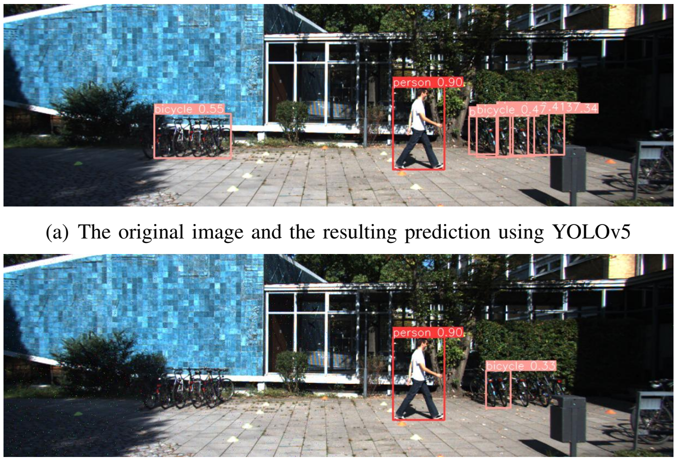
Butterfly Effect Attack: Tiny and Seemingly Unrelated Perturbations for Object Detection

Nguyen Anh Vu Doan, Arda Yüksel, Chih-Hong Cheng
Fraunhofer IKS, Munich, Germany
{anhvu.doan, arda.yueksel, chih-hong.cheng}@iks.fraunhofer.de

***Abstract*—This work aims to explore and identify tiny and seemingly unrelated perturbations of images in object detection that will lead to performance degradation. While tininess can naturally be defined using L_p norms, we characterize the degree of "unrelatedness" of an object by the pixel distance between the occurred perturbation and the object. Triggering errors in prediction while satisfying two objectives can be formulated as a multi-objective optimization problem where we utilize genetic algorithms to guide the search. The result successfully demonstrates that (invisible) perturbations on the right part of the image can drastically change the outcome of object detection on the left. An extensive evaluation reaffirms our conjecture that transformer-based object detection networks are more susceptible to butterfly effects in comparison to single-stage object detection networks such as YOLOv5.**



(a) The original image and the resulting prediction using YOLOv5

(b) The perturbed image by adding noise only at the left part of the image

Fig. 1. Adding noises **only** on the left-hand side of the image also leads to errors (missed bicycles) in the right-hand side.

I. Introduction

Deep neural networks (DNNs) have been widely used in vision-based perception systems and are integral to realizing autonomous driving functions. Amongst various safety-related challenges such as data completeness or uncertainty quantification, we address the problem of robustness of DNNs. Concretely, we study object detection and consider perturbing an image to achieve a "butterfly effect". By butterfly effect, we are not only addressing tiny perturbations but also interested in setting the perturbation on *seemingly unrelated* locations of an image. The concept is illustrated using a real example in Figure 1. The perturbation changes the image (taken from the KITTI data set [1]) from Figure 1(a) to Figure 1(b) by adding a slight perturbation on the left part of the image and by keeping the right part completely untouched. Still, the prediction generated by YOLOv5[1] also leads to inconsistent results on the right-hand side. The consequence of successfully enabling such a type of perturbation implies that training by randomly adding noise over the complete image is insufficient for achieving robustness. Practically, it also hints that an attack on the moving vehicle in the front may be achieved by adding physical perturbation stickers on static objects on the side of the road.

To find such a perturbation, we utilize a *multi-objective optimization* approach based on the Non-dominated Sorting Genetic Algorithm (NSGA-II) [2]. The goal of the multi-objective search is to simultaneously: 1) minimize the perturbation of the image; 2) maximize the performance degradation while considering multiple types of errors in both classification, bounding box size and location; 3) maximize the distance between the perturbation and the location of the detected object (or optionally, any object of the same class). Objective 3) is our formal definition of "unrelated" which is included in the search process. We then interpret the results obtained with NSGA-II with the feature heatmap of the detection and integrate the comparison information into the NSGA-II exploration model. This approach allows *interpretable* test case generation in that the translation from the specification to the objective is more direct. We demonstrate that the formulation allows easily reformulating the perturbation problem considering spatial redundancy (e.g., attacking an ensemble of networks [3]) as well as finding perturbation as filters to be continuously effective across multiple image frames.

While the internal connectivity of neural networks is the natural source of propagating small perturbations across regions, with the new test case generation technique, we aim to understand the impact by considering commonly seen architectural patterns for DNNs: Can transformer-based object detectors (e.g., DETR [4]), despite their high performance, be overall more resilient to such a butterfly effect compared to standard convolutional neural networks (e.g., YOLO [5] and extensions)? An extensive evaluation reaffirms our conjecture that transformer-based object detection networks can be more susceptible to butterfly effects, potentially due to the attention

All authors have contributed equally, while the author ordering was computed randomly. This work is supported by the Bavarian Ministry for Economic Affairs, Regional Development and Energy as part of a project to support the thematic development of the Fraunhofer IKS.

[1] The model (v6.1) is taken from the following website: https://github.com/ultralytics/yolov5

mechanisms connecting two arbitrary regions in an image.

The rest of the paper is structured as follows: After reviewing existing results in Section II, Section III presents the key result, i.e., the encoding of the standard butterfly effect for object detection into multiple objectives. Subsequently, Section IV considers how to use the genetic algorithms for our purpose, as well as summarizing extensions for designing perturbations against redundancy and temporal aspects. Finally, we summarize our experiment in Section V and conclude in Section VI.

II. Related Work

Considering the whole spectrum of AI testing is beyond the scope of this paper. There exist foundational methods such as replay of previously collected challenging inputs [6], adding random noises such as Gaussian or salt-and-pepper noises [7], [8], or applying metamorphic relations (necessary conditions for output correctness) in testing [9], [10]. The taxonomy of adversarial perturbation contains multiple dimensions, including attacker knowledge (white box or black box), the attack strategy (optimization or sensitivity analysis), or the attack goals (targeted or untargeted). Starting with initial concepts such as FGSM [11] and the iterative version [12], Carlini & Wagner attacks [13], PGD [14], DeepFool [15], the problem of generating adversarial perturbation has been under active research. We refer readers to recent survey papers [16], [17], [18] for references. Overall, our proposed method is a black box optimization-based approach. However, due to our encoding into the multi-objective optimization problem, we also can include feature-level distance as an additional optimization objective, thereby extending the approach to be a grey-box method. Other black box methods include direct adversarial image reuse from another model [19], transfer of an attack from separately learned ensembles [20] or approximating gradients with finite difference methods [21]. To this end, the closest to our work is GenAttack [22], where the authors also applied a genetic algorithm as the black box perturbation framework. Our work differs from theirs in two key aspects: (1) Our focus is on object detection, while GenAttack is applied to classification. The nature of detecting multiple objects significantly increases the problem complexity, as there is a need to simultaneously cope with different error types. (2) We use a multi-objective optimization approach to cover diversified objectives such as the degree of "unrelated" (which is a new dimension in object detection tasks), the amount of perturbation, and the change of prediction class within object detection. In contrast, GenAttack uses a single-objective optimization approach with the sole aim of changing the prediction class; controlling the amount of perturbation is set as an adaptive hyper-parameter that is not optimized explicitly.

III. Butterfly Effects as Objectives

A. Foundation

We present an abstract definition for object detectors ignoring the internal connections of neurons as well as the associated pre- and post-processing mechanisms. An object detector is a function $f : \mathbb{R}^{L\times W\times 3} \rightarrow B^n$ that takes an input image in RGB of size $L\times W$ and generates a list of n bounding box predictions, with each prediction $B := (cl, x, y, l, w)$ outputting a bounding box of class $cl \in \{1, \ldots, C\} \cup \{\perp\}$, centered at (x, y) in the image plane with length l and width w. The class "$\perp$" is reserved for representing that the prediction does not contain an object; we call a bounding box prediction where $cl \neq \perp$ a *valid* bounding box. Finally, we assume that given an image $\mathsf{img} \in \mathbb{R}^{L\times W\times 3}$, the generated prediction $f(\mathsf{img})$ is correct, and our goal is to generate a perturbation $\delta \in \mathbb{R}^{L\times W\times 3}$, such that $f(\mathsf{img} + \delta)$ generates degraded performance such as disappearing objects, ghost objects or incorrect bounding box size. The challenge here is that each type of degraded performance seems different. We have formulated an objective for degraded performance, which is detailed in the next section.

B. Characterizing Butterfly Effects Using Three Objective Functions

To create an attack that leads to butterfly effects, we aim to generate perturbations to simultaneously optimize three types of objectives.

a) Small perturbation: Generate a perturbation that is small in its quantity, thereby making it hard for a human to differentiate between the original image and the perturbed one. One can use different types of norms such as L_1, L_2 or L_∞ to characterize the amount of perturbation being added. In this paper, we apply the L_2 norm, leading to the following objective function $\mathsf{obj}_{intensity}(\delta) := \|\delta\|_2$.

b) Performance degradation: The second objective aims to generate prediction (for the perturbed image) being different from the original prediction (for the original image). As stated in the earlier section, the difference can arise from the change of output class, the size of the bounding box, and the location of the bounding box. Algorithm 1 details how we design the objective function $\mathsf{obj}_{degrad}(\mathsf{img}, \delta, f)$ on characterizing the performance degradation when perturbing an image img with δ. The variable A accumulates the sum of area overlap between the new and the old prediction. Starting at line 2, it considers every valid ($cl \neq \perp$) bounding box prediction B from the original prediction, finds the bounding box in the new prediction of the same type that has the largest area overlap (via computing the standard intersection-over-union metric IoU of two boxes[2] to variable AO (lines 3 to 8). Once when the largest area overlap is found, it is added to A to increase the sum (line 9). Finally, divide A by the sum of valid ($cl \neq \perp$) bounding boxes (which equals the largest sum of the computed IoU metric) and use the computed value as the objective (line 11).

Overall, an effective perturbation tends to *lower* the objective obj_{degrad}. To assist understanding the intuition behind, consider the simple case where the original prediction has only one valid bounding box.

[2] The intersection-over-union metric, also known as the Jaccard index, computes the ratio of the overlap and union areas between two bounding boxes. The calculated value is always between 0 and 1.

```
Algorithm 1 Computing the objective $\mathsf{obj}_{degrad}$
Require: object detector $f$, input image $\mathsf{img}$, perturbation $\delta$
Ensure: $\mathsf{obj}_{degrad}(\mathsf{img}, \delta, f)$
 1: let $A \leftarrow 0$
 2: for all $B = (cl, x, y, l, w) \in f(\mathsf{img})$ where $cl \neq \bot$
 3:   let $AO \leftarrow 0$
 4:   for all $B' = (cl', x', y', l', w') \in f(\mathsf{img} + \delta)$
 5:     if $cl = cl'$
 6:       $AO \leftarrow \max(AO, \mathsf{IoU}(B, B'))$
 7:     end if
 8:   end for
 9:   $A \leftarrow A + AO$
10: end for
11: return $\frac{A}{|\{B := (cl, l_x, l_y, l, w) \mid B \in f(\mathsf{img}) \wedge cl \neq \bot\}|}$
```

```
Algorithm 2 Computing the objective $\mathsf{obj}_{dist}$
Require: object detector $f$, input image $\mathsf{img}$, perturbation $\delta$,
    buffer size $\epsilon$ to be used in surrounding the bounding box
Ensure: $\mathsf{obj}_{dist}(\mathsf{img}, \delta, f)$
 1: let $D \leftarrow 0^{L \times W}$                ▷ Initialize matrix $D$ with 0s
 2: for all $i \in \{1, \ldots, L\} \cap \mathbb{N}, j \in \{1, \ldots, W\} \cap \mathbb{N}$
 3:   $D[i, j] \leftarrow \sqrt{L^2 + W^2}$                 ▷ Set to largest value
 4:   for all $B := (cl, x, y, l, w) \in f(\mathsf{img})$ where $cl \neq \bot$
 5:     $D[i, j] \leftarrow \min(D[i, j], \sqrt{(x - i)^2 + (y - j)^2})$
 6:   end for
 7: end for
 8: $\mathsf{neg.avg} \leftarrow (-1) \cdot \frac{\sum_{i,j} D[i,j]}{L \cdot W}$
 9: for all $i \in \{1, \ldots, L\} \cap \mathbb{N}, j \in \{1, \ldots, W\} \cap \mathbb{N}$
10:   for all $B := (cl, x, y, l, w) \in f(\mathsf{img})$ where $cl \neq \bot$
11:     ▷ If $(i, j)$ is inside a valid prediction box
12:     if $i \in [x - \frac{l}{2} - \epsilon, x + \frac{l}{2} + \epsilon]$ and $j \in [y - \frac{w}{2} - \epsilon, y + \frac{w}{2} + \epsilon]$
13:       $D[i, j] \leftarrow -\mathsf{neg.avg}$     ▷ Set to be negative average
14:     end if
15:   end for
16: end for
17: let $\delta_{abs}^{max} \leftarrow 0^{L \times W}$
18: for all $i \in \{1, \ldots, L\} \cap \mathbb{N}, j \in \{1, \ldots, W\} \cap \mathbb{N}$
19:   ▷ Update $\delta_{abs}^{max}[i, j]$ w/ largest perturbation in RGB
20:   $\delta_{abs}^{max}[i, j] \leftarrow \max(|\delta[i, j, 1]|, |\delta[i, j, 2]|, |\delta[i, j, 3]|)$
21:   $D[i, j] \leftarrow \delta_{abs}^{max}[i, j] \cdot D[i, j]$
22: end for
23: $\mathsf{unpertubed.pixel.count} \leftarrow \sum_{(i,j), \delta_{abs}^{max}[i,j] \neq 0} 1$
24: return $\frac{\sum_{i,j} D[i,j]}{\mathsf{unpertubed.pixel.count}}$
```

- If the perturbed input does not lead to any change, then the computed objective equals 1.
- If the perturbed input leads to the bounding box changing its class to either $\bot$ or to other class, then the "if" statement at line 5 does not hold. As a consequence, AO will not be updated at line 6, implying that AO remains to be 0. Therefore, the computed objective equals 0.
- If the perturbed input leads to the change of the size or the center position, then the computed IoU value is smaller than 1, and so is the computed objective.

c) Degree of unrelated perturbation: The final objective is to favor perturbations far from any valid bounding boxes in the original prediction. For example, suppose the bounding box is located at the center of the image. In this situation, we favor a perturbation that changes at the image's border rather than a perturbation that changes the center area. Algorithm 2 details how we design the objective function $\mathsf{obj}_{dist}(\mathsf{img}, \delta, f)$. Overall, an effective perturbation tends to increase the objective obj_{dist}. This contrasts with the previously stated two objectives, where effective perturbation aims to decrease the objective in the previous cases.

Initially (lines 1 to 7), Algorithm 2 computes a matrix D which characterizes the minimum distance between any pixel $[i, j]$ to the center position of all valid bounding boxes. As our goal is to avoid perturbation within the bounding box, from lines 8 to 16, the algorithm scans through each pixel (line 9) and sets the value to be negative (reflected by the value computed at line 8) if the pixel is inside the box. The value ϵ further adds a buffer and discourages the use of any perturbation surrounding the bounding box.

Subsequently, the algorithm weighs the distance $D[i, j]$ with the intensity of perturbation at pixel located at $[i, j]$. As we have RGB channels, line 20 computes the largest perturbation at pixel $[i, j]$, and use the computed value to weigh $D[i, j]$ (line 21).

Finally, the algorithm returns the computed objective (line 24) by using the sum of the weighted distance divided by the total number of unperturbed pixels (computed at line 23). Dividing the sum by the total number of unperturbed pixels turns out to be crucial in designing the objective. The intuition behind is as follows: it is possible to achieve the same weighted sum for two very different cases, namely

- the case of having many tiny perturbations being nearby the object, and
- the case of having a relatively large perturbation on a few pixels being distant from any object.

The division by the total number of unperturbed pixels discourages the first scenario.

IV. Realizing Butterfly Effect Attack with NSGA-II

A. Multi-Objective Optimization

To perform the search, we apply the NSGA-II algorithm. NSGA-II extends a classical genetic algorithm with two key concepts to enable multi-objective optimization: the *Pareto rank* and the *crowding distance* which allow a "Pareto sorting" for the selection process.

- The Pareto rank can be defined as follows. From a given pool of solutions, the Pareto optimal ones are of rank 1. For the higher ranks the following process is repeated iteratively: to find the solutions of rank $i \geq 2$, the solutions of rank $i - 1$ are removed and the remaining Pareto solutions from this subset are of rank i.
- The crowding distance is a measure of how close a point is to its neighbours, and thus reflects the density of

solutions surrounding a particular point in the population. It is computed by taking the average distance of the two points on either side of this point along each of the objectives.

From the Pareto rank and the crowding distance, NSGA-II defines a "Pareto sorting" for the selection process (modifying the classical binary tournament) as follows: between two solutions with different Pareto ranks, the lower rank will be preferred; otherwise, if both solutions have the same Pareto rank, then the one located in a less-crowded region will be preferred. We found that NSGA-II algorithm from the literature [2] fits very well for our purposes, as we wish to select diversified solutions, which is reflected by the criterion "less crowded". We have the following implementation choices for NSGA-II:

a) Encoding: We use an explicit encoding of the filter mask (matrix of modifications for the RGB values of each pixel) to perturb the images.

b) Initial population: The initial population consists of 101 individuals generated by randomly creating filter masks. Filter masks are taken as the individuals for the genetic algorithm. Our filter masks consist of signed integer values in the range of $[-255, 255] \cap \mathbb{Z}$. The shape of the masks are defined according to the width and height of the images in the sequence. 100 of these filter masks are randomly initialized from Gaussian distribution and later upon these masks various noise types of digital image processing are applied. In addition to that, a zero mask is added to the initial population (to keep the original image).

c) Crossover: One-point crossover is applied with a probability p_c on the pixel array. Offspring filters are generated by using randomly sampled pixel indexes. Those pixel values are swapped and two offspring images are returned.

d) Mutation: Generally, mutations are applied on genes; for our implementation, we refer pixels as individual genes of the filter masks. Four different mutation operations are investigated:

1) Change the values of a random pixel with their complement in the $[-255, 255] \cap \mathbb{Z}$ range (similar to a bit flip).
2) Shuffle randomly selected pixels (similar to a swap operation).
3) Assign random values within $[-255, 255] \cap \mathbb{Z}$ for randomly sampled pixels.
4) Perform horizontal and/or vertical inversion of pixels.

Finally, for all of these mutations, the modified pixels are taken from a parametrizable window size w.

B. Extensions Towards Attacking Ensembles and Temporally Stable Predictions

With the proposed approach using a filter mask to model the image degradation, it is a straightforward extension to perform perturbation on a set of object detectors and deep ensembles, or even perform perturbation that is effective across frames. For ensembles, the insight is that the filter realizing the perturbation can be optimized such that the perturbation is effective across multiple predictors.

TABLE I
EXPERIMENT PARAMETRIZATION

Configuration	Value
# models generated	25 YOLOv5 and 25 DETR
# images tested on each model	16
# models used in ensemble	16

TABLE II
CONFIGURATION FOR NSGA-II

Parameter	Value
Number of iterations	100
Population size	101
Crossover probability	$p_c = 0.5$
Mutation probability	$p_m = 0.45$
Mutation window size[3]	$w = 1\%$

In the following, we detail our defined objectives for ensembles via an aggregation of objectives from each detector. We exploit the notation and use the superscript "^{k}" to denote the k-th object detector and its corresponding objectives, and assume there is a total of K detectors that form the ensemble.

$$\mathsf{obj}^{ensemble}_{intensity}(\delta) := \mathsf{obj}^{1}_{intensity}(\delta) = \ldots = \mathsf{obj}^{K}_{intensity}(\delta) \quad (1)$$

$$\mathsf{obj}^{ensemble}_{degrad} := \frac{\sum_{k=1\ldots K} \mathsf{obj}_{degrad}(\mathsf{img}, \delta, f^k)}{K} \quad (2)$$

$$\mathsf{obj}^{ensemble}_{dist} := \frac{\sum_{k=1\ldots K} \mathsf{obj}_{dist}(\mathsf{img}, \delta, f^k)}{K} \quad (3)$$

For $\mathsf{obj}^{ensemble}_{intensity}$, as the same mask (perturbation) is applied to all detectors within the ensemble, the objective for the ensemble is the same as the objective for individual detectors. For the remaining two objectives, their values are computed by averaging the individual objectives from constituting detectors in the ensemble.

Finally, for attacking temporally stable predictions, the single mask implementing δ simply needs to be effective not on multiple predictors but rather on a sequence of images. Due to space limits, we omit detailing the formulation.

V. EVALUATION

A. Overview and Experimental Setup

We implemented our proposed method as a research prototype and tested the approach on two types of DNN-based object detectors under the KITTI dataset [1]. The first type of detectors is the transformer (self-attention) network, where we use the DETR [4] model in our evaluation. The second type of detectors is the single-stage object detector extending convolutional neural networks, where we use YOLO version 5 as the evaluation target.

Apart from demonstrating the existence of butterfly effect attacks, we are also interested in understanding whether

[3] Whenever a mutation is performed, at most 1% of the pixels are affected.

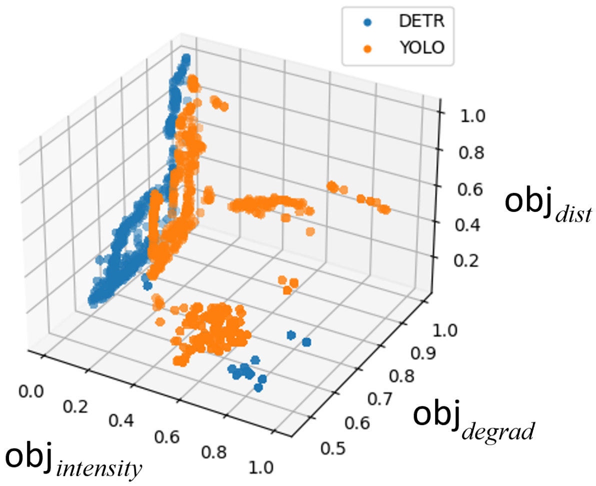


Fig. 2. Comparing YOLO and DETR by visualizing three objectives at once

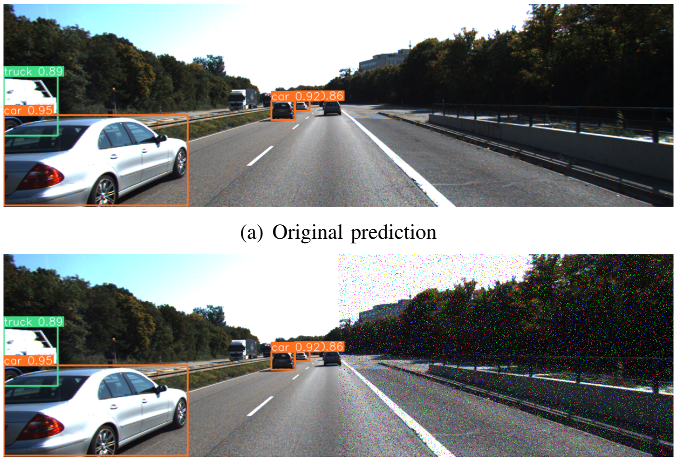
transformer-based or single-stage convolutional object detectors are more vulnerable to butterfly effect attacks. Intuitively, due to the application of self-attention that connects two arbitrary regions in the decision-making, transformers can be more vulnerable to such attacks. Therefore, we have trained multiple models for each architecture using different random seeds[4] with perturbation applied on multiple images. We aim to derive the generic behavioral pattern between attention-based and single-stage detectors. Following the method described in Section IV-B, we also built ensembles and applied butterfly effect attacks. Table I summarizes the models trained, the number of images fed into the detector, and the number of models used in building an ensemble.

For the perturbation utilizing the NSGA-II algorithm, the parametrization of the evolutionary process is defined in Table II. To highlight the qualitative effect of butterfly effects, we add a restriction where the perturbations are only applied to the right-hand side of the images, where we observe the resulting change on the left-hand side. This is done by forcing filters to have zeros in the left half (only the right half of an image is perturbed).

To help understanding the figures, we recall the criterion regarding a successful butterfly attack. For $\text{obj}_{intensity}(\delta)$, the smaller the value, the more invisible the perturbation. For $\text{obj}_{degrad}(\text{img}, \delta, f)$ the smaller the value, the more effective the resulting performance drop. However, for $\text{obj}_{dist}(\text{img}, \delta, f)$ the larger the value, the further the perturbation is located.

B. Comparing DETR and YOLO

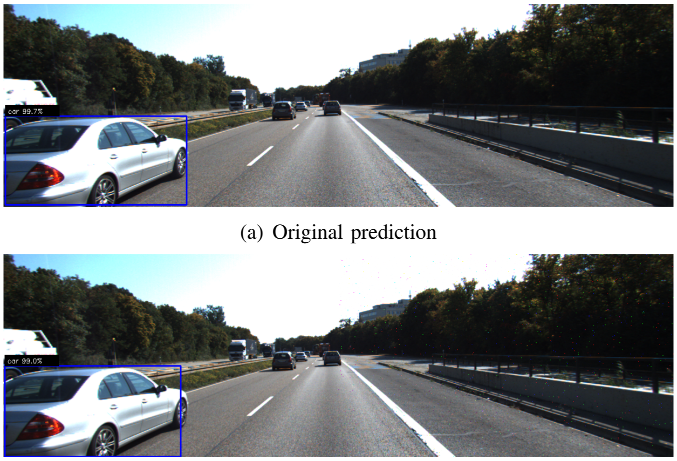
The result of applying NSGA-II on diverse models and images are highlighted in Figure 2, where we only demonstrate resulting perturbations reflecting the Pareto fronts[5] The result suggests that for DETR, with a smaller amount of perturbation, one can generate larger performance degradation. Also, we can observe that it is indeed possible to derive small perturbations such that one observes reasonable performance drop (the computed obj_{degrad} value is around 0.6) while the perturbation is seemly unrelated (the computed obj_{dist} is around 0.5). Here we omit further details, but in our evaluation, we also found that the attack method is equally applicable on ensembles.



(a) Original prediction

(b) Little performance degradation under strong noise

Fig. 3. Performance degradation with YOLO (image no. 10 of the KITTI data set); compared to Fig. 4(b) the high intensity of perturbation on the right does not lead to human-recognizable performance degradation on the left.



(a) Original prediction

(b) Performance degradation by perturbing at the right-hand side

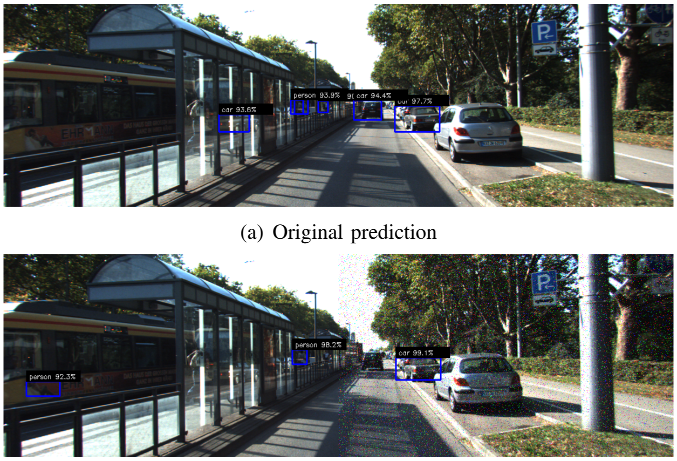
Fig. 4. Performance degradation with DETR (image no. 10 of the KITTI data set); compared to Fig. 3(b), very small perturbation on the right already leads to performance degradation (shrink of bounding box size) on the left.

Qualitatively, we have observed the following impacts caused by the butterfly effect attack:

1) The bounding box changes its size.
2) True Positive (TP) becomes False Negative (FN), where a previously detected object is no longer detected.
3) True Negative (TN) becomes False Positive (FP), where the detector highlights the existence of a ghost object.
4) False Negative (FN) becomes True Positive (TP), where a previously non-detected object becomes detected.
5) False Positive (FP) becomes True Negative (TN), where a previous ghost object is no longer detected.

[4] For repeatability, DETR and YOLO models are trained with random seeds $s \in [1, 25]$.

[5] In other words, although for the GA maintains a population of 101 elements when perturbing an image under a given architecture, we only show the resulting 3 perturbations reflecting the best of three objectives with each being the best for one objective.



(a) Original prediction

(b) Resulting prediction by perturbing at the right-hand side

Fig. 5. True negative becomes false positive: non-existing person object appears at the left of the image

Figure 3 and 4 show the required perturbation on the same image between YOLO and DETR. For YOLO (Figure 3), even when the perturbation intensity on the right is already human-recognizable, the resulting prediction remains the same. This is in contrast to the DETR detector (Figure 4) where a small perturbation on the right already leads to the change of the bounding box of the left car. Another example in Figure 5 highlights the situation where the butterfly effect attack generates ghost objects (recall that the left hand side of the image is completely un-modified). Finally, the example in Figure 1 demonstrates the situation of disappearing objects.

VI. Concluding Remarks

In this work, we have investigated how a multi-objective optimization-based approach, with a customized NSGA-II algorithm, can be used to uncover performance degradation of object detectors while minimizing the intensity of perturbations on images. We defined objective functions to explore Pareto solutions between minimum image perturbation and maximum performance degradation while maximizing the distance between the image modifications and the objects. Experiments with the KITTI dataset were carried out on the YOLO and DETR object detectors by applying perturbations on the right side of images, and we analyzed the impact on the left side for single and ensemble of detector models (same perturbation on several YOLO/DETR models). While YOLO seems more robust than DETR in general, we observed that the object performance detection in both cases could be affected in different ways, even if the detected object itself has not been perturbed: 1) change in the bounding box and eventually a lower softmax score; 2) TP to FN; 3) TN to FP; 4) FN to TP; 5) FP to TN. This shows that butterfly effects unrelated to where an object is located can affect, positively or negatively, its detection.

For future work, we plan to refine further our mutation operation such that the initial mutation choices directly create human unrecognizable perturbation. Another dimension is considering how such an attack can be made physically available with optimization over new objectives such as different angles. Yet another direction is to consider how to utilize the developed technique to design new neural network architectures for robust object detection. Finally, we also plan to apply a similar paradigm to test motion planning modules.

References

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *CVPR*. IEEE, 2012, pp. 3354–3361.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[3] T. Strauss, M. Hanselmann, A. Junginger, and H. Ulmer, "Ensemble methods as a defense to adversarial perturbations against deep neural networks," *arXiv preprint arXiv:1709.03423*, 2017.

[4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *ECCV*. Springer, 2020, pp. 213–229.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016, pp. 779–788.

[6] O. Zendel, K. Honauer, M. Murschitz, D. Steininger, and G. F. Dominguez, "Wilddash-creating hazard-aware benchmarks," in *ECCV*, 2018, pp. 402–416.

[7] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.

[8] C.-H. Cheng, C.-H. Huang, and G. Nührenberg, "nn-dependability-kit: Engineering neural networks for safety-critical autonomous driving systems," in *ICCAD*. IEEE, 2019, pp. 1–6.

[9] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *ASE*. IEEE, 2018, pp. 132–142.

[10] S. Wang and Z. Su, "Metamorphic testing for object detection systems," *arXiv preprint arXiv:1912.12162*, 2019.

[11] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[12] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[13] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *S&P*. Ieee, 2017, pp. 39–57.

[14] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[15] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *CVPR*, 2016, pp. 2574–2582.

[16] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, 2020.

[17] A. Serban, E. Poll, and J. Visser, "Adversarial examples on object recognition: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–38, 2020.

[18] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," *International Journal of Automation and Computing*, vol. 17, no. 2, pp. 151–178, 2020.

[19] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[20] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[21] A. N. Bhagoji, W. He, B. Li, and D. Song, "Exploring the space of black-box attacks on deep neural networks," *arXiv preprint arXiv:1712.09491*, 2017.

[22] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava, "Genattack: Practical black-box attacks with gradient-free optimization," in *GECCO*, 2019, pp. 1111–1119.