Adversarial Attacks on Computer Vision Algorithms using Natural Perturbations

Arvind Ramanathan  Laura Pullum
Computational Science and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
Email: {ramanathana, pullumll}@ornl.gov

Zubir Husein Sunny Raj Neslisah Torosdagli Sumanta Pattanaik Sumit K. Jha
Computer Science Department
University of Central Florida
4000 Central Florida Blvd
Orlando, Florida, USA
Email: {zhusein, sraj, neslisah, pattanaik, jha}@cs.ucf.edu

Abstract—Verifying the correctness of intelligent embedded systems is notoriously difficult due to the use of machine learning algorithms that cannot provide guarantees of deterministic correctness. In this paper, our validation efforts demonstrate that the OpenCV Histogram of Oriented Gradients (HOG) implementation for human detection is susceptible to errors due to both malicious perturbations and naturally occurring fog phenomena. To the best of our knowledge, we are the first to explicitly employ a natural perturbation (like fog) as an adversarial attack using methods from computer graphics. Our experimental results show that computer vision algorithms are susceptible to errors under a small set of naturally occurring perturbations even if they are robust to a majority of such perturbations. Our methods and results may be of interest to the designers, developers and validation teams of intelligent cyber-physical systems such as autonomous cars.

I. INTRODUCTION

Machine learning algorithms are increasingly being deployed in cyber-physical systems that directly expose human beings to the consequences of the errors in the design or implementation of such machine learning systems. The possibility of autonomous driving [1] has led to a number of car manufacturers (such as Tesla) and public ride-providers (such as Uber) rapidly exploring the market for autonomous vehicles through a number of innovative offerings. While such innovations are driven mostly by advances in sensor technology and computer vision algorithms supported by the scalability of Moore’s law [2], [3], researchers in safety, formal verification, and cyber-physical systems must explore the limits of such autonomous self-driving vehicles. Our work is an effort towards understanding the susceptibility of computer vision algorithms to both malicious and naturally occurring input perturbations.

The analysis of human detection code using traditional program analysis or model checking techniques is complicated by the tight coupling between data and control flow. For example, the human detection algorithm based on histogram of oriented gradients (HOG) [4] in OpenCV [5] generates more than three million feature values in our experiments; an exhaustive analysis of such a large state space is infeasible by current standards.

If current implementations of human detection algorithms were extremely proficient in correctly identifying human beings, such a huge state space would pose a serious challenge to their validation and verification. Unfortunately, we find that implementation of the histogram of oriented gradients (HOG) based human detection algorithm [4] in the popular OpenCV package is highly susceptible to both malicious and naturally occurring minor perturbations to its inputs. Hence, as we will demonstrate in Section IV, simulated annealing [6] based methods are readily able to discover counterexamples that cause the HOG-based human detection algorithm implemented in OpenCV to fail.

In this paper, we make the following contributions to the state of the art in validating human detection algorithms:

1) To the best of our knowledge, we are the first to explicitly employ a natural perturbation (like fog) as an adversarial attack on a computer vision algorithm by exploiting visualization algorithms from computer graphics. This is different from adversarial examples synthesized earlier that focussed either on introducing synthetic noise [7], [8], [9] or coupled the introduction of synthetic noise with reacquiring the synthetic image through a camera [10]. Existing literature does not focus on introducing naturally occurring perturbations into the image as is likely to be experienced by a computer vision system in the wild.

2) We propose a white-box algorithm for synthesizing counterexamples to the HOG-based human detection algorithm [4]. Our approach exploits the information encoded in the feature vector corresponding to a perturbed image and the decision boundary

---

Fig. 1. (a) OpenCV’s implementation of the Histogram of Oriented Gradients correctly determines that there is a human in the image on the left. (b) The identical OpenCV human detection implementation is not able to find the human in the image on the right. The human being is visible to the naked eye (the human vision system) in both the images and the two images are almost indistinguishable to the human eye. The image on the right is obtained by applying subtle perturbations to the image on the left.
of the support vector machine to guide the overall perturbation process using a simulated annealing approach [6]. As shown in Figure 1, our algorithm produces counterexamples that are almost indistinguishable from the original image.

3) We experimentally demonstrate that our new algorithm performs better than an earlier approach [9] that fails to transform many images into counterexamples within one hour. In contrast, our current algorithm computes counterexamples for all the images used in our study and produces these results in less than an hour. This is not surprising as the previous method [9] is a black-box approach with no knowledge of the HOG-based human detection algorithm.

II. RELATED WORK AND CURRENT CHALLENGES

Testing methods based on statistical analysis have been used to study the probabilistic correctness of machine learning algorithms [11]. Unit and integration testing methods have been employed to attempt the exploration of a significant portion of the input space of machine learning algorithms [12]. However, given the enormous space of all possible images and the difficulty in obtaining the ground truth for such a large number of images, both sampling and statistical analysis methods are unlikely to succeed in proving the correctness of computer vision algorithms.

Several critical pieces of hardware and software have been verified algorithmically using formal methods. The SLAM project [13] at Microsoft was one of the earliest to be practically deployed in the analysis of device drivers and ensure an improved stability of the Windows operating system. Intel has also explored the use of formal methods for replacing testing of key components of its processors with formal verification methods [14], [15]. The kernel of a modern operating system has been verified [16] against correctness properties using theorem proving methods.

While formal methods [17] such as model checking [18] have had great success in verifying abstraction of hardware and relatively simple software systems, these methods do not scale well when applied to computer vision algorithms for two reasons: (i) vision and other machine learning algorithms admit only a probabilistic notion of correctness and hence a single counterexample is not really of interest. (ii) probabilistic model checking methods [19] scale even more poorly than their deterministic counterparts [20] and hence cannot be applied to even moderate size machine learning methods.

Modern formal verification tools often use statistical methods to solve the probabilistic verification problem [21]. However, we note that no universally-adopted framework for formally verifying machine learning systems has been presented so far. Our work could be used as a starting point for developing statistical model checking methods that provide formal guarantees of correctness for machine learning algorithms.

There has been a series of experimental investigations [7], [8], [10] demonstrating that deep learning algorithms are highly susceptible to attacks. In some cases, deep learning vision algorithms see shapes and forms of real objects in images that look like pure noise to the human vision system. In other cases, deep learning algorithms are deeply affected by small perturbations and produce results not consistent with the human vision system. However, none of these methods employ computer graphics to generate naturally occurring perturbations that are likely to be experienced by the machine learning system in the wild and demonstrate the failure of the system on such naturally occurring perturbations.

III. VALIDATION USING SIMULATED ANNEALING

An image $I$ with $n$ horizontal rows of pixels and $m$ vertical columns of pixels, where each pixel has a depth of $d$, naturally corresponds to a tensor of size $n \times m \times d$. The space of all possible images can be thought of as members of this set of tensors $T$. Since we are interested in those counterexamples to human detection that are correctly classified by the human vision system as human beings, we only seek to explore an $\epsilon$-neighborhood of the image $I$ in the space of tensors $T$.

As $T$ is high-dimensional and has too many images, there is little hope of exploring all the possible images in the $\epsilon$-neighborhood of the image $I$. Ramanathan et al. [9] employed undirected search using a combination of statistical hypothesis testing and symbolic decision procedures to show that such a perturbed image exists for at least one input example. But, the approach does not discover perturbations of many other images, such as those reported in this paper, within one hour. This is not surprising as the previous method [9] is a black-box method with no knowledge of the HOG-based human detection algorithm and is performing a random exploration of the state space of image in the $\epsilon$-neighborhood of the image $I$.

![Fig. 2. If we draw a normal from the HOG feature vector of the original image $I$ (solid green circle) to the hyperplane $H$ and extend this normal to the other side of the hyperplane by a small margin, we obtain the closest feature vector $I'$ (dashed black circle) that will be incorrectly classified by the support vector machine. Ideally, one would like to invert this feature vector into an image; however, that is computationally challenging. Instead, we have used simulated annealing to explore the space of images in the $\epsilon$-neighborhood of the original image. The images and corresponding feature vectors explored during the simulated annealing process are illustrated by the dotted green circles. When the simulated annealing algorithm determines an image $I_{\Delta}$ whose feature vector lies on the other side of the hyperplane, we stop and report this image (illustrated by solid red circle).](image-url)
sub-problems. Our attempts at inverting the HOG transformation produced images that were readily recognized by the human vision system as having been significantly perturbed.

Algorithm 1 Generating counterexamples to the Histogram of Gradients (HOG) human detection algorithm

```plaintext
1: procedure COUNTEREXAMPLE2HOG(I, SVM Hyperplane \( T_{max} \), Maximum temperature \( T_{max} \), Maximum permitted perturbation \( D_{max} \))
2:   \( T \leftarrow T_{max} \) \( \triangleright \) Initialize temperature for simulated annealing
3:   \( c \leftarrow 0.99 \) \( \triangleright \) Cooling rate for simulated annealing
4:   \( I_{cur} \leftarrow I \) \( \triangleright \) Initialize current image to the input image
5:   \( V \leftarrow \text{HOG}(I_{cur}) \) \( \triangleright \) Compute the HOG feature vector
6:   repeat
7:     \( I_{\Delta} \leftarrow \text{Perturb}(I_{cur}, \Delta) \) \( \triangleright \) Perturb \( I \)
8:     \( V_{\Delta} \leftarrow \text{HOG}(I_{\Delta}) \) \( \triangleright \) Compute feature vector for \( I_{\Delta} \)
9:     \( D_{\Delta} \leftarrow \text{Distance}(V_{\Delta}, w^T x - a = 0) \) \( \triangleright \) Compute distance between \( V_{\Delta} \) and the SVM hyperplane
10:    if \( D_{\Delta} \leq D_{cur} \) then
11:      \( D_{cur} \leftarrow D_{\Delta} \)
12:      \( I_{cur} \leftarrow I_{\Delta} \)
13:    else if \( \text{rand}() \leq e^{-D_{\Delta}/T} \) then
14:      \( D_{cur} \leftarrow D_{\Delta} \)
15:      \( I_{cur} \leftarrow I_{\Delta} \)
16:    end if
17:   \( T \leftarrow cT \)
18:   until \( \text{HOG}(I_{\Delta}) = \text{False} \) or \( D_{cur} > D_{max} \)
19:   if \( D_{max} > D_{cur} \) then
20:     \( T_{max} \leftarrow T_{max}/2 \) \( \triangleright \) Decrease temperature to allow fewer perturbations
21:   until \( T_{max} < 100 \) or ( \( \text{HOG}(I_{\Delta}) = \text{False} \) and \( D_{cur} < D_{max} \))
22:   if \( \text{HOG}(I_{\Delta}) = \text{False} \) then
23:     return \( I_{\Delta} \) \( \triangleright \) Return perturbed image that causes HOG to fail
24:   else
25:     print No counterexample image found.
26:   end if
27: end procedure
```

Our approach (see Algorithm 1) for seeking counterexample images to the histogram of oriented gradients (HOG) human detection algorithm relies on performing an exploration of the \( c \) – neighborhood of the image \( I \) in the space of tensors \( T \) using a variant of simulated annealing [22], [23], [24]. The inputs to our algorithm include the image that is being perturbed \( I \), the SVM hyperplane \( H \) trained by the HOG framework, the permitted perturbation \( \Delta \) allowed by the simulated annealing at each step, the maximum temperature \( T_{max} \) for initializing simulated annealing and the maximum permitted perturbation \( D_{max} \) of the image.

In order to guide the search for the perturbed image \( I_{\Delta} \), we compute the distance of the feature vector \( F_{\Delta} \) of a perturbed image from the decision hyperplane \( H \) of the Support Vector Machine classifying the HOG feature vectors. Our goal is to minimize the distance of the HOG feature vectors corresponding to the perturbed images from the hyperplane and eventually find an image whose feature vector lies on the other side of the decision hyperplane.

Algorithm 1 only permits the image to change by a small margin \( \Delta \) at any step of the exploration of the space of possible images using simulated annealing. This ensures that the algorithm only makes small changes to the image during the search process. Further, the algorithm is not permitted to make more than \( D_{max} \) change in the image \( I \). High initial temperatures of simulated annealing can cause the algorithm to wander off and produce images that are too far away from the original image. In this case, the algorithm reduces the initial temperature by a factor of two, and restart the simulated annealing based search for a perturbed image within \( D_{max} \) distance of the original image which lies on the other side of the SVM’s decision hyperplane. It is also beneficial to only perturb pixels that lie within the bounding box of regions where the SVM has reported as containing human beings. This serves to reduce the possible search space by ignoring pixels with little to no influence on the SVM’s classification decision. A visualization of the perturbations produced by the algorithm is presented in Figure 3, we can see that our algorithm produces a maximum combined difference of six units for the RGB values, with the majority of perturbations being one unit. We also note that a majority of perturbations is in an area containing the human being.

IV. EXPERIMENTAL RESULTS

A. Physically Feasible Synthetic Perturbations

Ramanathan et al. have reported successfully attacking the Histogram of Oriented Gradients (HOG) based human detection algorithm on one example [9]. However, this approach does not successfully attack the HOG algorithm on other images reported in this paper within one hour. This is not surprising as their algorithm performs a black-box exploration of the HOG-based human detection algorithm and does not really exploit the details of the vision algorithm.

The algorithm [9] performs a random exploration of the space of images around the original image using a combination of symbolic methods and statistical model checking. While this random exploration allows the same method to analyze different machine learning algorithms such as k-means and human detection, this generalization comes at the cost of efficacy in terms of computational speed.
The algorithm introduced in the paper successfully transformed all the images in Figure 4 into perturbed images that the HOG-based human detection algorithm fails to classify as images containing human beings. Further, our algorithm obtains these results within 30 minutes while running on a standard desktop. It should be noted that our human vision system does not really detect any substantial difference between the original and the perturbed images (see Figure 4). However, the HOG-based vision system reports that more than 20% of all the HOG features have been perturbed by these subtle changes. Table I shows the number of perturbed HOG features for each of the images in Figure 4.

### Table I

<table>
<thead>
<tr>
<th>Image</th>
<th>Total # HOG Features</th>
<th>Perturbed # HOG Features</th>
<th>Percentage Perturbed Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 4a</td>
<td>3,016,440</td>
<td>813,414</td>
<td>26.97%</td>
</tr>
<tr>
<td>Figure 4c</td>
<td>3,016,440</td>
<td>891,216</td>
<td>29.54%</td>
</tr>
<tr>
<td>Figure 4e</td>
<td>3,016,440</td>
<td>714,420</td>
<td>23.68%</td>
</tr>
<tr>
<td>Figure 4g</td>
<td>3,016,440</td>
<td>707,076</td>
<td>23.44%</td>
</tr>
<tr>
<td>Figure 4i</td>
<td>3,016,440</td>
<td>1,004,269</td>
<td>33.29%</td>
</tr>
</tbody>
</table>

We also analyzed the difference between the natural and the perturbed images using the HDR-VDP metric and the results are reported in Table II. The HDR-VDP is a visual metric predicting the probability an average observer will notice the differences between two images [25]. We note that the difference between the original and perturbed images will be observed by an average individual with a probability of less than 0.001%.

### Table II

<table>
<thead>
<tr>
<th>Original Image</th>
<th>Perturbed Image</th>
<th>Percentage Perturbed Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 4a</td>
<td>Figure 4b</td>
<td>$7.9323 \times 10^{-4}$</td>
</tr>
<tr>
<td>Figure 4c</td>
<td>Figure 4d</td>
<td>$2.9054 \times 10^{-4}$</td>
</tr>
<tr>
<td>Figure 4e</td>
<td>Figure 4f</td>
<td>$2.8118 \times 10^{-4}$</td>
</tr>
<tr>
<td>Figure 4g</td>
<td>Figure 4h</td>
<td>$2.0906 \times 10^{-4}$</td>
</tr>
<tr>
<td>Figure 4i</td>
<td>Figure 4j</td>
<td>$2.1661 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

#### B. Computer Graphics Based Simulation of Fog Phenomena

Earlier work in creating counterexamples to computer vision algorithms has focussed entirely either on synthetic perturbations [7], [8], [9] or synthetic perturbations re-acquired through a physical camera [10]. To the best of our knowledge, we are the first to explicitly employ a natural perturbation (like fog) as an adversarial attack on a computer vision algorithm by exploiting visualization algorithms from computer graphics.

Light traveling in a medium containing scattering particles undergoes scattering and gets distributed away from its path. As a consequence, the surrounding area along the path of the light gets illuminated. But, the intensity reduces along the light propagation direction. We come across such scattering media in our day to day life in the form of fog, snow, rain, cloud, dust storm etc. Fog and rain in particular are important, as, they occur frequently and they occur in atmosphere close to the ground. Thus, they are likely to affect the behavior of autonomous unmanned aerial systems (UAS) and self-driving cars.

Fig. 4. The human beings in the images on the left were easily recognized by OpenCV’s Histogram of Oriented Gradients (HOG) human detection system. The images on the right have been obtained by perturbing the images on the left using our approach. OpenCV reports that there are no human beings in the images on the right even though they are clearly visible to the human eye.
Fog and rain are mostly composed of water droplets or ice particles that scatter light. The concentration of these scattering particles determines the amount of scattering and hence can affect visibility more or less significantly. Visibility is affected, because of the following reasons: (a) light from the points/objects of interest scatters away from the path and do not reach our eye, (b) scattered light from elsewhere (for example: sky) creates a veil of background illumination and hence hinders our ability to distinguish the object of our interest from background. Both of these components are key to fog rendering. They are separately computed and mixed. The former component, scattering loss of light intensity, can be physically modeled by Beer-Lambert law, which is simply an exponential decrease in intensity governed by two parameters: the distance of the emitting/reflecting object of interest from the eye/camera, and the density of the scattering particles. Accurate computation of the latter component, the light scattering-in to the eye/camera from the fog, is expensive and is often approximated.

In our experiments, we approximate this latter component as a user defined fog color. We consider a flat textured object imaged from a fixed distance, and hence the distance of the camera to each part of the object is almost the same. By varying the density of fog particles over the field of view we spatially vary the loss of intensity over the resulting image. We use Perlin’s noise [26], a procedural method, for creating spatially random variation of particle density.

In Figure 5, we report the results of applying our algorithm to five different images when the image perturbation function in Algorithm 1 is restricted to the introduction of a fog using Perlin’s noise [26]. The pictures on the left in Figure 5 are images where the HOG based human detection algorithm correctly identifies the presence of a human being while the HOG based human detection algorithm did not identify human beings in the images on the right. Both sets of images look almost identical to the human eye and have been obtained by adding imperceptibly different fogs to the original images. Table III reports the HDR-VDP metric [25] for the two sets of images. We again observe that an average person is going to distinguish between the corresponding images on the left and the right of Figure 5 with probability less than 0.001%.

<table>
<thead>
<tr>
<th>Original Image</th>
<th>Perturbed Image</th>
<th>Percentage Perturbed Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 5a</td>
<td>Figure 5b</td>
<td>$4.7398 \times 10^{-4}$</td>
</tr>
<tr>
<td>Figure 5c</td>
<td>Figure 5d</td>
<td>$4.3043 \times 10^{-4}$</td>
</tr>
<tr>
<td>Figure 5e</td>
<td>Figure 5f</td>
<td>$4.7877 \times 10^{-4}$</td>
</tr>
<tr>
<td>Figure 5g</td>
<td>Figure 5h</td>
<td>$5.5660 \times 10^{-4}$</td>
</tr>
<tr>
<td>Figure 5i</td>
<td>Figure 5j</td>
<td>$6.5642 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

V. Conclusions

Based on our experimental observations, we conclude that a small set of naturally occurring perturbations can cause computer vision systems to fail even if these systems perform robustly in presence of a large variety of naturally occurring perturbations. Our experimental results demonstrate the difficulty of predicting the performance of computer vision and other machine learning algorithms in the wild.
While we have worked only on one implementation of the HOG-based human detection algorithm, there is hope that our results may be generalizable to other algorithms. Studies on adversarial example transfer [27], [10], [7] have shown that examples readily transfer between different implementations of machine learning algorithms. Hence, our approach may be useful in creating adversarial examples for a variety of other human detection algorithms.

VI. ACKNOWLEDGEMENT

The authors would like to thank the US Air Force for support provided through the AFOSR Young Investigator Award to Sumit Jha. The authors acknowledge support from the National Science Foundation Software & Hardware Foundations #1438989 and Exploiting Parallelism & Scalability #1422257 projects. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-16-1-0255.

This research was partially supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

REFERENCES