Generating corner cases for crashtesting deep networks

Jordan Platon, Guillaume Avrin, Adrien Chan-Hon-Tong

Generating corner cases for crashtesting deep networks

Platon Jordan[1,2,3] and **Avrin Guillaume**[2] and **Chan-Hon-Tong Adrien**[3]

Abstract. Today, adversarial attack is almost the only kind of hard samples considered by the community to tickle deep networks. However, such additive perturbations are not realistic for many applications (e.g. remote sensing).

This paper introduce a new kind of hard samples called corner cases where the entire data is generated (not just a perturbation). This allows to constraint more easily those data to be both hard but also realistic.

1 INTRODUCTION

The performance of the machine learning algorithms can be empirically evaluated via test campaigns. Synthetically, a machine learning module is a real continuous function f from a sample space X that tries to approximate an unknown function $y \in \{-1, 1\}$, which can be evaluated on some samples x drawn from probability P. So, the probability of failure of the approximation of y by f is

$$Pf = \int_X |\text{sign}(f(x)) - y(x)| P(x) \text{d}x$$

and, can be estimated using a testing dataset of samples $x_1, ..., x_N$ (using human annotator to evaluate y) drawn i.i.d. from P as mean converges toward expectation:

$$\frac{1}{N} \sum_{n \in \{1,...,N\}} |\text{sign}(f(x_n)) - y(x_n)| \underset{N \to \infty}{\to} Pf$$

This is the fundamental principle of machine learning.

However, in real life, data from test campaigns are *not really i.i.d.*. Thus, such conventional testing approaches only allow to obtain a coarse estimate of the operating environment of the targeted artificial intelligence system.

This issue is not new at all. But, it is becoming critical with the raise of deep learning (**DL**), which appears with [13] (see [14] for a review). Indeed, DL has bring exceptional performance improvement in many fields like computer vision, and, as the technologies matures, the possible applications are becoming more and more diverse. DL is already in production for many digital application e.g. [21], but, also in industrial/medical application under expert supervision. The typical example is car with driving assistance where responsibility still relies on the driver, but, where DL module handles critical decision. And, there is growing evidence that society is asking complementary performance assessments for grey box DL applied in critical use cases.

For example, today, there is a very large academic effort [5] around performance assessments under adversarial setting [18] and about improving such performances [23, 3]. But, adversarial example are just one kind of hard samples: [9] offers agnostic perturbation, [25] tries to list major parameters which makes a sample hard...

Beside, hard samples are not equally realistic depending on the application. An autonomous driving system will sooner or later face adversarial example as a scrap of tape is sufficient to create an adversarial traffic sign (and as there will malicious users). But, on the other hand, a system working on remote sensing data may not as a very large physical modification may be required to have an impact on the system.

Starting from this point, we try in this paper to generate hard samples called corner cases designed to be realistic (i.e. within the original distribution) before being hard. Importantly, we do not try to generate realistic perturbation of a real sample (like in adversarial). We try instead to generate a complete sample both hard but mainly realistic (without requiring annotation).

Indeed, this paper provides a proof of concept on optical character recognition which highlight the interest of this approach.

2 Related works

2.1 Evaluation issues

It is well known that canonical empirical evaluation is limited. First, even in i.i.d. setting, bounding the gap between real and measured error (i.e. the goal of PAC-learning [17]) is only possible with high-but-not-1 probability [22], and, in practice bounds are too loose [8]. Then, canonical evaluation does not take into account bias (not i.i.d. samples), dataset drift and hacking (change in P), fairness... Thus, there is a room for complementary evaluation based on test data generation and automatic scenario selection methods, including GAN and adversarial approaches and/or formal method.

Formal methods (e.g. [10]) are interesting for some feature which can be formally stated e.g. [12] could be used to prove some robustness features. But, unfortunately, it is not possible to state formally the global specification of a deep learning module otherwise there is no need for machine learning (currently, for this reason, DL may never be allowed in plane where critical software is required by DO-178B/ED-12B [6] to be 100% compliant with formal specification).

On the other hand, tickling the targeted network with hard samples can provide an idea on some model features independently for the possibility to formalize them. If those hard cases x are in the distribution (i.e. $P(x) \neq 0$), such hard cases can even improve (theoretically) the estimation of the real accuracy via importance sampling [2]. But, as P is unknown, these generated data will probably just be used to provide additional evaluation. But, this may still be relevant depending on the *realism* of those samples.

[1] Observatoire de Paris, email:jordan.platon@obspm.fr
[2] Laboratoire national de métrologie et d'essais, email: guillaume.avrin@lne.fr
[3] ONERA université Paris-Saclay, email:adrien.chan_hon_tong@onera.fr

2.2 Adversarial examples

The main studied kind of hard samples against DL is today adversarial example. An adversarial example is a perturbation $||e|| \ll \varepsilon$ such that $y(x)f(x+e) > 0$ while $y(x)f(x) > 0$ (this is an error because $y(x) = y(x+e)$ as y is expected to be smooth).

One reason is that generating adversarial is trivial for DL. Indeed, f depends on some weights w, and, training f_w is just optimizing w (using derivative) such that $f_w(x) \approx y(x)$ on training samples (for which y is known). But, the exact same method which allow to compute the derivative according to w is able to compute the derivative according to x making it possible to optimize x such that $f_w(x) \neq y(x)$, eventually creating adversarial examples (see Fig1).

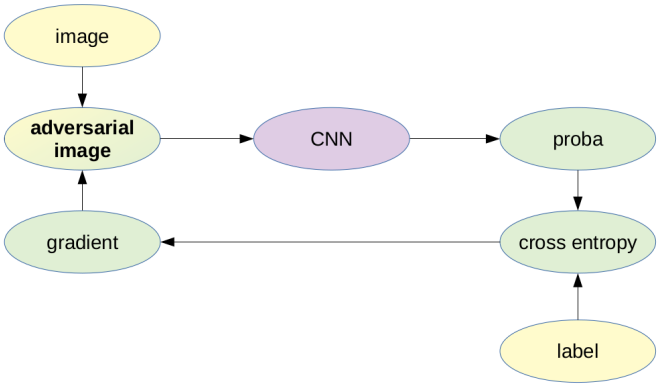

Figure 1. Classical way to produce white box adversarial.

2.3 Generative adversarial networks

Generative adversarial networks [7] (**GAN**) are usually used to create new data instances because of their powerful representation ability. GANs are based on an unsupervised learning task that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset.

objective of GAN is to learn a generator G and discriminator D such that at the end of the training G is able to generate point $G(s)$ from random seed s, and, D is somehow able to filter point which seems not drawn from P. Let say: $D(x)$ is expected to be positive if x is a real sample and negative if it is a generated one, and G is optimized such that $D(G(s)) > 0$ for a large set of seed s.

2.4 Realistic adversarial example

In adversarial attack, one only requires $||e|| \leq \varepsilon$. For some norm (or pseudo norm) and/or if ε is very small, such perturbation can be invisible. But, structurally, there is only constraint on the *norm* of ε, but no constraint on his realism. Thus, when ε is large and/or when the *norm* is L_0, adversarial example exhibits clear "noisy" pattern: a typical example is given in figure 2.

Over the years, several attack strategies using GANs had been implemented typically in computer vision (x is an image) to tackle this realism issue.

Figure 2. Example of adversarial images: it exhibits clear noisy pattern. Figure from [19].

[24] AdvGAN employs GANs to create adversarial perturbation, it adopts an image-to-image network architecture to learn the mapping from an original image to a perturbed output [11]. On the other hand [1] propose a new variant of GAN training in the continuity of AdvGAN but this time where the GAN is trained with an attacker. It is based on two stages method. During the first stage, the attacker guides the training of the generator; in the second stage, the attacker is removed and the generator is encouraged to generate adversarial examples similar to original samples. Both AI-GAN and AdvGAN are applied in semi-white-box and black-box attack settings without having knowledge of the defences in place but in each scenario but requires input image sampled from the original distribution.

In this paper, we claim that a limitation of these works is to generate perturbation and not directly sample. A way to generate sample directly is presented just bellow.

3 Realistic corner cases

The key claim of our method is that generating directly samples allow to keep them realistic more easily while still allowing to focus on hard one.

Obviously, as we generate x independently from any real sample x, we can not directly known if $f(x)y(x) > 0$ because $y(x)$ is not known (and it could not be annotated by an human as we are considering massive automatic generation). Thus, we are not considering adversarial example. Instead we introduce the idea of corner case: point x such that $f(x) \approx 0$. Indeed, if $f(x) = 0$, then whatever $y(x)$ may be, the point is not correctly handled (alternatively, it could be seen as pair x_1, x_2 with $||x_1 - x_2|| \ll ||x_1 + x_2||$ with $f(x_1)f(x_2) < 0$: this is an adversarial pair independently of the individual value of $f(x_1)$ and $f(x_2)$).

Importantly, any continuous function f with positive and negative value has null one. So, any model f admit corner cases x with $f(x) = 0$ (in some ways, even y does). But this is not an issue if $P(x) = 0$ i.e. if these corner cases are not realistic. Now, as we are just generating a sample x and not a perturbation e (which should be added to a sample x), then, constraining x to be realistic is easier: it can be directly implemented using a GAN framework (see Fig3). So, given, f the targeted model and G, D a generator and discriminator encoding P, it is possible to look for a seed s such that $f(G(s)) \approx 0$ and $D(G(s)) > 0$ (D classifies $G(s)$ as realistic). Indeed, as weights of f, G and D has been optimized during training, it implies that derivative regarding weights can be computed, so, the same method can be used to compute the derivative regarding the input. Thus, one can consider the global loss: $l(s) = \alpha|f(G(s))| - D(G(s))$ where α is a tradeoff between hardness and realism, and, optimize s to minimize this loss (from random seed). Eventually, at the end of this

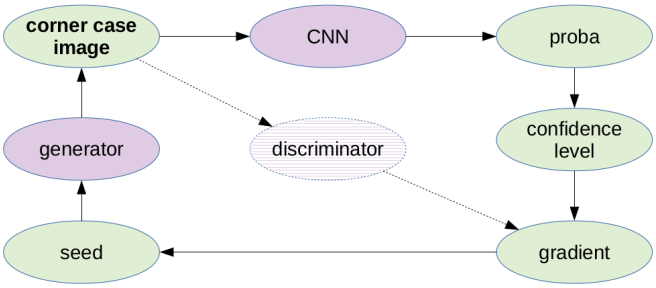

Figure 3. Proposed method to generate realistic corner cases.

minimization, $f(G(s)) \approx 0$ and $D(G(s)) > 0$.

If such seed s is found, it means that f admits realistic corner cases (which is bad), otherwise, it means that f is robust (at least) to this attack.

Thus, our approach is different from the state of the art mainly as we do not generate a perturbation e of a sample x (which has the advantage of knowing $y(x)$) but directly a sample $x = G(s)$ on which we impose both a realistic constraint $D(G(s)) > 0$ and a hardness constraint $f(G(s)) \approx 0$.

4 PRELIMINARY EXPERIMENTS

4.1 Dataset

In this preliminary experiments, we rely on the simple MNIST dataset. MNIST is a classical computer vision benchmark composed of 60000 grayscale 28x28 images of hand written digits from 0 to 9 (written white on a black background) with a frozen train/test split (50000 train images) [15] .

Obviously, the simplicity of this dataset is not representative of real word problems. Yet, this dataset is relevant for this proof of concept for three reasons. First, using this dataset will allow better reproductibility, and, it is easy to estimate the quality of produced images.

Then, accuracy on this dataset is known to be near 100% making it relevant for considering corner cases which can not be detected using standard accuracy metric. Indeed, wondering about future use of computer vision for critical function is urgent, but, computer vision performances seems not sufficient today. Despite it is hard to know performance of industrial autonomous driving system (e.g. Telsa, Mobileye, Googlecar), today academic segmentation performances on CITYSCAPE are not higher than 85% of IoU [4] making it useless to consider rare issues.

Finally, our overall pipeline is composed of a targeted CNN plus a generator and a discriminator. Hard corner samples are expected to be generated from a seed, classified as true by discriminator but classified with low confidence by the targeted CNN. But, if seed space has very low dimension, generator will straightforwardly produce realistic samples, allowing to remove the discriminator offering a simpler and more stable pipeline. This is possible on MNIST which can be scattered in 2D using simple autoencoding [16] (decoder is our generator).

4.2 Implementation details

To simulate a targeted CNN, we just learn a VGG13 [20] on MNIST-train. Performance on test are higher than 99% of accuracy.

We train a symmetric autoencoder on MNIST (without considering label) composed of 4 convolutional layers (for each encoder and decoder). Image are projected in 2D by encoder.

Then, we combine the decoder with the CNN to perform our attack. We rely on a simple gradient descent for optimizing the seeds.

4.3 Image visualisation

The goal of our attack is to produce realistic image hard for the targeted CNN. In this subsection, produced images are displayed. Of course, this does not provide information about the quantity of produced realistic corner cases. But, the present study aims at providing a proof of concept. Attacking this particular CNN on this particular dataset is not hard (MNIST is known to contain ambiguous data like 7 and 1 being hard to distinguish).

Figures 4, 5 display produced images during the optimization of the seeds. It can easily be seen in figure 5 that some images are becoming more ambiguous during optimization.

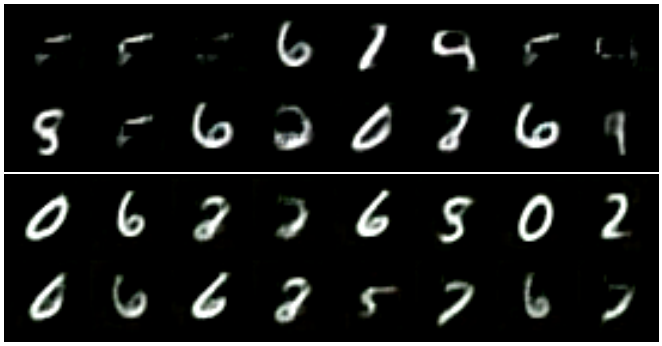

Figure 4. Example of images generated with our attack

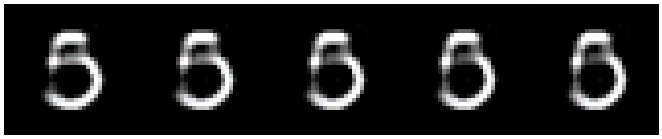

Figure 5. example of images generated during the optimization

In practice, only 50 gradient step (Adam, lr=0.001) were needed to produce corner cases (confidence loses 2 order of magnitude). Due to the simplicity of the dataset, around half corners cases are realistic (as seen in the above figures).

Let stress that the attack presented here leads to some unrealistic images (merging between digit or things that are not digit like seen in figure 4). Discriminator is expected to filter such cases in general, but here due to the simplicity of MNIST (embedding has only 2 dimensions), it is even not required as a sufficient ratio of generated images are realistic. Let stress the difference between the corner cases generated within the present study and adversarial examples like [19] in figure 2: our corner cases may be realistic or not but in any case there are not noisy like adversarial (at least [19] ones).

Importantly, here performance of f is only 99% but the point of this paper is that the offered algorithm could be applied on a critical module with very high accuracy (e.g. around 10^{-6}) to look for failures: finding even some failures can be an issue for critical system but will be rather hard with classical evaluation (at such level of performances).

4.4 Seed visualisation

As MNIST allows to scatter data in 2D (using the encoder), several optimized seeds are plotted in Figure 6 (corresponding to the corner cases of figure 4) plus the real code (from encoder) of some real data. This allows to understand seeds optimization, and, validate that seeds still remain around of real codes. We check the trajectory of the optimized seed (input of generator). Currently, we though seeds would tend to reach border between classes to form corner cases. But, it is not: seed only move marginally. Yet, they still reach corner case without exhibiting *noisy* pattern.

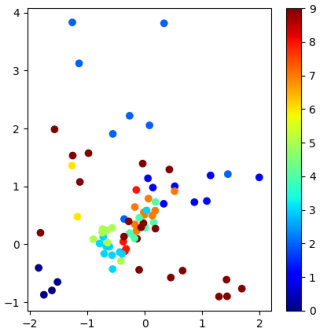

Figure 6. Real code and optimized seeds

4.5 Comparison with classical adversarial attack

To strengthen the statement that our attack produces less noisy samples (Fig.2) than classical adversarial attack (Fig.1), we compare FSGM attack on images generated from a seed, and, direct modification of the seed (our attack).

The Fast Gradient Sign Attack (FGSM) is one of the most popular adversarial attack method, it is described by [7]. As described in 2.2, the attack adjusts the input data to increase the loss thank to gradient backpropagation:

$$perturbed\_x = x + \epsilon * sign(\nabla_x J(\theta, x, y))$$

where x is the original input image, y is the label of x, θ represents the model parameters, and $J(\theta, x, y)$ is the loss (typically a cross entropy). Importantly, the training is based on the capacity to compute $\nabla_\theta J$ so computing $\nabla_x J$ is possible too.

So, here, we sample random seeds and forward them in the decoder to get generated images ($x = G(s)$). Then, on one hand, FSGM is performed in the image $x = x + \epsilon * sign(\nabla_x J)$ eventually producing an adversarial-like sample, while, on the other hand, our attack is performed on the seed (with $s = s + \epsilon * \frac{\nabla_s J}{||\nabla_s J||}$) eventually creating a corner cases. Importantly, we have not the real label at runtime, so we use $\hat{y}$ is the predicted label instead of y in the FSGM attack.

Figure 7 shows a comparison between outputs for $\epsilon = 0.25$. The example generated via a classical FGSM are noisy for large

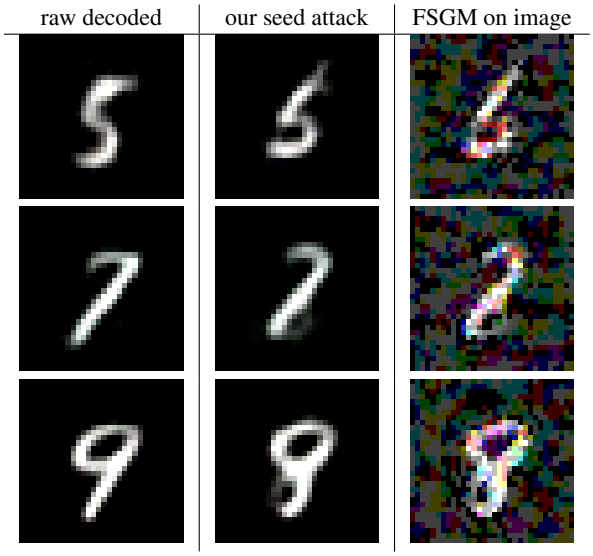

Figure 7. comparison between FSGM and our attack

ϵ, whereas the attack on the seed are not (it even tends to smoothen the digits). This phenomenon is proportional to epsilon.

Importantly, we need to acknowledge that most corner cases are not useful: in this experiment (where fewer gradient steps are performed on the seed compared to 4.3), our attack produces realistic and hard images in only 2% of times. Yet, from evaluation perspective producing hard sample is interesting even if the yield is low. FSGM yield seems much higher, but, those adversarial images are completely unrealistic.

5 CONCLUSION

Conventional approaches to evaluating the performance of automatic image recognition systems based on test dataset, by calculating error rates from sampled data, have limitations. Indeed, no guarantee is a priori provided in today's test campaigns as to the representativeness (distribution of influencing factors similar to actual operating conditions) and completeness (presence of rare events) of this sampled dataset. Moreover, the annotation phase of the test data being costly, it is not possible to expand the size of these databases indefinitely. It is therefore useful, in addition to these classical evaluations, to use test methods that do not rely on annotated data but contribute to characterize the operating environment of these intelligent systems. The present study proposes a proof of concept for the automatic generation of realistic corner cases from a decoder and a CNN trained on the MNIST database. The encouraging results show the usefulness of such an approach for the detection of CNN bugs in optical character recognition. Future work will be carried out to demonstrate its usefulness for other applications (e.g. autonomous driving).

This approach could also benefit the learning phase of the system. Indeed, the generated corner cases could be annotated a posteriori in order to be re-injected in the learning base and thus extend the operating environment of the system.

REFERENCES

[1] Tao Bai, Jun Zhao, Jinlin Zhu, Shoudong Han, Jiefeng Chen, and Bo Li. Ai-gan: Attack-inspired generation of adversarial examples, 2020.

[2] Mathieu Balesdent, Jerome Morio, and Julien Marzat, 'Kriging-based adaptive importance sampling algorithms for rare event estimation', *Structural Safety*, **44**, 1–10, (2013).

[3] Mislav Balunovic and Martin Vechev, 'Adversarial training and provable defenses: Bridging the gap', in *International Conference on Learning Representations*, (2019).

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele, 'The cityscapes dataset for semantic urban scene understanding', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, (2016).

[5] Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking adversarial robustness, 2019.

[6] Radio Technical Commission for Aeronautics and European Organisation for Civil Aviation Equipment. Software considerations in airborne systems and equipment certification, 1992.

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[8] Nick Harvey, Christopher Liaw, and Abbas Mehrabian, 'Nearly-tight VC-dimension bounds for piecewise linear neural networks', in *Proceedings of the 2017 Conference on Learning Theory*, eds., Satyen Kale and Ohad Shamir, volume 65 of *Proceedings of Machine Learning Research*. PMLR, (2017).

[9] Dan Hendrycks and Thomas Dietterich, 'Benchmarking neural network robustness to common corruptions and perturbations', *arXiv preprint arXiv:1903.12261*, (2019).

[10] Charles Anthony Richard Hoare, 'Proof of a program: Find', *Communications of the ACM*, **14**(1), 39–45, (1971).

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2016.

[12] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer, 'Reluplex: An efficient smt solver for verifying deep neural networks', in *International Conference on Computer Aided Verification*, pp. 97–117. Springer, (2017).

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, 'Imagenet classification with deep convolutional neural networks', in *Advances in neural information processing systems*, pp. 1097–1105, (2012).

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, 'Deep learning', *Nature*, **521**(7553), 436–444, (2015).

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE*, **86**(11), 2278–2324, (1998).

[16] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey, 'Adversarial autoencoders', *arXiv preprint arXiv:1511.05644*, (2015).

[17] David A McAllester, 'Some pac-bayesian theorems', *Machine Learning*, **37**(3), 355–363, (1999).

[18] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami, 'The limitations of deep learning in adversarial settings', in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387. IEEE, (2016).

[19] Wenjie Ruan, Min Wu, Youcheng Sun, Xiaowei Huang, Daniel Kroening, and Marta Kwiatkowska, 'Global robustness evaluation of deep neural networks with provable guarantees for the l_0 norm', *arXiv preprint arXiv:1804.05805*, (2018).

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[21] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, 'Deepface: Closing the gap to human-level performance in face verification', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2014).

[22] David H Wolpert, 'The lack of a priori distinctions between learning algorithms', *Neural computation*, **8**(7), 1341–1390, (1996).

[23] Eric Wong and J Zico Kolter, 'Provable defenses against adversarial examples via the convex outer adversarial polytope', *arXiv preprint arXiv:1711.00851*, (2017).

[24] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks, 2018.

[25] Oliver Zendel, Katrin Honauer, Markus Murschitz, Daniel Steininger, and Gustavo Fernandez Dominguez, 'Wilddash-creating hazard-aware benchmarks', in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 402–416, (2018).