SITAR: Evaluating the Adversarial Robustness of Traffic Light Recognition in Level-4 Autonomous Driving

Bo Yang[1] and Jinqiu Yang[2]

***Abstract*— Traffic Light Recognition (TLR) is vital for Autonomous Driving Systems as it supplies critical information at intersections. Modern TLRs leverage camera and geolocation data, incorporating complex pre-(post)-processing steps and multiple deep learning (DL) models for detecting, recognizing, and tracking traffic lights. While the adversarial robustness of standalone DL models has been extensively studied, the robustness of a modern TLR system, i.e., a complex software component with code and DL models, is rarely studied and hence requires research efforts.**

In this work, we propose a novel testing framework (namely SITAR) targeting TLR modules from a representative Level-4 ADS, such as Baidu Apollo and Autoware. We design a novel adversarial attack loss function to evaluate and improve the adversarial robustness of modern TLR systems. We applied SITAR on Apollo TLR and compared our novel loss function with the state-of-the-art approaches that can effectively attack object detection and image recognition models. SITAR is shown to be effective and our novel loss function performs better than previous SOTAs with a 93% to 100% success rate with a maximum of five-step iteration and eight pixels per perturbation.

I. Introduction

Traffic Light Recognition (TLR) is critical in autonomous driving because the status (color) of TLs guides the behavior of the vehicles at intersections. Therefore, accurately recognizing the status of TLs is of paramount importance for ensuring an autonomous driving system makes accurate decisions. A typical TLR is composed of detecting the locations of the TLs within a picture, recognizing the status of the TLs, and tracking the status continuously. Modern TLRs employ deep learning models to detect and recognize the TLs.

However, classic DL models developed for object detection tasks face challenges when being applied to TLRs. First, TLs are small or medium-sized objects consisting of a limited number of pixels [15, 29]. Such small-sized objects are prone to low detection rates in traditional object detection models. Furthermore, TLRs need to output more detailed information than traditional object detection. TLRs need to report the color of the detected TL [4]. Previous studies attempted to tackle this issue by modifying the structure of the model [17], dividing the detection and recognition into two distinct stages [3], etc. Baidu Apollo [1], a leading level-4 autonomous driving, designs a robust TLR by incorporating various novel research findings and techniques. First, it utilizes the practices for small object detection, for example, detecting based on features at different resolutions. Second, it adopts the two-stage solution for detecting and recognizing TLs, which breaks the complex problem into simple subtasks. Last, engineering innovations are applied to enrich the DL models through pre- and post-processing operations, which will be further elaborated in Section II-A. Despite the efforts to improve reliability, prior studies still exposed its weaknesses [22, 28].

[1]Bo Yang is with the Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada bo_yang20@encs.concordia.ca

[2]Jinqiu Yang is with the Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada jinqiu.yang@concordia.ca

Therefore, in this work, we investigate the robustness of the Apollo TLR from a software perspective. In particular, we look for adversarial examples that make the entire TLR system fail. Prior research in the DL community focuses on the adversarial robustness of standalone DL models. Our study targets the entire Apollo TLR, a sophisticated software system propelled by DL models, which exhibits a distinctive trait of resilience to inaccurate detections which are considered as failures at the model's level. The nuanced engineering enhancements in Apollo TLR elucidated in Section II-A, afford a degree of tolerance to imprecise model outputs, challenging the conventional definition of failure and emphasizing the pivotal role of system-level robustness in real-world applications. To the best of our knowledge, we are the first to exploit the Apollo TLR using an adversarial attack. Our main contributions are as follows: (1) We formulated an attack goal aimed at exposing more realistic vulnerabilities within the Apollo TLR, (2) We designed a novel attack framework, SITAR, based on the design of the Apollo TLR to apply different adversarial attacks flexibly, (3) We designed a novel loss function to attack Apollo TLR effectively.

II. Background and Related Work

We introduce the background of TLR in the literature and industry-grade ADS, adversarial robustness on object detection, and other related work on testing ADS.

A. Traffic Light Detection and Recognition for Autonomous Driving

Traditional TLRs use heuristic-based feature extraction to detect and recognize TLs [7]. Lately, advanced object detection models with near real-time efficiency such as Faster RCNN [20] are applied to detect and recognize TLs in modern ADS. However, object detection techniques are known for the challenge of detecting small objects, e.g., TLs may only occupy 0.02% of the camera pictures. To improve the performance, [13, 8, 2] integrate prior knowledge to narrow down the detection range. For example, [2] estimate the TL location based on geographic information such as a map and the vehicle location. Apollo TLR adopts a similar strategy which we will discuss later in this section.

An Industry-Grade TLR pipeline. Apollo TLR has four major components, namely pre-processing, detection, post-processing, and recognition. Spiritually similar to prior works, Apollo TLR leverages HDMap and Localization modules, which achieve centimeter-level precision [14, 24], to accurately identify regions with high confidence containing TLs in camera images. We show the pipeline in Figure 1.

- **Pre-processing** crops several Regions-Of-Interests (ROIs) containing TLs with high confidence based on geographic information. In step ①, TLs in the HDMap are projected in the image, i.e., the purple bounding boxes (bboxes). Note that such projections could be inaccurate due to errors in geographic information. In step ②, Apollo calculates ROIs centered around the projections based on heuristics. It is common for one TL to be included in several *ROIs* since TLs are close to each other. Step ③ produces a list of *ROI*s to be processed by the **Detection** stage.

- **Detection** leverages Faster RCNN [20] to process each of the ROI. Given one ROI, the backbone CNN extracts features (④), regional proposal network (RPN) identifies regions (i.e., proposals) that may contain TLs (⑤), and the Fast-RCNN layer produces refined locations and layouts of TLs. In step ⑤, RPN calculates the objectness scores (*obj_cls_vec* in Figure 1) and regressions for a list of pre-defined bboxes at each position of the features extracted by the backbone CNN. For each pre-defined bbox, its objectness scores indicate the probability that the bbox contains a TL, and the regressions refine its location. Finally, RPN selects top-*n* refined bboxes with *objectness scores* above a threshold and size fitting a criteria as *proposals*. In step ⑥, Fast-RCNN refines the locations and classifies the layouts of the *proposals*. For each proposal, it calculates the layout classification scores (*layout_cls_vec*) which consists of four scores, indicating the layout of the *proposal* as one of the following: *Unknown*, *Vertical*, *Quadratic*, or *Horizontal*, where the *Unknown* refers to the invalid TLs. Similar to RPN, Fast-RCNN refines the locations of the *proposals* and picks the top-*n proposals* as *detections* (*ROI_detects_** in the illustration).

- **Post-processing** integrates the *detections* from different *ROI*s and selects a list of *detections* as the final detection results. In step ⑦, Apollo merges the *detections* from different ROIs (*merged_detects*). In step ⑧, it removes the duplicates by NMS (non-maximum suppression) and outputs distinctive *detections* (*nms_detects*). In step ⑨, Apollo matches the *detections* with the projections based on the distance and confidence scores. The matched *detections* (*matched_detects*) are going to be recognized in the **Recognition**, while the non-matched cases are discarded.

- **Recognition** recognizes the colours of the detected TLs. Based on its layout, Apollo applies the corresponding CNN recognizer to each detected TL to recognize its colour.

The recognition results of the matched *detections* are broadcast to a channel to be used by other modules in the ADS. Apollo also has a tracker, which simply stores the recognition of each frame, and provides the recognition of the previous frame under some failure situations.

B. Adversarial Robustness on Object Detection

We present the attack approaches working for the detector of Apollo TLR, Faster RCNN. [27] view the object detection task as a multi-targets classification problem and their attack objective is causing the model to misclassify all of the objects. [26] considers both regression and classification of object detection and attacks Faster RCNN by causing either inaccurate location or incorrect classification for the objects. [6] propose a unified attack approach that works for the three dominant types of object detectors. Our work introduced an attack framework named ***SITAR***, alongside a novel semantic attack designed specifically for Apollo TLR.

C. Robustness of Autonomous Driving Systems

ADS is a complex system involving different forms of data from diverse sensors (e.g., camera and LiDAR) and heavily depend on DL models to process the data [18]. Numerous testing approaches are proposed at module and system levels, targeting different aspects of ADS [23]. Our study specifically focuses on module-level testing within the perception module, contributing to the comprehensive testing landscape of ADS.

System-level testing. [12] exposed safety violations on Baidu Apollo by perturbing the environment. [21] added physical adversarial patterns to the road to fail the OpenPilot.

Perception module testing. The perception module leverages DL models to process the information from multiple sensors to perceive various elements like traffic lights and obstacles. [5] attacked MSF-based perception systems with 90%+ success rates using adversarial attacks. [28] exploit the camera weakness by irradiating the camera, which induces a physical color change in the image and causes recognition failure. [22] applies GPS spoofing to fail the localization of Apollo, which further fails the pre-processing of the Apollo TLR. Differently, we evaluate the robustness of the Apollo TLR from a software perspective.

Prediction and planning modules. [30] found that adversarial perturbed trajectory could lead ADS to make huge prediction errors and make unsafe driving decisions consequently. [25] exploited the conservative design of ADS to apply **D**enial **o**f **S**ervice (DoS) attack.

III. SITAR: A Framework to Evaluate the Adversarial Robustness of Apollo TLR

Attack goal. We formulate our attack goal considering both the detection and recognition in Figure 1. [9] suggests considering detections with $IoU(bbox_{det}, bbox_{gt}) \geq 0.5$[1] as true positives, and assessing overall performance through precision, recall, and AUC. Recent object detection works prefer average precision (AP), the averaged precision across a range of IoU thresholds (typically starting from 0.5), e.g., COCO evaluation metrics[2]. However, we argue that a minimum of

[1]Intersection over Union: $IoU(box1, box2) = \frac{area(box1 \cap box2)}{area(box1 \cup box2)}$

[2]https://cocodataset.org/#detection-eval

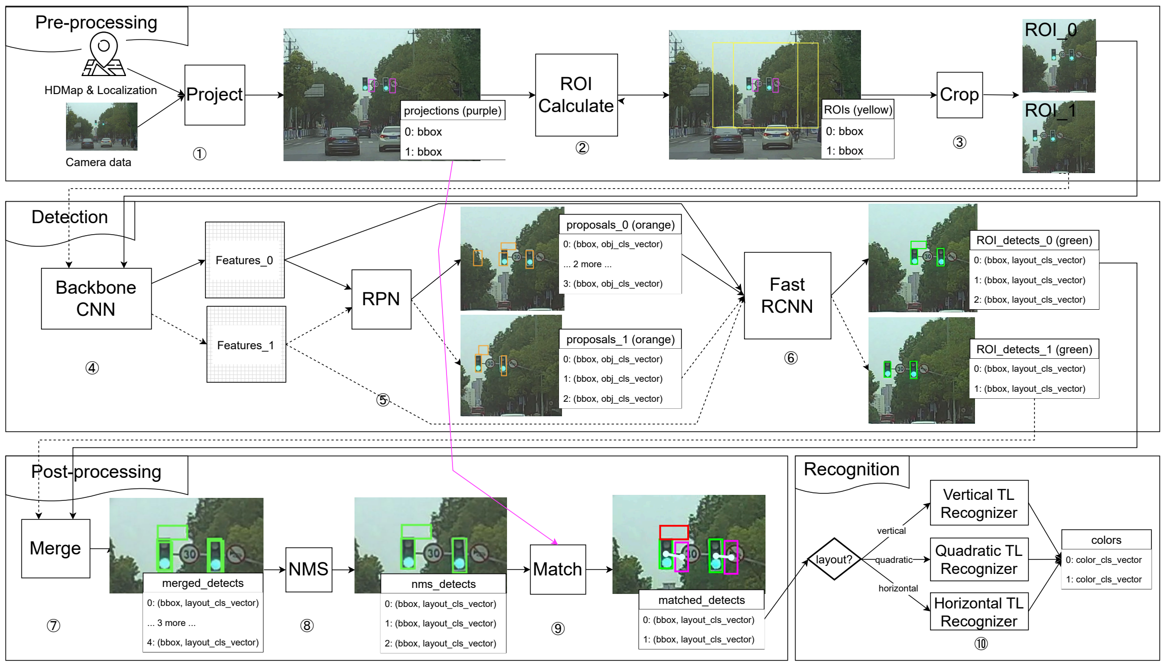

Fig. 1: Apollo TLR pipeline with an example. Boxes of the same colours in previous stages remain consistent, while those of different colours represent transformations. For example, the RPN output is marked in orange, and the Fast-RCNN output in green, meaning they are at different locations even if they are spatially close. An exception to colour usage occurs in the *Match* output, where red boxes refer to the same boxes in the previous stage which are discarded.

0.5 IoU is too strict for assessing the vulnerabilities in Apollo TLR. CNNs exhibit robust performance in image classification, even with occluded or small objects, suggesting that Apollo TLR can handle inaccurate detections. Additionally, TLs are small so minor inaccuracies can lead to low IoU with ground truth annotations. A wrong classification is a failure for recognition, but a correct classification does not mean success if the detected region does not include the TL. We formulate the attack goal as satisfying one of the following conditions: (1) The TL is not matched with any detection, (2) The matched detection does not overlap with the ground truth location of the TL, (3) Regardless of the detection result, the recognition is incorrect.

A. Adversarial attack design

As discussed in Section II-A, Apollo TLR detects TLs within multiple ROIs which may overlap. The adversarial patches mined for each ROI separately may counteract one another. This sets Apollo TLR apart from conventional object detection problems that typically operate on the entire image. We conceptualize the problem as a multi-object optimization challenge and introduce an innovative framework, ***SITAR***, to simultaneously attack the detection of each ROI, so the shared perturbation is optimized to impact both ROIs.

Formally, we denote the input image by $x \in [0, 255]^m$ and the TL targets in the x by $\mathcal{T} = \{t_1, t_2, ..., t_N\}$. Each TL target t_i comprises the ground truth location bb_{t_i} and the color c_{t_i}. We denote the detection and recognition results in all ROIs by $\mathcal{D} = \{d_1, d_2, ..., d_M\}$ and the *Match* (step ⑨ in Figure 1) result of t_i by $match(t_i, \mathcal{D})$, which is either a detection d_{t_i} or **N**ot **D**etected (ND). The location and the corresponding recognition color of d_{t_i} are denoted as $bb_{d_{t_i}}$ and $c_{d_{t_i}}$, respectively. We define

$$t_i \neq d_{t_i} \iff IoU(bbox_{t_i}, bbox_{d_{t_i}}) = 0 \lor c_{t_i} \neq c_{d_{t_i}} \quad (1)$$

The attack goal is expressed as

$$Min\ D(x, x'),\ \ s.t., \forall t_i \in \mathcal{T},\ match(x', t_i) \neq t\_i \quad (2)$$

where $x' \in [0, 255]$ is the adversarial image to be mined, $D(x, x')$ is the distance metric to measure the perturbation introduced by x'. Since formula (2) cannot be solved directly, we approximate it by

$$Min\ L(x', \mathcal{T}, TLR),\ s.t.,\ ||x' - x||_\infty \leq \epsilon \quad (3)$$

where $TLR(\cdot)$ is the Apollo TLR, $L(\cdot)$ is a loss function that leads to an approximation solution of formula (2) when it is minimized, and $||\cdot||_\infty$ is the L_∞ norm. Algorithm 1 shows the approach of ***SITAR*** to address the problem. Lines 1 to 3 set the initial states, and the while loop (lines 4 to 10) is the pivotal component for discovering the adversarial image. Specifically, line 1 introduces an initial perturbation to the input image. Line 2 executes Apollo TLR on the initially perturbed image, yielding the detection set $\mathcal{D}$, encompassing detections from each ROI. Line 3 initializes the iteration step to 0, and line 4 marks the commencement of the while loop, terminating when no TL is correctly detected or the maximum iteration is reached. In line 5, we use an instrument method *get_data* to get all of the produced data when running Apollo TLR on x'. In line 6, we use the loss function

$loss\_fn$ to construct the loss using the data produced in line 5. Subsequently, the adversarial image is updated through the optimization method $update$ (line 7). Finally, we update the detections in line 8 and increment the iteration number. It should be noted that the $init$, $loss\_fn$, and $update$ are changeable, offering flexibility in their application. In this work, we use the random perturbation as the $init$ method and Projected Gradient Descent (PGD) [16] as the optimization method $update(\cdot)$. We use our novel loss function and two baselines from previous works as the $loss\_fn$.

Algorithm 1: The algorithm of SITAR to generate adversarial attacks.

```
Input  : input image $x$, TL targets $\mathcal{T}$, max iteration $max\_iter$
Output: adversarial image $x'$
1  $x' \leftarrow init(x)$
2  $\mathcal{D} \leftarrow TLR(x')$
3  $iter \leftarrow 0$
4  while $\exists t_i \in \mathcal{T},\ match(t_i, \mathcal{D}) = t_i$ and $iter < max\_iter$ do
5      $data \leftarrow$ get_data($TLR, x'$) // Extract intermediate and output data
6      $loss \leftarrow loss\_fn(data)$
7      $x' \leftarrow update(x', loss)$
8      $\mathcal{D} \leftarrow TLR(x')$
9      $iter \leftarrow iter + 1$
10 end
11 return $x'$
```

1) Loss functions design: We designed an effective novel loss function to find adversarial examples that cause Apollo TLR to fail. First, we design a detection removal loss function, L_{RM}, to directly suppress the correct detections by turning the layout classification to *Unknown*, which Apollo TLR will discard. Recall that the Fast-RCNN takes the proposals from RPN, and generates detections by refining the locations of proposals and classifying their layouts. Then it selects the top-n detections based on their confidence scores. Denote the pre-selection detections by $\mathcal{D}^{pre} = \{d_1^{pre}, d_2^{pre}, ..., d_m^{pre}\}$, the label for *Unknown* layout by Unk. Then

$$L_{RM} = \sum_{i=1}^{N}\sum_{j=1}^{m} CELoss(d_j^{pre}.layout, Unk) * [IoU(t_i, d_j^{pre}) > 0] \quad (4)$$

where $CELoss$ is the cross-entropy loss, and $[\cdot]$ is the Iverson bracket. In other words, for all the detections that overlap with some TL targets, we try to minimize their confidence scores by increasing the score of *Unknown* class. Then, we design a detection fabrication loss function L_{FAB} which indirectly suppresses detections by fabricating detections that do not overlap with any TL. Recall the work mechanism of the Fast-RCNN (step ⑥) and Match (step ⑨) illustrated in section II-A: The Fast-RCNN outputs top-n confident detections and the Match matches the detections and TL targets based on the confidence scores and their distances. So fabricating false positives can indirectly suppress good detection by competing for the top-n positions, and matching with projections. So we define

$$L_{FAB} = \sum_{i=1}^{N}\sum_{j=1}^{m} CELoss(d_j^{pre}.layout, argmax(d_j^{pre}.layout \setminus Unk)) * [IoU(t_i, d_j^{pre}) = 0] \quad (5)$$

where the $argmax$ finds the layout with the highest confidence. In other words, we try to increase confidence scores for any detection not overlapping with any TL. Finally, we combine them as our novel loss function $L_{RM\&FAB} = L_{RM} + L_{FAB}$.

IV. Evaluation and results

In this section, we conduct a thorough evaluation of our framework and novel loss functions. As baseline comparisons, we apply the loss function proposed by [26] for Faster RCNN (the detection model in Apollo TLR) and the loss function from [11] for CNN (the recognition model in Apollo TLR). Both [26] and [11] utilize PGD as the optimization method and have demonstrated excellent results, ensuring a fair and consistent basis for comparison.

Experiment Setup. We set the perturbation budget $\epsilon = \{2, 4, 8, 16\}$ as in [11], iteration budget $max\_iter = \{5, 10\}$, and step size $\alpha = 3$ as in [26]. The motivation to use different settings is to explore the effects of our approach with varying perturbation and computation budgets. We conducted experiments on the 400 images that Apollo TLR performs best, comprising 200 with a resolution of 720 * 1,280 (S2TLD720) and 200 with a resolution of 1,080 * 1,920 (S2TLD1080), from the S2TLD dataset [29].

A. Attack Effectiveness

We evaluate the effectiveness of the loss functions in two ways: mAP (mean average precision) and success rate. mAP is a common object detection metric and success rate is defined based on our attack goal. We believe mAP is not suitable for the Apollo TLR due to the distinct fault-tolerant design, i.e., a TL could be recognized even if the detection is inaccurate. Therefore, we also calculated the attack success rates for each TL based on our attack goal. We use the results for the setting $\epsilon = 8, max\_iter = 5, \alpha = 3$ to conduct the main analysis, and will discuss other settings in section IV-B.

The quantitative results of ***mAP drop*** and ***success rates*** are presented in Table I and the adversarial examples mined by each loss function for a representative case are visualized in Figure 2. In our analysis, we distinguish TLs based on whether their ROIs overlap with other ROIs. We term TLs with overlapping ROIs as *OLP TLs* and those without overlap as *NO_OLP TLs* for brevity.

Overall, the different loss functions yield a mAP drop ranging from 87% to 98% (excluding Baseline 2 [11]) and a success rate spanning from 26% to 100%. Baseline 2 [11] only targets the recognizer of the Apollo TLR while others

TABLE I: Adversarial attack results for the setting $\epsilon = 8, max\_iter = 5, \alpha = 3$. mAP & drop shows the mAP of the detection on the adversarial images and the drop compared to the mAP of the clean images. OLP SUCC shows the number and percentage of successful attacks on the *OLP TLs*. NO_OLP SUCC, on the contrary, shows the results of *NO_OLP TLs*.

loss_fn	S2TLD720			S2TLD1080		
	mAP & drop	OLP SUCC	NO_OLP SUCC	mAP & drop	OLP SUCC	NO_OLP SUCC
Baseline 1 [26]	6% [93% ↓]	103 (68%)	212 (90%)	4% [96% ↓]	11 (92%)	280 (92%)
Baseline 2 [11]	99% [0% ↓]	40 (26%)	53 (23%)	100% [0% ↓]	7 (58%)	80 (26%)
L_{RM}	**1% [98% ↓]**	140 (93%)	227 (97%)	**2% [98% ↓]**	**12 (100%)**	299 (98%)
L_{FAB}	10% [89% ↓]	62 (41%)	121 (51%)	13% [87% ↓]	9 (75%)	189 (62%)
$L_{RM\&FAB}$	**1% [98% ↓]**	**141 (93%)**	**232 (99%)**	4% [96% ↓]	11 (92%)	**304 (100%)**

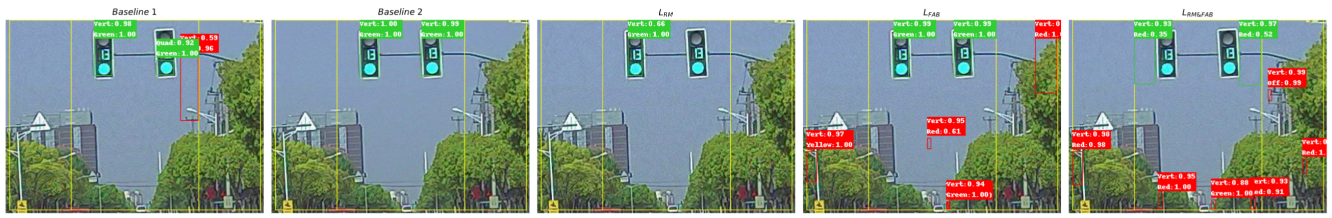

Fig. 2: Adversarial examples for a representative case. The yellow, green and red boxes represent the ROIs, matched and discarded detections, respectively. Each detection is annotated with layout, recognition and confidence scores.

target the detector, so it does not cause any mAP drop. Among the all of the loss functions, our $L_{RM\&FAB}$ is the most effective, while Baseline 2 is the least effective. Adversarial attacks, known for their efficacy in attacking image classification [11, 16], seem ineffective for Apollo TLR. The two-stage design significantly simplifies the recognition thus the recognizer is much more robust. Our $L_{RM\&FAB}$ is significantly better (25% higher) than Baseline 1 [26] in *OLP TLs* in S2TLD720 and slightly better in others (from 0 to 9% higher). As Figure 2 shows, Baseline 1 causes almost no effect on the left TL and a significant effect on the right TL, however, both TLs are recognized correctly and confidently. In contrast, the adversarial image found by our $L_{RM\&FAB}$ causes the detections not to overlap with any TL and the recognition of these detections are random guesses with low confidence scores (red:0.35 and red:0.52 respectively).

For the two sub-loss functions, L_{RM} closely aligns with $L_{RM\&FAB}$, while L_{FAB} is worse yet useful in certain challenging cases. For instance, in Figure 2, L_{RM} fails to suppress the left TL as its confidence (0.66) is above the threshold (0.1). The fabricated false positives by L_{FAB} are rejected by Apollo TLR despite their high confidence, as true positives have higher confidence scores. When combined, the fabricated false positives are selected by Apollo TLR because L_{RM} reduced the confidence of the true positives. As a result, $L_{RM\&FAB}$ successfully suppressed all TLs.

Regarding the effects of overlapping between ROIs, we observed that all approaches perform better on *OLP TLs* than *NO_OLP TLs* in S2TLD720, except for Baseline 2 [11] as it only affects the TL area and is unaffected by overlapping ROIs. However, for S2TLD1080, we cannot conclude which set of attacks performs better due to the small number of cases (only 12 TLs from 6 pictures) of *OLP TLs*, making the statistics less stable.

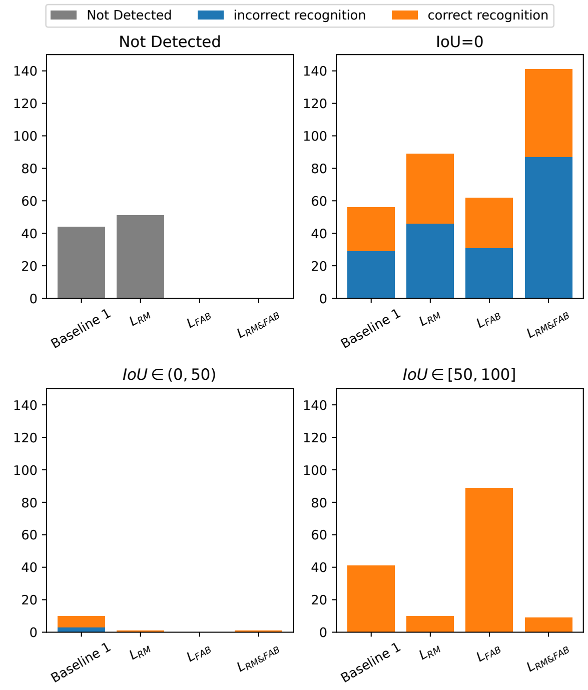

Fig. 3: Categorization of the detection and recognition of the adversarial examples mined by each loss function for OLP TLs in S2TLD720.

Our framework SITAR is effective in evaluating the adversarial robustness of a TLR module from a level-4 autonomous driving system. Our novel loss function surpasses the baselines, particularly when TLs have overlapping ROIs.

TABLE II: Success rates under different experiment settings.

loss_fn	S2TLD720		S2TLD1080	
	OLP	NO_OLP	OLP	NO_OLP
Baseline 1 [26]	37 (25%)	**79 (34%)**	3 (25%)	**172 (56%)**
Baseline 2 [11]	31 (21%)	41 (17%)	3 (25%)	61 (20%)
L_{RM}	**43 (28%)**	75 (32%)	**4 (33%)**	169 (55%)
L_{FAB}	5 (3%)	9 (4%)	0 (0%)	18 (6%)
$L_{RM\&FAB}$	41 (27%)	55 (23%)	3 (25%)	135 (44%)

(a) $\epsilon = 2, max\_iter = 5$

loss_fn	S2TLD720		S2TLD1080	
	OLP	NO_OLP	OLP	NO_OLP
Baseline 1 [26]	77 (51%)	193 (82%)	**8 (67%)**	268 (88%)
Baseline 2 [11]	41 (27%)	45 (19%)	5 (42%)	81 (27%)
L_{RM}	**114 (75%)**	**206 (88%)**	7 (58%)	269 (88%)
L_{FAB}	14 (9%)	47 (20%)	3 (25%)	89 (29%)
$L_{RM\&FAB}$	113 (75%)	**206 (88%)**	**8 (67%)**	**272 (89%)**

(b) $\epsilon = 4, max\_iter = 5$

loss_fn	S2TLD720		S2TLD1080	
	OLP	NO_OLP	OLP	NO_OLP
Baseline 1 [26]	123 (81%)	220 (94%)	10 (83%)	293 (96%)
Baseline 2 [11]	45 (30%)	54 (23%)	9 (75%)	83 (27%)
L_{RM}	**147 (97%)**	234 (100%)	**11 (92%)**	**304 (100%)**
L_{FAB}	83 (55%)	171 (73%)	9 (75%)	226 (74%)
$L_{RM\&FAB}$	146 (97%)	**235 (100%)**	**11 (92%)**	**304 (100%)**

(c) $\epsilon = 16, max\_iter = 5$

loss_fn	S2TLD720		S2TLD1080	
	OLP	NO_OLP	OLP	NO_OLP
Baseline 1 [26]	139 (92%)	232 (99%)	**12 (100%)**	303 (99%)
Baseline 2 [11]	40 (26%)	54 (23%)	7 (58%)	81 (27%)
L_{RM}	149 (99%)	**235 (100%)**	**12 (100%)**	303 (99%)
L_{FAB}	105 (70%)	176 (75%)	8 (67%)	253 (83%)
$L_{RM\&FAB}$	**151 (100%)**	**235 (100%)**	11 (92%)	**305 (100%)**

(d) $\epsilon = 8, max\_iter = 10$

B. Qualitative Analysis

In this section, we conduct a fine-grained analysis of the effects of each loss function on the Apollo TLR. To provide a more detailed view, we categorize the attack results into four finer-grained categories:

1) $Not\ Detected$. Refers to TLs that do not have a corresponding detection.
2) $IoU = 0$. Refers to the TLs that do not overlap with the matched detection.
3) $IoU \in (0, 50)$. Refers to TLs that inaccurately overlap with the matched detection (i.e., IoU $<$ 50%).
4) $IoU \in [50, 100]$. Refers to TLs that accurately overlap with the matched detection (i.e., IoU $>$ 50%).

The categories (1) and (2) correspond to successful attacks, meeting the attack goals defined in Section III. For categories (3) and (4), we further distinguish them by the recognition results. An incorrect recognition is considered a successful attack; otherwise, it is unsuccessful. Baseline 2 [11] is beyond the scope of the analysis in this section as it only targets the recognizers. Due to space limitations, we only show the results for *OLP TLs* in S2TLD720 in Figure 3, with the rest available in our replication package.

In general, Apollo TLR performs excellently in recognition when the detection contains the necessary information. As Figure 3 shows, all of the accurately detected TLs ($IoU \in [50, 100]$) and the majority (9 out of 12) of the inaccurately detected TLs ($IoU \in (0, 50)$) are recognized correctly. For example, the second TL of Baseline 1 in Figure 2 is partially detected but is correctly recognized as the lit area is contained in the detection. In contrast, when the detections do not contain sufficient information (i.e., $IoU = 0$), the recognitions are just random guesses as shown by the nearly 50:50 ratio between correct and incorrect recognitions.

Adversarial examples found by Baseline 1 mainly fall in $Not\ Detected$, $IoU = 0$ and $IoU \in [50, 100]$ categories, with a few in $IoU \in (0, 50)$. Baseline 1 increases the training loss of the Faster RCNN consisting of both localization and classification losses. As the localization loss increases, the detection gradually deviates from the ground truth location, leading to increasingly inaccurate detections. Differently, although the classification scores gradually change, the classification label is altered suddenly when the index of the highest score changes. This leads to three possible results: unconfident detections, incorrect layout detections and invalid detections (i.e., *Unknown* layout). Consequently, the adversarial examples fall in $Not\ Detected$ when the classification of true positives is turned to *Unknown* and in $IoU = 0$ when the localization fully deviates from the ground truth location. However, given the limited attack budgets, it may only find adversarial examples with inaccurate locations ($IoU \in (0, 50)$) or unconfident classifications ($IoU \in [50, 100]$) for some cases.

In contrast, most of the adversarial examples found by our $L_{RM\&FAB}$ are in the $IoU = 0$ category and few in others, contributing to its high success rates. $L_{RM\&FAB}$ consists of L_{RM} and L_{FAB}. L_{RM} spends all computations to misclassify the true positives into the invalid class by directly increasing the score for the *Unknown* class. Consequently, Apollo TLR detects no TLs ($Not\ Detected$) or only false positives ($IoU = 0$) on the adversarial examples found by L_{RM}. L_{FAB} fabricates false positives with high confidence by increasing the scores of valid classes. However, it does not affect the true positives which retain their original confidence. Consequently, Apollo TLR detects only false positives ($IoU = 0$) when false positives surpass true positives or original true positives ($IoU \in [50, 100]$) when true positives remain more confident than fabricated cases. When combined, the cases falling into $Not\ Detected$ caused by L_{RM} are counteracted by the fabricated false positives. Most true positives remaining in $IoU \in [50, 100]$ by L_{FAB} are removed by L_{RM}, and true positives failing to be removed by L_{RM} in $IoU \in [50, 100]$ are surpassed by the fabricated cases by L_{FAB} (such cases are not many but do exist).

Our novel loss function outperforms Baseline 1 [26] due to its direct and effective suppression of correct detections. The two sub-losses complement each other.

In addition to the settings used for the primary analysis, we conducted four additional experiments with different settings of perturbation budgets ϵ and computation budgets max_iter. The results are shown in Table II. Highermagnitude perturbations perform more powerful attacks but cause a more significant corruption of the images, and vice versa. When the max_iter is fixed at 5 and ϵ is variable, our novel loss function $L_{RM\&FAB}$ outperforms the two baselines under all settings (especially for *OLP TLs*), except the setting with $\epsilon = 2$, where our approach performs worse than Baseline 1 [26] on *NO_OLP TLs*. We attribute this performance difference to the sub-loss L_{FAB}, which is highly influenced by the ϵ. The success rates of L_{FAB} are poor with $\epsilon \leq 4$. For instance, the success rates drop below 10% in S2TLD720 with $\epsilon = 2$, indicating the difficulty of fabricating false positives with high confidence scores. In contrast, L_{RM} is less affected by ϵ, so when $\epsilon \leq 4$, the sub-loss L_{RM} performs equally well or better than $L_{RM\&FAB}$. Since the denseness of the false positives to be fabricated is much higher than that of the true positives to be suppressed, $L_{RM\&FAB}$ invests more computations in L_{FAB} instead of L_{RM}. Limiting the number of false positives fabricated in L_{FAB} is a potential improvement. When $\epsilon > 4$, the effect of L_{FAB} becomes positive. When we fix $\epsilon = 8$ and increase the max_iter to 10, we found that our $L_{RM\&FAB}$ successfully attacks Apollo TLR in almost all cases, with a single exception on *NO_OLP TLs* in S2TLD1080.

Our novel loss function $L_{RM\&FAB}$ outperforms baselines in most settings. L_{FAB} is highly sensitive to perturbation budgets and negatively affects the $L_{RM\&FAB}$ under very limited perturbation budgets.

V. Threats to validity

Internal Validity. Given that Apollo TLR has open-sourced its engineering code in C++ and the model with customized layers in TensorRT, our attempts to conduct adversarial experiments were hindered because TensorRT is an inference framework. We replicated the entire system using PyTorch. We utilized the LISA [10, 19] dataset containing more than 110k traffic lights to verify our replication. We only found around 30 discrepant cases when comparing our replication with Apollo TLR in Apollo v7.0. We attribute these errors to the different underlying implementations of PyTorch and TensorRT. We have released our replication[3] of Apollo TLR.
External Validity. Our work is tailored to assess the adversarial robustness of Apollo TLR which has a unique combination of deep learning models and engineering code. While we identified a similar system in Autoware TLR, the open-sourced model is only available in ONNX format, lacking the necessary model structure for performing adversarial attacks. We hope that future collaborations or the availability of the model structure could enhance the scope of our research. Additionally, we employed two traffic light datasets with different resolutions, underscoring that our approach extends beyond specific picture formats. Furthermore, our novel loss functions are designed based on Faster RCNN and can be applied to other perception works (e.g., traffic sign detection) with the same model architecture.

[3]https://doi.org/10.5281/zenodo.8390340

VI. Conclusions and Future Work

In this study, we conducted a comprehensive analysis of the Apollo TLR, a sophisticated system integrating multiple deep-learning models and engineering code. To evaluate the adversarial robustness of Apollo TLR, we developed an attack framework named SITAR, accompanied by a novel loss function $L_{RM\&FAB}$ designed to effectively and efficiently discover adversarial examples. We compared our strategy with two baselines from previous works which are shown to be effective for Faster RCNN (the object detector used by Apollo TLR) and CNN (the colour recognizer used by Apollo TLR). Subsequently, we performed a detailed comparative analysis of our loss function and those from previous works. In future work, we aim to enhance our adversarial attack strategies based on our findings to achieve improved performance and to design defence strategies for the Apollo TLR against such attacks.

References

[1] Baidu Apollo. *https://en.apollo.auto/*.

[2] Dan Barnes, Will Maddern, and Ingmar Posner. "Exploiting 3D semantic scene priors for online traffic light interpretation". In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. 2015, pp. 573–578.

[3] Karsten Behrendt, Libor Novak, and Rami Botros. "A deep learning approach to traffic lights: Detection, tracking, and classification". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1370–1377.

[4] Azzedine Boukerche and Zhijun Hou. "Object Detection Using Deep Learning Methods in Traffic Scenarios". In: *ACM Comput. Surv.* 54.2 (2021).

[5] Yulong Cao, Ningfei Wang, Chaowei Xiao, et al. "Invisible for both Camera and LiDAR: Security of Multi-Sensor Fusion based Perception in Autonomous Driving Under Physical-World Attacks". In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 176–194.

[6] Ka-Ho Chow, Ling Liu, Margaret Loper, et al. "Adversarial Objectness Gradient Attacks in Real-time Object Detection Systems". In: *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. 2020, pp. 263–272.

[7] Moises Diaz, Pietro Cerri, Giuseppe Pirlo, et al. "A survey on traffic light detection". In: *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops: ICIAP 2015 International Workshops, BioFor, CTMR, RHEUMA, ISCA, MADiMa, SBMI, and QoEM, Genoa, Italy, September 7-8, 2015, Proceedings 18*. Springer. 2015, pp. 201–208.

[8] Nathaniel Fairfield and Chris Urmson. "Traffic light mapping and detection". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 5421–5426.

[9] Morten Bornø Jensen, Mark Philip Philipsen, Andreas Møgelmose, et al. “Vision for Looking at Traffic Lights: Issues, Survey, and Perspectives”. In: *IEEE Transactions on Intelligent Transportation Systems* 17.7 (2016), pp. 1800–1815.

[10] Morten Bornø Jensen et al. “Vision for looking at traffic lights: Issues, survey, and perspectives”. In: *IEEE Transactions on Intelligent Transportation Systems* 17.7 (2016), pp. 1800–1815.

[11] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. “Adversarial examples in the physical world”. In: *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[12] Guanpeng Li, Yiran Li, Saurabh Jha, et al. “AV-FUZZER: Finding Safety Violations in Autonomous Driving Systems”. In: *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. 2020, pp. 25–36.

[13] F. Lindner, U. Kressel, and S. Kaelberer. “Robust recognition of traffic signals”. In: *IEEE Intelligent Vehicles Symposium, 2004*. 2004, pp. 49–53.

[14] Rong Liu, Jinling Wang, and Bingqi Zhang. “High Definition Map for Automated Driving: Overview and Analysis”. In: *The Journal of Navigation* 73.2 (2020), pp. 324–341.

[15] Yang Liu et al. “A survey and performance evaluation of deep learning methods for small object detection”. In: *Expert Systems with Applications* 172 (2021), p. 114602.

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, et al. “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *International Conference on Learning Representations*. 2018.

[17] Julian Müller and Klaus Dietmayer. “Detecting traffic lights by single shot detection”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 266–273.

[18] Zi Peng et al. “A First Look at the Integration of Machine Learning Models in Complex Autonomous Driving Systems: A Case Study on Apollo”. In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020, pp. 1240–1250.

[19] Mark Philip Philipsen, Morten Bornø Jensen, Andreas Møgelmose, et al. “Traffic light detection: A learning algorithm and evaluations on challenging dataset”. In: *intelligent transportation systems (ITSC), 2015 IEEE 18th international conference on*. IEEE. 2015, pp. 2341–2345.

[20] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).

[21] Takami Sato, Junjie Shen, Ningfei Wang, et al. “Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under Physical-World Attack”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 3309–3326.

[22] Kanglan Tang, Junjie Shen, and Qi Alfred Chen. “Fooling perception via location: a case of region-of-interest attacks on traffic light detection in autonomous driving”. In: *NDSS Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*. 2021.

[23] Shuncheng Tang, Zhenya Zhang, Yi Zhang, et al. “A Survey on Automated Driving System Testing: Landscapes and Trends”. In: *ACM Trans. Softw. Eng. Methodol.* 32.5 (2023).

[24] Guowei Wan et al. “Robust and Precise Vehicle Localization Based on Multi-Sensor Fusion in Diverse City Scenes”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 4670–4677.

[25] Ziwen Wan, Junjie Shen, Jalen Chuang, et al. “Too Afraid to Drive: Systematic Discovery of Semantic DoS Vulnerability in Autonomous Driving Planning under Physical-World Attacks”. In: *Network and Distributed Systems Security Symposium*. 2022.

[26] Yutong Wang et al. “Adversarial attacks on Faster R-CNN object detector”. In: *Neurocomputing* 382 (2020), pp. 87–95.

[27] Cihang Xie, Jianyu Wang, Zhishuai Zhang, et al. “Adversarial examples for semantic segmentation and object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1369–1378.

[28] Chen Yan et al. “Rolling colors: Adversarial laser exploits against traffic light recognition”. In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 1957–1974.

[29] Xue Yang, Junchi Yan, Wenlong Liao, et al. “Scrdet++: Detecting small, cluttered and rotated objects via instance-level feature denoising and rotation loss smoothing”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[30] Qingzhao Zhang, Shengtuo Hu, Jiachen Sun, et al. “On Adversarial Robustness of Trajectory Prediction for Autonomous Vehicles”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 15159–15168.