

---

# RobustLight: Improving Robustness via Diffusion Reinforcement Learning for Traffic Signal Control

---

Mingyuan Li<sup>1</sup> Jiahao Wang<sup>1</sup> Guangsheng Yu<sup>2</sup> Xu Wang<sup>2</sup> Qianrun Chen<sup>3</sup> Wei Ni<sup>2</sup> Lixiang Li<sup>1</sup>  
Haipeng Peng<sup>1</sup>

## Abstract

Reinforcement Learning (RL) optimizes Traffic Signal Control (TSC) to reduce congestion and emissions, but real-world TSC systems face challenges like adversarial attacks and missing data, leading to incorrect signal decisions and increased congestion. Existing methods, limited to offline data predictions, address only one issue and fail to meet TSC's dynamic, real-time needs. We propose RobustLight, a novel framework with an enhanced, plug-and-play diffusion model to improve TSC robustness against noise, missing data, and complex patterns by restoring attacked data. RobustLight integrates two algorithms to recover original data states without altering existing TSC platforms. Using a dynamic state infilling algorithm, it trains the diffusion model online. Experiments on real-world datasets show RobustLight improves recovery performance by up to 50.43% compared to baseline scenarios. It effectively counters diverse adversarial attacks and missing data. The relevant datasets and code are available at [GitHub](#).

## 1. Introduction

### 1.1. Motivation

Improving traffic efficiency through Traffic Signal Control (TSC) has been established as an effective strategy (Wei et al., 2019b). Traditional TSC systems depend heavily on static, predefined expert system controls, lacking the flexibility of dynamically responding to fluctuating traffic conditions (Lowrie, 1990; Hunt et al., 1982; Webster, 1958). Recently, by integrating the methods of Reinforcement Learning (RL), TSC has demonstrated its superiority

<sup>1</sup>Beijing University of Posts and Telecommunications  
<sup>2</sup>University of Technology Sydney <sup>3</sup>Lanzhou University. Correspondence to: Haipeng Peng <penghaipeng@bupt.edu.cn>.

*Proceedings of the 42<sup>nd</sup> International Conference on Machine Learning*, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

to conventional expert systems in improving vehicular traffic flow (Wei et al., 2018). Deploying TSC algorithms in real-world settings typically involves using sensors, such as cameras and radar, to monitor traffic states, including vehicle counts and speeds. Sensors exposed in public areas are susceptible to noise attacked (Chen et al., 2018; Chowdhury et al., 2023b), leading to potential adversarial attacks. Moreover, in extremely adverse weather conditions, these sensors are prone to physical damage (Laszka et al., 2016b; Lee & Park, 2012). This study delves into these sensor security challenges in real-world TSC systems.

Adversarial attacks on the sensors can range from Gaussian noise and uniformly distributed random noise (U-rand) to more sophisticated strategies, including maximum action-difference attack (MAD) and minimum Q-value attack (MinQ) (Tang et al., 2016). When testing a TSC algorithm, its performance can significantly decline (Gershenson, 2004; Chen et al., 2020; Zhang et al., 2022b) within the CityFlow (Zhang et al., 2019) simulation environment under various attacks. Such attacks compromise sensors, leading to inaccurate state observations and consequent malfunctions in TSC systems. When the sensors are intruded on or compromised, traffic disruptions could potentially occur, even leading to traffic safety accidents and substantial economic losses. To this end, it is crucial to develop innovative defense algorithms to mitigate the impact of sensor-related anomalies for TSC.

### 1.2. Challenges

Various solutions have been proposed in the context of online applications to address security issues faced by sensors (Sun et al., 2021). However, these methods typically only address a single type of attack. For instance, Zhang et al. (Zhang et al., 2021) proposed a method to address an adversarial attack, and it significantly decreases performance as noise levels increase, especially in complex, high-dimensional state environments (Yang et al., 2022). Lin et al. (Lin et al., 2017) employed model-based methods, e.g., a Multi-Layer Perceptron (MLP) for data prediction to address a data loss problem. Similarly, Mei et al. (Mei et al., 2023) advocated for interpolation techniques during training

to solve a missing data problem. Yang et al. (Yang et al., 2023) proposed an offline method to solve the security problem of robotics. However, in a TSC system that requires real-time decision based on traffic volume, the above methods need to collect a large amount of data offline, which cannot meet the demand of real-time dynamic change of TSC data. The challenges faced by the existing TSC systems can be summarized as follows:

- Recent TSC algorithms demonstrate significant performance degradation when undergoing adversarial attacks or sensor damage, often resulting in traffic congestion, indicating a lack of security resilience.
- Existing defense methods address only one or two types of attacks and lack a comprehensive framework to address multiple security issues holistically.
- Current offline methods typically rely on the collection of large datasets to train static models, which struggle to make accurate decisions when faced with untrained real-time data.

### 1.3. Contributions

Diffusion models have recently achieved great results in image generation and RL control (Janner et al., 2022; Wang et al., 2022; Yang et al., 2023). The training of a diffusion model consists of the noise addition and denoising processes. The denoising process has the potential to evade noise attacks undergone by TSC systems. Moreover, the diffusion models have a strong data generation ability that has the potential to solve the missing data problem in TSC systems. Our contributions in addressing these challenges can be summarized as follows:

- We propose RobustLight, a framework to enhance the robustness of online TSC systems, consisting of a TSC agent and a dynamic state filling (DSI) agent. The DSI agent uses a model-free RL algorithm with an enhanced diffusion model to recover TSC data in real-time, ensuring optimal strategies without altering the original TSC algorithms. This architecture prevents data missing without changing the TSC algorithms.
- The RobustLight framework adopts two algorithms, de-noise and repaint, which leverage the trained diffusion model of DSI to defend against adversarial attacks and handle missing data in TSC systems.
- Experiments show that RobustLight improves the average travel time of existing TSC algorithms by 50.43% under various adversarial attacks and missing data scenarios, with state recovery closely matching pre-attack distributions, demonstrating its enhanced robustness.

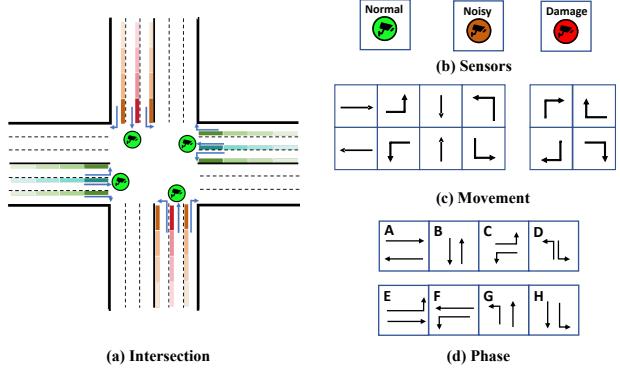


Figure 1. Definition of the TSC.

## 2. Preliminary

### 2.1. Definitions of TSC

We use a four-way intersection, as shown in Figure 1, to illustrate the concepts and summarize the definition of TSC.

**Intersection.** Each road network consists of multiple intersections, each with multiple, e.g.,  $N$ , road segments, denoted by  $(Inter_1, \dots, Inter_N)$ .

**Traffic perception.** Each intersection has four directional sensors (e.g., cameras, radars) monitoring vehicles in three lanes, with different security states indicated by colors: orange for noise attacks, red for sensor damage, and green for normal operation, as shown in Figure 1(b).

**Traffic movement.** The vehicle passes through the intersection from entering the lane ( $lane_{in}$ ) to leaving the lane ( $lane_{out}$ ). This traffic movement is represented as  $TM = (lane_{in}, lane_{out})$ , as shown in Figure 1(c).

**Traffic signal phase.** Two movements i.e.,  $(TM_i, TM_j, i \neq j)$  form a phase, represented as  $p_w = (TM_i, TM_j)$ , as depicted in Figure 1(d).

### 2.2. Adversarial Attack and Physical Sensor Damage

We introduce the concepts of four classic adversarial attacks and the definition of TSC sensor attacks.

**Gaussian noise attack.** The Gaussian noise attack adds Gaussian random noise  $\mathcal{N}$  within the scale or intensity  $k$  to the state  $s$ , represented as  $\tilde{s}_t = s_t + k \cdot \mathcal{N}(\mu, \sigma^2)$

**U-rand attack.** The uniform random noise (U-rand) attack adds U-rand noise  $\mathcal{U}$  within the intensity  $k$  to the state  $s$ , i.e.,  $\tilde{s}_t = s_t + k \cdot \mathcal{U}(\mathcal{I}, \mathcal{I})$ , where  $\mathcal{I}$  is the identity matrix.

**MAD attack.** The maximum action-difference (MAD) attack selects noise within a given range  $k$  to maximize the difference between two policies  $\pi_\phi(\cdot|s)$  and  $\pi_\phi(\cdot|\tilde{s})$  in the policy space, denoted as  $\tilde{s}_t = s_t +$

$\arg \max_{\tilde{s} \in \mathbf{B}_d(s, k)} D(\pi_\phi(\cdot|s) \| \pi_\phi(\cdot|\tilde{s}))$ , where  $\mathbf{B}_d(s, k)$  is the  $\ell_\infty$  ball centered at state  $s$  with radius  $k$ .

**MinQ attack.** The Minimum Q-value (adversarial) attack selects the minimum  $Q$ -value within a certain range  $k$  and adds it as noise to the original state  $s$ , denoted as  $\tilde{s}_t = s_t + \arg \min_{\tilde{s} \in \mathbf{B}_d(s, k)} Q(\tilde{s}_t, \pi_\phi(\cdot|\tilde{s}))$ .

**Physical sensor damage.** For each intersection, sensor damage due to weather or human factors causes state dimensions to be unobserved, represented as  $\tilde{s}_t = Mask \cdot s_t$ .

### 2.3. Diffusion Process and Guided Diffusion

The diffusion model (Ho et al., 2020b) is a probabilistic deep learning model that generates samples through diffusion and inverse diffusion processes.

**Forward process.** In this process, the data is deformed by introducing random noise, which makes the data gradually lose structured information and eventually turn into random noise. The forward diffusion chain gradually adds noise to the data  $x_0$  (sampled from the distribution  $q(x_0)$ ) over  $T$  steps using a pre-defined variance schedule  $\beta_t \{\beta_t \in (0, 1)\}_{t=1}^T$ , as given by

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}), \quad (1)$$

where  $\alpha_t = 1 - \beta_t$  and  $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$ . Equation (1) allows for the use of reparametrization (Gu et al., 2022) to directly obtain the noisy data  $x_t$  corresponding to a specific timestep  $t$  from the initial data without the need for multi-step iteration.

**Reverse process.** This process is assumed to follow a Gaussian distribution probability, as given by

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)), \quad (2)$$

where  $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$  represents the mean of the Gaussian distribution that needs to be predicted by the neural network, and  $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)$  is the pre-defined variance. After training, the neural network predicts the noise term. We sample data from an isotropic Gaussian noise and run the reverse diffusion process from  $t = T$  to  $t = 0$ .

Guided Diffusion (Kim & Oh, 2022) extends diffusion by introducing external guidance during the reverse process. It controls the generation of specific content. Classifier-free guidance (Ho & Salimans, 2022) is a widely considered type of guided diffusion, which adds condition  $c$  to the  $\boldsymbol{\mu}_\theta$  network in the diffusion reverse process, as given by

$$\boldsymbol{\mu}_\theta^{guided}(x_t, t, c) = \boldsymbol{\mu}_\theta(x_t, t) + \omega(\boldsymbol{\mu}_\theta(x_t, t, c) - \boldsymbol{\mu}_\theta(x_t, t)), \quad (3)$$

where  $\omega$  is a weighting coefficient that controls the trade-off between conditional and unconditional generations.

## 3. RobustLight

In this section, we elaborate on the proposed RobustLight framework, which integrates a new dynamic state infilling (DSI) algorithm, and two new processes, namely, the no-attack training process and the attacked testing process, into any type of the existing TSC algorithms to protect from data false injection and missing data. Specifically, we propose DSI to train an improved diffusion model. The denoise and repaint algorithms use the trained diffusion model of DSI to solve adversarial attacks and missing data problems. The integration of these algorithms improve the robustness of TSC in unreliable real-time traffic environments.

### 3.1. TSC Algorithm

A TSC algorithm implements the basic functions of signal control, serving as a TSC agent to interact with the traffic environment. This allows for the use of various TSC algorithms, including both traditional and RL-based algorithms.

In what follows, we elaborate on the TSC algorithms (namely, MaxPressure (Cools et al., 2013), Advanced-Maxpressure (Zhang et al., 2022b), Colight (Wei et al., 2019a), Advanced-Colight (Zhang et al., 2022b), Mp-light (Chen et al., 2020), and Advanced-Mplight (Zhang et al., 2022b)). The specific definitions of state, action, and reward are given by

- **State:** The Efficient Pressure (EP) (Wu et al., 2021) and Running Vehicle (RV) are the input state  $s$ .
- **Action:** The traffic signal phase is action  $a$ .
- **Reward:** The negative of the queue length is the reward  $r$ .

We represent the TSC state trajectory  $\tau$  as a sequence, as given by

$$\tau_t^s = \{s_1, s_2, \dots, s_{t-1}, s_t\}, \quad (4)$$

where  $t$  indicates the RL timestep. We can update the RL-based TSC agent according to the Bellman function. Then, the Q-value in the RL-based TSC evolves as

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[ r + \gamma \max_{a' \in a} Q'(s', a') - Q(s, a) \right], \quad (5)$$

where  $\alpha$  represents the learning rate,  $\gamma$  is the discount factor, and  $Q'(s', a')$  denotes the target Q-value for the next state-action pair. For the TSC algorithms based on RL, a replay buffer is used to store tuples  $(s, a, r, s')$ .

### 3.2. Dynamic State Infilling (DSI) Algorithm

We design a DSI algorithm using an improved diffusion model as the policy to reduce noise in online model-free RL. The TSC agent's replay buffer is used to update the policy and adapt to real-time environments. The state, action, and reward of the DSI agent are defined as follows:

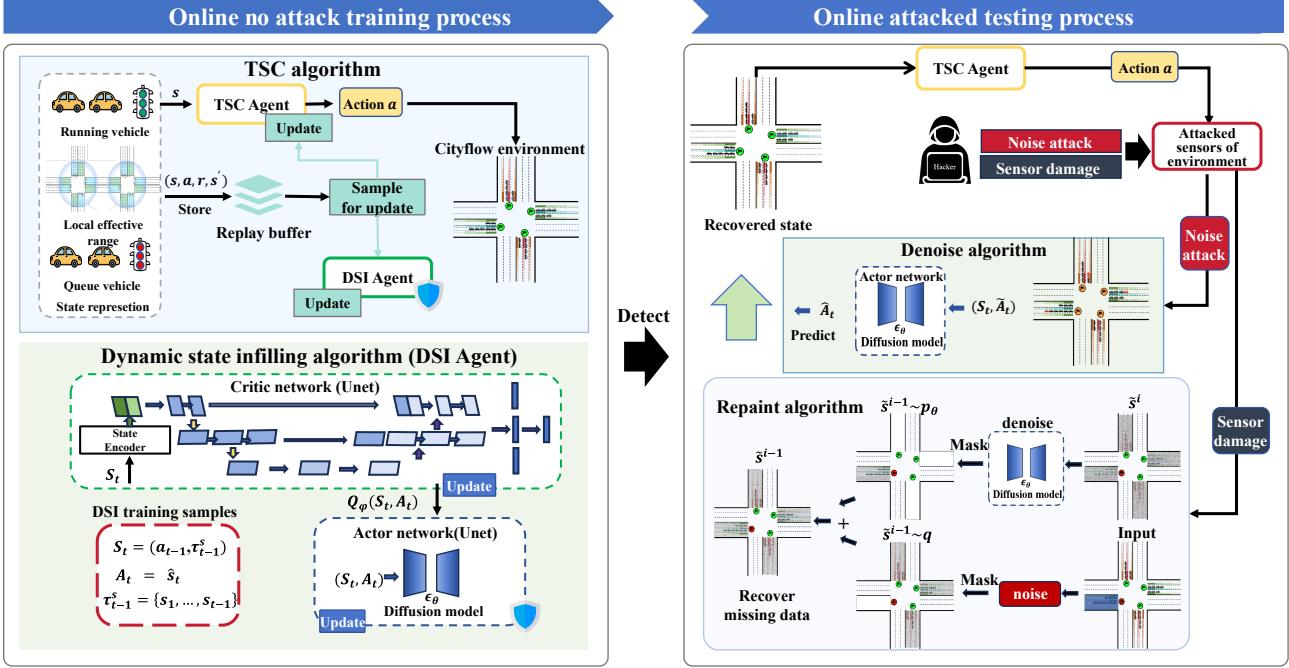


Figure 2. RobustLight uses a two-process framework with four algorithms: the TSC and DSI algorithms train on normal data to enhance efficiency and robustness, while the Denoise and Repaint algorithms mitigate noise and infer missing data during attack testing.

• **State.** We represent the DSI agent’s state as  $S_t = (a_{t-1}, \tau_{t-1}^s)$ , where  $a_{t-1}$  is the previous TSC action and  $\tau_{t-1}^s = \{s_1, \dots, s_{t-1}\}$  is the TSC state trajectory.

• **Action.** The DSI denoises the noisy or missing state  $\tilde{s}_t$  to the recovered state  $\hat{s}_t$ . Hence, the action at time  $t$  can be represented as  $\hat{A}_t = \hat{s}_t$ .

• **Reward.** The reward is the negative of the queue length. The DSI agent consists of an actor and a critic network. **Critic network.** In the critic network parameterized by  $\varphi$ , a U-Net architecture (Ronneberger et al., 2015) identical to that of the actor network is employed. The final layer of the critic network produces the  $Q$  value. The network is updated using the Bellman error, as given by

$$L_\varphi = r + \lambda Q_\varphi(S_{t+1}, A_{t+1}) - Q_\varphi(S_t, A_t), \quad (6)$$

where  $\lambda$  is the discount factor.

**Actor network.** For the actor network, we use action gradient ascent to update the control policy (Yang et al., 2023)

$$A_t = A_t + \eta \nabla_A Q_\varphi(S_t, A_t), \quad (7)$$

where  $\eta$  is the learning rate of the action gradient ascent. We employ an improved diffusion model to denoise and predict the original state  $\hat{A}_t = \hat{s}_t$  through the reverse process:

$$\hat{A}_t \sim p_\theta(\hat{A}_t^{0:k} | S_t) = f_k(\hat{A}_t) \prod_{i=1}^k p_\theta(\hat{A}_t^{i-1} | \hat{A}_t^i, S_{t-1}), \quad (8)$$

where  $f_k(\hat{A}_t) = \sqrt{\alpha_k} \hat{A}_t$ , and  $\prod_{i=1}^k p_\theta(\hat{A}_t^{i-1} | \hat{A}_t^i, S_{t-1})$  can be modeled as Gaussian distribution, as follows:

$$\mathcal{N}(\hat{A}_t^{i-1}; \mu_\theta(\hat{A}_t^i, S_{t-1}, i), \Sigma_\theta(\hat{A}_t^i, S_{t-1}, i)). \quad (9)$$

According to (Ho et al., 2020b), the denoising process for each step in the diffusion is expressed as

$$\hat{A}_t^{i-1} | \hat{A}_t^i = \frac{\hat{A}_t^i}{\sqrt{\alpha_i}} - \frac{\beta_i}{\sqrt{\alpha_i(1-\alpha_i)}} \epsilon_\theta(\hat{A}_t^i, S_t, i) + \sqrt{\tilde{\beta}_i} \epsilon. \quad (10)$$

To handle denoising tasks with improved diffusion, we use the beta schedule (Zhihe & Xu)  $\beta_i = 1 - \alpha_i = e^{-\frac{b}{i+a+c}}$ ,  $\bar{\alpha}_k = \prod_{i=1}^k \alpha_i$ , and  $\tilde{\beta}_i = \frac{1-\bar{\alpha}_{i-1}}{1-\bar{\alpha}_i} \beta_i$  to address small to medium-scale noises, where  $\epsilon_\theta$  is the noise network. Unlike other beta (Xiao et al., 2021a; Nichol & Dhariwal, 2021a; Ho et al., 2020a) schedules that generate data from pure noise, our approach employs the U-Net architecture and a conditional generative diffusion model for noise prediction.

**Lemma 3.1.** *The lemma basis of diffusion for denoising is to minimize the sum entropy of the denoised data:*

$$L_\theta = \sum_{t=1}^T \mathbb{E}_{q(A_t)}[-\log p_\theta(\hat{A}_t^0 | S_{t-1})], \quad (11)$$

which can be optimized using the variational lower bound

**Algorithm 1** The training process of RobustLight

**Initialize:** DSI algorithm  $Q_\varphi$  critic network,  $Q_\theta$  actor network, Replay buffer  $\mathcal{R}$  and target network  $Q_{\varphi-}, Q_{\theta-}$ .

- 1: **for**  $j = 1$  to  $E$  **do**
- 2:   Get init state  $s_0$  from Cityflow
- 3:   **for**  $t = 1$  to  $T$  **do**
- 4:     Execute TSC algorithms, including Colight, Advanced-Colight, Maxpressure,..., etc.
- 5:     Training the TSC agent by Equation 5.
- 6:     Sample batch  $S = (\tau^s, a)$  and  $A = s$  from  $\mathcal{R}$ .
- 7:     Update critic network  $Q_\varphi$  by Equation 6.
- 8:     **for**  $n = 1$  to  $N$  **do**
- 9:       Execute gradient ascend by Equation 7.
- 10:      **end for**
- 11:     Update actor network  $Q_\theta$  by Equation 12.
- 12:     Update target critic network by:
- 13:        $\varphi^- = \eta\varphi + (1 - \eta)\varphi^-$
- 14:       Update target actor network by
- 15:        $\theta^- = \eta\theta + (1 - \eta)\theta^-$
- 16:      **end for**
- 17:     **end for**
- 18:    Return Actor network  $Q_\theta$

(VLB). For the detailed derivation process, please refer to (Zhihe & Xu; Ho et al., 2020a).

Expanding  $L_\theta$ , we obtain the following Non-Markov loss to update the actor network:

$$L_\theta = \mathbb{E}_{i \sim \mathcal{U}_K, \epsilon_t \sim \mathcal{N}(0, I), (A_{t-N}, \dots, A_{t+M-1}) \in D_v} \left\| \epsilon_\theta(\tilde{A}_t^i, S_{t-1}, i) - \epsilon_t^i \right\|_2 + \sum_{m=t+1}^{t+M-1} \left\| \epsilon_\theta(\tilde{A}_m^i, \hat{S}_{m-1}, i) - \epsilon_m^i \right\|_2 \quad (12)$$

where  $\hat{S}_{m-1} = (a_{m-1}, \tau_{m-1}^{\hat{s}})$ , and  $\tau_{m-1}^{\hat{s}}$  is the predicted state trajectory. This loss function  $L_\theta$  balances the trade-off between the current RL timestep and future RL timesteps, aiming to minimize the accumulated error over time. Detailed training algorithm is summarized in Algorithm 1.

### 3.3. Denoise Algorithm

In the testing phase, we may unintentionally expose the sensors within the TSC algorithm to various attacks, including Gaussian, MAD, U-rand, and MinQ attacks.

In the denoise algorithm, we use the trained actor diffusion model from the DSI algorithm to recover the original TSC state  $\hat{A}_t = \hat{s}_t$ . We use the diffusion reverse process to denoise. As described in Equation (10), we input  $(S_t, \tilde{A}_t)$  and the diffusion time step  $i$  into the actor diffusion network. In each diffusion timestep, the data currently under a noise attack is denoised to obtain the data for the next diffu-

**Algorithm 2** Repaint algorithm of RobustLight

- 1: Input  $\tilde{A}_t^i, S_{t-1}, m$
- 2: **for**  $i = 1$  to  $K$  **do**
- 3:   **for**  $u = 1$  to  $U$  **do**
- 4:      $\epsilon \sim \mathcal{N}(0, I)$  if  $i > 1$ , else  $\epsilon = 0$
- 5:     Get  $\tilde{A}_{t,known}^{i-1}$  by Equation (13)
- 6:      $z \sim \mathcal{N}(0, I)$  if  $i > 1$ , else  $z = 0$
- 7:     Get  $\tilde{A}_{t,unknown}^{i-1}$  by Equation (14)
- 8:     Get recovered  $\hat{A}_t^{i-1}$  by Equation (15)
- 9:     **if**  $u < U$  and  $i > 1$  **then**
- 10:        $\tilde{A}_t^i \sim \mathcal{N}(\sqrt{1 - \beta_{i-1}}\hat{A}_t^{i-1}, \beta_{i-1}I)$
- 11:     **end if**
- 12:   **end for**
- 13: **end for**
- 14: Return  $\hat{A}_t = \hat{A}_t^0$

sion timestep. We perform the denoising process in each  $i$  timesteps to predict the original state  $\hat{s}_t^0$  using Equation (8).

### 3.4. Repaint Algorithm

The concept of utilizing diffusion for image repaint is inspired by (Lugmayr et al., 2022). The core idea is to use the known sensor data to infer the unknown sensor data. We adapt and refine the image repaint process to the TSC and introduce the repaint algorithm to interpolate the damaged or missing TSC state, as illustrated in Figure 2.

We use a well-trained conditional denoising diffusion probabilistic model, based on DSI (distinct from (Zhihe & Xu)), to effectively restore the original state  $\hat{A} = \hat{s}$ , where the unknown part is represented by  $m \odot \tilde{A}^{i-1}$  with  $m$  being the mask matrix, and the known part is  $(1 - m) \odot \tilde{A}^{i-1}$ , stands for unmask matrix. We note that the reverse process of diffusion from  $\hat{A}^i$  to  $\tilde{A}^{i-1}$  depends solely on  $\hat{A}^i$ , as long as we maintain the correctness properties of the corresponding distribution. Therefore, we update the known state  $(1 - m) \odot \tilde{A}^{i-1}$ . According to Equation (1), we sample the known state at any diffusion timestep  $i$ , and use Equation (2) for the unknown state. We use the following expression for one reverse step:

$$\tilde{A}_{t,known}^{i-1} = \sqrt{\bar{\alpha}_i} \tilde{A}_{t,known} + \sqrt{1 - \bar{\alpha}_i} \epsilon, \quad (13)$$

$$\tilde{A}_{t,unknown}^{i-1} = \frac{1}{\sqrt{\bar{\alpha}_i}} \tilde{A}_{t,unknown}^i - \frac{\beta_i}{\sqrt{\bar{\alpha}_i(1 - \bar{\alpha}_i)}} \epsilon_\theta(\tilde{A}_t^i, S_{t-1}, i) + \sqrt{\beta_i} z, \quad (14)$$

$$\tilde{A}_t^{i-1} = m \odot \tilde{A}_{t,known}^{i-1} + (1 - m) \odot \tilde{A}_{t,unknown}^{i-1}. \quad (15)$$

Thus,  $\tilde{A}_{t,known}^{i-1}$  is sampled using the known state in the given state  $m \odot A^0$ , and  $\tilde{A}_{t,unknown}^{i-1}$  is sampled from the condition model given the previous iteration  $\tilde{A}^i$ . These are then combined into the new sample  $\tilde{A}^{i-1}$ . The repaint algorithm is summarized in Algorithm 2.

**Table 1. Performance of ATT in JiNan, HangZhou.** “-” implies that traditional algorithms are not adapted to MinQ and MAD attacks. Our RobustLight recovers the state of traditional and RL-based TSC algorithms to evaluate the performance.

Dataset	Noise Type	Noise Scale	FixedTime		MaxPressure		Advanced-MaxPressure		Advanced-MpLight		CoLight		Advanced-CoLight	
			base	base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight	base
<i>JiNan</i> <sub>1</sub>	Gaussian	3.5		301.97±2.14	<b>296.01±2.18</b>	285.43±1.82	<b>283.48±1.22</b>	356.86±19.78	<b>302.16±2.86</b>	287.76±1.52	<b>282.35±1.96</b>	320.57±2.82	<b>294.57±2.02</b>	
		4.0		305.88±2.92	<b>298.95±2.69</b>	290.28±2.17	<b>289.71±1.59</b>	377.91±38.59	<b>311.59±4.69</b>	314.93±28.28	<b>288.07±2.27</b>	338.93±15.32	<b>298.23±5.16</b>	
	U-rand	3.5		325.09±2.65	<b>319.01±2.17</b>	312.27±2.92	<b>306.12±1.79</b>	538.66±132.69	<b>349.89±22.56</b>	312.27±4.07	<b>407.41±24.57</b>	449.13±21.14	<b>358.22±7.02</b>	
		4.0	428.11±0.00	331.58±3.57	<b>322.93±2.67</b>	319.18±2.91	<b>311.32±1.97</b>	549.41±120.08	<b>364.39±28.71</b>	352.63±7.85	<b>320.48±3.29</b>	460.46±22.41	<b>360.82±4.79</b>	
	MAD	3.5		-	-	-	-	321.24±14.89	<b>280.45±2.99</b>	487.69±50.75	<b>277.37±1.16</b>	479.63±8.82	<b>283.13±1.56</b>	
		4.0		-	-	-	-	338.14±24.71	<b>287.04±2.69</b>	555.71±64.93	<b>279.15±1.22</b>	520.09±7.12	<b>288.07±2.27</b>	
	MinQ	3.5		-	-	-	-	313.73±1.87	<b>291.58±8.12</b>	683.38±94.91	<b>277.93±3.47</b>	394.39±50.27	<b>323.25±20.54</b>	
		4.0		-	-	-	-	321.55±12.64	<b>295.12±6.06</b>	716.58±134.92	<b>281.13±1.96</b>	394.63±47.37	<b>331.52±10.64</b>	
<i>JiNan</i> <sub>3</sub>	Gaussian	3.5		275.32±1.24	<b>270.16±1.70</b>	267.76±0.93	<b>263.81±0.94</b>	383.41±58.34	<b>272.12±2.25</b>	289.03±10.62	<b>258.33±2.06</b>	410.96±132.05	<b>288.87±3.59</b>	
		4.0		279.69±1.92	<b>274.18±1.25</b>	272.17±1.26	<b>271.87±2.03</b>	550.13±188.98	<b>279.86±2.31</b>	293.55±1.86	<b>261.63±1.66</b>	461.57±126.58	<b>294.84±5.45</b>	
	U-rand	3.5		299.34±2.22	<b>288.71±2.56</b>	293.46±1.22	<b>289.66±2.09</b>	454.32±138.56	<b>307.59±20.82</b>	407.89±50.61	<b>300.36±7.67</b>	987.04±42.62	<b>561.02±62.23</b>	
		4.0	383.01±0.00	306.02±2.36	<b>292.13±2.65</b>	298.58±2.19	<b>294.64±2.42</b>	500.65±122.68	<b>313.02±17.33</b>	554.51±80.09	<b>306.03±6.59</b>	1015.29±41.75	<b>596.24±67.48</b>	
	MAD	3.5		-	-	-	-	498.88±208.66	<b>259.84±1.52</b>	534.43±87.67	<b>263.73±2.57</b>	431.77±17.35	<b>260.78±1.42</b>	
		4.0		-	-	-	-	594.48±218.45	<b>264.52±1.38</b>	550.26±93.93	<b>267.03±2.09</b>	474.16±17.21	<b>260.78±1.42</b>	
	MinQ	3.5		-	-	-	-	652.05±230.01	<b>283.01±34.19</b>	726.85±282.56	<b>267.34±3.65</b>	515.34±167.83	<b>309.65±9.99</b>	
		4.0		-	-	-	-	733.62±233.21	<b>295.12±57.76</b>	744.21±274.72	<b>273.13±4.52</b>	516.23±167.39	<b>329.81±14.76</b>	
<i>HangZhou</i> <sub>1</sub>	Gaussian	3.5		332.03±2.01	<b>322.45±1.25</b>	327.37±1.94	<b>324.84±1.79</b>	564.65±103.29	<b>351.68±40.82</b>	356.33±6.38	<b>322.62±5.96</b>	480.38±22.93	<b>327.98±2.45</b>	
		4.0		335.97±2.49	<b>326.58±1.36</b>	331.93±1.57	<b>329.25±1.21</b>	490.72±93.84	<b>366.62±50.39</b>	371.97±12.44	<b>337.58±5.68</b>	510.38±25.17	<b>331.21±3.65</b>	
	U-rand	3.5		356.53±3.72	<b>338.42±1.74</b>	354.91±3.31	<b>297.34±1.39</b>	373.88±34.05	<b>328.18±3.17</b>	647.64±54.89	<b>435.33±20.02</b>	717.12±70.08	<b>473.85±32.68</b>	
		4.0	495.57±0.00	361.58±4.37	<b>341.16±2.21</b>	359.61±3.34	<b>350.86±3.12</b>	384.75±42.24	<b>330.47±3.11</b>	475.49±29.78	<b>441.33±17.05</b>	738.51±63.18	<b>487.74±31.44</b>	
	MAD	3.5		-	-	-	-	803.38±92.02	<b>325.49±28.86</b>	402.67±17.42	<b>324.55±6.63</b>	463.64±33.44	<b>309.24±0.94</b>	
		4.0		-	-	-	-	782.23±121.05	<b>320.91±2.73</b>	419.07±18.26	<b>331.47±8.62</b>	491.64±39.78	<b>352.58±3.71</b>	
	MinQ	3.5		-	-	-	-	778.04±78.91	<b>420.46±87.75</b>	466.14±70.37	<b>317.36±3.96</b>	449.54±8.22	<b>406.32±6.41</b>	
		4.0		-	-	-	-	764.88±78.52	<b>435.53±69.87</b>	460.29±62.45	<b>331.93±3.91</b>	482.98±7.29	<b>430.53±8.49</b>	

## 4. Experiments

We conduct experiments with real-world datasets to evaluate RobustLight’s generalization and efficiency on NVIDIA P100 hardware, using the hyperparameters in Table 10.

### 4.1. Datasets

We use real-world traffic flow and road topology datasets for our experiments, with Cityflow (Zhang et al., 2019) as the simulator to evaluate Average Travel Time (ATT) and exit points with a simulation time of 60 minutes for all vehicles. The datasets include vehicle start and end points, following a fixed motion model. Seven traffic datasets from three cities JiNan and HangZhou (China) and New York (Zheng et al., 2019) (USA) are used.

**JiNan Datasets:** The JiNan road network consists of 12 intersections (in a  $3 \times 4$  grid). It includes three traffic flow datasets: *JiNan*<sub>1</sub>, *JiNan*<sub>2</sub>, and *JiNan*<sub>3</sub>.

**HangZhou Datasets:** The HangZhou network encompasses 16 intersections (in a  $4 \times 4$  grid) and features two datasets: *HangZhou*<sub>1</sub> and *HangZhou*<sub>2</sub>.

**New York Datasets:** The New York network features a more complex structure with 192 intersections ( $28 \times 7$  grid) and includes two datasets: *Newyork*<sub>1</sub> and *Newyork*<sub>2</sub>.

### 4.2. Compared Methods

**Traditional Methods:** These methods include Fixed-Time (Webster, 1958), which uses a fixed green phase time;

Advanced-Maxpressure (Zhang et al., 2022a), which uses running and waiting vehicles to choose the phase; and Max-pressure (Gershenson, 2004), which uses waiting vehicles to choose the phase.

**RL-based Methods:** For RL benchmarks, we consider Co-Light (Wei et al., 2019a), which uses waiting and neighboring vehicles to select the signal phase; Advanced-CoLight (Zhang et al., 2022a), which employs waiting and running vehicles with a graph attention neural network; and Advanced-Mplight (Zhang et al., 2022a), which uses the FRAP (Zheng et al., 2019) model for signal phase selection.

**RobustLight Method:** RobustLight integrates the base methods of traditional and RL-based TSC algorithms to recover data in real-time, and then evaluates the ATT under different sensor noise attacks and sensor damage. Results are presented as the average of ten independent runs.

### 4.3. Results

This subsection presents the results of our experiments, evaluating RobustLight’s performance under various conditions, including resilience to noise attacks and sensor damage, using ATT on real-world traffic datasets.

#### 4.3.1. NOISE ATTACK ON STATE RESULTS

Table 15 summarizes our experimental results for the *JiNan* and *HangZhou* datasets based on the ATT metric. The noise scale range is based on the value of  $k$  for state noise attacks, as described in Section 2.2. The re-

**Table 2. ATT in JiNan and HangZhou: 25% refers to missing data in  $sensor_W$ , and 50% refers to  $sensor_W$  and  $sensor_E$ .**

Dataset	Mask Scale	FixedTime		MaxPressure		Advanced-MaxPressure		Advanced-MpLight		CoLight		Advanced-CoLight	
		base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight
JiNan1	25%	428.11±0.00	386.74±0.00	<b>324.31±22.74</b>	353.04±0.00	<b>302.84±9.43</b>	398.27±86.93	<b>337.63±50.23</b>	400.93±19.66	<b>372.93±7.43</b>	326.14±16.52	<b>298.35±16.32</b>	
	50%	798.90±0.00	<b>613.23±108.81</b>	1061.92±0.00	<b>548.84±152.33</b>	1052.83±102.63	<b>642.62±111.28</b>	849.61±77.93	<b>766.91±81.83</b>	732.37±52.93	<b>680.37±97.32</b>		
JiNan2	25%	368.76±0.00	272.51±0.00	<b>273.78±3.61</b>	323.13±0.00	<b>274.84±4.74</b>	701.85±200.84	<b>259.36±6.43</b>	637.94±285.96	<b>308.56±22.59</b>	290.86±24.79	<b>259.56±7.59</b>	
	50%	836.81±0.00	<b>748.23±132.12</b>	1209.97±0.00	<b>751.53±268.33</b>	985.42±64.04	<b>665.92±148.31</b>	880.23±52.93	<b>754.23±173.95</b>	725.57±57.89	<b>639.94±100.49</b>		
JiNan3	25%	383.01±0.00	289.81±0.00	<b>288.74±9.83</b>	340.81±0.00	<b>286.54±10.85</b>	824.35±228.64	<b>368.62±104.37</b>	367.94±9.36	<b>315.43±9.67</b>	363.84±54.43	<b>301.63±38.26</b>	
	50%	823.48±0.00	<b>574.97±69.0</b>	1109.57±0.00	<b>592.56±222.45</b>	947.84±150.34	<b>567.37±75.16</b>	816.73±56.68	<b>382.39±37.83</b>	756.37±162.69	<b>407.68±78.39</b>		
HangZhou1	25%	495.57±0.00	369.77±0.00	<b>350.8±20.22</b>	513.15±0.00	<b>371.64±28.35</b>	510.37±92.75	<b>372.12±21.49</b>	490.36±13.84	<b>424.36±13.22</b>	401.87±48.31	<b>328.26±12.67</b>	
	50%	722.43±0.00	<b>714.32±76.93</b>	1186.56±0.00	<b>752.84±399.84</b>	1170.97±71.83	<b>515.77±180.92</b>	786.32±40.86	<b>542.91±108.24</b>	842.29±235.36	<b>541.56±122.96</b>		
HangZhou2	25%	406.65±0.00	372.12±0.00	<b>353.85±3.66</b>	405.27±0.00	<b>356.64±4.64</b>	362.93±15.36	<b>350.26±10.62</b>	397.88±20.68	<b>390.43±10.37</b>	378.86±26.86	<b>375.64±22.32</b>	
	50%	533.81±0.00	<b>447.91±70.34</b>	781.57±0.00	<b>542.37±161.07</b>	639.52±127.45	<b>522.92±157.32</b>	663.72±80.82	<b>470.91±34.82</b>	508.93±30.27	<b>444.37±61.24</b>		

sults reveal that the incorporation of Robustlight leads to improved performance across all methods, demonstrating its effectiveness in mitigating various types of noise attacks.

#### 4.3.2. SENSOR DAMAGE ON STATE RESULTS

We compare traditional and RL-based TSC algorithms, focusing on deliberate sensor attacks on  $sensor_W$  and  $sensor_E$ , with data from these directions masked to simulate damage. We use the Repaint algorithm within RobustLight for data completion, allowing us to evaluate the TSC algorithm's performance before and after this process, as shown in Table 2. Our findings lead to several key conclusions. Under noisy conditions, MaxPressure outperforms Advanced-MaxPressure with an 11.6% average improvement across all datasets. Advanced-CoLight surpasses Co-Light with an 18.4% improvement. The RobustLight algorithm enhances performance in nearly all methods, with Advanced-CoLight showing a 10.3% improvement. Following sensor damage in  $sensor_W$  and  $sensor_E$ , all methods perform worse than FixedTime, but our algorithms effectively recover missing data and still outperform FixedTime with damage in one direction, highlighting the importance of addressing sensor damage in real-world TSC deployments.

**Table 3. Performance comparison with other benchmarks.**

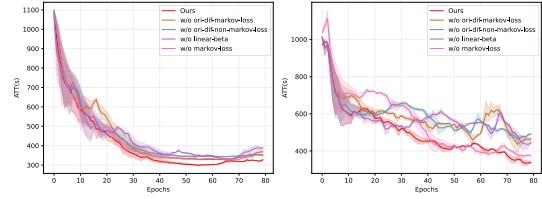
Dataset	Noise Type	Noise Scale	Advanced-CoLight			
			base	RobustLight	Diffusion Linear-Beta	LSTM
JiNan1	Gaussian	3.5	320.57±2.82	<b>294.57±2.02</b>	324.04±3.04	892.0±213.7
		4.0	338.93±15.32	<b>298.23±5.16</b>	326.83±6.56	914.18±215.7
	Sensor	25%	326.14±16.52	<b>298.35±16.32</b>	350.96±17.32	1035.39±156.69
HangZhou1	Damage	50%	732.37±52.93	<b>680.37±97.32</b>	689.13±98.43	1088.77±160.11
	Gaussian	3.5	480.38±22.93	<b>327.98±2.45</b>	480.53±33.55	1123.4±341.28
		4.0	510.38±25.17	<b>331.21±3.65</b>	510.16±22.34	1140.26±325.02
	Sensor	25%	401.87±48.31	<b>328.26±12.67</b>	386.13±29.28	1581.35±37.31
	Damage	50%	842.29±235.36	<b>541.56±122.96</b>	551.44±40.16	1638.32±14.1

#### 4.3.3. OTHER METHODS COMPARISON EXPERIMENTS

We compare the native Diffusion model with three Beta schedule methods (Xiao et al., 2021b; Ho et al., 2020b; Nichol & Dhariwal, 2021b) and LSTM model (Sun et al., 2021) on real-world datasets. As shown in Table 3, our RobustLight outperforms the native Diffusion model in all noise attack scenarios, demonstrating its effectiveness.

**Table 4. Performance Comparison with DiffLight**

Dataset	Noise/Mask	Method	RobustLight	PSNR	MAE	ATT
JiNan1	U-rand Noise (3.5)	DiffLight	No	6.71	6.26	310.92
		Advanced-MaxPressure	Yes	<b>7.20</b>	<b>5.42</b>	<b>304.34</b>
HangZhou1	U-rand Noise (3.5)	DiffLight	No	6.85	7.93	361.31
		Advanced-MaxPressure	Yes	<b>7.71</b>	<b>5.12</b>	<b>297.34</b>
JiNan1	Sensor Damage (25%)	DiffLight	No	7.66	1.05	366.05
		Advanced-CoLight	Yes	<b>9.34</b>	<b>0.89</b>	<b>304.13</b>
HangZhou1	Sensor Damage (25%)	DiffLight	No	18.05	1.84	372.53
		Advanced-CoLight	Yes	<b>22.96</b>	<b>1.17</b>	<b>306.56</b>

**Figure 3. Ablation of RobustLight based on Advanced-CoLight in JiNan1 and HangZhou1 under Gaussian noise (scale 3.5).**

To validate our approach with DiffLight (Chen et al., 2024), we setup involved randomly masking data from Kriging Missing (12.5%, single-intersection-sensor failure) and Random Missing (12.5%, full-intersection failure). As demonstrated in Table 4, our method effectively addresses data missing scenarios (Kriging and random missing) while also exhibiting robust performance under noisy data conditions.

#### 4.3.4. ABLATION EXPERIMENTS

We conduct ablation experiments on RobustLight to assess the impact of each component. “w/o linear beta” refers to the experiment without the linear beta schedule, “w/o markov-loss” excludes the Markov loss, “w/o ori-dif-markov-loss” omits the original diffusion model with Markov loss, and “w/o ori-dif-non-markov-loss” excludes the original diffusion model with non-Markov loss. Showing the effectiveness of RobustLight.

#### 4.3.5. ROBUSTNESS RECOVERY ANALYSIS

We analyze RobustLight’s effectiveness in recovering the original data using t-SNE (Van der Maaten & Hinton, 2008) plots. In the plots, yellow represents the original state distribution, blue represents the recovered states, and red shows

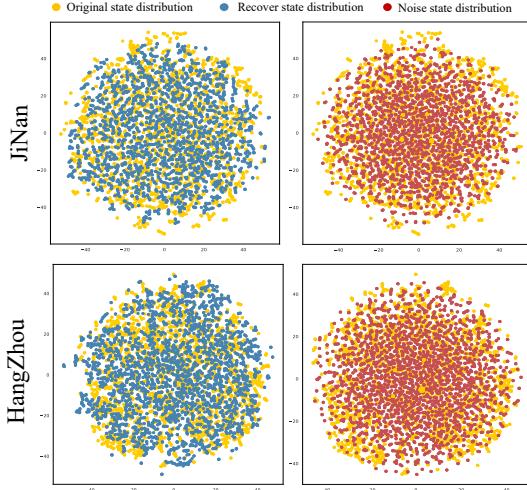


Figure 4. State visualization of RobustLight based on Advanced-Colight in *HangZhou<sub>1</sub>* and *JiNan<sub>1</sub>*.

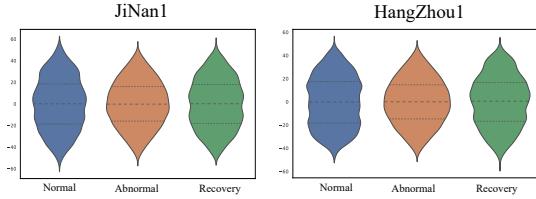


Figure 5. Violin figure of normal, noise and recovery data.

the distribution with Gaussian noise (scale 3.5). Figure 4 shows that the blue distribution is closer to the original, demonstrating RobustLight’s strong robustness. Additionally, violin plots in Figure 5 show that the recovered data distribution closely matches the normal data.

To directly evaluate the recovery effect, we define the metric  $E_{denoise} = \frac{1}{N} \sum_1^T |\hat{s} - s|$ . Where  $N$  is the number of intersections,  $T$  is the running count,  $\hat{s}$  is the recovered state, and  $s$  represents the original state. This metric measures the ability to restore data by the absolute difference between the original and reconstructed states. As shown in Table 5, RobustLight exhibits strong recovery performance.

#### 4.3.6. MODEL GENERALIZATION

To assess generalization, we train a diffusion model based on the Advanced-Maxpressure algorithm using the *JiNan<sub>1</sub>*

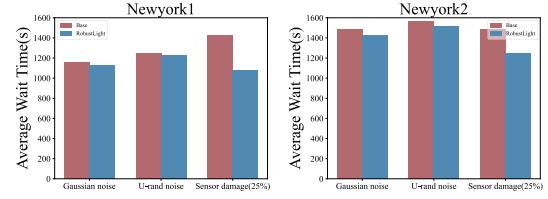


Figure 6. Performance of RobustLight based on Advanced-Maxpressure in *Newyork* transfer by *JiNan<sub>1</sub>*.

dataset and test it on the *Newyork* dataset. As shown in Figure 6, transferring the model to *Newyork* effectively mitigates noise attacks and demonstrates strong state restoration in a different urban traffic environment.

## 5. Discussion

### Q1. Is there any challenge to apply the diffusion model?

We evaluate the time consumption of the DSI algorithm (Table 17) to assess its computational costs and impact on TSC algorithms. As the noise range increases, more diffusion steps are needed, resulting in longer recovery times. A key challenge in large-scale scenarios is slow inference, driven by both the diffusion steps and the increasing number of intersections. To address this, we plan to reduce diffusion steps using DDIM and divide intersections into smaller sub-regions, each with its own actor diffusion model, to speed up inference. This discussion highlights the trade-off between accuracy and speed in diffusion models for real-time TSC systems and suggests ways to improve efficiency in large-scale deployments.

**Q2. Is there any report about cyber attacks on traffic control systems?** Traffic sensor attacks (Chen et al., 2019; Chowdhury et al., 2023a; Laszka et al., 2016a), demonstrated at DEFCON 22 by Cesar Cerrudo, highlight vulnerabilities in traffic control systems, where hackers target street magnetic sensors. Discussions on platforms like Quora (“hack traffic lights”) further emphasize concerns about sensor safety. As AI systems become more prominent, ensuring their security is crucial, especially given the additional risks from extreme weather interference.

**Q3. How to detect noise and missing data?** To detect noise and missing data in our study, we use a combination of attack detection models and statistical methods. We assume that attacks, caused by factors such as weather conditions and radio interference, have been pre-detected, with the attack detection model identifying noisy or missing data based on abnormal sensor readings. Noise is detected through outlier detection and variance analysis, while missing data is identified by checking for null or NaN values and analyzing missingness patterns across sensors. Once detected, we handle noise using denoise algorithm and missing data through repaint algorithm of RobustLight, ensuring data

Table 5.  $E_{denoise}$  the smaller the value, the better.

Dataset	Noise Type	Noise Scale	CoLight		Advanced-CoLight	
			base	RobustLight	base	RobustLight
<i>JiNan<sub>1</sub></i>	Gaussian	3.5	399.15±20.31	<b>284.46±21.26</b>	808.76±16.84	<b>484.15±13.26</b>
	U-rand	3.5	899.85±15.23	<b>628.36±13.82</b>	1814.56±31.36	<b>1186.43±24.31</b>
<i>HangZhou<sub>1</sub></i>	Gaussian	3.5	399.15±21.35	<b>284.30±16.37</b>	1077.09±25.82	<b>320.57±13.82</b>
	U-rand	3.5	1198.21±13.37	<b>535.67±16.31</b>	2417.44±19.96	<b>1202.61±38.74</b>

**Table 6. Detection Rate and Throughput Comparison**

Detection Rate	Dataset	Base	ATT	Base	Throughput
80%	<i>JiNan</i> <sub>1</sub>	487	<b>297</b>	5812	<b>6154</b>
	<i>HangZhou</i> <sub>1</sub>	463	<b>326</b>	2888	<b>2938</b>
60%	<i>JiNan</i> <sub>1</sub>	487	<b>328</b>	5812	<b>6131</b>
	<i>HangZhou</i> <sub>1</sub>	463	<b>331</b>	2888	<b>2930</b>

integrity and robustness in the presence of environmental disturbances.

**Q4. If the RobustLight rely on the accuracy of detection?** To evaluate its reliance on detection accuracy, we integrated TP-FDS (Sarteshnizi et al., 2023), a method that detects anomalies by comparing new data distributions with historical data from the same period, achieving an AUC of 96% and an F1 score of 76%. TP-FDS identifies anomalies through multi-sensor cross-referencing (e.g., cameras and radar) or rule-based methods, such as detecting a queue increase from 3 to 5 during a north-south green light. Minor fluctuations, like queue changes from 3 to 5, minimally impact system efficiency. To assess RobustLight’s dependence on detection accuracy, we conducted experiments simulating real-world scenarios with detection rates of 80% and 60%, as shown in Table 6. The results show that RobustLight has a robust performance in most detection rates.

**Q5. What are the practical considerations and potential barriers to real-world implementation?** Our framework supports both distributed and centralized deployments: the distributed approach employs a federated learning architecture with central training and edge-based parameter updates, while the centralized solution addresses scalability through high-performance servers and algorithmic optimizations like DDIM (Song et al., 2020) acceleration. For comprehensive coverage of intersection failures (Kriging) and sensor-specific issues (Random missing), we recommend a hybrid approach, implementing centralized data missing recovery algorithms alongside edge-based noise reduction using cost-effective hardware for accelerated denoising. This balanced strategy ensures robustness, real-time performance, and attack resilience across all deployment scenarios.

## 6. Related Works

**TSC Algorithms.** Since the introduction of static fixed-time plans in 1958 (Webster, 1958), TSC systems have evolved significantly. Systems like SCOOT (Hunt et al., 1982) and SCATS (Lowrie, 1990) rely on expert-designed plans with predefined thresholds, lacking dynamic adaptability to changing traffic conditions. The advent of RL methods has marked a paradigm shift, leveraging real-time traffic data to optimize signal management through trial-and-error, outperforming traditional approaches. RL models

in TSC vary from value-based (Abdulhai et al., 2003; Wei et al., 2018), policy-based (Mousavi et al., 2017), to actor-critic frameworks (Aslani et al., 2018; Wu et al., 2022), with state and reward designs incorporating features like queue length (Varaiya, 2013; Wu et al., 2021; Li et al., 2025), vehicle counts (Wei et al., 2019a; Xu et al., 2021), or travel time (Zheng et al., 2019). Advanced-Colight method (Zhang et al., 2022a) stands out, achieving state-of-the-art results by using running and waiting vehicles to model lane capacity relationships.

**Robust RL.** In RL, disturbances cause errors, leading to Robust RL for improved reliability, split into “training-time” and “testing-time” robustness. “Training-time” adds noise during training for adaptability (Zhang et al., 2022c; Ye et al., 2023). “Testing-time” trains in clean settings and tests under disruptions (Yang et al., 2022; Panaganti et al., 2022). Our RobustLight focuses on “testing-time” robustness for real-world attack resilience. “Testing-time” robustness covers state, action, and transition/reward perturbations. State perturbation uses neural networks and SA-MDP (Zhang et al., 2021; 2020) or conservative actions (Yang et al., 2022). Action perturbation employs adversarial training (Tan et al., 2020) or optimal policies (Liu et al., 2023). Transition/reward perturbations use Markov games (Pinto et al., 2017; Gleave et al., 2019) or robust Bellman operators (Panaganti et al., 2022). We prioritize “testing-time” robustness for real-world attack resilience.

**Diffusion Model.** Diffusion models, first used for image generation (Ho et al., 2020b), excel in RL control tasks. Diffusion RL splits into online and offline settings. Online, Yang et al. (Yang et al., 2023) applied diffusion as a policy with model-free control. Offline, Ajay et al. (Ajay et al., 2022) generated trajectories for decisions, and Wang et al. (Wang et al., 2022) introduced Diffusion-QL, merging TD3+BC (Fujimoto & Gu, 2021) with behavior cloning. Online TSC algorithms (Wei et al., 2019a; Chen et al., 2020; Zhang et al., 2022a) falter under state attacks and missing data, risking congestion. RobustLight, our proposed algorithm, enhances online TSC resilience by recovering data without altering existing systems, offering a novel solution for research and industry.

## 7. Conclusion.

In this paper, we introduced RobustLight, designed to address abnormal TSC tasks. By leveraging the denoising properties of the diffusion model, RobustLight effectively handles noise interference and incomplete state information in real-world environments. Empirical results demonstrate its robustness and effectiveness, significantly enhancing the security of RL-based TSC strategies and strengthening traditional TSC algorithms, thereby improving the safety and integrity of TSC systems.

## Acknowledgements

We sincerely thank our collaborators for their invaluable feedback and meticulous proofreading of this manuscript. This work was supported in part by the National Key R&D Program of China (Grant No. 2024YFB2906503) and the National Natural Science Foundation of China (Grant No. 62032002).

## Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

## References

- Abdulhai, B., Pringle, R., and Karakoulas, G. J. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering-asce*, 129:278–285, 2003.
- Ajay, A., Du, Y., Gupta, A., Tenenbaum, J., Jaakkola, T., and Agrawal, P. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.
- Aslani, M., Mesgari, S., Seipel, S., and Wiering, M. Developing adaptive traffic signal control by actor-critic and direct exploration methods. *Proceedings of the Institution of Civil Engineers - Transport*, 172:1–25, 01 2018. doi: 10.1680/jtran.17.00085.
- Chen, C., Wei, H., Xu, N., Zheng, G., Yang, M., Xiong, Y., Xu, K., and Li, Z. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3414–3421, 2020.
- Chen, H., Jiang, Y., Guo, S., Mao, X., Lin, Y., and Wan, H. Difflight: a partial rewards conditioned diffusion model for traffic signal control with missing data. *Advances in Neural Information Processing Systems*, 37:123353–123378, 2024.
- Chen, Q. A., Yin, Y., Feng, Y., Mao, Z. M., and Liu, H. X. Exposing congestion attack on emerging connected vehicle based traffic signal control. In *NDSS*, 2018.
- Chen, X., He, Z., and Sun, L. A bayesian tensor decomposition approach for spatiotemporal traffic data imputation. *Transportation research part C: emerging technologies*, 98:73–84, 2019.
- Chowdhury, A., Karmakar, G., Kamruzzaman, J., Das, R., and Newaz, S. S. An evidence theoretic approach for traffic signal intrusion detection. *Sensors*, 23(10):4646, 2023a.
- Chowdhury, A., Karmakar, G., Kamruzzaman, J., Das, R., and Newaz, S. S. An evidence theoretic approach for traffic signal intrusion detection. *Sensors*, 23(10):4646, 2023b.
- Cools, S.-B., Gershenson, C., and D’Hooghe, B. *Self-Organizing Traffic Lights: A Realistic Simulation*, pp. 45–55. 2013. ISBN 978-1-4471-5113-5.
- Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.
- Gershenson, C. Self-organizing traffic lights. *arXiv preprint nlin/0411066*, 2004.
- Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., and Russell, S. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.
- Gu, S., Chen, D., Bao, J., Wen, F., Zhang, B., Chen, D., Yuan, L., and Guo, B. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10696–10706, 2022.
- Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.
- Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020a.
- Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020b.
- Hunt, P., Robertson, D., Bretherton, R., and Royle, M. C. The scoot on-line traffic signal optimisation technique. *Traffic Engineering & Control*, 23(4):190–192, 1982.
- Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.
- Kim, K. and Oh, S. J. Diffusion graph convolutional networks with adaptive diffusion time. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*, 2022.
- Laszka, A., Potteiger, B., Vorobeychik, Y., Amin, S., and Koutsoukos, X. Vulnerability of transportation networks to traffic-signal tampering. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPs)*, pp. 1–10. IEEE, 2016a.

- Laszka, A., Potteiger, B., Vorobeychik, Y., Amin, S., and Koutsoukos, X. Vulnerability of transportation networks to traffic-signal tampering. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPs)*, pp. 1–10. IEEE, 2016b.
- Lee, J. and Park, B. Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. *IEEE transactions on intelligent transportation systems*, 13(1):81–90, 2012.
- Li, M., Wang, J., Du, B., Shen, J., and Wu, Q. Fuzzylight: A robust two-stage fuzzy approach for traffic signal control works in real cities. *arXiv preprint arXiv:2501.15820*, 2025.
- Lin, Y.-C., Liu, M.-Y., Sun, M., and Huang, J.-B. Detecting adversarial attacks on neural network policies with visual foresight. *arXiv preprint arXiv:1710.00814*, 2017.
- Liu, G., Zhou, Z., Liu, H., and Lai, L. Efficient action robust reinforcement learning with probabilistic policy execution uncertainty. *arXiv preprint arXiv:2307.07666*, 2023.
- Lowrie, P. R. Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic. *Roads and Traffic Authority NSW*, pp. 28, 1990.
- Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11461–11471, 2022.
- Mei, H., Li, J., Shi, B., and Wei, H. Reinforcement learning approaches for traffic signal control under missing data. *arXiv preprint arXiv:2304.10722*, 2023.
- Mousavi, S. S., Schukat, M., and Howley, E. Traffic light control using deep policy-gradient and value-function-based reinforcement learning. *IET Intelligent Transport Systems*, 11(7):417–423, 2017.
- Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021a.
- Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021b.
- Panaganti, K., Xu, Z., Kalathil, D., and Ghavamzadeh, M. Robust reinforcement learning using offline data. *Advances in neural information processing systems*, 35: 32211–32224, 2022.
- Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pp. 2817–2826. PMLR, 2017.
- Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.
- Sarteshnizi, I. T., Bagloee, S. A., Sarvi, M., and Nassir, N. Traffic anomaly detection: Exploiting temporal positioning of flow-density samples. *IEEE Transactions on Intelligent Transportation Systems*, 25(5):4166–4180, 2023.
- Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.
- Sun, Y., Zheng, R., Liang, Y., and Huang, F. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep rl. *arXiv preprint arXiv:2106.05087*, 2021.
- Tan, K. L., Esfandiari, Y., Lee, X. Y., Sarkar, S., et al. Robustifying reinforcement learning agents via action space adversarial training. In *2020 American control conference (ACC)*, pp. 3959–3964. IEEE, 2020.
- Tang, B., Yan, J., Kay, S., and He, H. Detection of false data injection attacks in smart grid under colored gaussian noise. In *2016 IEEE Conference on Communications and Network Security (CNS)*, pp. 172–179. IEEE, 2016.
- Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- Varaiya, P. *The Max-Pressure Controller for Arbitrary Networks of Signalized Intersections*, pp. 27–66. 2013. ISBN 978-1-4614-6243-9. doi: 10.1007/978-1-4614-6243-9\_2. URL [https://doi.org/10.1007/978-1-4614-6243-9\\_2](https://doi.org/10.1007/978-1-4614-6243-9_2).
- Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.
- Webster, F. V. Traffic signal settings. Technical Report 39, 1958.
- Wei, H., Zheng, G., Yao, H., and Li, Z. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2496–2505, 2018.

- Wei, H., Xu, N., Zhang, H., Zheng, G., Zang, X., Chen, C., Zhang, W., Zhu, Y., Xu, K., and Li, Z. J. Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM ’19, pp. 1913–1922, 2019a.
- Wei, H., Zheng, G., Gayah, V., and Li, Z. A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117*, abs/1904.08117, 2019b.
- Wu, Q., Zhang, L., Shen, J., Lü, L., Du, B., and Wu, J. Efficient pressure: Improving efficiency for signalized intersections. *arXiv preprint arXiv:2112.02336*, 2021.
- Wu, Q., Wu, J., Shen, J., Du, B., Telikani, A., Fahmideh, M., and Liang, C. Distributed agent-based deep reinforcement learning for large scale traffic signal control. *Knowledge-Based Systems*, 241:108304, 2022.
- Xiao, Z., Kreis, K., and Vahdat, A. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021a.
- Xiao, Z., Kreis, K., and Vahdat, A. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021b.
- Xu, B., Wang, Y., Wang, Z., Jia, H., and Lu, Z. Hierarchically and cooperatively learning traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 669–677, 2021.
- Yang, L., Huang, Z., Lei, F., Zhong, Y., Yang, Y., Fang, C., Wen, S., Zhou, B., and Lin, Z. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.
- Yang, R., Bai, C., Ma, X., Wang, Z., Zhang, C., and Han, L. Rorl: Robust offline reinforcement learning via conservative smoothing. *Advances in Neural Information Processing Systems*, 35:23851–23866, 2022.
- Ye, C., Yang, R., Gu, Q., and Zhang, T. Corruption-robust offline reinforcement learning with general function approximation. *arXiv preprint arXiv:2310.14550*, 2023.
- Zhang, H., Feng, S., Liu, C., Ding, Y., Zhu, Y., Zhou, Z., Zhang, W., Yu, Y., Jin, H., and Li, Z. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The world wide web conference*, pp. 3620–3624, 2019.
- Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., and Hsieh, C.-J. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems*, 33: 21024–21037, 2020.
- Zhang, H., Chen, H., Boning, D., and Hsieh, C.-J. Robust reinforcement learning on state observations with learned optimal adversary. *arXiv preprint arXiv:2101.08452*, 2021.
- Zhang, L., Wu, Q., Shen, J., Lü, L., Du, B., and Wu, J. Expression might be enough: representing pressure and demand for reinforcement learning based traffic signal control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 26645–26654, 2022a.
- Zhang, L., Wu, Q., Shen, J., Lü, L., Du, B., and Wu, J. Expression might be enough: Representing pressure and demand for reinforcement learning based traffic signal control. In *International Conference on Machine Learning*, pp. 26645–26654. PMLR, 2022b.
- Zhang, X., Chen, Y., Zhu, X., and Sun, W. Corruption-robust offline reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 5757–5773. PMLR, 2022c.
- Zheng, G., Xiong, Y., Zang, X., Feng, J., Wei, H., Zhang, H., Li, Y., Xu, K., and Li, Z. Learning phase competition for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM ’19, pp. 1963–1972, 2019.
- Zhihe, Y. and Xu, Y. Dmbp: Diffusion model-based predictor for robust offline reinforcement learning against state observation perturbations. In *The Twelfth International Conference on Learning Representations*.

## A. Appendix: Settings.

### A.1. Unet Network Structure

The unet structure is shown in Tables 7, 8, and 9.

Table 7. TemporalUnet Structure

Layer	Layer (input size, output size)
<b>state_encoder</b>	Linear(state_dim, hidden_size) Mish() Linear(hidden_size, state_dim)
<b>action_encoder</b>	Linear(action_dim, hidden_size) Mish() Linear(hidden_size, action_dim)
<b>time_mlp</b>	SinusoidalPosEmb(hidden_size,hidden_size) Linear(hidden_size, hidden_size*2) Mish() Linear(hidden_size*2, hidden_size)
<b>downs</b>	ResidualTemporalBlock(hidden_size,hidden_size*2) ResidualTemporalBlock(hidden_size*2,hidden_size*2) Identity() Downsample1d(hidden_size*2,hidden_size*2) ResidualTemporalBlock(hidden_size*2,hidden_size*4) ResidualTemporalBlock(hidden_size*4,hidden_size*2) Identity() Identity()
<b>ups</b>	ResidualTemporalBlock(hidden_size*8,hidden_size*2) ResidualTemporalBlock(hidden_size*2,hidden_size*2) Identity() ConvTranspose1d(hidden_size*2,hidden_size*2)
<b>mid_block1</b>	ResidualTemporalBlock(hidden_size*4,hidden_size*4) ResidualTemporalBlock
<b>mid_attn</b>	Identity()
<b>mid_block2</b>	ResidualTemporalBlock(hidden_size*4,hidden_size*4)
<b>final_conv</b>	Conv1dBlock(hidden_size*2,hidden_size*2) Conv1d(hidden_size*2,hidden_size/4)
<b>mid_layer</b>	Linear(out_horizon*hidden_size/4+ (hidden_size*3)/2+hidden_size,hidden_size*2) Mish() Linear(hidden_size*2, hidden_size*2) Mish() Linear(hidden_size*2, hidden_size*2) Mish()
<b>final_layer</b>	Linear(hidden_size*2, hidden_size/2)

Table 8. ResidualTemporal Block Structure

Layer		Details
ResidualTemporalBlock	blocks	Conv1dBlock(input_dim,output_dim) Conv1dBlock(output_dim,output_dim)
	time_mlp	Mish() Linear(output_dim, output_dim*2) Rearrange('batch t - batch t 1')
	residual_conv	Conv1d(input_dim, output_dim)

Table 9. Conv1dBlock Structure

Layer	Details
Conv1dBlock	Conv1d(input.dim,output.dim,Kernel Size, Stride, Padding) Rearrange('batch channels horizon' → 'batch channels 1 horizon' 'batch channels 1 horizon' → 'batch channels horizon') GroupNorm(output.dim, Group, eps, Affine: True) Mish()

## A.2. Hyperparameter

By effectively tuning these hyperparameters, users can optimize RobustLight performance for their specific data recovery tasks, achieving better accuracy and robustness in handling missing or corrupted data. The detailed settings are summarized in Table 10.

Table 10. Hyperparameters

Hyperparameter type	Diffusion policy Hyperparameter	Setting
UNet hyperparameter	embed_dim	64
	state_dim	12/24
	action_dim	4
Diffusion training hyperparameter	non_markovian_step	6
	condition_length	4
	beta schedule $a, b, c$	2.1190,25.06,-2.5446
	discount( $\gamma$ )	0.99
	target critic( $\tau$ )	0.005
	diffusion timestep	100
	batch size	64
	buffer capacity	12000
	optimizer	Adam
	learning rate	0.0003
TSC RL agent training hyperparameter	epochs	50
	action gradient steps	20
	discount( $\gamma$ )	0.8
	target critic( $\tau$ )	0.95
	buffer capacity	12000
	epochs	100
	batch_size	20
	learning_rate	0.001
	target update time	5
	normal factor	20
	loss function	MSE
	optimizer	Adam

**Table 11.** Performance in terms of ATT. %25 means the *sensor<sub>N</sub>*, %50 means the *sensor<sub>N</sub>*, *sensor<sub>E</sub>* missing data.

Dataset	Mask Scale	FixedTime		MaxPressure		Advanced-MaxPressure		Advanced-MpLight		CoLight		Advanced-CoLight	
		base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight
JiNan1	25%	428.11±0.00	286.75±0.00	<b>279.52±1.07</b>	265.36±0.0	<b>263.07±1.09</b>	524.48±199.99	<b>260.36±4.47</b>	316.76±15.24	<b>276.02±3.44</b>	253.58±1.84	<b>252.88±1.3</b>	
	50%	428.11±0.00	308.17±0.0	<b>299.56±2.88</b>	277.53±0.0	<b>275.13±1.83</b>	278.37±2.47	<b>270.44±5.09</b>	312.07±4.91	321.85±5.63	313.69±12.85	<b>304.13±11.42</b>	
JiNan2	25%	368.76±0.00	252.12±0.00	283.78±10.61	253.71±0.0	<b>250.77±0.85</b>	268.98±34.52	<b>242.73±0.47</b>	267.43±2.42	<b>260.02±3.18</b>	237.03±1.09	243.49±1.86	
	50%	368.76±0.00	276.93±0.0	<b>272.67±3.52</b>	269.6±0.0	<b>266.67±1.58</b>	288.02±1.38	<b>242.01±25.74</b>	279.01±4.46	<b>271.44±2.54</b>	267.85±6.47	282.31±7.6	
JiNan3	25%	383.01±0.00	251.22±0.00	<b>250.09±8.83</b>	248.68±0.0	<b>245.66±0.74</b>	259.62±0.43	<b>238.43±3.43</b>	267.17±3.04	253.9±1.16	242.36±1.77	<b>239.41±1.7</b>	
	50%	383.01±0.00	273.32±0.0	<b>270.64±4.39</b>	265.72±0.0	<b>267.77±1.53</b>	255.55±1.54	343.01±5.26	288.54±5.05	273.83±2.21	304.03±26.17	<b>268.88±9.01</b>	
HangZhou1	25%	495.57±0.00	306.21±0.00	<b>307.82±1.58</b>	397.12±0.0	<b>301.53±4.38</b>	505.87±9.42	<b>305.12±7.49</b>	376.22±19.3	<b>320.05±6.62</b>	303.94±10.84	<b>290.02±7.84</b>	
	50%	495.57±0.00	392.9±0.0	528.5±0.0	<b>433.36±38.17</b>	428.21±4.29	<b>46.02±10.38</b>	467.12±12.2	<b>362.18±9.7</b>	511.01±56.7	375.96±47.84		
HangZhou2	25%	406.65±0.00	352.17±0.00	<b>349.81±1.19</b>	378.94±0.0	<b>330.82±3.91</b>	382.15±10.52	<b>349.4±2.56</b>	383.97±15.87	<b>362.11±8.16</b>	347.53±18.69	<b>334.01±7.22</b>	
	50%	406.65±0.00	425.14±0.0	<b>405.95±6.37</b>	452.73±0.0	<b>422.18±12.57</b>	436.27±18.12	<b>413.07±13.63</b>	434.66±7.18	<b>415.51±7.82</b>	444.82±41.36	<b>383.46±17.09</b>	

**Table 12.** Performance of in terms of ATT. %25 means the *sensor<sub>E</sub>*, %50 means the *sensor<sub>E</sub>*, *sensors* missing data.

Dataset	Mask Scale	FixedTime		MaxPressure		Advanced-MaxPressure		Advanced-MpLight		CoLight		Advanced-CoLight	
		base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight
JiNan1	25%	428.11±0.00	308.68±0.0	<b>292.73±4.19</b>	288.06±0.0	<b>276.14±2.23</b>	535.51±110.15	<b>360.76±4.58</b>	353.3±4.7	<b>328.99±2.28</b>	291.08±4.77	298.17±8.72	
	50%	428.11±0.00	704.67±0.0	877.64±81.64	1138.58±40.0	<b>837.9±100.32</b>	1168.06±45.85	<b>838.9±76.63</b>	967.5±21.19	<b>955.55±1.28</b>	635.65±21.05	<b>615.93±31.48</b>	
JiNan2	25%	368.76±0.00	265.92±0.0	<b>259.68±6.24</b>	273.48±0.0	<b>257.17±1.6</b>	817.22±231.9	<b>275.46±46.36</b>	281.06±2.07	264.35±5.53	<b>254.08±4.13</b>		
	50%	368.76±0.00	758.94±0.0	<b>641.38±38.7</b>	1106.91±0.0	<b>959.47±69.72</b>	1178.75±106.23	<b>809.77±64.74</b>	1125.51±37.74	<b>1044.55±16.55</b>	659.29±39.5	<b>650.12±39.04</b>	
JiNan3	25%	383.01±0.00	261.32±0.0	<b>260.4±1.35</b>	264.86±0.0	<b>252.11±1.38</b>	545.89±179.58	<b>325.41±44.77</b>	305.41±7.81	<b>271.38±3.22</b>	260.69±2.7	<b>253.03±4.76</b>	
	50%	383.01±0.00	668.31±0.0	701.76±53.94	1254.97±0.0	<b>727.59±91.6</b>	1144.57±64.26	<b>761.78±143.2</b>	1022.03±23.29	<b>801.03±88.64</b>	643.5±45.69	<b>596.76±42.72</b>	
HangZhou1	25%	406.53±0.00	426.5±0.0	<b>426.5±51.56</b>	450.99±0.0	<b>300.44±2.19</b>	619.52±133.36	<b>410.38±60.16</b>	432.58±6.51	<b>335.26±10.09</b>	381.1±62.66	<b>340.48±22.93</b>	
	50%	406.53±0.00	636.51±0.0	754.16±33.52	1005.01±0.0	<b>781.38±69.53</b>	825.5±49.93	<b>424.8±55.41</b>	912.27±43.76	<b>462.48±37.71</b>	602.16±109.41	<b>516.18±19.13</b>	
HangZhou2	25%	406.53±0.00	381.51±0.0	<b>371.48±6.28</b>	414.65±0.0	<b>351.19±4.67</b>	396.42±23.46	<b>353.43±2.89</b>	426.9±2.24	<b>379.94±7.1</b>	377.86±28.22	<b>352.87±10.5</b>	
	50%	406.53±0.00	554.02±0.0	552.24±23.82	718.93±0.0	<b>534.73±38.6</b>	687.16±62.93	<b>447.47±17.12</b>	693.78±34.59	<b>471.14±12.57</b>	501.53±48.97	<b>446.97±11.96</b>	

**Table 13.** Performance in terms of ATT. %25 means the *sensors*, %50 means the *sensor<sub>W</sub>*, *sensor<sub>S</sub>* missing data.

Dataset	Mask Scale	FixedTime		MaxPressure		Advanced-MaxPressure		Advanced-MpLight		CoLight		Advanced-CoLight	
		base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight	base	RobustLight
JiNan1	25%	428.11±0.00	307.99±0.02	318.51±9.13	294.96±0.0	<b>272.62±1.45</b>	314.89±29.39	456.42±158.17	563.6±108.06	<b>347.75±13.07</b>	277.76±4.93	297.89±6.94	
	50%	428.11±0.00	422.76±0.0	537.52±16.47	320.54±0.0	<b>319.24±5.6</b>	326.15±6.18	327.87±5.09	783.84±298.76	<b>377.02±25.84</b>	379.57±13.3	431.91±25.96	
JiNan2	25%	368.76±0.00	263.82±0.0	<b>258.16±2.01</b>	268.66±0.0	<b>256.44±1.63</b>	266.23±20.64	<b>249.71±1.88</b>	286.31±5.87	<b>266.66±2.43</b>	258.47±2.06	<b>257.17±3.87</b>	
	50%	368.76±0.00	306.72±0.0	407.54±7.79	294.71±0.0	<b>288.55±4.1</b>	293.48±18.01	304.9±2.08	342.81±36.26	<b>295.67±6.86</b>	307.66±14.51	377.25±28.18	
JiNan3	25%	383.01±0.00	270.41±0.0	275.75±2.51	262.33±0.0	<b>254.85±1.56</b>	475.95±92.19	<b>250.51±1.99</b>	315.87±7.89	<b>273.27±5.88</b>	277.3±28.08	<b>261.94±5.56</b>	
	50%	383.01±0.00	323.21±0.0	407.54±9.48	306.58±0.0	<b>286.77±4.06</b>	330.86±22.93	337.54±4.54	320.0±2.34	331.79±18.83	341.43±9.31	365.48±16.22	
HangZhou1	25%	495.57±0.00	313.71±0.0	319.2±21.11	445.59±0.0	<b>364.62±14.37</b>	416.28±22.38	<b>391.16±55.98</b>	434.97±1.84	<b>396.72±25.87</b>	408.48±21.16	<b>345.09±11.53</b>	
	50%	495.57±0.00	430.52±0.0	569.06±7.55	632.61±0.0	<b>484.92±19.65</b>	460.54±12.55	<b>388.82±11.75</b>	577.86±20.19	<b>511.12±23.68</b>	656.46±29.04	<b>514.43±46.55</b>	
HangZhou2	25%	406.65±0.00	361.71±0.0	364.52±2.2	394.35±0.0	<b>369.94±28.47</b>	398.64±50.26	<b>379.31±8.41</b>	406.21±4.92	<b>396.05±8.35</b>	393.66±12.79	<b>361.81±7.71</b>	
	50%	406.65±0.00	387.58±0.0	461.11±4.51	446.4±0.0	<b>411.78±3.03</b>	458.86±16.85	467.39±0.68	417.73±16.48	<b>411.47±13.55</b>	462.53±17.42	<b>419.92±9.52</b>	

**Table 14.** Performance of ATT comparison between the native diffusion model, LSTM model and our improved diffusion model.

Dataset	Noise Type	Noise Scale	Colight						Advanced-Colight					
			base	RobustLight	Diffusion	Lstm-Beta	Diffusion	LSTM	base	RobustLight	Diffusion	Lstm-Beta	Diffusion	LSTM
JiNan1	3.5	287.76±1.52	<b>282.63±1.96</b>	287.08±2.23	286.89±1.7	278.63±1.60	1210.56±7.02	320.57±8.82	<b>294.57±2.02</b>	324.04±3.04	304.3±1.04	324.3±6.66	392.04±21.37	
	4.0	314.91±2.48	<b>288.07±2.27</b>	294.76±1.35	1289.13±40.58	296.62±3.28	328.93±15.32	298.23±5.16	326.83±6.56	1230.28±6.59	342.47±8.48	91.18±21.57		
U-rand	3.5	407.41±24.57	<b>312.27±4.27</b>	407.38±2.23	424.39±14.73	405.22±23.74	1102.77±134.8	449.13±1.14	359.82±2.70	433.95±0.03	437.44±12.43	445.68±22.08	786.75±153.39	
	4.0	352.62±3.78	<b>320.48±3.2</b>	431.15±2.49	449.54±7.18	426.82±2.03	1134.31±19.32	460.46±2.41	360.82±4.79	456.95±6.03	453.61±18.91	458.69±21.52		
JiNan3	3.5	487.69±30.75	<b>277.37±1.16</b>	471.97±3.31	630.83±37.37	734.38±1.03	1303.95±1.19	479.63±8.82	283.13±1.56	370.09±5.50	356.21±5.30	360.58±6.34	968.35±150.25	
	4.0	682.38±44.91	<b>277.93±3.47</b>	713.04±19.43	485.78±18.87	497.04±0.25	1296.72±16.26	394.39±50.27	<b>322.25±20.54</b>	508.66±21.33	514.2±30.03	523.33±17.34	1095.13±163.77	
MinQ	4.0	716.58±4.29	<b>281.13±1.94</b>	811.24±23.13	529.54±32.45	553.46±2.36	1296.73±16.26	394.63±4.73	331.52±10.64	532.73±12.30	570.18±4.03	576.11±15.04	1091.69±158.39	
	5.0%	400.93±19.66	<b>372.93±7.43</b>	409.63±3.37	1293.77±45.93	538.01±18.39	1290.52±110.7	326.14±16.52	<b>298.35±16.32&lt;/b</b>					

**Table 15. Performance of ATT in JiNan, HangZhou.** “-” implies that traditional algorithms are not adapted to MinQ and MAD attacks. Our RobustLight recovers the state of traditional and RL-based TSC algorithms to evaluate the performance.

Dataset	Noise Type	Noise Scale	FixedTime	MaxPressure	Advanced-MaxPressure	Advanced-MpLight	CoLight	Advanced-CoLight
			base	base	RobustLight	base	RobustLight	base
<i>JiNan<sub>2</sub></i>	Gaussian	3.5	279.05±1.56	<b>276.14±1.49</b>	275.95±1.22	<b>274.67±1.58</b>	383.41±58.34	<b>265.21±2.02</b>
		4.0	282.99±1.22	<b>279.89±1.41</b>	279.63±1.15	<b>276.56±1.74</b>	455.56±110.18	<b>281.88±2.91</b>
	U-rand	3.5	301.51±2.82	<b>290.84±2.46</b>	299.75±1.16	<b>297.34±1.39</b>	548.04±98.46	<b>323.93±11.34</b>
		4.0	306.29±3.09	<b>294.39±3.25</b>	304.75±1.09	<b>302.97±1.91</b>	514.01±16.27	<b>327.93±11.35</b>
	MAD	3.5	-	-	-	-	285.67±11.23	<b>263.21±1.17</b>
		4.0	-	-	-	-	297.97±18.32	<b>268.94±1.13</b>
	MinQ	3.5	-	-	-	-	283.37±3.09	<b>264.48±1.83</b>
		4.0	-	-	-	-	289.43±3.78	<b>270.98±1.83</b>
<i>HangZhou<sub>2</sub></i>	Gaussian	3.5	360.56±2.45	<b>358.57±1.71</b>	345.27±1.23	<b>342.52±1.23</b>	564.65±103.29	<b>362.27±3.85</b>
		4.0	363.64±2.77	<b>359.93±2.02</b>	348.40±1.11	<b>346.66±1.02</b>	418.41±12.43	<b>371.71±3.65</b>
	U-rand	3.5	371.21±3.44	<b>365.23±2.07</b>	362.11±2.13	<b>389.62±2.09</b>	504.63±9.71	<b>437.93±14.49</b>
		4.0	373.89±3.09	<b>366.52±2.14</b>	366.69±3.53	<b>359.17±2.02</b>	513.83±11.21	<b>445.99±11.53</b>
	MAD	3.5	-	-	-	-	345.74±29.73	<b>344.52±4.49</b>
		4.0	-	-	-	-	349.48±26.63	<b>348.61±5.77</b>
	MinQ	3.5	-	-	-	-	347.04±24.07	<b>343.38±9.54</b>
		4.0	-	-	-	-	350.15±24.27	<b>347.44±9.43</b>

**Table 16. Performance Compared with DiffLight.**

Dataset	Noise Type	Noise Scale	DiffLight	Advanced-Colight-RobustLight
<i>JiNan<sub>1</sub></i>	Gaussian	3.5	289.45±3.37	<b>294.57±2.02</b>
	U-rand	3.5	311.53±4.17	<b>358.22±7.02</b>
	MAD	3.5	384.71±13.25	<b>283.13±1.56</b>
	MinQ	3.5	321.98±6.37	<b>323.25±20.54</b>
	Mask (Kriging and Random)	25%	353.45±34.31	<b>320.31±3.21</b>
<i>HangZhou<sub>1</sub></i>	Gaussian	3.5	325.05±2.58	<b>327.98±2.45</b>
	U-rand	3.5	<b>365.05±20.67</b>	473.85±32.68
	MAD	3.5	366.14±5.34	<b>309.24±0.94</b>
	MinQ	3.5	426.14±10.56	<b>406.32±6.41</b>
	Mask (Kriging and Random)	25%	346.05±41.27	<b>296.69±4.67</b>

## B. Appendix: Other Experiments

**Table 17. Recover time comparison.**

Dataset	Noise type	Noise scale	Diffusion step	Advanced	Advanced
				Colight	Maxpressure
<i>JiNan<sub>1</sub></i>	Gaussian	1.0	20	376.16	343.06
		0.5	20	202.98	178.95
		1.0	16	262.89	233.45
		0.5	8	142.05	124.33
	U-rand	1.0	240	3792.56	-
		0.5	120	1959.03	-
		1.0	240	3779.34	-
		0.5	120	1976.44	-
<i>HangZhou<sub>2</sub></i>	Sensor damage	25%	200	2987.34	2765.36
		50%	200	2997.61	2768.24
		1.0	40	417.26	447.84
		0.5	20	221.38	233.92
	U-rand	1.0	16	281.60	302.86
		0.5	8	156.32	160.16
		1.0	240	5220.45	-
		0.5	120	3581.67	-
	MinQ	1.0	240	5348.28	-
		0.5	120	2889.14	-
		25%	200	3232.13	3552.44
		50%	200	3223.55	3559.41

### C. Appendix: Model Generalization Analysis

In this section, we analyze the model's generalization ability. We transfer the algorithm trained on the *JiNan<sub>1</sub>* dataset to other datasets to observe whether our RobustLight can effectively recover the data under various attacks.

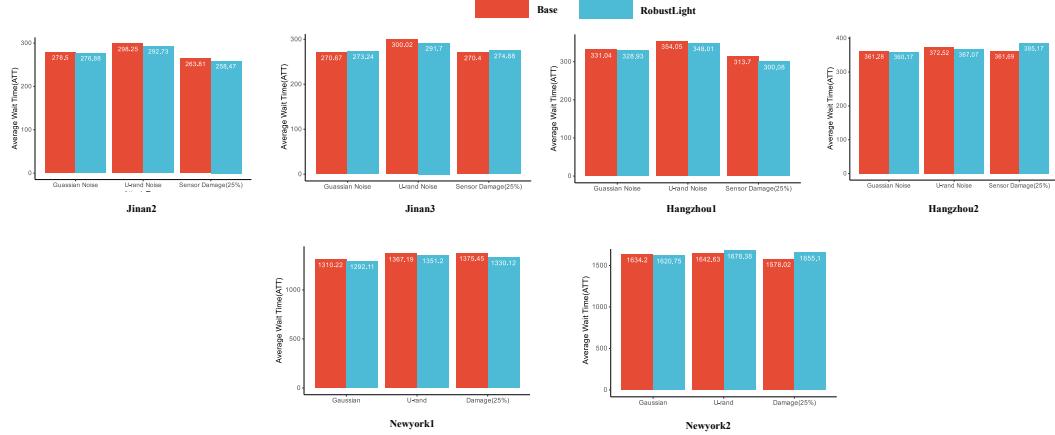


Figure 7. Performance of RobustLight based on Maxpressure transfer by *JiNan<sub>1</sub>*.

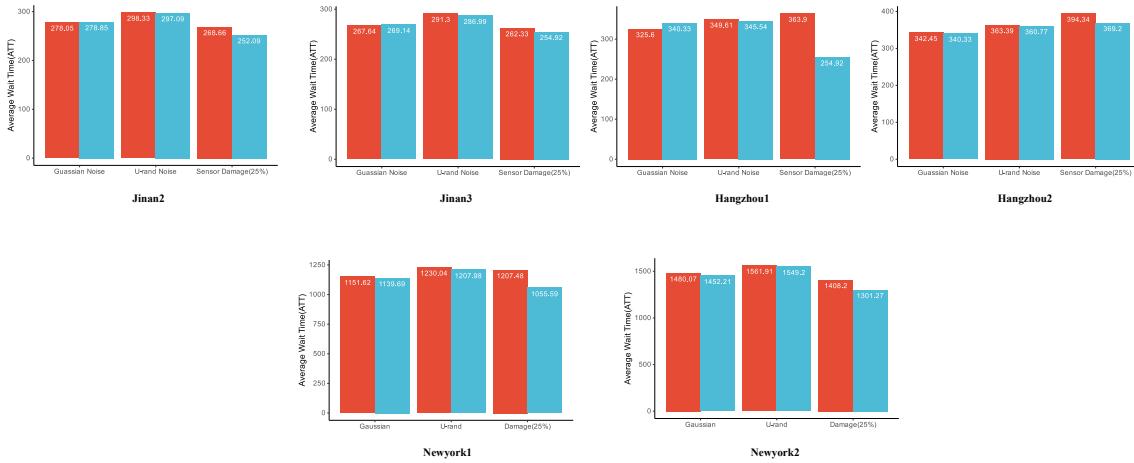


Figure 8. Performance of RobustLight based on Advanced-Maxpressure transfer by *JiNan<sub>1</sub>*.

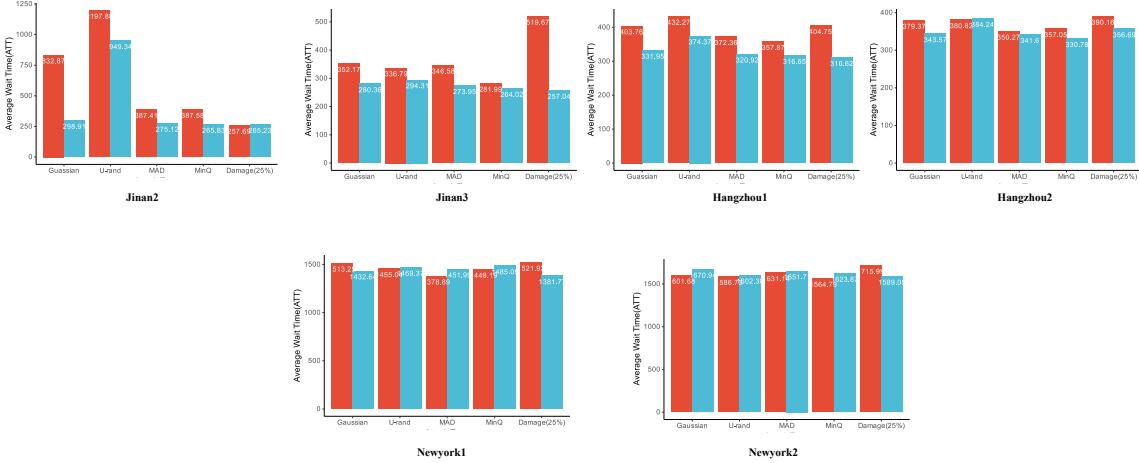


Figure 9. Performance of RobustLight based on Advanced-Mplight transfer by JiNan1.

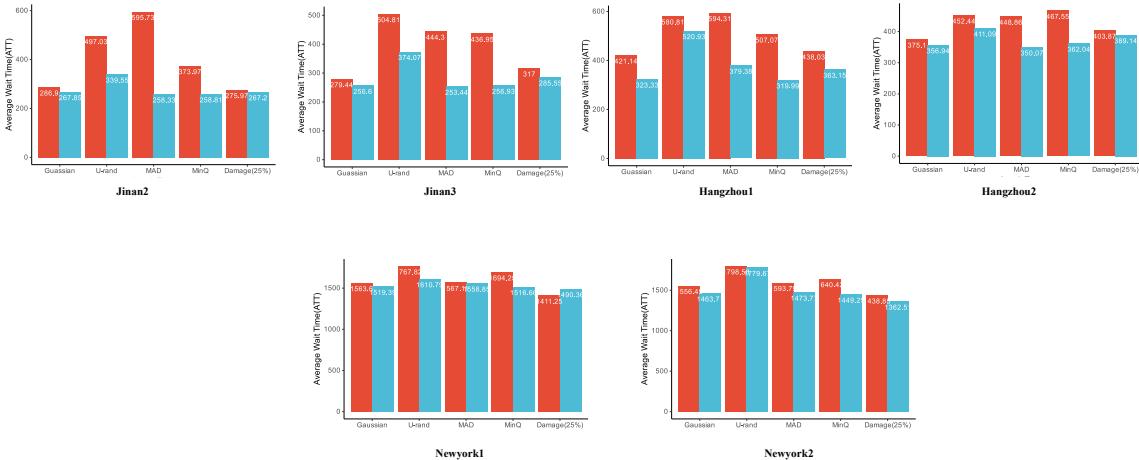


Figure 10. Performance of RobustLight based on Colight transfer by JiNan1.

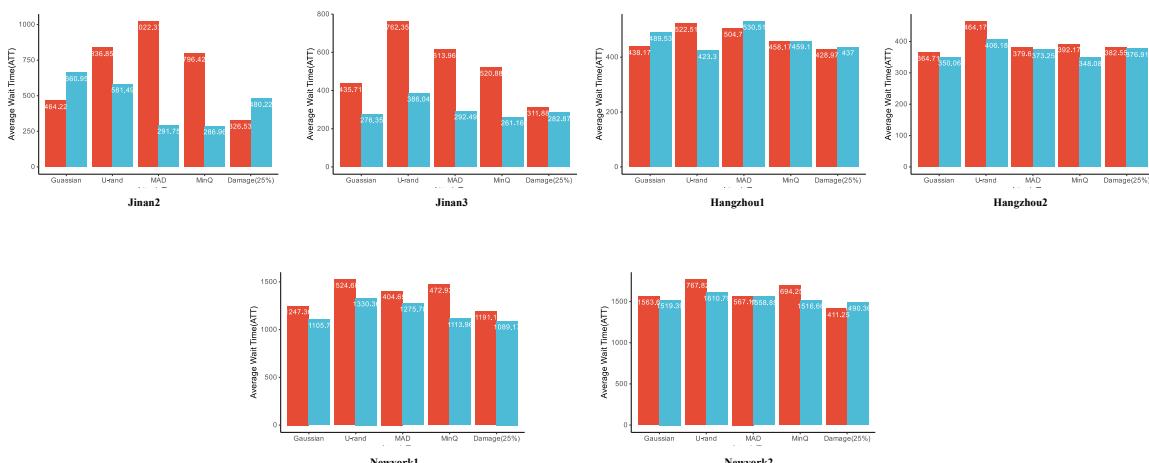
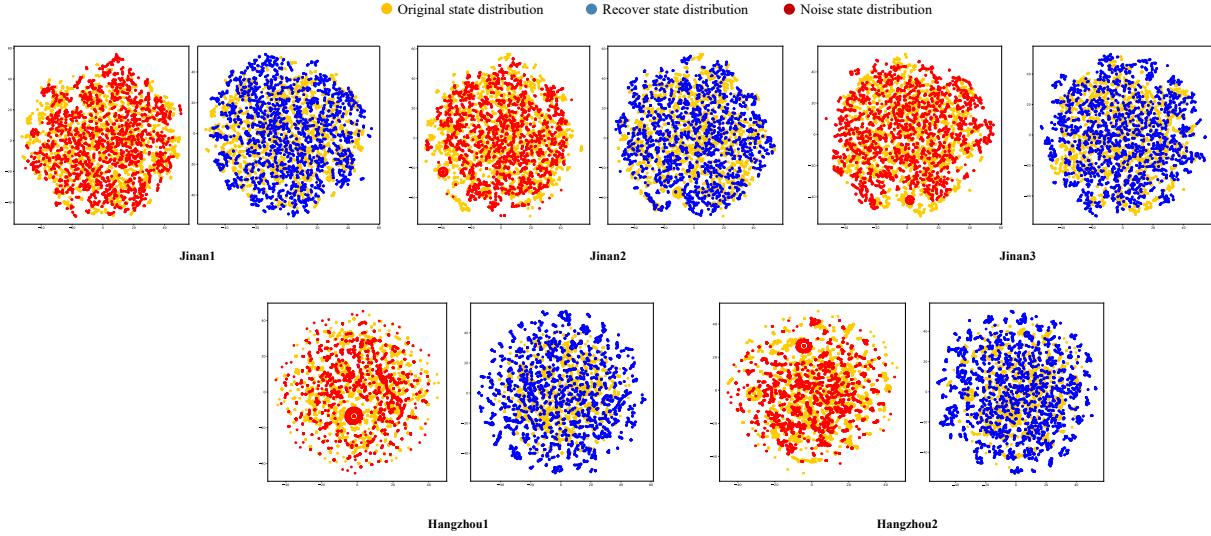


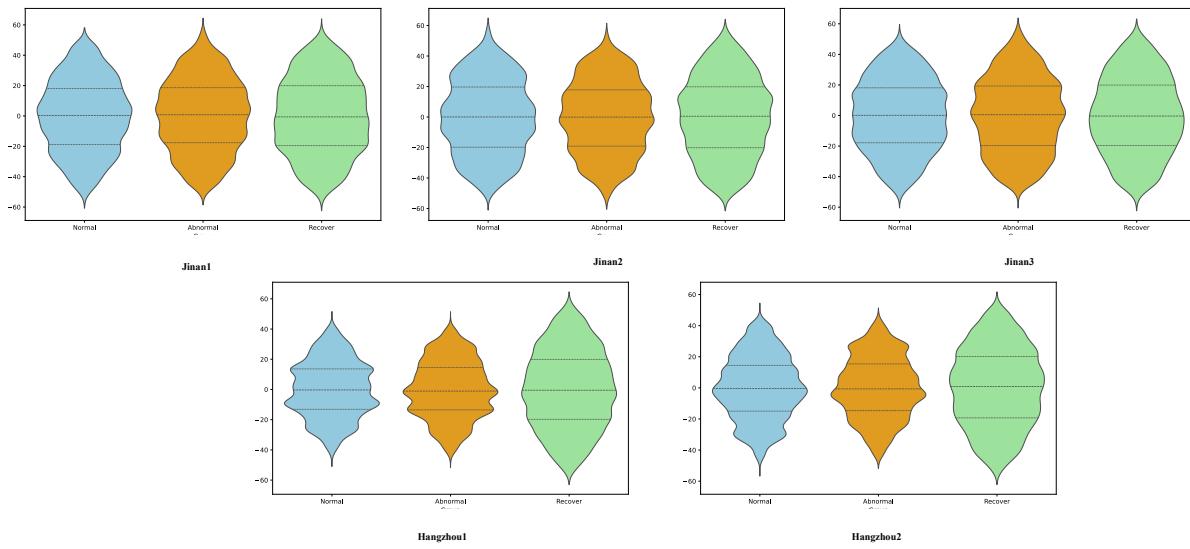
Figure 11. Performance of RobustLight based on Advanced-Colight transfer by JiNan1.

## D. Appendix: Robustness Analysis

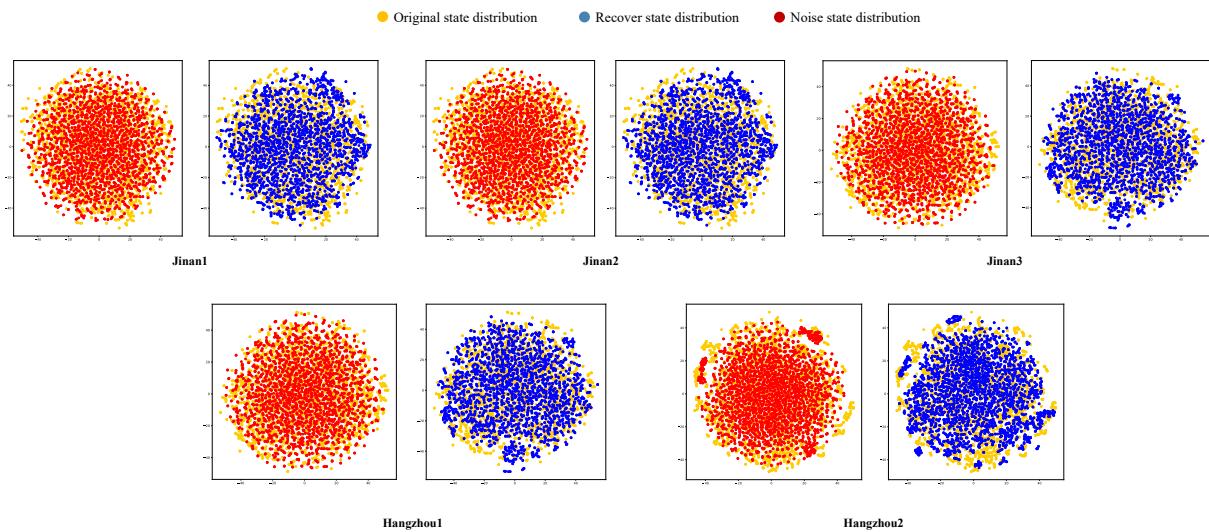
In this section, we provide detailed visualizations of the state t-SNE distributions and violin plots after dimensionality reduction under Gaussian noise attacks for different algorithms. As observed, for different algorithms, our RobustLight shows a significantly better recovery of the state distribution in most datasets, making it more closely resemble the original state distribution compared to the attacked state distribution.



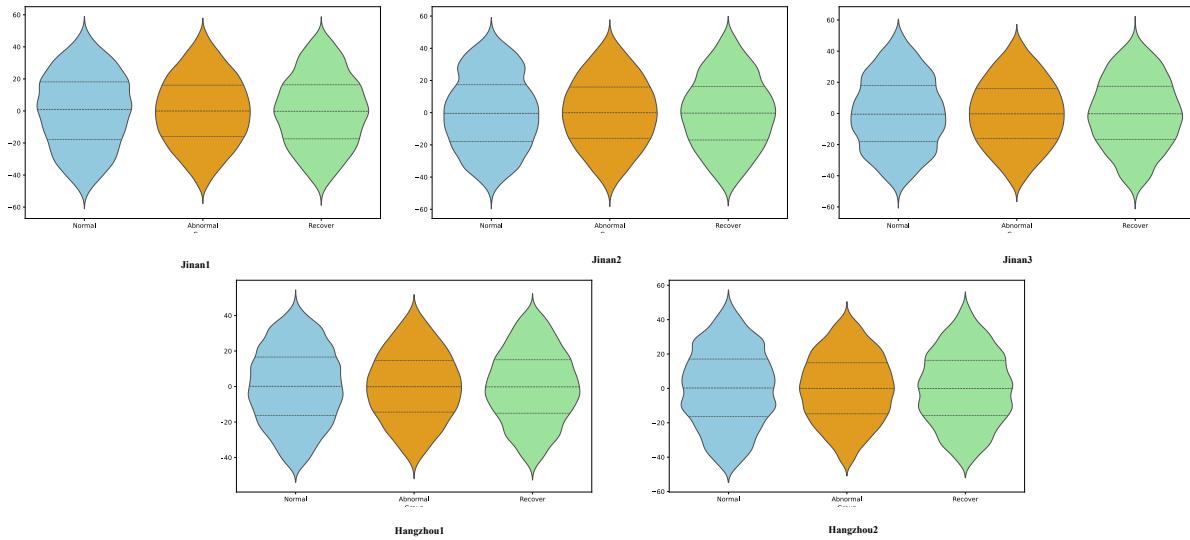
**Figure 12. State t-SNE visualization of RobustLight based on Maxpressure in Hangzhou and Jinan. Compared to the state distribution after the noise attack, the state distribution is much closer to the original state distribution after using our algorithm. RobustLight has the ability of state recovery.**



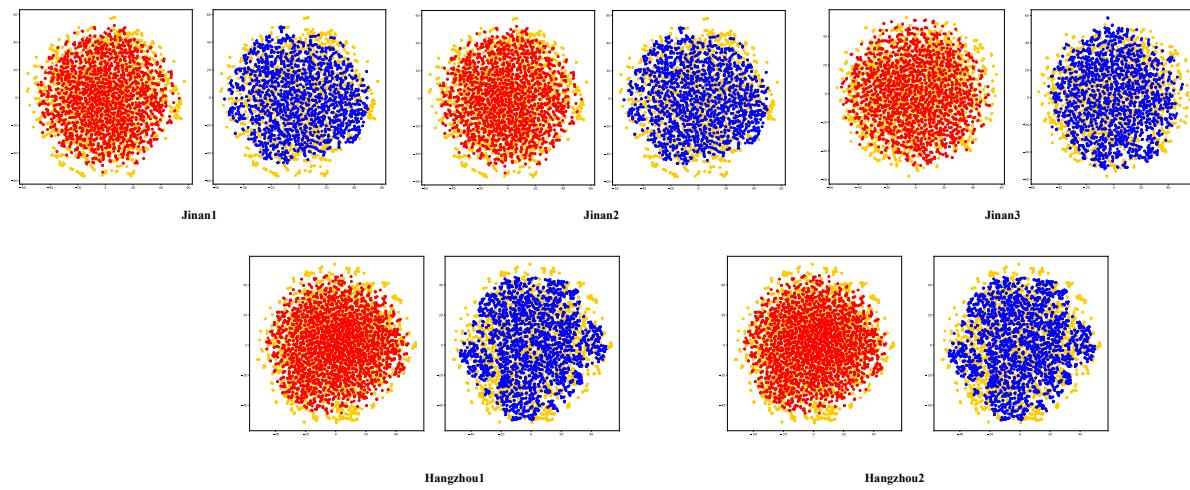
**Figure 13. State violin visualization of RobustLight based on Maxpressure in Hangzhou and Jinan.** Compared to the state distribution after the noise attack, the state distribution is much closer to the original state distribution after using our algorithm. RobustLight has the ability of state recovery.



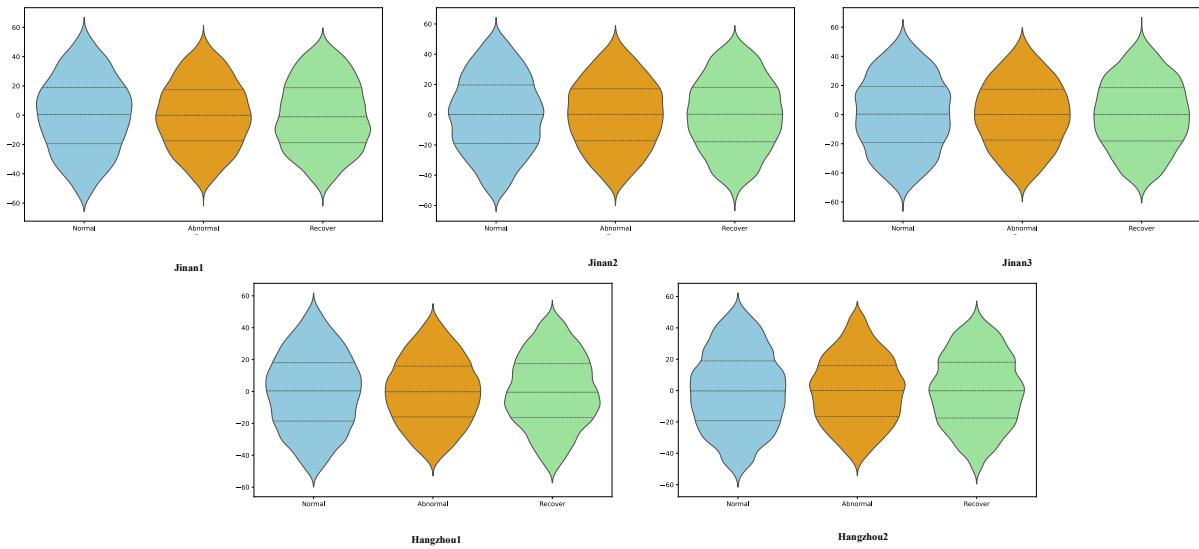
**Figure 14. State t-SNE visualization of RobustLight based on AdvancedMaxpressure in Hangzhou and Jinan.** Compared to the state distribution after the noise attack, the state distribution is much closer to the original state distribution after using our algorithm. RobustLight has the ability of state recovery.



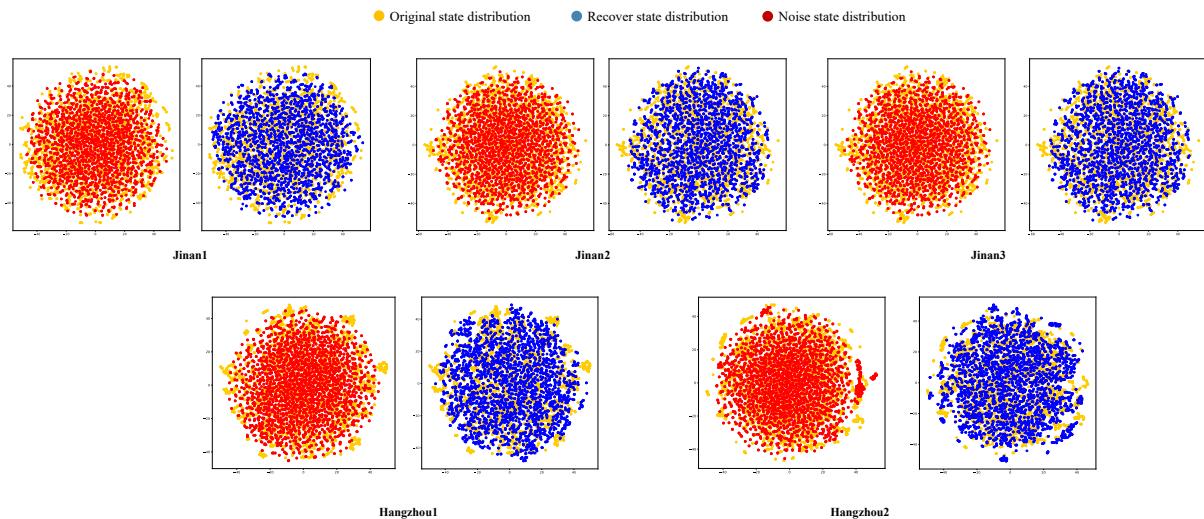
**Figure 15. State violin visualization of RobustLight based on AdvancedMaxpressure in HangZhou and JiNan.** Compared to the state distribution after the noise attack, the state distribution is much closer to the original state distribution after using our algorithm. RobustLight has the ability of state recovery.



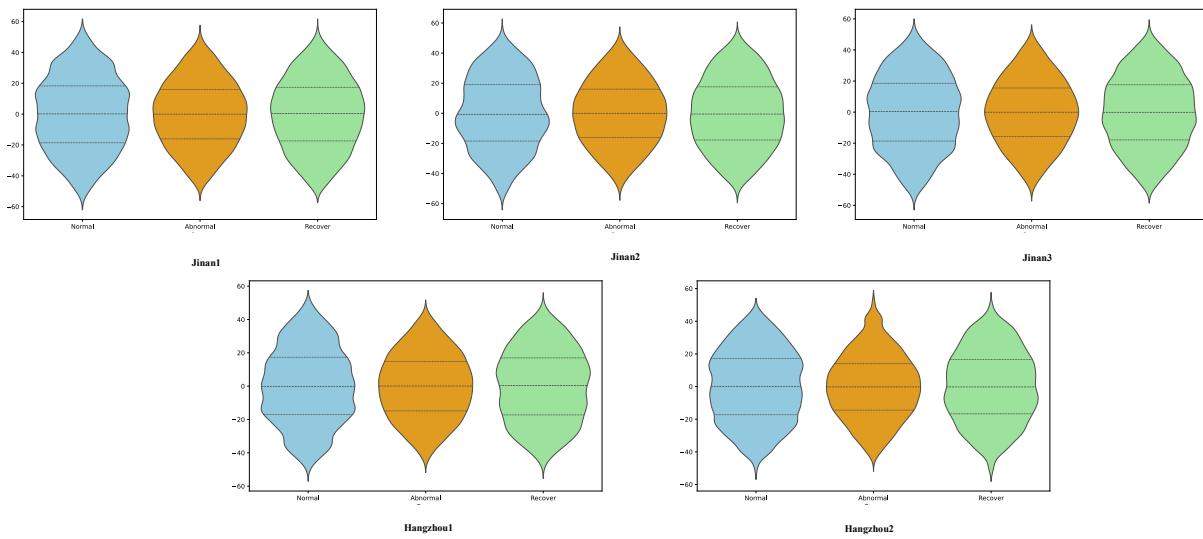
**Figure 16. State t-SNE visualization of RobustLight based on Colight in HangZhou and JiNan.** Compared to the state distribution after the noise attack, the state distribution is much closer to the original state distribution after using our algorithm. RobustLight has the ability of state recovery.



**Figure 17. State visualization of RobustLight based on Colight in HangZhou and JiNan.** Compared to the state distribution after the noise attack, the state distribution is much closer to the original state distribution after using our algorithm. RobustLight has the ability of state recovery.



**Figure 18. State t-SNE visualization of RobustLight based on AdvancedColight in HangZhou and JiNan.** Compared to the state distribution after the noise attack, the state distribution is much closer to the original state distribution after using our algorithm. RobustLight has the ability of state recovery.



**Figure 19. State visualization of RobustLight based on AdvancedColight in HangZhou and JiNan. Compared to the state distribution after the noise attack, the state distribution is much closer to the original state distribution after using our algorithm. RobustLight has the ability of state recovery.**