323.69±1.02	296.02±0.55	303.62±2.31	318.10±1.32
		AWT ↓	77.14±1.31	81.90±1.03	50.02±0.61	56.34±2.29	68.25±1.26
	CoLight+CRC	ATT ↓	**310.12±0.63**	**314.59±1.26**	293.79±1.11	299.27±0.90	312.98±0.73
		AWT ↓	**69.45±0.55**	**73.05±1.12**	48.21±1.07	52.45±0.87	64.37±0.64
	MPLight+CRC	ATT ↓	319.44±0.98	328.63±1.50	297.99±2.97	302.03±1.59	316.10±0.99
		AWT ↓	76.60±0.98	84.27±1.35	51.71±2.78	54.31±1.60	65.73±1.01
	PressLight+CRC	ATT ↓	317.60±0.77	326.00±0.81	291.88±1.39	297.98±1.21	309.37±1.11
		AWT ↓	74.77±0.68	81.87±0.70	45.65±1.46	50.27±1.19	59.11±0.94

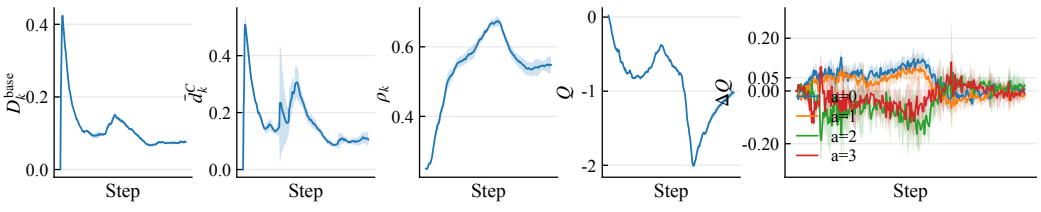


Figure 21: Test-time adaptation behavior of FRAP+CRC on JN1 under peak boost (×1.6).

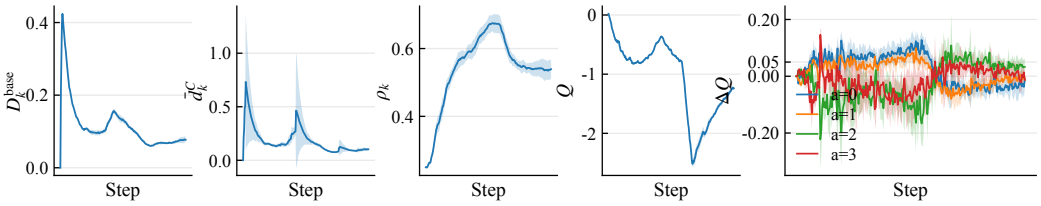


Figure 22: Test-time adaptation behavior of FRAP+CRC on JN1 under peak boost (×2.0).

Table 10: Detailed performance comparison on individual valley- and peak-boost evaluation scenarios with different drift intensities (Part 4/4). Results are reported as mean ± standard deviation over five runs. This table provides the fine-grained results underlying Table 6. Lower values are better for both ATT and AWT. The best result is shown in **bold**, and the second-best result is underlined.

Category	Method	Metric	JN3-Peak ×1.2	JN3-Peak ×1.6	JN3-Peak ×2.0	JN3-Valley ×1.2	JN3-Valley ×1.6	JN3-Valley ×2.0
LLM-based Methods	LLMLight	ATT ↓	288.06±0.21	294.39±0.41	309.81±0.69	288.78±0.35	292.15±0.03	302.43±1.25
		AWT ↓	50.56±0.18	55.36±0.24	67.12±0.57	50.98±0.21	53.57±0.14	61.40±0.92
	Traffic-R1	ATT ↓	**285.59±0.20**	**292.74±0.91**	309.80±0.08	**286.53±0.15**	**291.07±0.06**	302.10±0.12
		AWT ↓	**46.08±4.41**	55.83±0.58	66.49±0.16	**49.18±0.46**	55.34±1.44	61.81±0.23
	CoLLMLight	ATT ↓	289.46±0.18	293.86±1.37	307.17±0.54	289.53±1.21	292.82±1.74	301.87±1.40
		AWT ↓	52.55±0.00	55.73±1.38	**65.32±0.48**	52.55±0.97	54.82±1.35	61.41±0.98
Meta-RL-based Methods	MetaLight	ATT ↓	372.52±1.90	382.51±0.00	400.33±0.00	376.56±0.00	384.58±0.00	401.55±0.01
		AWT ↓	135.51±1.87	143.79±0.00	157.19±0.00	139.05±0.00	146.23±0.00	160.51±0.04
	MetaGAT	ATT ↓	318.38±0.94	363.61±0.00	534.89±0.00	320.43±0.00	326.62±0.00	349.35±0.00
		AWT ↓	81.09±0.95	122.88±0.00	274.57±0.00	82.35±0.00	87.76±0.00	108.38±0.00
	MetaVIM	ATT ↓	526.54±138.61	580.93±142.60	593.34±130.22	502.03±120.81	546.01±139.99	601.50±162.88
		AWT ↓	297.60±145.61	325.72±138.96	342.73±132.77	255.34±117.96	324.22±152.96	357.73±165.79
TTRL-based Methods	FRAP+TTRL	ATT ↓	323.37±0.95	328.45±1.16	349.74±0.85	325.59±1.35	329.48±0.63	345.03±0.65
		AWT ↓	87.13±0.66	90.84±0.91	108.05±0.53	88.94±1.14	91.74±0.83	105.06±0.60
	CoLight+TTRL	ATT ↓	332.11±1.17	341.54±4.47	362.41±6.73	382.76±21.46	340.66±4.66	356.79±10.14
		AWT ↓	97.09±2.96	103.05±5.12	119.56±6.57	157.59±21.54	103.24±4.37	124.66±12.74
	MPLight+TTRL	ATT ↓	327.94±12.96	329.30±0.79	342.41±0.38	329.50±1.28	330.86±3.75	347.07±10.44
		AWT ↓	92.23±13.34	92.33±0.68	101.43±0.43	93.50±1.66	93.31±3.64	107.36±10.42
	PressLight+TTRL	ATT ↓	304.57±4.32	313.15±6.38	356.35±8.50	303.55±1.13	324.87±28.97	348.56±24.89
		AWT ↓	67.89±4.23	75.68±6.11	113.06±7.59	66.93±0.77	84.39±24.55	105.90±23.95
CRC (Ours)	FRAP+CRC	ATT ↓	290.84±0.77	299.08±2.41	318.57±2.39	291.90±0.88	296.25±1.50	308.17±2.50
		AWT ↓	53.66±0.70	60.73±2.38	76.70±2.33	54.37±0.74	57.83±1.50	67.31±2.45
	CoLight+CRC	ATT ↓	288.43±0.84	294.43±1.22	**306.97±0.58**	288.90±0.90	292.68±0.64	**300.29±0.63**
		AWT ↓	51.85±0.80	56.61±1.10	66.29±0.48	52.07±1.02	55.15±0.68	60.81±0.59
	MPLight+CRC	ATT ↓	290.14±1.04	296.17±1.34	311.58±2.28	290.07±1.66	294.46±1.43	304.39±0.79
		AWT ↓	52.31±0.98	56.94±1.24	68.56±2.11	52.09±1.43	55.44±1.34	62.88±0.75
	PressLight+CRC	ATT ↓	286.85±0.28	293.90±0.89	310.49±1.33	287.38±0.54	291.87±0.78	302.10±1.01
		AWT ↓	49.28±0.27	**54.63±0.79**	67.18±1.18	49.55±0.50	**53.02±0.62**	**60.71±0.83**

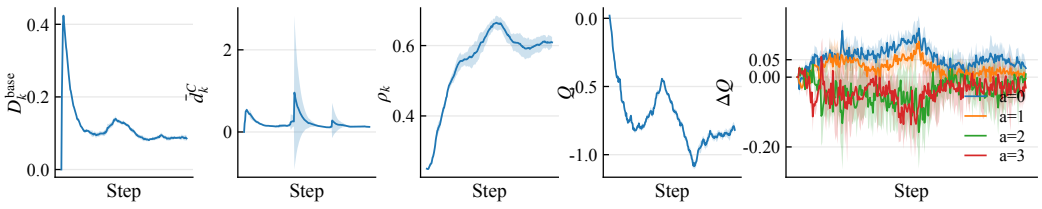


Figure 23: Test-time adaptation behavior of FRAP+CRC on JN1 under valley boost (×1.2).

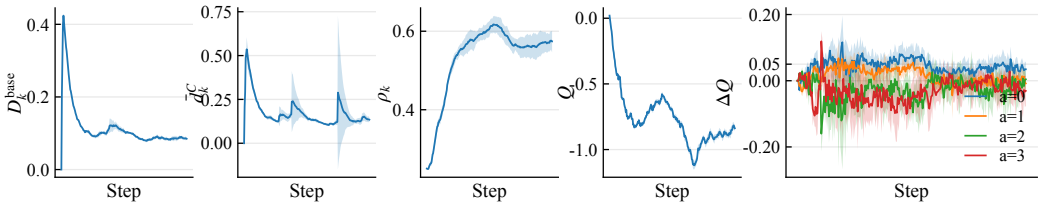


Figure 24: Test-time adaptation behavior of FRAP+CRC on JN1 under valley boost (×1.6).

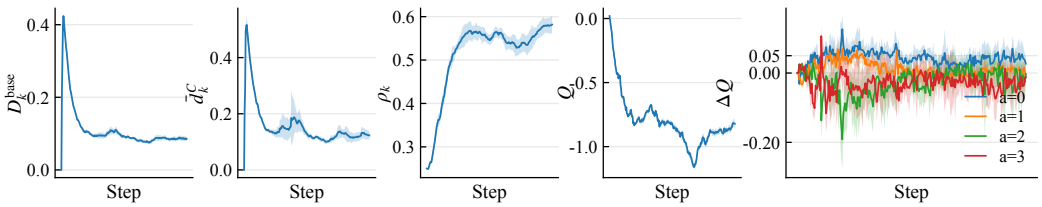


Figure 25: Test-time adaptation behavior of FRAP+CRC on JN1 under valley boost ($\times 2.0$).

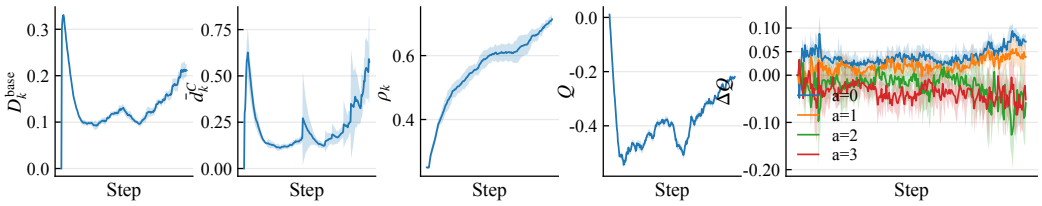


Figure 26: Test-time adaptation behavior of FRAP+CRC on JN2 under peak boost ($\times 1.2$).

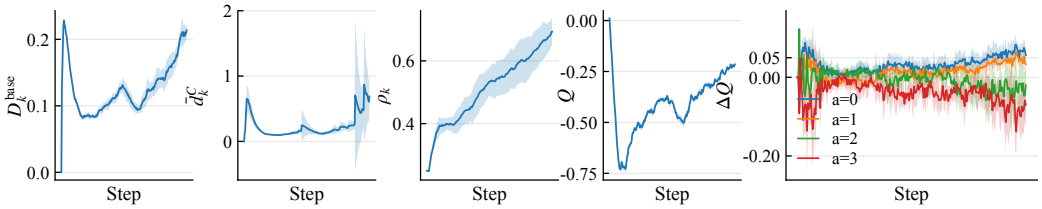


Figure 27: Test-time adaptation behavior of FRAP+CRC on JN2 under peak boost ($\times 1.6$).

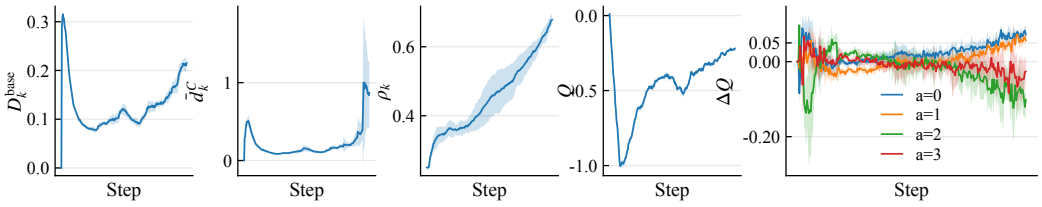


Figure 28: Test-time adaptation behavior of FRAP+CRC on JN2 under peak boost ($\times 2.0$).

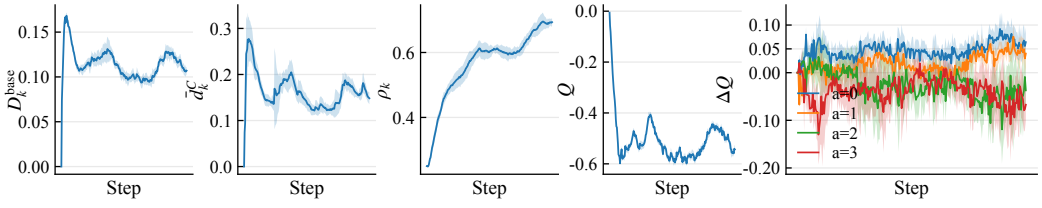


Figure 29: Test-time adaptation behavior of FRAP+CRC on JN3 under peak boost ($\times 1.2$).

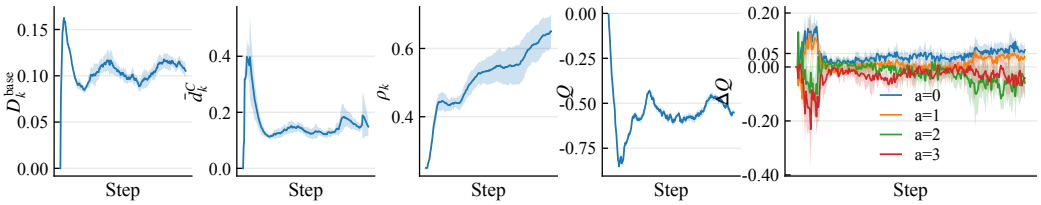


Figure 30: Test-time adaptation behavior of FRAP+CRC on JN3 under peak boost ($\times 1.6$).

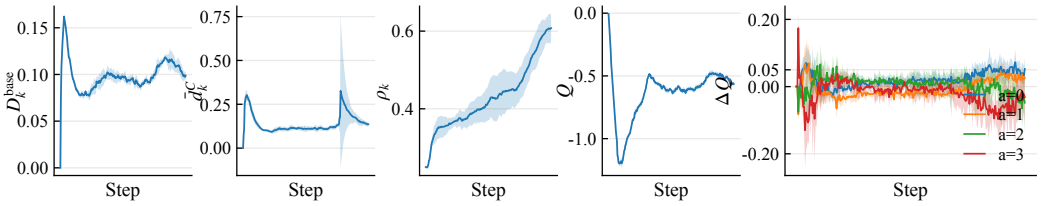


Figure 31: Test-time adaptation behavior of FRAP+CRC on JN3 under peak boost ($\times 2.0$).

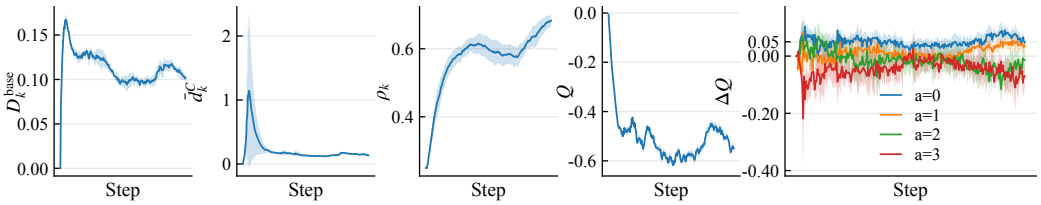


Figure 32: Test-time adaptation behavior of FRAP+CRC on JN3 under valley boost ($\times 1.2$).

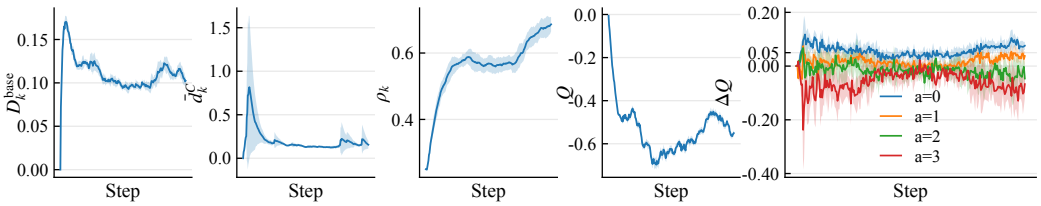


Figure 33: Test-time adaptation behavior of FRAP+CRC on JN3 under valley boost ($\times 1.6$).

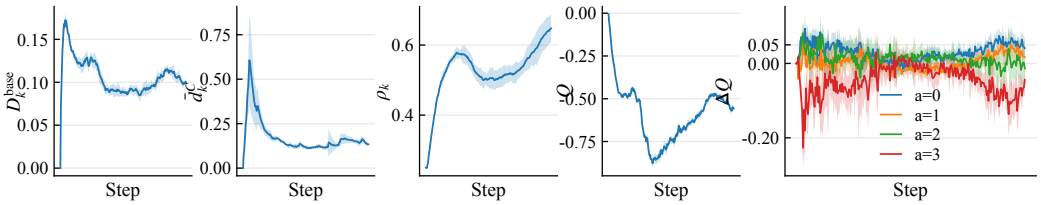


Figure 34: Test-time adaptation behavior of FRAP+CRC on JN3 under valley boost ($\times 2.0$).

Table 11: Detailed ablation results of CRC on HZ1 and HZ2. For each benchmark, results are averaged across the corresponding valley- and peak-boost scenarios with different drift intensities. The compared variants respectively remove action-wise credibility (*w/o Cred*), dynamic replay (*w/o Dyn*), action routing (*w/o Route*), or the entire CRC module (*Base*). Lower values are better for both ATT and AWT. The best result is shown in **bold**, and the second-best result is underlined.

Base Controller	Variant	HZ1		HZ2	
		ATT ↓	AWT ↓	ATT ↓	AWT ↓
FRAP	Base	363.38	64.81	455.93	162.23
	w/o Route	368.51±1.97	67.63±1.44	448.73±2.72	154.69±3.82
	w/o Dyn	358.76±1.88	59.04±2.15	445.76±24.12	145.43±25.65
	w/o Cred	359.54±0.65	59.60±0.56	439.28±0.29	141.02±1.52
	CRC (Ours)	**347.47±3.25**	**46.98±2.39**	**425.10±1.06**	**115.83±0.25**
CoLight	Base	348.96	50.10	423.57	111.58
	w/o Route	348.76±0.85	50.10±0.69	422.06±0.14	109.11±0.52
	w/o Dyn	358.61±18.39	56.77±13.80	450.90±56.58	128.11±41.33
	w/o Cred	349.45±0.73	50.25±0.63	422.15±1.67	109.36±2.14
	CRC (Ours)	**342.34±0.56**	**43.65±0.54**	**416.07±1.14**	**98.62±2.76**

Table 12: Sensitivity analysis of the threshold hyperparameters in CRC. Each hyperparameter is varied while the others are fixed. Lower values are better for both ATT and AWT. The best result is shown in **bold**, and the second-best result is underlined.

Parameter	**Value**	**ATT ↓**	**AWT ↓**
τ_D	0.05	**423.11±0.55**	**115.24±0.61**
	0.10	427.51±1.04	119.67±1.89
	0.15	429.59±1.72	121.26±3.05
	0.20	429.58±3.10	122.82±2.19
	0.30	436.21±2.08	130.71±0.38
τ_C	0.10	**415.72±0.62**	**104.18±1.40**
	0.15	420.64±6.51	104.99±0.96
	0.20	423.75±5.51	112.11±7.70
	0.24	420.01±0.44	108.40±1.86
	0.40	427.06±1.64	119.12±2.10
τ_ρ	0.35	430.03±2.54	121.61±3.75
	0.45	428.31±3.94	119.44±4.22
	0.55	427.73±3.41	118.06±2.40
	0.65	426.27±1.51	119.05±3.02
	0.75	**424.58±1.60**	**114.55±2.41**

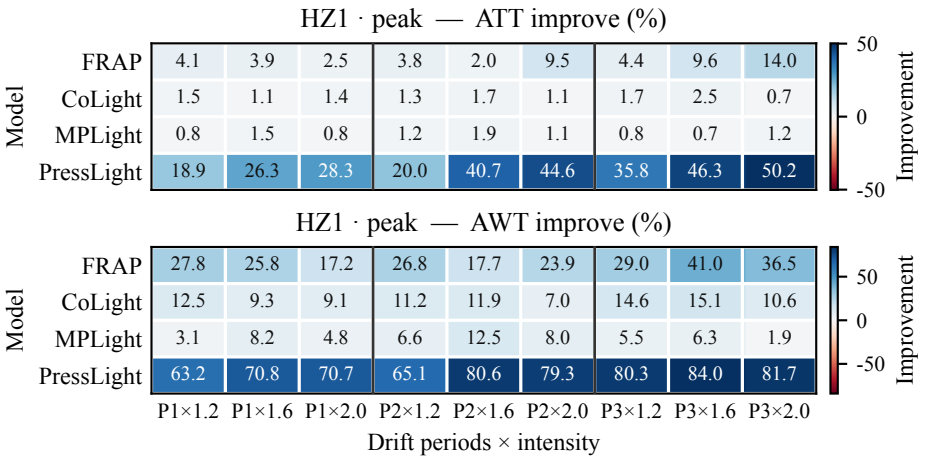


Figure 35: Relative performance improvement of CRC under different drift intensities and numbers of drift periods on the HZ1 peak-boost scenario. Higher values indicate larger reductions in ATT or AWT.

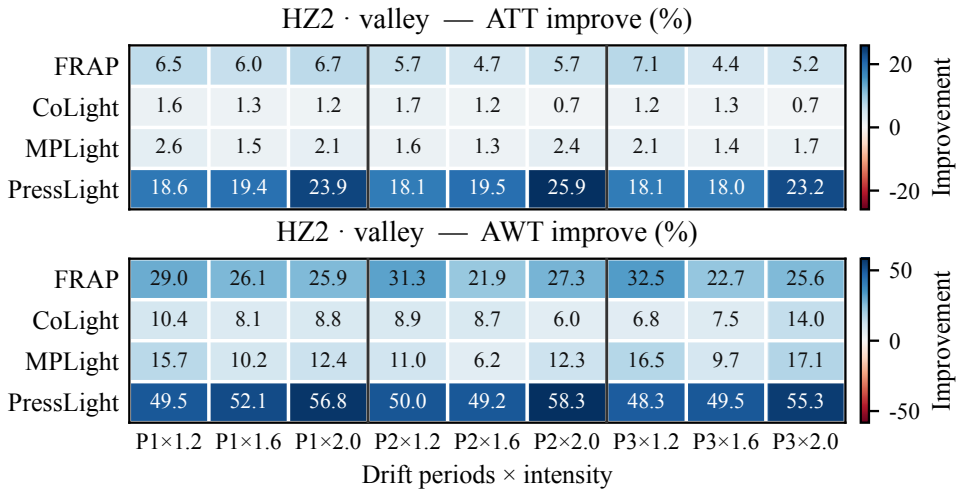


Figure 36: Relative performance improvement of CRC under different drift intensities and numbers of drift periods on the HZ2 valley-boost scenario. Higher values indicate larger reductions in ATT or AWT.

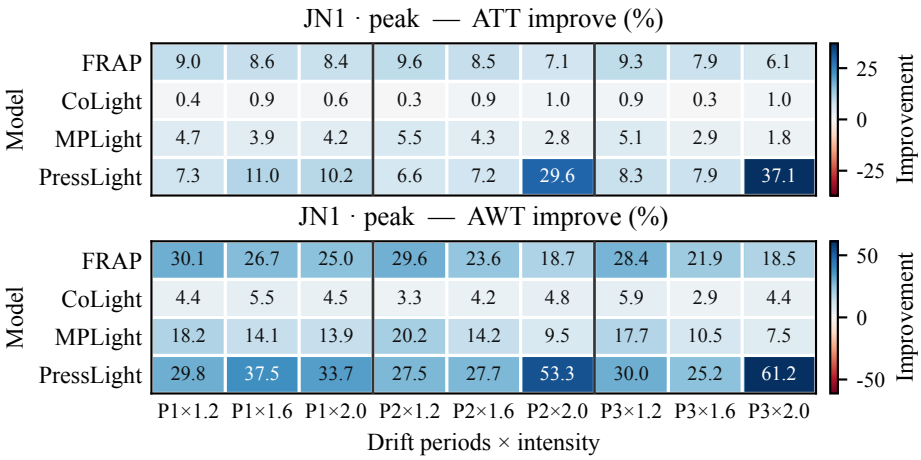


Figure 37: Relative performance improvement of CRC under different drift intensities and numbers of drift periods on the JN1 peak-boost scenario. Higher values indicate larger reductions in ATT or AWT.

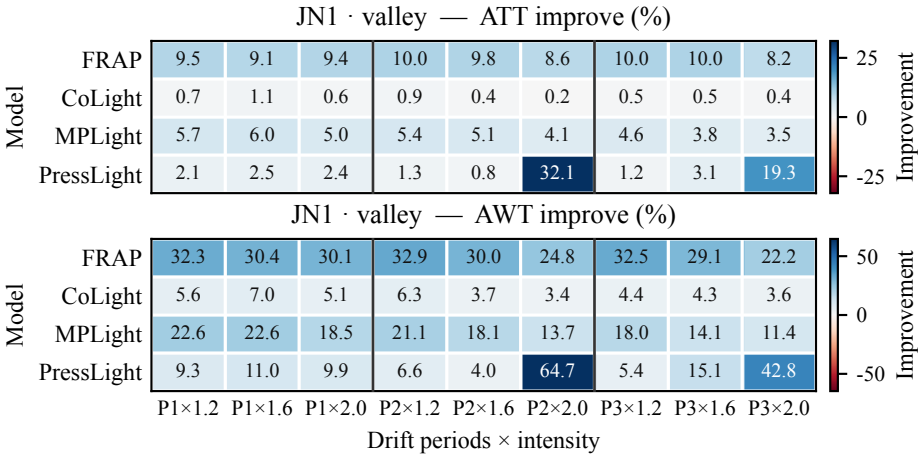


Figure 38: Relative performance improvement of CRC under different drift intensities and numbers of drift periods on the JN1 valley-boost scenario. Higher values indicate larger reductions in ATT or AWT.

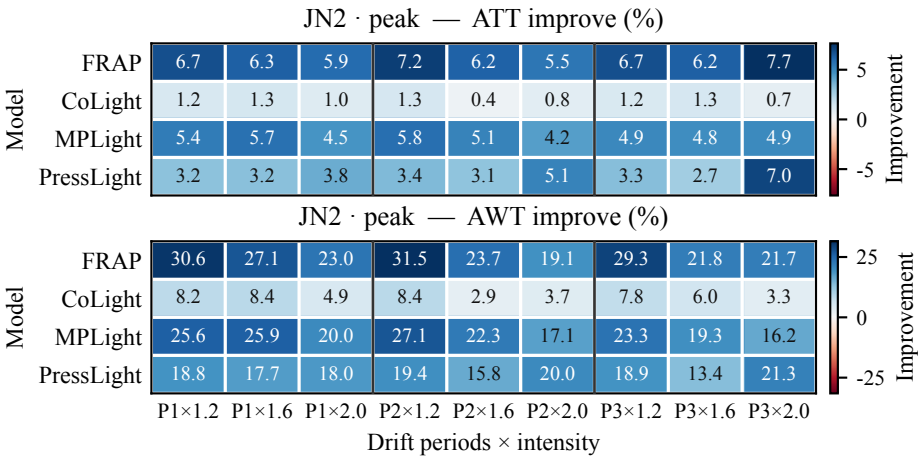


Figure 39: Relative performance improvement of CRC under different drift intensities and numbers of drift periods on the JN2 peak-boost scenario. Higher values indicate larger reductions in ATT or AWT.

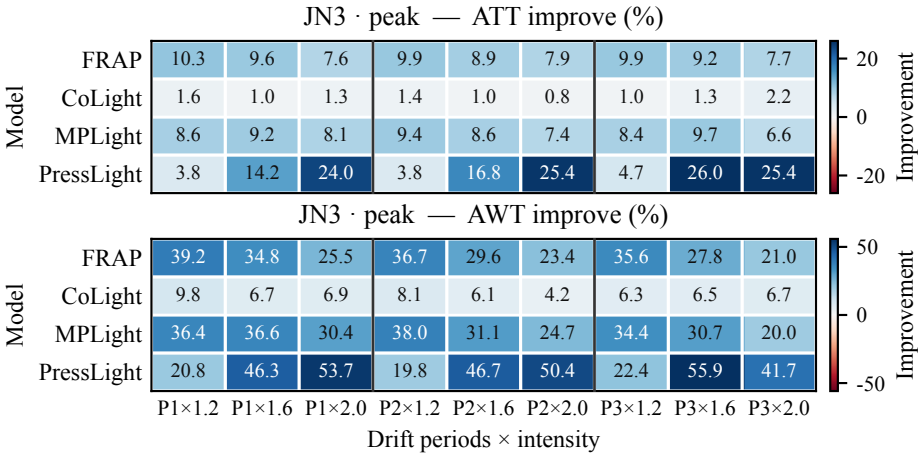


Figure 40: Relative performance improvement of CRC under different drift intensities and numbers of drift periods on the JN3 peak-boost scenario. Higher values indicate larger reductions in ATT or AWT.

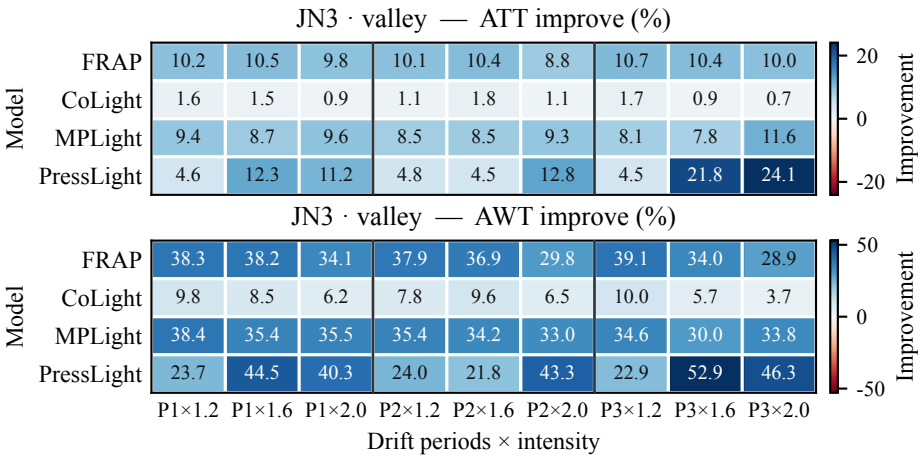


Figure 41: Relative performance improvement of CRC under different drift intensities and numbers of drift periods on the JN3 valley-boost scenario. Higher values indicate larger reductions in ATT or AWT.