CATI – An Open-Source Framework to Evaluate Attacks on Cameras for Autonomous Vehicles

Michael Kühr, Maximilian Mittmann, Mohammad Hamad, Sebastian Steinhorst
TUM School of Computation, Information and Technology, Technical University of Munich, Germany
{firstname.lastname}@tum.de

***Abstract*—Cameras and the subsequently applied perception algorithms are essential for the safe operation of autonomous vehicles. While many attacks on this processing pipeline are known, their impact is often evaluated on generic, non-automotive Machine Learning models. Although such models are still widely used in research, a realistic attack evaluation is not possible with them. A central problem for security researchers is the lack of realistic open-source Machine Learning models that represent autonomous driving functionalities. In our work, we propose CATI, an open-source framework to evaluate attacks on cameras in autonomous vehicles. Besides two trained models for automotive object detection and traffic sign detection, it is designed to be modular to include further models for other tasks. The two integrated models are specifically trained versions of the established object detection model YOLO. We show different attacks that successfully trick commonly available implementations of YOLO but not our trained models. Additionally, we highlight how the model robustness benefits in challenging real-world scenarios.**

***Index Terms*—Autonomous Vehicle, Security, ML Model Resilience**

I. INTRODUCTION

Cameras are among the most important sensors in autonomous vehicles [1] and are the input for Machine Learning (ML) models that can perform tasks like object detection or traffic sign detection [2]. Such functionality not only enhances passenger comfort but also plays an important role in the safety of people inside and outside the vehicle. Malfunctions or attacks on cameras can limit their operation and, therefore, result in severe consequences.

Different attacks against cameras and the subsequently applied perception algorithms in autonomous vehicles are well-researched [3], [4], [5], [6], [7], [8]. Nevertheless, many attacks [3], [4], [5], [9], [10], [11], [12], [13] evaluate their attack success on generic ML models that were trained on a multitude of different object types [14]. Although these commonly available trained models can achieve high accuracy for generic object types, they contain types that are not common for the specific tasks of an autonomous vehicle. As a result, automotive object types, such as "car," "truck," or "person," are misdetected into non-automotive classes, such as "bottle" [11], "cake" [12], or "donut" [4]. Such misclassifications are not realistic and can lead to wrong conclusions. While we acknowledge the attack principles discussed in these research studies, we emphasize that *attacks on autonomous driving algorithms using camera images must be evaluated under more realistic conditions.* However, to the best of our knowledge, there is no open-source framework available that enables security researchers to assess their attacks or defenses in a realistic setup where a properly trained ML model is used for evaluation.

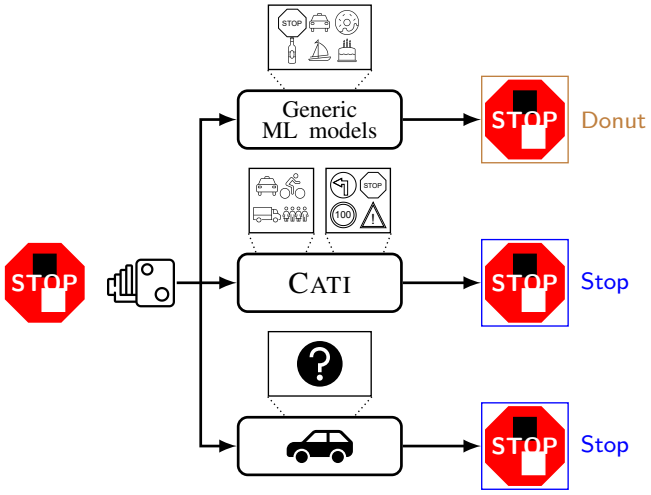


Fig. 1: Attacks on cameras can cause malicious behavior on ML models used by researchers. Nevertheless, the impact on autonomous vehicles might be less severe since they use proprietary specialized models. CATI aims to bridge this gap by providing an open-source framework with more realistically trained ML models to evaluate attacks on cameras.

Although modifications of model architectures can lead to better performance for specific tasks [15], [16], adapting such models to changing conditions or application use cases can require significant effort [17]. Another method to improve the performance of ML models is training and fine-tuning existing models on a limited set of object types [18]. However, existing implementations of this approach have key limitations, such as excluding relevant objects (e.g., bicycles, buses, and motorcycles), focusing only on object detection without traffic signs, and not open-sourcing their code [19].

To overcome these limitations, we introduce CATI (**C**ommon **A**utomotive **T**asks on **I**mages), a framework that contains not only ML models performing representative computer vision tasks of real autonomous vehicles but also allows the inclusion of other customized ML models. With this framework, we want to provide security researchers with more realistic ML models than the commonly available models.

Therefore, we train a state-of-the-art version of You Only Look Once (YOLO) [20] for two common automotive tasks, namely: (i) automotive object detection, and (ii) traffic sign detection. Additionally, we provide an interface for researchers to integrate further models, such as lane detection [21]. We evaluate our framework on existing datasets containing automotive object types and traffic signs and demonstrate that CATI outperforms commonly available implementations of YOLO [22]. Additionally, we show the positive impact of customized training on different attacks and challenging real-world scenarios. We make the following contributions:

- **Providing trained ML models for common automotive tasks on images.** As shown in Figure 1, we provide automotive object detection and traffic sign detection that security researchers can use to evaluate their attack more realistically than generic ML models.
- **Enabling researchers to integrate further customized models.** With our modular framework CATI, security researchers have the possibility to include further customized models covering their respective research areas.
- **Testing both the positive security and robustness impact of specifically-trained ML models.** We analyze different attacks by two attack mechanisms as well as challenging real-world images and show that our trained models can provide higher accuracy.

We provide CATI as an open-source framework, including the two trained models on automotive object detection and traffic sign detection: https://github.com/tum-esi/CATI.

II. BACKGROUND

A variety of attacks is known in research that target perception of cameras in automotive applications, such as traffic sign detection [23], [24], [6], [25], [26], [3] and object detection [9], [27], [28], [29]. These attacks mostly use publicly available trained ML models and datasets for evaluation.

In practice, car manufacturers and suppliers use custom, closed-source models [30] with only a few exceptions [31]. These custom models are often trained on specific datasets for certain applications [32]. By contrast, academia relies on established computer vision models for object detection tasks, such as YOLO [20] or Faster R-CNN [33]. Commonly available implementations of object detectors such as YOLO [22] or Faster R-CNN [34] are trained on the Common Objects in Context (COCO) dataset [14]. This dataset contains *"91 objects types that would be easily recognizable by a 4 year old."* [14]. Among these 91 objects, there are types that can be found in automotive applications, such as "car," "person," or "motorcycle," but it also contains types that do not relate to the automotive domain, such as "frisbee," "donut," or "pizza."

Various datasets are available for automotive object detectors, each containing different object types.

Automotive object detection: Besides the mentioned COCO dataset [14] that contains 328,000 images with 2.5 million objects of all 91 object types, other datasets offer automotive object types. The PASCAL Visual Object Classes (VOC) dataset [35] contains 20 object types that are similarly

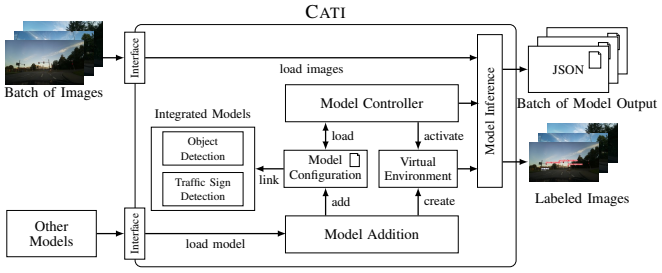


Fig. 2: High-level architecture of CATI. The model addition and model controller modules support the integration of other ML models.

generic as the COCO dataset, but some important object types, such as "traffic light," are missing. A different approach based on videos instead of images is covered by the BDD100K dataset, which consists of 100,000 videos recorded with a resolution of 1280×720 [36]. In addition to automotive object detection, it can also be used for other tasks, such as lane detection or semantic segmentation. Lastly, KITTI contains not only camera images recorded with 1392×512 pixels but also LiDAR data from 39.2km of public street [37].

Traffic sign detection: The German Traffic Sign Detection Benchmark (GTSDB) dataset consists of 900 images containing 1206 German traffic signs with an image resolution of 1360×800 pixels [38]. Similarly, the Tsinghua-Tencent 100K (TT100K) dataset consists of 100,000 images containing 30,000 Chinese traffic signs with an image resolution of 2048×2048 pixels [39]. As a more comprehensive dataset, the Mapillary Traffic Sign Dataset consists of 52,000 fully annotated images with 400 classes of international traffic signs with varying image resolutions from 1MP to 16MP [40]. Lastly, the LISA traffic sign dataset contains 49 different US traffic signs distributed over 6610 images with 7855 signs in total, captured with varying resolutions up to 1024×522 [41].

Datasets that contain many objects or images from many different high-resolution cameras, such as COCO or the Mapillary Traffic Sign Dataset, can, therefore, significantly improve the inference accuracy of ML models [42].

III. CATI

We introduce CATI as a modular open-source framework to enable security researchers to evaluate their attack on a more realistic automotive setup.

A. Architecture

CATI is a modular tool that includes trained ML models for automotive object detection and traffic sign detection. Its architecture supports the integration of additional models to represent other automotive tasks. Figure 2 depicts the modular architecture, which requires a trained model and an input image for testing. The output consists of a human- and machine-readable file containing detected objects and their details and, optionally, an image with the detection results.

The internal architecture of CATI consists of the following main components:

- **Model Addition:** The main task of this component is the creation of virtual environments containing the specific external requirements for a new model. This allows a separation of possible conflicting dependencies for certain models. Additionally, it adds the new model to the configuration.
- **Integrated Models:** As a default configuration, we deploy CATI with an automotive object detection model and a traffic sign detection model. These models can be directly used without considering the interface.
- **Model Controller:** This component parses the desired model from the user input and loads the correct virtual environment with all dependencies needed.
- **Model Inference:** This component executes the selected models on the given batch of images. It outputs one JavaScript Object Notation (JSON) file for each image with all labels and, if selected, the labeled image. It will generate one labeled image for each selected model.

Since test images by security researchers can differ significantly in size and their testing environments can also differ, CATI can work on arbitrary image sizes and aspect ratios and works platform-independent on both standard processors and Graphics Processing Units (GPUs). This allows a fast processing time on GPUs and tests on potentially resource-constrained test stands.

B. Automovie Object Detection

As the first integrated ML model, we deploy automotive object detection, a fundamental feature of autonomous vehicles. As an underlying model architecture, we select YOLOv10 in its medium version [22], representing a state-of-the-art object detector. While commonly available versions are trained on the generic object types of the COCO dataset, we train it specifically on automotive object classes that can typically be found in automotive scenarios: "Person," "Car," "Bicycle," "Motorcycle," "Bus," "Truck," and "Traffic Light." With these object types, we result in a subset of COCO with 70,922 training images and 3001 validation images, including at least one instance of the previously mentioned classes.

We train the model with the parameters and information given in Table I and enable the "mosaic mode" for the first ten epochs, combining multiple images into a larger one. If not otherwise mentioned, we use the default parameters for YOLOv10.

The training results are presented in Table II. The automotive object detection model training required nearly the specified maximum amount of epochs to reach good precision and recall. A typical quality metric is the mean average precision (mAP) with an intersection of union threshold of 50%. Further comparison with YOLOv10m, trained on the COCO dataset, is available in Table III as part of the evaluation.

C. Traffic Sign Detection

The second integrated ML model of CATI is traffic sign detection, representing another functionality of autonomous vehicles. Although the object types differ significantly from the automotive object detection, the task of detecting and classifying objects is similar. Therefore, we use the same medium-sized YOLOv10 model but train it with the Mapillary Traffic Sign Dataset [40]. This dataset contains a large number of international traffic signs captured with different cameras, making it ideal for CATI to handle images of different image sensors. We select 11,054 training and 1627 validation images that show German traffic signs for training, allowing us to compare our model with other established datasets, such as GTSDB [38]. An overview of the 51 traffic signs CATI can detect is available in Appendix A. We also trained using the GTSDB dataset [38], but its limited dataset size and smaller image resolutions limit its application in CATI.

As shown in Table I, we rescale the images to a higher resolution than automotive object detection since traffic signs can be smaller than other object types. With the same rescaling as for the automotive object detection, we observed a bad detection performance, especially for smaller traffic signs in the distance. The exact amount of 1280 pixels is given by the smallest available images in the Mapillary dataset [40]. Except for the dataset, we train the model with the same parameters as the automotive object detection. The training stopped after fewer training epochs compared to the automotive object detection, as shown in Table II. We achieve good results in these common testing metrics but refer to Table IV as part of the evaluation for further performance comparison.

TABLE I: Training parameters for the automotive object detection and the traffic sign detection.

	Automotive Object Detection	**Traffic Sign Detection**
Model architecture	YOLOv10m [43]	YOLOv10m [43]
Dataset	COCO* [14]	Mapillary [40]
Training epochs	300	300
Patience	25	25
Image rescaling	640px	1280px
Batch size	12	12

* Only a subset of COCO is used

TABLE II: Training results of the ML for the automotive object detection and the traffic sign detection.

	Automotive Object Detection	**Traffic Sign Detection**
Training epochs	292	174
mAP50	70.4%	70.3%
Precision	76.1%	72.6%
Recall	63.4%	62.7%

IV. EVALUATION

We perform all evaluations with an Intel Xeon Gold 5220R and an NVIDIA RTX A5000, though we emphasize that CATI works both on CPUs and GPUs. In the first step, we compare the inference quality and latency of the models integrated in CATI with existing standard models that detect the same object types in Section IV-A. Later, we highlight the positive security

TABLE III: Comparison of standard YOLOv10m and our automotive object detection model. For selected automotive object types our model shows a better performance than the standard model.

	YOLOv10m	YOLOv10 in CATI
Precision	57.0%	**61.8%**
Recall	47.3%	**51.8%**
mAP50	47.3%	**49.8%**
mAP75	41.0%	**41.8%**
Latency	39.4ms	39.1ms

TABLE IV: Comparison of Faster R-CNN, specifically trained on the GTSDB dataset [44] and our YOLOv10-based traffic sign detection in CATI.

Model Backbone	Faster R-CNN ResNet-50 [44]	YOLOv10 in CATI –
Precision	–	89.2%
Recall	–	86.0%
mAP50	**83.1%**	70.7%
mAP75	67.3%	**68.8%**
Latency	–	18.4ms

and robustness impact in Sections IV-B and IV-C. For all tests, we set the detection threshold to 0.5.

A. Comparison with Standard Models

Automotive Object Detection: Since our model is also trained on a subset of the COCO dataset, we can use a similar subset of its validation dataset to compare our model against the standard medium-sized YOLOv10 model. We select 3000 images containing at least one object of the selected automotive object types shown in Section III-B.

As depicted in Table III, our automotive object detection model in CATI demonstrates a better overall performance compared to the standard medium-sized YOLOv10 model with no negative impact on the latency. This shows in a direct comparison that our fine-tuned model benefits from the reduced number of classes.

Traffic Sign Detection: For the performance evaluation of the traffic sign detection, we cannot compare our model in CATI with the default YOLO because of different object types. Therefore, we use the complete GTSDB dataset with all 900 images since it represents an unknown input for our model. Since this dataset also contains German traffic signs, most traffic sign classes overlap with our selected ones of the Mapillary dataset. We compare our performance with a Faster R-CNN [33] implementation trained on GTSDB [44] and consider their result with a ResNet-50 backbone since it is also used in commonly available trained models [34].

Table IV shows that our deployed version of YOLOv10 in CATI shows a better performance than the Faster R-CNN model with the ResNet-50 backbone [44] that is available in common implementations [34] when considering the stricter mAP75 metric. Although the mAP50 metric shows better results for Faster R-CNN, we emphasize that it is typically slower than YOLOv10 [45], [46].

(a) YOLOv10m detects the adversarial sticker as a "bench."

(b) CATI detects only correct object types.

Fig. 3: Physical adversarial example from Apricot dataset [47], tricking the commonly available YOLOv10m but not YOLOv10m in CATI.

(a) YOLOv10m

(b) CATI automotive object detection

(c) CATI traffic sign detection

Fig. 4: Misclassification of an image from Apricot [47], tricking the commonly available YOLOv10m but not CATI.

B. Security Impact

We use 50 example images from the Apricot dataset [47] that contain automotive object types to evaluate the impact of physical adversarial stickers designed to attack image-based object detectors. Each image has a resolution of 12MP. As a comparison model, we use YOLOv10m, trained on the COCO dataset. Our automotive object detection in CATI uses the same model architecture and dataset but was trained only on a relevant subset of object types.

Figure 3 shows a physical adversarial example from the Apricot dataset. While the commonly available version of YOLOv10m is tricked and classifies the adversarial sticker as a "bench," our model in CATI does not get tricked. Another example is Figure 4, where CATI classifies the bus correctly and even detects the stop sign. By contrast, YOLOv10m misclassifies the bus as a "train." This shows that YOLOv10m trained on COCO dataset can be attacked with physical adversarial samples, leading to the appearance of non-existing objects or misidentification. While we do not claim to be secure against such attacks, YOLOv10m in CATI shows no false-positive in all tested images of the Apricot dataset [47].

As another attack example, we capture real-world images of a blinding attack [48]. Figure 5 shows the region of interest

(a) Original YOLOv10m does not detect the sign

(b) Traffic sign detection in CATI detects the sign

Fig. 5: Example of a blinding attack with a stop sign.

of this blinding attack where the camera was blinded with a red laser to hide the traffic sign. Although the COCO dataset contains a class "stop sign" [14], YOLOv10m does not detect it. By contrast, our trained model of the traffic sign detection inside CATI detects it with high confidence. Additional tests with the automotive object detection in CATI show no false-positive.

C. Robustness Impact

In addition to the security impact of our trained models, we highlight the impact on robust detection of automotive objects and traffic signs in challenging scenarios without malicious intent. We use images from the GTSDB dataset [38], which were initially designed to challenge traffic sign detection models but also contain typical street scenes. We compare both models deployed in CATI with the commonly available YOLOv10m trained on the COCO dataset.

Figure 6 shows two example images from the GTSDB dataset. While Figure 6a shows a misdetection of the traffic sign by YOLOv10m as a "stop sign," CATI detects the traffic sign correctly in Figure 6b. This is especially critical since a pure filtering of non-automotive object types would still lead to a misdetected sign. Figures 6c and 6d show a challenging scenario with large overexposed areas in the image. YOLOv10m detects the traffic sign as a "frisbee," while our traffic sign detection model detects the correct object type. An example of the object detection use-case is available in Figure 7. While CATI detects small cars even in challenging environmental conditions, YOLOv10m detects only larger objects in closer proximity.

Lastly, we demonstrate CATI on the dataset by Ceccarelli *et al.* [49], containing altered images of the KITTI dataset [37] with common failures on cameras. Figure 8 shows an example of these images, mimicking a broken lens. While YOLOv10m does not detect the preceding truck, CATI detects it.

V. RELATED WORK

Existing approaches that aim to bring more realistic automotive camera perception functionality to researchers focus on ML models trained for specific tasks. Especially for the detection of small objects, different improved versions of the existing YOLO model are available [50], [51], [52]. These works aim to increase the detection performance of the actual

(a) YOLOv10m

(b) Traffic sign detection in CATI

(c) YOLOv10m

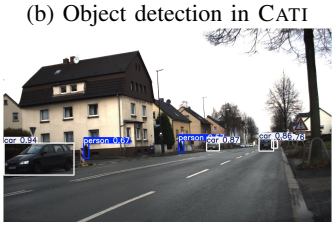
(d) Traffic sign detection in CATI

Fig. 6: Example images from GTSDB [38] with misdetections by YOLOv10m and correct detections by CATI.

(a) YOLOv10m

(b) Object detection in CATI

(c) YOLOv10m



(d) Object detection in CATI

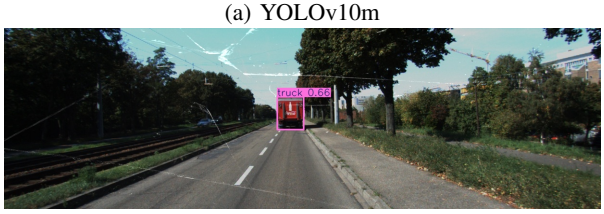
Fig. 7: Example images from GTSDB [38] with less detected cars by YOLOv10m and correct detections by CATI.

model rather than providing specific automotive features. Azevedo and Santos [53] show the deployment of YOLO on edge devices to perform automotive object detection. Other work that aims to include automotive object types like cars and street signs is not available as an extensible open-source framework [54], [19] or uses outdated versions of ML models [55].

Similarly, research on traffic sign detection using YOLO is available but uses outdated model architectures [56], [16], [57]. In contrast to automotive object detection, the dataset on which the model is trained can significantly limit the possible application since it might only be capable of detecting traffic signs of certain geographic regions [58], [57]. Other model architectures like Faster R-CNN can also be customized to achieve improved results for tasks such as traffic sign detection [44].

In contrast to CATI, they do not provide trained state-of-the-art models that: (i) perform image-based automotive object

(a) YOLOv10m



(b) Object detection in CATI

Fig. 8: Example images from Ceccarelli *et al.* [49] with a broken lens. While YOLOv10m does not detect the truck, CATI detects it correctly.

detection and traffic sign detection; (ii) come with a modular framework for possible extensions, and (iii) are available as open-source software that allows security researchers to analyze their attacks on camera images.

VI. CONCLUSION

Commonly available trained models are often trained on generic datasets with multiple non-automotive object types [14]. Although attacks on cameras in autonomous vehicles have been known for multiple years [3], their evaluation on realistic ML models remains a challenge. This is especially critical since many manufacturers often use proprietary algorithms optimized for specific tasks [32].

With our open-source framework CATI, we provide security researchers with a modular tool that includes an image-based automotive object detection ML model and a traffic sign detection model. Both models are specifically trained versions of the state-of-the-art ML model architecture YOLOv10m [20], [22]. With its provided interfaces, CATI allows the inclusion of further models for other specialized tasks, such as lane detection [21].

We have shown that CATI does not only provide better results for automotive object detection tasks than the commonly available versions of YOLO [22] but also shows a better performance when exposed to different attack scenarios. Additionally, specifically trained models in CATI can lead to more robust object detection in challenging environmental conditions. We evaluated CATI using well-established datasets [38], [47], [49] to highlight realistic use cases. By providing CATI as a modular open-source framework, we enable security researchers to test their camera attacks on a more realistic setup comparable to car manufacturers and suppliers.

ACKNOWLEDGMENT

This work is supported by the European Union-funded project CyberSecDome (Agreement No.: 101120779).

REFERENCES

[1] C. Gao, G. Wang, W. Shi, Z. Wang, and Y. Chen, "Autonomous Driving Security: State of the Art and Challenges," *IEEE Internet of Things Journal*, vol. 9, no. 10, 2022.

[2] B. Ranft and C. Stiller, "The Role of Machine Vision for Intelligent Vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, 2016.

[3] D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramèr, A. Prakash, and T. Kohno, "Physical Adversarial Examples for Object Detectors," in *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, (Baltimore, MD), USENIX Association, 2018.

[4] S.-T. Chen, C. Cornelius, J. Martin, and D. H. Chau, "ShapeShifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector," in *Machine Learning and Knowledge Discovery in Databases*, vol. 11051, Cham: Springer International Publishing, 2019.

[5] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang, and K. Chen, "Seeing isn't Believing: Towards More Robust Adversarial Attack Against Real World Object Detectors," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, (London United Kingdom), ACM, 2019.

[6] G. Lovisotto, H. Turner, I. Sluganovic, M. Strohmeier, and I. Martinovic, "SLAP: Improving Physical Adversarial Examples with Short-Lived Adversarial Perturbations," in *30th USENIX Security Symposium (USENIX Security 21)*, USENIX Association, 2021.

[7] A. Gnanasambandam, A. M. Sherman, and S. H. Chan, "Optical Adversarial Attack," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, (Montreal, BC, Canada), IEEE, 2021.

[8] M. Kühr, M. Hamad, P. MohajerAnsari, M. D. Pesé, and S. Steinhorst, "SoK: Security of the Image Processing Pipeline in Autonomous Vehicles," September 2024. arXiv:2409.01234 [cs].

[9] J. Wang, A. Liu, Z. Yin, S. Liu, S. Tang, and X. Liu, "Dual Attention Suppression Attack: Generate Adversarial Camouflage in Physical World," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Nashville, TN, USA), IEEE, 2021.

[10] A. Zolfi, M. Kravchik, Y. Elovici, and A. Shabtai, "The Translucent Patch: A Physical and Universal Attack on Object Detectors," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Nashville, TN, USA), IEEE, 2021.

[11] X. Ji, Y. Cheng, Y. Zhang, K. Wang, C. Yan, W. Xu, and K. Fu, "Poltergeist: Acoustic Adversarial Machine Learning against Cameras and Computer Vision," in *2021 IEEE Symposium on Security and Privacy (SP)*, (San Francisco, CA, USA), IEEE, 2021.

[12] Y. Zhang, H. Foroosh, P. David, and B. Gong, "CAMOU: Learning Physical Vehicle Camouflages to Adversarially Attack Detectors in the Wild," in *International Conference on Learning Representations*, 2019.

[13] W. Zhu, X. Ji, Y. Cheng, S. Zhang, and W. Xu, "TPatch: A Triggered Physical Adversarial Patch," in *32nd USENIX Security Symposium (USENIX Security 23)*, (Anaheim, CA), USENIX Association, 2023.

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, vol. 8693, Cham: Springer International Publishing, 2014.

[15] L. Ma, Q. Wu, Y. Zhan, B. Liu, and X. Wang, "Traffic Sign Detection Based on Improved YOLOv3 in Foggy Environment," in *Proceeding of 2021 International Conference on Wireless Communications, Networking and Applications*, Singapore: Springer Nature Singapore, 2022.

[16] J. Wang, Y. Chen, Z. Dong, and M. Gao, "Improved YOLOv5 network for real-time multi-scale traffic sign detection," *Neural Computing and Applications*, vol. 35, no. 10, 2023.

[17] N. Shazeer, K. Fatahalian, W. R. Mark, and R. T. Mullapudi, "HydraNets: Specialized Dynamic Architectures for Efficient Inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (Salt Lake City, UT), IEEE, June 2018.

[18] Z. Mai, A. Chowdhury, P. Zhang, C.-H. Tu, H.-Y. Chen, V. Pahuja, T. Berger-Wolf, S. Gao, C. Stewart, Y. Su, and W.-L. Chao, "Fine-Tuning is Fine, if Calibrated," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, (Vancouver, BC, Canada), 2024.

[19] M. N. Teli and S. Oh, "Resilience of Autonomous Vehicle Object Category Detection to Universal Adversarial Perturbations," in *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, (Toronto, ON, Canada), IEEE, 2021.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, NV, USA), IEEE, 2016.
[21] D. Vu, B. Ngo, and H. Phan, "HybridNets: End-to-End Perception Network," March 2022. arXiv:2203.09035 [cs].
[22] Ultralytics Inc., "YOLOv10 - Ultralytics YOLO Docs," January 2025.
[23] T.-F. Hsiao, B.-L. Huang, Z.-X. Ni, Y.-T. Lin, H.-H. Shuai, Y.-H. Li, and W.-H. Cheng, "Natural Light Can Also be Dangerous: Traffic Sign Misinterpretation Under Adversarial Natural Light Attacks," in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, (Waikoloa, HI, USA), IEEE, 2024.
[24] J. Yan, H. Yin, B. Ye, W. Ge, H. Zhang, and G. Rigoll, "An Adversarial Attack on Salient Regions of Traffic Sign," *Automotive Innovation*, vol. 6, no. 2, 2023.
[25] Y. Li, X. Xu, J. Xiao, S. Li, and H. T. Shen, "Adaptive Square Attack: Fooling Autonomous Cars With Adversarial Traffic Signs," *IEEE Internet of Things Journal*, vol. 8, no. 8, 2021.
[26] D. Guo, Y. Wu, Y. Dai, P. Zhou, X. Lou, and R. Tan, "Invisible Optical Adversarial Stripes on Traffic Sign against Autonomous Vehicles," in *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, (Minato-ku, Tokyo Japan), ACM, 2024.
[27] J. Zhou, L. Lyu, D. He, and Y. Li, "RAUCA: A Novel Physical Adversarial Attack on Vehicle Detectors via Robust and Accurate Camouflage Generation," February 2024. arXiv:2402.15853 [cs].
[28] H. Wen, S. Chang, L. Zhou, W. Liu, and H. Zhu, "OptiCloak: Blinding Vision-Based Autonomous Driving Systems Through Adversarial Optical Projection," *IEEE Internet of Things Journal*, vol. 11, no. 17, 2024.
[29] W. Wang, Y. Yao, X. Liu, X. Li, P. Hao, and T. Zhu, "I Can See the Light: Attacks on Autonomous Vehicles Using Invisible Lights," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, (Virtual Event Republic of Korea), ACM, 2021.
[30] A. Karpathy, "Workshop on Autonomous Driving at CVPR 2021," June 2021.
[31] BMW Group, "BMW Group shares AI algorithms used in production," tech. rep., Munich, December 2019.
[32] Robert Bosch GmbH, "Computer Vision | Bosch Global," January 2025.
[33] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015.
[34] The Linux Foundation, "Models and pre-trained weights — Torchvision main documentation," January 2025.
[35] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, 2010.
[36] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Seattle, WA, USA), IEEE, 2020.
[37] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, (Providence, RI), IEEE, 2012.
[38] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German traffic sign detection benchmark," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, (Dallas, TX, USA), IEEE, 2013.
[39] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-Sign Detection and Classification in the Wild," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, NV, USA), IEEE, 2016.
[40] C. Ertler, J. Mislej, T. Ollmann, L. Porzi, G. Neuhold, and Y. Kuang, "The Mapillary Traffic Sign Dataset for Detection and Classification on a Global Scale," in *Computer Vision – ECCV 2020*, vol. 12368, Cham: Springer International Publishing, 2020.
[41] A. Møgelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, 2012.
[42] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era," in *2017 IEEE International Conference on Computer Vision (ICCV)*, (Venice), IEEE, Oct. 2017.
[43] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "GitHub - THU-MIG/yolov10: YOLOv10: Real-Time End-to-End Object Detection [NeurIPS 2024]," February 2025.
[44] O. N. Manzari, A. Boudesh, and S. B. Shokouhi, "Pyramid Transformer for Traffic Sign Detection," in *2022 12th International Conference on Computer and Knowledge Engineering (ICCKE)*, (Mashhad, Iran, Islamic Republic of), IEEE, 2022.
[45] Y. Li, W. Leong, and H. Zhang, "YOLOv10-Based Real-Time Pedestrian Detection for Autonomous Vehicles," in *2024 IEEE 8th International Conference on Signal and Image Processing Applications (ICSIPA)*, (Kuala Lumpur, Malaysia), IEEE, 2024.
[46] A. Sharma, V. Kumar, and L. Longchamps, "Comparative performance of YOLOv8, YOLOv9, YOLOv10, YOLOv11 and Faster R-CNN models for detection of multiple weed species," *Smart Agricultural Technology*, vol. 9, 2024.
[47] A. Braunegg, A. Chakraborty, M. Krumdick, N. Lape, S. Leary, K. Manville, E. Merkhofer, L. Strickhart, and M. Walmer, "APRICOT: A Dataset of Physical Adversarial Attacks on Object Detection," in *Computer Vision – ECCV 2020*, vol. 12366, Cham: Springer International Publishing, 2020.
[48] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, no. 2015, 2015. Publisher: Amsterdam, Netherlands.
[49] A. Ceccarelli and F. Secci, "RGB Cameras Failures and Their Effects in Autonomous Driving Applications," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, 2023.
[50] H. Lou, X. Duan, J. Guo, H. Liu, J. Gu, L. Bi, and H. Chen, "DC-YOLOv8: Small-Size Object Detection Algorithm Based on Camera Sensor," *Electronics*, vol. 12, no. 10, 2023.
[51] H. Wang, C. Liu, Y. Cai, L. Chen, and Y. Li, "YOLOv8-QSD: An Improved Small Object Detection Algorithm for Autonomous Vehicles Based on YOLOv8," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, 2024.
[52] X. Han, J. Chang, and K. Wang, "Real-time object detection based on YOLO-v2 for tiny vehicle object," *Procedia Computer Science*, vol. 183, 2021.
[53] P. Azevedo and V. Santos, "YOLO-Based Object Detection and Tracking for Autonomous Vehicles Using Edge Devices," in *ROBOT2022: Fifth Iberian Robotics Conference*, vol. 589, Cham: Springer International Publishing, 2023.
[54] N. M. Alahdal, F. Abukhodair, L. H. Meftah, and A. Cherif, "Real-time Object Detection in Autonomous Vehicles with YOLO," *Procedia Computer Science*, vol. 246, 2024.
[55] A. Sarda, S. Dixit, and A. Bhan, "Object Detection for Autonomous Driving using YOLO [You Only Look Once] algorithm," in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, (Tirunelveli, India), IEEE, 2021.
[56] S. Saxena, S. Dey, M. Shah, and S. Gupta, "Traffic sign detection in unconstrained environment using improved YOLOv4," *Expert Systems with Applications*, vol. 238, 2024.
[57] J. Zhang, M. Huang, X. Jin, and X. Li, "A Real-Time Chinese Traffic Sign Detection Algorithm Based on Modified YOLOv2," *Algorithms*, vol. 10, no. 4, 2017.
[58] Y. Wu, Y. Liu, J. Li, H. Liu, and X. Hu, "Traffic sign detection based on convolutional neural networks," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, (Dallas, TX, USA), IEEE, 2013.

APPENDIX A
TRAFFIC SIGN CLASSES

A list of all traffic signs that CATI can detect is available in Table V. For each traffic sign it shows the unique identification (ID), its label name, and an example image.

TABLE V: Subset of traffic sign types from the Mapillary dataset [40] that are available in CATI.

ID	Name	Image	ID	Name	Image	ID	Name	Image
1	danger		18	give-way-to-oncoming-traffic		35	end-of-no-overtaking	
2	bicycles		19	stop		36	end-of-no-overtaking-trucks	
3	children		20	roundabout		37	no-trucks	
4	priority-next-intersection		21	go-straight-or-right		38	road-closed-to-all-vehicles	
5	give-way-next-intersection		22	go-straight-or-left		39	no-entry	
6	wild-animals		23	go-right		40	give-way	
7	icy-road		24	go-left		41	priority-road	
8	warning-pedestrians-crossing		25	go-left-or-right		42	end-of-prohibition	
9	pedestrians		26	go-straight		43	speed-limit-120	
10	roadworks		27	keep-right		44	speed-limit-100	
11	traffic-signals		28	keep-left		45	speed-limit-80	
12	slippery-road		29	one-way-right		46	speed-limit-70	
13	two-way-traffic		30	one-way-left		47	speed-limit-60	
14	curve-left		31	pedestrians-crossing		48	speed-limit-50	
15	curve-right		32	dead-end		49	speed-limit-40	
16	double-curve-first-left		33	no-overtaking		50	speed-limit-30	
17	double-curve-first-right		34	no-overtaking-trucks		51	speed-limit-20	