REAP: A Large-Scale Realistic Adversarial Patch Benchmark

Nabeel Hingun*
UC Berkeley
nabeel126@berkeley.edu

Chawin Sitawarin*
UC Berkeley
chawins@berkeley.edu

Jerry Li
Microsoft
jerrl@microsoft.com

David Wagner
UC Berkeley
daw@cs.berkeley.edu

Abstract

Machine learning models are known to be susceptible to adversarial perturbation. One famous attack is the *adversarial patch*, a sticker with a crafted pattern that makes the model incorrectly predict the object it is placed on. This attack presents a critical threat to cyber-physical systems such as autonomous cars. Despite the significance of the problem, conducting research in this setting has been difficult; evaluating attacks and defenses in the real world is exceptionally costly while synthetic data are unrealistic. In this work, we propose the REAP (REalistic Adversarial Patch) Benchmark, a digital benchmark that allows the user to evaluate patch attacks on real images, and under real-world conditions. Built on top of the Mapillary Vistas dataset, our benchmark contains over 14,000 traffic signs. Each sign is augmented with a pair of geometric and lighting transformations, which can be used to apply a digitally generated patch realistically onto the sign, while matching real-world conditions. Using our benchmark, we perform the first large-scale assessments of adversarial patch attacks under realistic conditions. We release our benchmark publicly at https://github.com/wagner-group/reap-benchmark.

1 Introduction

Research has shown that machine learning models lack robustness against adversarially chosen perturbations. One particularly concerning type of attack is the *adversarial patch attack* [Brown et al., 2018, Eykholt et al., 2018, Karmon et al., 2018, Sitawarin et al., 2018, Chen et al., 2019, Liu et al., 2019b, Patel et al., 2019, Sharma et al., 2019, Zhao et al., 2019, Huang et al., 2020, Wu et al., 2020]. These are real-world attacks, where the objective of the attacker is to print out a patch, physically place it in a scene, and cause a vision network processing the scene to malfunction. These attacks are especially concerning because of the impact they could have on autonomous vehicles which have already been demonstrated both in academic settings [Liu et al., 2019a, Evtimov et al., 2017, Sato et al., 2021], as well as in the real world [Tencent Keen Security Lab, 2019].

Despite the significant risks that adversarial patch attacks pose, research on these attacks has stalled. This is in large part because quantitatively evaluating this threat is challenging. Researchers turn to one of two techniques: either, physically create their attacks and try them out on real objects, or digitally evaluate patch attacks using images containing simulated patches. Both of these approaches have major drawbacks. Although the former approach is more realistic, the sample size is so small

*Equal contribution.

ML Safety Workshop, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

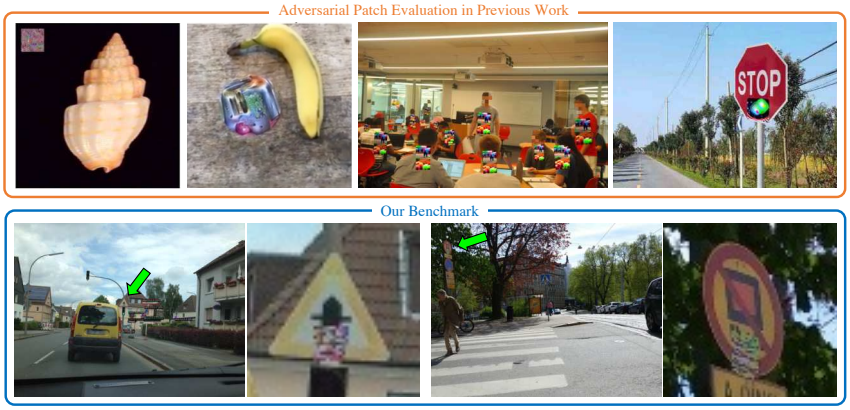


Figure 1: Past work (top row) ignores many real-world factors which might yield a misleading evaluation. Our benchmark (bottom row) supports a more realistic rendering of the patch attack on road signs, simulating the effect of the pose, location, and lighting on rendering and image capture.

that it is impossible to compare the results across different papers. In contrast, a digital simulation can allow us to obtain some quantitative measures, but it is difficult to ensure that the simulation accurately captures all of the challenges that arise in the real world. See Fig. 1 (top row) for examples of past works, where the patch is not constrained to be on the target object, does not respect the pose and shape of the object, and/or ignores the lighting conditions.

The REAP Benchmark: In this work, we propose the REalistic Adversarial Patch Benchmark (REAP), the first large-scale standardized benchmark for security against patch attacks. Motivated by the aforementioned shortcomings of prior evaluations, REAP is designed to have:

1. **Large scale evaluation:** REAP consists of a collection of 14,651 images of road signs drawn from the Mapillary Vistas dataset. This allows us to use REAP to draw quantitative conclusions
2. **Realistic patch rendering:** REAP comes with tooling to realistically render any digital patch onto every road sign in the dataset. It accounts for location, camera angle, lighting conditions, etc.
3. **Realistic image distribution:** REAP consists of images of signs taken under realistic conditions, mimicking for instance what a self-driving car would see from its sensors.

Additionally, with our new benchmark, we have found some interesting preliminary results. (i) Existing patch attacks are, in fact, not that effective. (ii) Performance on synthetic data is not reflective of performance on REAP. We believe that REAP will help support research in patch attacks by enabling a more accurate evaluation. We plan to release REAP publicly before the conference.

2 Related Work

Small scale, real-world tests. A common methodology used to test the adversarial patch is to print it out, physically place it onto an object, and capture pictures or videos of the patch for evaluation [Brown et al., 2018, Eykholt et al., 2018, Sitawarin et al., 2018, Chen et al., 2019, Zhao et al., 2019, Hoory et al., 2020, Huang et al., 2020, Wu et al., 2020]. While realistic, this method has some downsides. First, it is time-consuming and hence limits the number of images that can be used for testing. Consequently, one cannot extract quantitative conclusions from the results of these tests. Additionally, the methodologies vary across different papers so it is hard to compare their results.

Digital simulation. The other approach is to simulate the effects of the adversarial patch by digitally inserting it into the image. The advantage is scalability. Most of existing work simply apply the patch to the image at some random position, and with some simple transformations [Karmon et al., 2018, Liu et al., 2019b, Hoory et al., 2020, Zhang et al., 2020, Rao et al., 2020, McCoyd et al., 2020, Xiang and Mittal, 2021, Wang et al., 2021]. Our benchmark is arguably more realistic as shown in Fig. 1. The benchmarks most similar to the one we propose are the ones in Zhao et al. [2019] and Huang et al. [2020]. Zhao et al. [2019] consider the realistic orientation of the signs and the patches, but they only use synthetic stop signs. Huang et al. [2020] produce a benchmark of 20 fully virtual scenes,

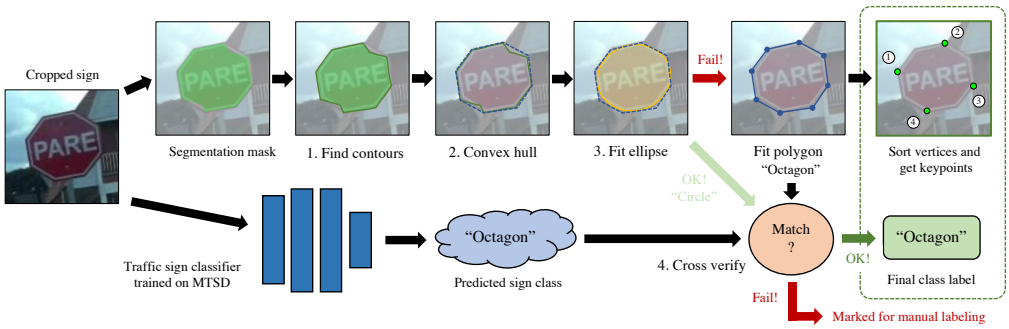


Figure 2: The automated procedure we use to extract the keypoints from each traffic sign.

Figure 3: The rightmost stop sign has a patch rendered with a perspective and relighting transform making it more realistic. The first and second images have patches that are too bright whereas the first and third images have patches that do not respect the sign's orientation.

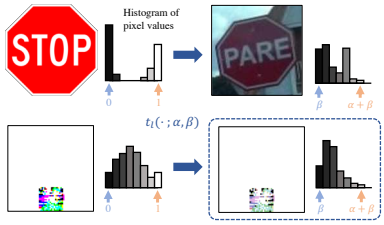


Figure 4: Computing relighting parameters (top) and applying the transform (bottom).

over which a user can control multiple cameras and lighting conditions. However, their dataset is small and does not consider road signs or corresponding attacks on autonomous vehicles

3 Adversarial Patch Benchmark

Our dataset, built on top of Mapillary Vistas [Neuhold et al., 2017], consists of a collection of images containing traffic signs. Each sign comes with a segmentation mask, and we label them with one of 11 classes based on their shape and size, which is by design also associated with their semantics. However, the main additional feature of our benchmark is the associated rendering transformation. This transformation allows us to apply a digital patch on the sign in a way that respects the scaling, orientation, and lighting of the sign in the image. Moreover, this process is fully differentiable, which allows our dataset to be used to generate patch attacks and to apply defenses such as adversarial training. Fig. 2 and Fig. 4 give an overview of the process to obtain these transformations which we will describe in Section 3.1. In total, we label 14,651 traffic signs across 8,433 images.

3.1 Transformations

When digitally applying a patch to a sign, we utilize two types of transformations: 3D or *perspective transform* to the patch to simulate the orientation and *relighting transform* to account for the varying lighting conditions. The importance of these transformations is highlighted in Fig. 3.

Geometric transformation. To determine the parameters of the perspective transform, we need *four* keypoints for each sign in our dataset. We infer the keypoints for a particular traffic sign using only its segmentation mask (which is provided in the Mapillary Vistas dataset) by following these steps (also visualized in Fig. 2): first, we find the contour of the segmentation mask, and then compute a convex hull around it to correct minor annotation errors. Next, we fit an ellipse and/or a polygon to the convex full to identify the shape of the signs. Lastly, we match the vertices to the canonical ones and then simply take the four predefined vertices as the keypoints.

Relighting transform. Each traffic sign in our dataset has two associated relighting parameters, $\alpha, \beta \in \mathbb{R}$. Given a patch $\mathbf{P}$, its relighted version $\mathbf{P}_{\text{relighted}} = \alpha\mathbf{P} + \beta$ is rendered on the scene as depicted on the bottom row of Fig. 4. We infer α, β by matching the histogram of the original sign (e.g., the real stop sign on the upper-right of Fig. 4) to a canonical image (e.g., the synthetic stop sign

Table 1: ASR and FNR of the adversarial patches on the two traffic sign detectors. One adversarial patch is generated per class of traffic signs. For sign-specific metrics, see Table 4.

Patch Size	Benchmarks	Faster R-CNN				YOLOv5			
		mFNR	$mFNR_w$	mASR	$mASR_w$	mFNR	$mFNR_w$	mASR	$mASR_w$
No patch	Synthetic	11.8	17.4	n/a	n/a	10.8	9.6	n/a	n/a
	Ours	16.2	17.4	n/a	n/a	14.2	14.5	n/a	n/a
Small (10"×10")	Synthetic	54.4	67.3	49.6	61.8	75.9	73.6	72.2	70.4
	Ours	35.1	30.5	24.7	17.7	46.1	32.5	44.4	24.9
Medium (10"×20")	Synthetic	69.7	87.3	67.7	85.6	88.2	90.8	85.0	88.3
	Ours	47.9	42.6	40.5	32.4	60.1	48.6	58.7	42.6
Large (two 10"×20")	Synthetic	98.1	98.1	99.5	99.5	100	100	100	100
	Ours	79.3	84.2	76.2	81.7	85.9	90.0	85.0	89.3

on the upper-left): in particular, we set β as the 10th percentile of all the pixel values on that sign and α as the difference between the 10th and 90th percentile.

4 Experiments on our Benchmark

Our benchmark can be used with a broad range of threat models. However, in this paper, we only focus on an attacker that tries to make traffic signs disappear or be misclassified.

4.1 Experiment Setup

Traffic sign detectors. We experiment with two models: Faster R-CNN [Ren et al., 2015] and YOLOv5 [Jocher et al., 2022]. Both models are trained on the MTSD dataset [Ertler et al., 2020]. We report false negative rate (FNR) along with the attack success rate (ASR), defined in Appendix C.2.

Synthetic benchmark. We use canonical synthetic signs, one per class, as a baseline to compare our REAP benchmark to. Similarly to Eykholt et al. [2018], the synthetic sign is placed at a random location in the image and randomly rotated between 0 and 15 degrees. The synthetic benchmark can be used for both generating and testing the adversarial patch. For testing, we use a total of 5,000 images randomly selected from our REAP benchmark.

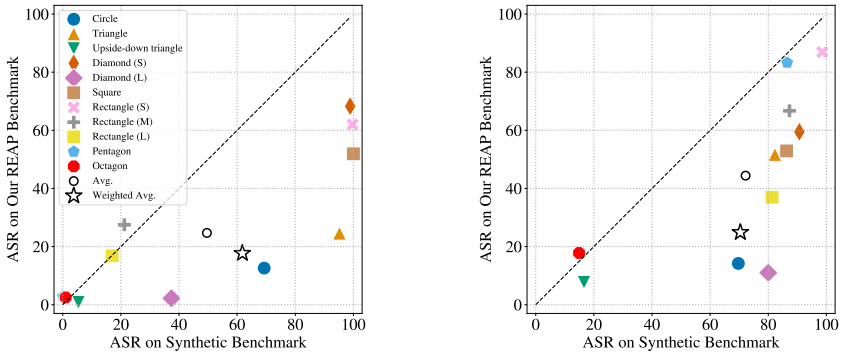
Attack algorithms. We use the RP2 attack [Eykholt et al., 2018] to generate adversarial patches for YOLOv5, and the Shapeshifter attack [Chen et al., 2019] for Faster R-CNN, as each attack was created for a specific type of model. We generate one patch per one sign class. We also generate an adversarial patch using a synthetic sign as has been done in prior work, and we use whichever patch performs better. For more detail on the setup, please see Appendix C.

4.2 Results

Patch attacks against road signs are less effective than previously believed. From Table 1, a 10"×10" adversarial patch only succeeds for about 18% and 25% of the signs on our realistic benchmark. Even though we use undefended models (normally trained without any special data augmentation), attacks are not very effective. For comparison, a universal adversarial perturbation under ℓ_2 and ℓ_∞ norms achieves above 80% success rate [Moosavi-Dezfooli et al., 2017].

We also experimented with a "per-image" attack where we generate one adversarial patch for each instance of a traffic sign, as opposed to one patch per class (see Table 8). ASR for the "per-image" attack is much higher than a per-class patch (40% relative increase on average). While such attacks are harder to generate and might be more fragile in practice, this result suggests that better attack algorithms may exist and that a specifically targeted adversary could still be an important threat.

ASR measured on synthetic data is not predictive of ASR measured on our realistic benchmark. We compared our benchmark to a synthetic benchmark (described earlier) intended to be representative of the methodology often found in prior work. Table 1 show that there is a large difference



(a) Faster R-CNN (b) YOLOv5

Figure 5: Comparison of ASR on synthetic benchmark vs on our realistic benchmark for the two models. The dashed line marks the points with an equal ASR on both synthetic and our benchmarks.

between metrics as measured on such a synthetic benchmark compared to our benchmark. The gap can be up to 50 percentage points on average.

Fig. 5a compares ASR on the two benchmarks by class of the traffic signs. If the two ASRs were similar, all data points would be close to the diagonal dashed line. Instead, most of the data points are below the line, suggesting that the synthetic benchmark consistently overestimates the ASR. Moreover, there is no clear relationship between the two measurements of ASR as the ordering is not preserved, and the gap varies significantly among different sign classes. Additional results and samples from our benchmark can be found in Appendix D and Appendix E, respectively.

Acknowledgements

This research was supported by the Hewlett Foundation through the Center for Long-Term Cybersecurity (CLTC), by Microsoft through the Berkeley Artificial Intelligence Research (BAIR), by the Berkeley Deep Drive project, and by generous gifts from Open Philanthropy.

References

T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer. Adversarial patch. *arXiv:1712.09665 [cs]*, May 2018. 1, 2

S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau. ShapeShifter: Robust physical adversarial attack on faster R-CNN object detector. In M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, and G. Ifrim, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 52–68, Cham, 2019. Springer International Publishing. ISBN 978-3-030-10925-7. 1, 2, 4

C. Ertler, J. Mislej, T. Ollmann, L. Porzi, and Y. Kuang. The mapillary traffic sign detection and classification around the world. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 4

I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song. Robust physical-world attacks on machine learning models. 2017. 1

K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramèr, A. Prakash, T. Kohno, and D. Song. Physical adversarial examples for object detectors. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, Baltimore, MD, Aug. 2018. USENIX Association. 1, 2, 4

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90. 10

S. Hoory, T. Shapira, A. Shabtai, and Y. Elovici. Dynamic adversarial patch for evading object detection models. *arXiv:2010.13070 [cs]*, Oct. 2020. 2

L. Huang, C. Gao, Y. Zhou, C. Xie, A. L. Yuille, C. Zou, and N. Liu. Universal physical camouflage attacks on object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2

G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, K. Michael, J. Fang, imyhxy, Lorna, C. Wong, Z. Yifu, A. V, D. Montes, Z. Wang, C. Fati, J. Nadar, Laughing, UnglvKitDe, tkianai, yxNONG, P. Skalski, A. Hogan, M. Strobel, M. Jain, L. Mammana, and xylieong. Ultralytics/yolov5: V6.2 - yolov5 classification models, apple M1, reproducibility, ClearML and deci.ai integrations. Zenodo, Aug. 2022. 4

D. Karmon, D. Zoran, and Y. Goldberg. LaVAN: Localized and visible adversarial noise. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2507–2515. PMLR, July 2018. 1, 2

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 D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1. 12

A. Liu, X. Liu, J. Fan, Y. Ma, A. Zhang, H. Xie, and D. Tao. Perceptual-sensitive gan for generating adversarial patches. In *AAAI Conference on Artificial Intelligence*, 2019a. 1

X. Liu, H. Yang, Z. Liu, L. Song, H. Li, and Y. Chen. DPatch: An adversarial patch attack on object detectors. *arXiv:1806.02299 [cs]*, Apr. 2019b. 1, 2

M. McCoyd, W. Park, S. Chen, N. Shah, R. Roggenkemper, M. Hwang, J. X. Liu, and D. Wagner. Minority reports defense: Defending against adversarial patches. In J. Zhou, M. Conti, C. M. Ahmed, M. H. Au, L. Batina, Z. Li, J. Lin, E. Losiouk, B. Luo, S. Majumdar, W. Meng, M. Ochoa, S. Picek, G. Portokalidis, C. Wang, and K. Zhang, editors, *Applied Cryptography and Network Security Workshops*, pages 564–582, Cham, 2020. Springer International Publishing. ISBN 978-3-030-61638-0. 2

S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 4

G. Neuhold, T. Ollmann, S. R. Bulo, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5000–5009, Venice, Oct. 2017. IEEE. ISBN 978-1-5386-1032-9. doi: 10.1109/ICCV.2017.534. 3

N. Patel, P. Krishnamurthy, S. Garg, and F. Khorrami. Adaptive adversarial videos on roadside billboards: Dynamically modifying trajectories of autonomous vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5916–5921, Nov. 2019. doi: 10.1109/IROS40897.2019.8968267. 1

S. Rao, D. Stutz, and B. Schiele. Adversarial training against location-optimized adversarial patches. In A. Bartoli and A. Fusiello, editors, *Computer Vision – ECCV 2020 Workshops*, pages 429–448, Cham, 2020. Springer International Publishing. ISBN 978-3-030-68238-5. 2

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 91–99, Cambridge, MA, USA, 2015. MIT Press. 4

E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. Kornia: An open source differentiable computer vision library for PyTorch. In *Winter Conference on Applications of Computer Vision*, 2020. 10

T. Sato, J. Shen, N. Wang, Y. Jia, X. Lin, and Q. A. Chen. Dirty road can attack: Security of deep learning based automated lane centering under physical-world attack. In *USENIX Security*, 2021. 1

P. Sharma, D. Austin, and H. Liu. Attacks on machine learning: Adversarial examples in connected and autonomous vehicles. In *2019 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–7, 2019. doi: 10.1109/HST47167.2019.9032989. 1

C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal. DARTS: Deceiving autonomous cars with toxic signs. *arXiv:1802.06430 [cs]*, May 2018. 1, 2

Tencent Keen Security Lab. Experimental Security Research of Tesla Autopilot, 2019. URL `https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf`. 1

Y. Wang, H. Lv, X. Kuang, G. Zhao, Y.-a. Tan, Q. Zhang, and J. Hu. Towards a physical-world adversarial patch for blinding object detection models. *Information Sciences*, 556:459–471, 2021. ISSN 0020-0255. doi: 10.1016/j.ins.2020.08.087. 2

Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2, 2019. 9

Z. Wu, S.-N. Lim, L. S. Davis, and T. Goldstein. Making an invisibility cloak: Real world adversarial attacks on object detectors. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 1–17, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58548-8. 1, 2

C. Xiang and P. Mittal. PatchGuard++: Efficient provable attack detection against adversarial patches. *arXiv:2104.12609 [cs]*, Apr. 2021. 2

Z. Zhang, B. Yuan, M. McCoyd, and D. Wagner. Clipped BagNet: Defending against sticker attacks with clipped bag-of-features. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 55–61, San Francisco, CA, USA, May 2020. IEEE. ISBN 978-1-72819-346-5. doi: 10.1109/SPW50608.2020.00026. 2

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*, CCS '19, pages 1989–2004, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 978-1-4503-6747-9. doi: 10.1145/3319535.3354259. 1, 2

A X-Risk Sheet

A.1 Long-Term Impact on Advanced AI Systems

In this section, please analyze how this work shapes the process that will lead to advanced AI systems and how it steers the process in a safer direction.

1. **Overview.** How is this work intended to reduce existential risks from advanced AI systems?
   Answer: This work is intended to serve as a new benchmark for future research on adversarial patches, both attacks, and defenses. We believe that a realistic and diverse benchmark would be an important driving force in achieving a more adversarially robust system. Since realism and practicality are the main factors of adversarial patches (even more so than other types of attacks), we design our benchmark to specifically prioritize these properties.
2. **Direct Effects.** If this work directly reduces existential risks, what are the main hazards, vulnerabilities, or failure modes that it directly affects?
   Answer: This work does not come with a method to directly reduce AI existential risks. They serve as an evaluation tool and a standardized benchmark for future work.
3. **Diffuse Effects.** If this work reduces existential risks indirectly or diffusely, what are the main contributing factors that it affects?
   Answer: Our main contribution in this work is the realistic benchmark which can be used for many different purposes. Naturally, it can be used as the main and standardized dataset for comparing different attack and defense algorithms. With more considerations, it can also be part of safety team resources, test requirements, safety constraints, or future standards in the field.
4. **What's at Stake?** What is a future scenario in which this research direction could prevent the sudden, large-scale loss of life? If not applicable, what is a future scenario in which this research direction be highly beneficial?
   Answer: We believe that our research here will have a large impact on future academic research in the domain of physical adversarial examples. Ultimately, this line of research hopes to come up with robust and reliable cyber-physical systems, e.g., autonomous vehicles, that can truly benefit and safely coexist with humans in society. We hope that our work will be an important step towards that goal.
5. **Result Fragility.** Do the findings rest on strong theoretical assumptions; are they not demonstrated using leading-edge tasks or models; or are the findings highly sensitive to hyperparameters? ☐
6. **Problem Difficulty.** Is it implausible that any practical system could ever markedly outperform humans at this task? ☒
7. **Human Unreliability.** Does this approach strongly depend on handcrafted features, expert supervision, or human reliability? ☐
8. **Competitive Pressures.** Does work towards this approach strongly trade off against raw intelligence, other general capabilities, or economic utility? ☐

A.2 Safety-Capabilities Balance

In this section, please analyze how this work relates to general capabilities and how it affects the balance between safety and hazards from general capabilities.

9. **Overview.** How does this improve safety more than it improves general capabilities?
   Answer: The benchmark we created has almost the sole intention of improving the robustness against adversaries. It can also be used for research on general robustness.
10. **Red Teaming.** What is a way in which this hastens general capabilities or the onset of x-risks?
    Answer: This benchmark can be used by both attackers and defenders alike. However, it does not necessarily benefit attackers who wish to launch such an attack in practice since the adversary in this case would be better off collecting his or her own data from a specific target sign and scene. Rather, our benchmark is better suited for testing the capabilities of attack and defense algorithms in diverse settings.
11. **General Tasks.** Does this work advance progress on tasks that have been previously considered the subject of usual capabilities research? ☐

12. **General Goals.** Does this improve or facilitate research towards general prediction, classification, state estimation, efficiency, scalability, generation, data compression, executing clear instructions, helpfulness, informativeness, reasoning, planning, researching, optimization, (self-)supervised learning, sequential decision making, recursive self-improvement, open-ended goals, models accessing the Internet, or similar capabilities? □
13. **Correlation With General Aptitude.** Is the analyzed capability known to be highly predicted by general cognitive ability or educational attainment? □
14. **Safety via Capabilities.** Does this advance safety along with, or as a consequence of, advancing other capabilities or the study of AI? □

A.3 Elaborations and Other Considerations

15. **Other.** What clarifications or uncertainties about this work and x-risk are worth mentioning?
Answer: N/A

B Additional Details of REAP Benchmark

B.1 Basic Usage

REAP benchmark provides simple tooling for rendering adversarial patch onto desired traffic signs. This tool consists of three main components:

1. `reap_annotations.csv`: The annotation for each sign in our REAP benchmark corresponds to one row in this `csv` file. We load it in as a `pandas.DataFrame` object and read from it as we iterate through each sample from the dataset.
2. `RenderObject` class: We create one `RenderObject` for each sign we want to apply an adversarial patch to. `RenderObject` holds the parameters of the geometric and the relighting transforms for that sign and also applies the transformations when called by `RenderImage`. We use two separate subclasses to implement `RenderObject` for the real signs in REAP benchmark and for the synthetic data.
3. `RenderImage` class: We wrap each sample in a `RenderImage` object. `RenderImage` holds the original image and a dictionary of multiple `RenderObject`'s. Once `RenderObject.apply_objects()` is called, it loops through all of its `RenderObject`'s and applies the transformed adversarial patches to the image.

Below we show a snippet of how this tool is generally used for applying patches for the REAP benchmark. This process can be applied during both the attack generation and the evaluation. Our Github repository (`https://github.com/wagner-group/reap-benchmark`) also contains a real example of how this is used with the `Detectron2` framework [Wu et al., 2019].[1]

```
# Given input parameters
sample: Dict[str, Any]  # An input sample (e.g., loaded by Detectron2)
img_df: pandas.DataFrame  # DataFrame of REAP annotation for this image
adv_patch: torch.Tensor  # Generated adversarial patch
patch_mask: torch.Tensor  # Binary mask of adversarial patch
obj_id: int  # ID of object to apply patch to

# Create RenderImage object around a given sample
rimg: RenderImage = RenderImage(sample, img_df, **rimg_kwargs)

# Create RenderObject of obj_id in rimg
rimg.create_object(obj_id, **robj_kwargs)

# Load adv patch to associated RenderObject
robj: RenderObject = rimg.get_object(obj_id)
robj.load_adv_patch(adv_patch=adv_patch, patch_mask=patch_mask)
```

[1] `https://github.com/facebookresearch/detectron2`

Table 2: Dimension of the sign by classes in the REAP benchmark.

Traffic Sign Class	Width (mm)	Height (mm)	Number of Samples in REAP
Circle	750	750	7971
Triangle	900	789	636
Upside-down triangle	1220	1072	824
Diamond (S)	600	600	317
Diamond (L)	915	915	1435
Square	600	600	1075
Rectangle (S)	458	610	715
Rectangle (M)	762	915	544
Rectangle (L)	915	1220	361
Pentagon	915	915	133
Octagon	915	915	637

```
# Render adv_patch on rimg and post process it into desired format
img_render: torch.Tensor = rimg.apply_objects()
img_render = rimg.post_process_image(img_render)

# Perform inference on rendered image
outputs: Dict[str, Any] = predict(img_render)
```

The operations on the image and the adversarial patch are implemented with `PyTorch` and the `Kornia` package [Riba et al., 2020], which also uses `PyTorch` underneath, so the entire process can be executed on a GPU or a CPU and is entirely differentiable. Our hardware setup (one Nvidia Tesla V100 with six CPU cores) can evaluate anywhere between one to two images per second with the default resolution of 1536×2048 pixels. This includes data loading, preprocessing, applying a patch to at least one sign on that image, and running an inference. The total evaluation of our REAP benchmark is about five hours. The total time depends mostly on the number of CPU cores (e.g., using twice as many CPU cores cuts down the total runtime by about half) and not on the GPU specs since we evaluate one image at a time and not in batch. In the next section (Appendix B.5), we provide additional details of how the transforms are applied and what is going on inside of `RenderObject`.

B.2 Traffic Sign Classification

Our first step is to simplify the attack setting by grouping traffic signs of a similar shape and size together. The original MTSD dataset has over 300 classes so we hope to take a subset of them with a common and fairly standardized size. For example, we do not take generic directional signs because there is no standard size for them at all. In the end, we end up with 11 classes in total plus one background class where all the remaining signs belong. Then, we assign the dimension to each class according to the official guideline published by the U.S. Department of Transportation. Table 2 summarizes the dimension we assign to all the signs.

This dimension only approximates the true size which we have no way of measuring given that the camera specifications and the distance to the signs are not known. This leads to two limitations: First, signs within a single class may actually be of different sizes because each sign has more than one standard size which mostly depends on the type of road it is placed on. The second is that both MTSD and Mapillary Vistas contain signs from all over the world, not only from the US. Hence, the dimension may not be consistent across countries. Nevertheless, we argue that this approximation is more realistic than naively specifying the patch size relative to the sign size in pixels (e.g., "each patch covers 10% of the sign") because sign sizes vary significantly between classes.

After mapping the original MTSD classes to our new 11 classes, we train a ResNet-18 [He et al., 2016] on all of the signs from MTSD. These signs are cropped by leaving 10% padding between the sign border and the patch border on all four sides. The cropped patches are then resized to 128×128 pixels. We trained the ResNet-18 with a batch size of 128, a learning rate of 0.1, and a weight decay of 5×10^{-4}. We use the validation set to early stop the training where the model achieves slightly above 97% accuracy. Lastly, we use the trained ResNet to classify traffic signs in the Mapillary

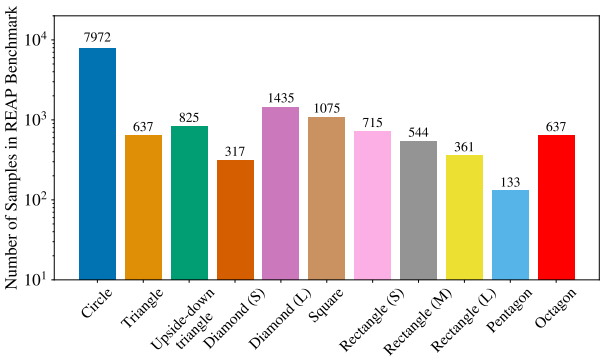


Figure 6: Class distribution of the traffic signs in our REAP benchmark.

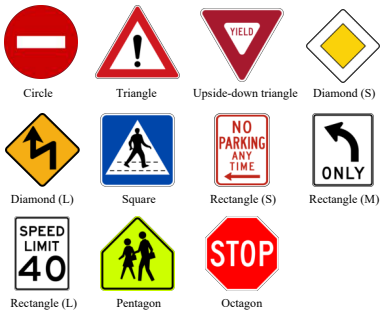


Figure 7: Images of the synthetic signs we use for generating and evaluating adversarial patches under the synthetic benchmark.

dataset. We also combine the training and the validation sets together. We ignore signs that are classified as the background class and discard images that do not contain any non-background sign.

B.3 Geometric Transformation

To determine the parameters of the perspective transform, we need *four* keypoints for each sign in our dataset. We infer the keypoints for a particular traffic sign using only its segmentation mask (which is provided in the Mapillary Vistas dataset) by following the four steps below (also visualized in Fig. 2):

1. *Find contours*: First, we find the edge or the contour of the segmentation mask.
2. *Compute convex hull*: Then, we find the convex hull of the contour to correct annotation errors and occlusion. This does not affect the already correct masks which should already be convex.
3. *Fit polygon and ellipse*: We try to fit an ellipse to the convex hull, to find circular signs. If the fitted ellipse results in a larger error than some threshold, we know that the sign is not circular and therefore fit a polygon instead.
4. *Cross verify*: We verify that the shape obtained from the previous step matches with the ResNet's prediction. If not, the sign is flagged for manual inspection.

The last step is finding the keypoints. For polygons, we first match the vertices to the canonical ones and then simply take the four predefined vertices as the keypoints. For circular signs, we use the ends of their major and minor axes as the four keypoints. Then, we use these keypoints to infer a perspective transform appropriate for this sign. Triangular signs are a special case as we can only identify a maximum of three keypoints which means we can only infer a unique affine transform (six degrees of freedom). Note that this transform is a linear transformation, and hence is fully differentiable. Lastly, we manually check all annotations and correct any errors.

B.4 Relighting Transformation

Each traffic sign in our dataset has two associated relighting parameters, $\alpha, \beta \in \mathbb{R}$. Given a patch $\mathbf{P}$, its relighted version $\mathbf{P}_{\text{relighted}} = \alpha\mathbf{P} + \beta$ is rendered on the scene as depicted on the bottom row of Fig. 4. We infer α, β by matching the histogram of the original sign (e.g., the real stop sign on the upper-right of Fig. 4) to a canonical image (e.g., the synthetic stop sign on the upper-left): in particular, we set β as the 10th percentile of all the pixel values (aggregated over all three RGB channels) on that sign and α as the difference between the 10th and 90th percentile. In doing so, we make an assumption that the relighting can be approximated with a linear transform where α and β represent contrast and brightness adjustments, respectively. We choose the 10th and the 90th percentiles, instead of the 1st and the 99th, to filter noise in the real image. Finally, note that like before, since this transformation is linear, it is differentiable.

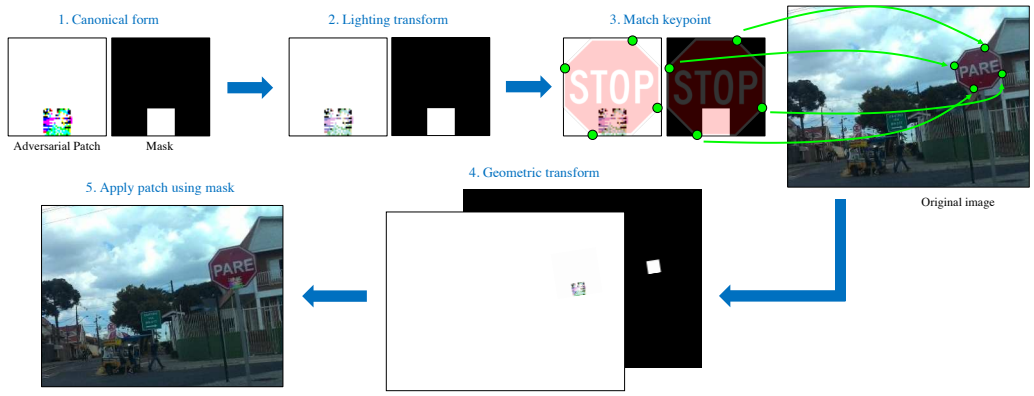


Figure 8: REAP's procedure for applying lighting and geometric transforms to a digitally generated adversarial patch.

Table 3: Performance of the models on two different datasets without attacks. "MTSD" represents the test set of the MTSD dataset on which the models are trained. mAP is the mean average precision, commonly used to represent performance in object detection tasks. We follow COCO's method for computing mAP [Lin et al., 2014]. mAP_w is the class-weighted version analogous to mFNR and mFNR_w which are defined in Appendix C.2.

Datasets	Faster R-CNN				YOLOv5			
	mFNR ↓	mFNR_w ↓	mAP ↑	mAP_w ↑	mFNR ↓	mFNR_w ↓	mAP ↑	mAP_w ↑
MTSD	29.9	29.7	55.0	56.0	17.2	14.7	69.3	71.3
Our Benchmark	16.2	17.4	67.0	64.4	14.2	14.5	69.7	69.6

B.5 Applying the Transforms

Fig. 8 summarizes the steps to apply an adversarial patch using our REAP benchmark. Given a patch and a corresponding mask with respect to the canonical sign, we first apply relighting transform on the patch. Then, we use the annotated keypoints of the target sign to determine the parameters of the perspective transform which is then applied to both the patch and the mask. Throughout this paper, we use bilinear interpolation for any geometric transform. Finally, the transformed patch is applied to the image using the transformed mask.

To be precise, let $\boldsymbol{X}$, $\boldsymbol{P}$, and $\boldsymbol{M}$ denote the original image, the adversarial patch, and the patch mask, respectively. The final image $\boldsymbol{X}'$ is obtained by the following equation

$$\boldsymbol{X}' = t_g\left(\boldsymbol{M}\right) \odot t_g\left(t_l\left(\boldsymbol{P}\right)\right) + \left(1 - t_g\left(\boldsymbol{M}\right)\right) \odot \boldsymbol{X} \tag{1}$$

where $t_g(\cdot)$ and $t_l(\cdot)$ are the geometric and the relighting transforms which in fact, depend on the annotated parameters associated with $\boldsymbol{X}$.

We note that the mask is concatenated to the patch, and both are applied with the same geometric transform and interpolation. Therefore, $t_g\left(\boldsymbol{M}\right)$ is no longer a binary mask like $\boldsymbol{M}$. This creates an effect where the transformed patch blends in more cleanly with the sign than the nearest interpolation does. Additionally, we also clip the pixel values after applying each transform to ensure that they always stay between 0 and 1.

C Detailed Experiment Setup

C.1 Attack Algorithms

Here, we describe hyperparameters of the attack as well as the benchmarks. We re-implement both the RP2 and the ShapeShifter attacks based on the description provided in the paper. The ShapeShifter

Figure 9: Examples of Small (10" x 10"), Medium (10" x 20") and Large (two 10" x 20") patches applied to random signs from our benchmark.

Table 4: Attack success rates by sign classes under synthetic vs our REAP benchmarks. "*Avg*" is an average, and "*WAvg*" is a weighted average by the number of samples in each class in our benchmark. The patch size is $10'' \times 10''$.

Models	Bench	Circ	Tri	UTri	Dia(S)	Dia(L)	Squ	Rec(S)	Rec(M)	Rec(L)	Pen	Oct	*Avg*	*WAvg*
Faster R-CNN	Syn	69.3	95.2	5.4	98.9	37.4	100.0	99.7	21.2	17.0	0.1	1.0	49.6	61.8
	REAP	12.7	24.6	0.8	68.3	2.2	52.0	62.0	27.4	16.7	2.8	2.5	24.7	17.7
YOLOv5	Syn	69.7	82.3	16.6	90.7	80.0	86.3	98.6	87.3	81.3	86.5	14.9	72.2	70.4
	REAP	14.2	51.7	7.7	59.5	11.0	52.9	86.9	66.7	37.0	83.3	17.8	44.4	24.9

attack has an official and publicly available implementation in Tensorflow so we are able to compare our code to theirs directly.[2]

For both attacks, our default hyperparameters include 64×64 patch and object dimension, EoT rotation with a maximum 15 degrees, and no color jitter for EoT. We use Adam optimizer with a step size of 0.01 for 1,000 iterations. The λ parameter used to encourage low-frequency patterns is set to 10^{-5}. We find that the choices of the patch dimension and λ do not affect the ASR when varying in a reasonable range. We use the same set of hyperparameters when generating adversarial patches from both our REAP and the synthetic benchmark.

We assume that the adversary has access to 50 held-out images from our benchmark. Each of the 11 classes in our dataset has a specific set of 50 images which are all guaranteed to contain at least one sign of the class that is being attacked.

C.2 Evaluation Metrics

Here, we define *a successful attack* as a patch that makes the sign either (i) undetected or (ii) classified to a wrong class (i.e., one of the other 11 classes, or the background class). Similarly to previous work, we measure the effectiveness of an attack by the *attack success rate* (ASR), defined as follows. Given a list of signs $\{x_i\}_{i=1}^N$ and the corresponding perturbed version (i.e., with an adversarial patch applied to it) $\{x_i'\}_{i=1}^N$,

$$\text{Attack Success Rate} = \frac{\sum_{i=1}^N \mathbf{1}_{x_i \text{is detected}} \ \wedge \ \mathbf{1}_{x_i' \text{is not detected}}}{\sum_{i=1}^N \mathbf{1}_{x_i \text{is detected}}}. \tag{2}$$

Additionally, we also report false negative rate (FNR), which is simply the fraction of signs that the model fails to detect and classify correctly. We report both metrics for each class of the signs as well as their average (mFNR and mASR). We also compute an average of the metrics weighted by the number of samples in each class (mFNR_w and mASR_w).

D Additional Experiments

ASR by Classes for All Patch Sizes. Table 4, Table 5, and Table 6 contain a breakdown of ASR by sign class for the three patch sizes in Table 1. It is evident that the synthetic data do not overestimate the ASR on average but consistently across almost every traffic sign class. The trend is also consistent for all patch sizes. The gap becomes narrower for the two 10×20 patches as the ASR reaches 100% on both the synthetic and our benchmarks. However, it is almost undeniable that a patch of this size covers the majority of the sign area and is, in no way, inconspicuous.

[2] https://github.com/shangtse/robust-physical-attack

Table 5: Attack success rates by sign classes under synthetic vs our REAP benchmarks with the patch size is 10"×20".

Models	Bench	Circ	Tri	UTri	Dia(S)	Dia(L)	Squ	Rec(S)	Rec(M)	Rec(L)	Pen	Oct	*Avg*	*WAvg*
Faster R-CNN	Syn	98.0	99.7	0.3	100.0	75.7	100.0	100.0	92.7	71.0	1.6	5.4	67.7	85.6
	REAP	27.8	63.2	0.6	89.7	7.1	64.7	87.4	54.8	39.2	6.9	3.6	40.5	32.4
YOLOv5	Syn	96.2	95.0	21.6	99.8	71.3	100.0	100.0	97.3	99.2	99.6	55.3	85.0	88.3
	REAP	36.7	69.4	7.8	78.0	13.1	82.6	92.4	78.8	58.0	91.3	37.9	58.7	42.6

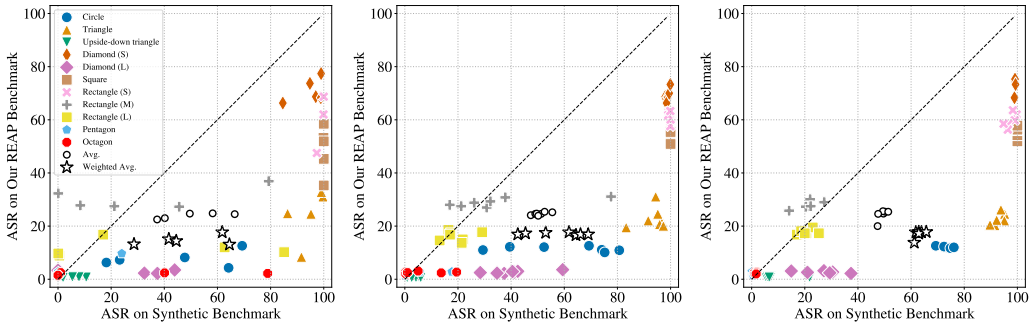
Table 6: Attack success rates by sign classes under synthetic vs our REAP benchmarks with two patches of size 10"×20".

Models	Bench	Circ	Tri	UTri	Dia(S)	Dia(L)	Squ	Rec(S)	Rec(M)	Rec(L)	Pen	Oct	*Avg*	*WAvg*
Faster R-CNN	Syn	100	100	99.9	100	100	100	100	100	79.6	100	100	98.1	99.5
	REAP	88.2	86.7	20.1	99.6	53.5	99.7	100.0	95.5	40.5	63.9	90.3	76.2	81.7
YOLOv5	Syn	100	100	100	100	100	100	100	100	100	100	100	100	100
	REAP	97.7	85.3	37.6	99.0	64.5	98.6	100.0	98.4	58.0	98.4	97.9	85.0	89.3

ASR under varying hyperparameters of the synthetic benchmark. We conduct an additional ablation study to compare the effects of the hyperparameters of the synthetic benchmark as mentioned in Appendix C. Fig. 10 shows a similar scatter plot to Fig. 5a but with all the hyperparameters we have swept. This plot further strengthens the conclusions that the synthetic benchmark overestimates ASR and that it is not predictive of the ASR on our REAP benchmark. These observations persist across all the hyperparameter choices.

Additionally, Fig. 10 demonstrates that there is a large variation in the ASRs measured by the synthetic benchmark when the hyperparameters vary. For instance, changing the rotation can affect the ASR up to 20%–40% for many signs. This emphasizes that results reported on a synthetic benchmark are sensitive to its hyperparameters, and we should take special care when using one. On the other hand, our REAP benchmark does not have a similar set of hyperparameters to sweep over since the transformations as well as the sign sizes are fixed with respect to each image.

The lighting transform affects the attack's effectiveness more than the geometric transform. Table 7 shows results from an ablation study showing how the transformations our benchmark applies to the patch affect its ASR. For both YOLOv5 and Faster R-CNN, our realistic lighting transform has a much larger effect than the geometric transform. Without the lighting transform, the average ASR increases by approximately 10 percentage points for YOLOv5, and 15 pp for Faster R-CNN. This



(a) Synthetic object size (b) Rotation degrees (c) Color jitter intensity

Figure 10: A scatter plot similar to Fig. 5a where each point denotes a pair of ASRs measured by the synthetic and our REAP benchmark for each class of the signs and also for each hyperparameter choice for the synthetic benchmark. Particularly, we sweep three hyperparameters that control (a) patch and object dimension, (b) the range of rotation degree used in EoT, and (c) the color jitter intensity used in EoT.

Table 7: Attack success rate on our realistic benchmark when other choices of transformations are applied. "No 3D" means the perspective transform is replaced by only translation and scaling. "No light" means there is no relighting transform, but we still apply the perspective transform.

Models	Transform	Circ	Tri	UTri	Dia(S)	Dia(L)	Squ	Rec(S)	Rec(M)	Rec(L)	Pen	Oct	*mASR*	$mASR_w$
Faster R-CNN	No 3D	13.0	24.8	0.3	62.6	2.7	55.2	62.0	25.6	17.2	5.6	2.7	24.7	18.0
	No light	30.8	44.1	1.5	77.0	6.3	62.5	80.8	41.7	47.6	2.8	6.1	36.5	32.3
	Ours	12.7	24.6	0.8	68.3	2.2	52.0	62.0	27.4	16.7	2.8	2.5	24.7	17.7
YOLOv5	No 3D	16.6	49.8	8.1	57.2	11.4	61.3	90.6	66.2	39.9	88.9	16.0	46.0	26.8
	No light	26.7	60.0	8.5	70.1	14.8	62.8	92.6	72.1	55.0	86.5	22.6	52.0	34.6
	Ours	14.2	51.7	7.7	59.5	11.0	52.9	86.9	66.7	37.0	83.3	17.8	44.4	24.9

Table 8: Attack success rate of the *per-image* attack on our realistic benchmark. The model is Faster R-CNN, and the patch size is $10'' \times 10''$.

Metrics	Circ	Tri	UTri	Dia(S)	Dia(L)	Squ	Rec(S)	Rec(M)	Rec(L)	Pen	Oct	*Avg.*	*WAvg.*
FNR	34.2	42.4	7.6	87.5	16.6	76.1	94.1	53.9	42.5	15.4	9.6	43.6	37.9
ASR	20.1	35.3	1.7	86.4	3.7	72.5	88.1	44.2	22.0	8.3	5.0	35.2	26.4
AP	46.3	55.7	63.9	29.4	65.1	26.3	3.9	46.2	56.2	61.5	77.5	48.4	47.1

observation explains why the synthetic benchmark as well as synthetic evaluations in previous works overestimate ASR, as prior works do not consider lighting.

Per-image attack. As another ablation study, we experiment with the worst-case possible attack on our benchmark where the adversary can generate a unique adversarial patch for each image and is also aware of how the patch will appear in the image exactly. This setting is similar to the commonly studied "white-box" attack in the adversarial example literature. This threat model is particularly unrealistic for patch attacks because, in the real world, the adversary cannot predict apriori how the video or the image of the patch will be taken. Nonetheless, theoretically, this measurement is useful because the ASR in this setting should be the upper bound of any other setting including the "per-class" threat model we have considered throughout the paper.

Table 8 reports the per-image ASR on our REAP benchmark. We only compute the ASR for Faster R-CNN because this experiment is computationally expensive even when we reduce the attack iterations from 1,000 to 200. This experiment takes about 10 days to finish on an Nvidia GTX V100 GPU. On average, the per-image attack results in about 10 percentage points higher ASR than the per-class attack. This is a significant increase (about 40% relatively).

However, in the absolute sense, the ASR is only 35%, i.e., the patch attack only succeeds about one-third of the time in the worst-case scenario. There are two ways to interpret this observation: first, it could mean that an object detection model may be more robust to physical attacks than the researchers expect, and this makes coming up with an effective defense easier. The second way to view this result is that the previously proposed attack algorithms are far from optimal, and there is a large room for improvement from the attacker's side.

E Additional Visualization of the Benchmark

In Fig. 12, we select six images from our benchmark with the 10"×10" patch applied, one from each sign class.

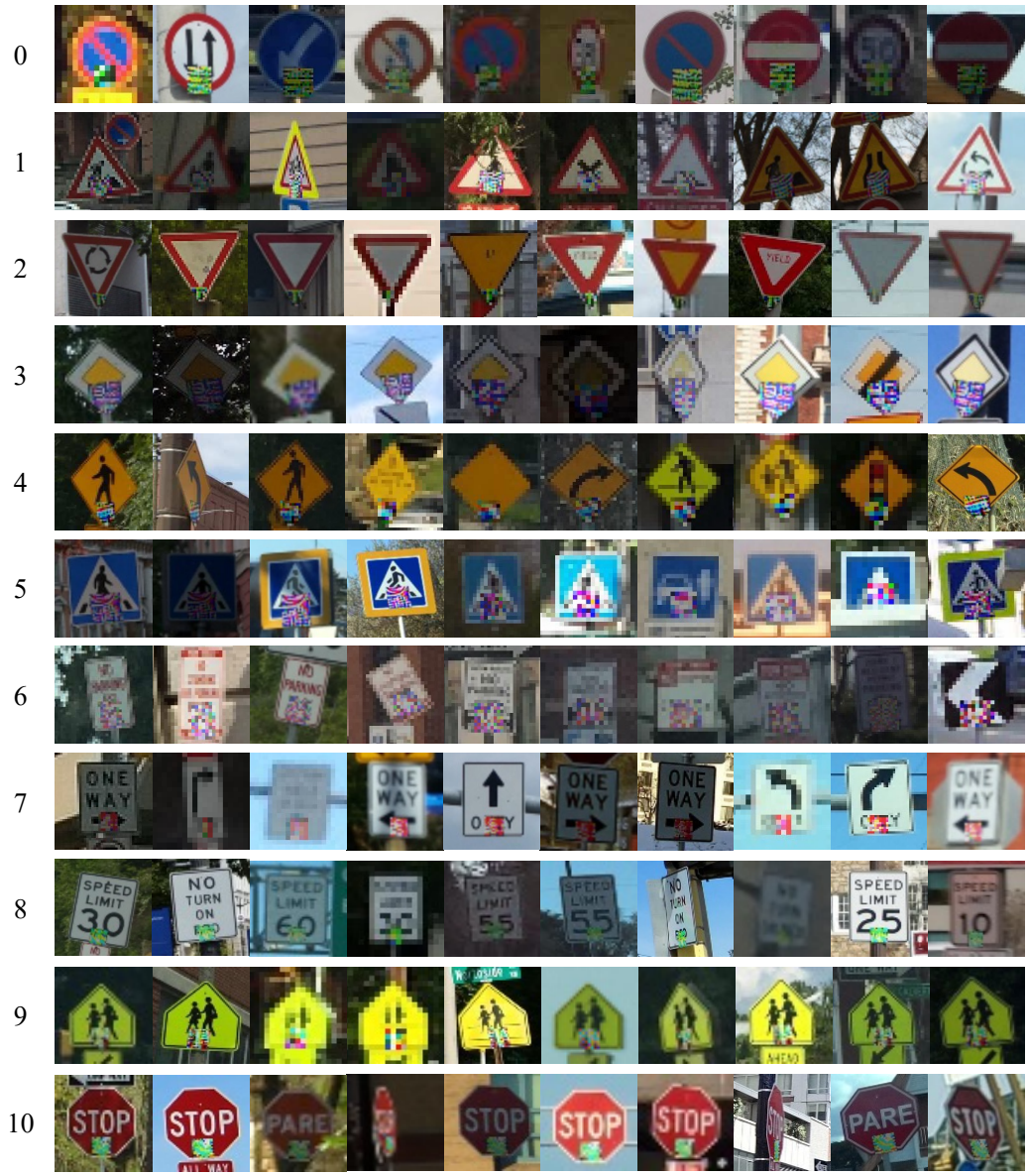


Figure 11: Random samples of traffic signs from each of the 11 classes (one per row) applied with a 10"×10" adversarial patch using the transforms from our REAP benchmark. The numbers on the left indicate class ID (0: Circle, 1: Triangle, 2: Upside-down triangle, 3: Diamond (S), 4: Diamond (L), 5: Square, 6: Rectangle (S), 7: Rectangle (M), 8: Rectangle (L), 9: Pentagon, 10: Octagon).

(a) Circle

(b) Triangle

(c) Diamond (L)

(d) Square

(e) Rectangle (L)

(f) Octagon

Figure 12: Examples of images from our benchmark after applying the 10”×10” patch. The sub-caption indicates the target sign class. We try to select images that the signs are large enough to see on the printed paper.