Dynamic Black-Box Attacks Against Neural Networks using genetic algorithms

Mathis Hammel
contact@mathishammel.com

Abstract. In the past decade, neural networks have emerged as an almost universal method for implementing AI systems. Their exceptional capacity for generalization and abstraction across a wide range of domains sets them apart from more traditional statistical models. However, the complexity of neural networks are an open door to a range of payloads called adversarial attacks.

Similar to optical illusions that deceive the brain into perceiving a non-existent pattern or movement in an image, adversarial attacks consist of small targeted manipulations to the input of a neural network which result in an incorrect detection or classification.

An adversarial attack can easily be crafted on any neural network when provided direct knowledge about its internal values and access to bit-perfect manipulation of the input. In this article, we explore more challenging conditions with a black-box attack on image classifiers in real-world settings, where the injection has to be robust through a live camera feed.

Our approach features a novel way of performing such adversarial attacks on vision systems, leveraging genetic algorithms to iteratively solve a constrained optimization problem, along with simulated graphics to ensure robustness in the real world.

We then demonstrate the attack on a live ResNet-34 neural network trained to recognize road signs. After applying square stickers covering 3% of a stop sign, the neural network detects a "speed limit 50 km/h" sign from any angle.

1 Introduction

In the last decade, neural networks have become the best performing model on nearly all complex tasks in artificial intelligence such as speech recognition [1], object classification [2] or text-to-image generation [3], sometimes even exceeding human performance.

However, their high complexity and weak explainability make them vulnerable to a family of attacks called "adversarial attacks" or "adversarial examples". When given partial or full control over the input data provided to the neural network, an adversary can compute a targeted modification of the input that looks inconspicuous but has a strong effect on the model's output.

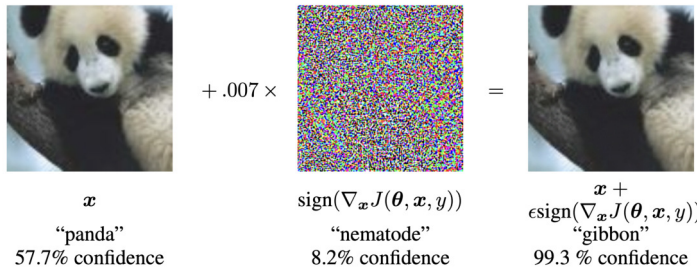

Fig. 1. Adversarial image generated using the *Fast Gradient Sign Method* [4]. Each pixel component is modified by 2/256 or less.

1.1 Definition of an adversarial attack

We model the adversarial attack on an image classifier as follows: let's consider a neural network denoted as a function $f : [0,1]^N \rightarrow [0,1]^M$. It takes an image consisting of N normalized pixel values as its input, and produces a vector of M classification probabilities, each representing a possible prediction class. We are also given an image $img \in [0,1]^N$, belonging to class $src \in \{1, 2, ..., M\}$, which will be the base of the attack.

We expect our classifier to predict the correct class on img, meaning that $argmax(f(img)) = src$.

In its most standard form, the goal of the attack is to compute a perturbation $atk \in [0,1]^N$ such that $argmax(f(img)) \neq argmax(f(img + atk))$, i.e. the predicted class of the modified image is incorrect.

2 Classification of adversarial attacks

There exist many classes of adversarial attacks, which depend on the capabilities provided to the attacker over the model and the desired effect of the attack. [5] provides a complete taxonomy, but the following distinctions are sufficient to understand our proposed attack methodology.

2.1 Model knowledge

- *White-box* adversarial attacks assume a full disclosure of information about the model f, most notably the neural network's architecture and weights.

- *Black-box* adversarial attacks are based on no particular knowledge other than the output values, which in most cases corresponds to the

access level a normal user would have. Black-box attacks are either based on raw output values $f(x)$ or the output class $argmax(f(x))$.

- Used less frequently, the term *grey-box* describes any setup that discloses partial information about the trained model, such as the training dataset or access to only a part of the model parameters [6].

2.2 Target specificity

- For classifier models, *targeted* attacks try to make the model output a specific predicted class, i.e. $argmax(f(img + atk)) = target$ for a chosen $target$ class.

- On the other hand, *non-targeted* attacks impose more relaxed constraints, having $argmax(f(img + atk)) \neq src$. They are often slightly easier to perform depending on the structure of the output space, given that some classes may be similar to each other.

2.3 Static vs. dynamic

In nearly all seminal research on the topic, adversarial attacks are conducted in a *static* manner: the perturbation is introduced with bit-perfect precision and evaluated on a single data sample.

However, models in the real world often acquire their input data through a noisy channel (microphone, camera, ...). Without additional measures, printing an adversarial image and presenting it to a camera will render the attack completely inefficient. The notion of *dynamic* attack emphasizes this robustness to channel-induced noise and other external conditions, such as capturing an object from multiple angles [7].

Another difference between these two classes is that static attacks also often use continuous real numbers, whereas actual images are usually encoded in discrete measurements (24 bits per pixel) which removes the ability of applying extremely fine changes.

2.4 Constraints on the perturbation

The amplitude and localization of the input perturbation needs to be constrained. If not, the attack would be as simple as providing an image from the desired class: depending on the environment, this would either be trivial behavior of the system, or a form of injection unrelated to adversarial attacks.

The goal of the attacker is usually to find small modifications to the original image, that would be nearly invisible to the human eye or

inconspicuous enough by blending in its environment. This additional constraint is modeled by a distance function $d : [0,1]^N \times [0,1]^N \to \mathbb{R}$, which is bound by a value t fixed by the attacker when comparing the original input and the modified input: $d(img, img + atk) \leq t$.

The distance function usually comes from one of 4 main categories:

- l_∞ norm is the maximum difference in value between the original and modified input across all pixels (also called *box constraint*). Figure 1 uses l_∞ norm with a threshold $t = 0.07$.

- l_1 norm is the sum of all absolute differences in values.

- l_0 norm is the count of pixels that were modified. Since it does not take into account the magnitude of atk, adversarial attacks based on this metric generally tend to produce a few significant extreme pixel values in the resulting image (similar to *salt-and-pepper noise*). Such an attack is illustrated later in this article.

- Metrics based on perceptual similarity [8, 9], which require more complex attack modeling but are designed to match human perception better than the other metrics mentioned above.

The attacker can either try to maximize the misclassification score while maintaining the perturbation magnitude below the threshold (*constraint perturbation*), or try to find the lowest possible limit that results in satisfying misclassification (*optimized perturbation*).

3 Design of the attack

Our goal is to successfully execute one of the hardest adversarial attack types: a black-box, targeted, dynamic attack.

The task we choose is road sign classification, because the risks associated with successful exploitation of such a model are very easy to understand for a broad audience. We aim to modify a stop sign in order to make the neural network recognize a different type of road sign, which could result in disaster if applied to a model embedded in an autonomous vehicle.

Attacking road sign classifiers has been explored in prior research such as [10], which has been an inspiration for this article but only works in white-box conditions.

3.1 Model setup

In order to work in a controlled environment, we start by training a ResNet-34 neural network [11] on the GTSRB dataset [12] which contains 39,209 pictures of German road signs in 43 different classes.

Fig. 2. Sample of the 43 classes in the GTSRB dataset.

After training for 30 epochs, our model reaches 97.7% accuracy on the test dataset. While some specialized neural networks can surpass human performance (98.8%) on the same task, we do not really care about reaching state-of-the-art performance here: Resnet-34 is a very common architecture, meaning that our attack would be more likely to generalize on other tasks where a similar architecture is used.

In order to increase the robustness of predictions, our live-video system performs classification over a sliding window of 20 consecutive frames (0.67 seconds) and uses the mean of the prediction vector.

Street signs in Germany are slightly different from the ones available to our lab setup in France, as seen in figure 3. Manual testing shows this is not a problem, as our classifier performs very well on the French version of the sign: on our set of 60 training and validation images, all classification probabilities are above 99.9% for the correct class.

Fig. 3. Comparison between standard stop signs in Germany (left) and France (right).

3.2 Attack setup

For demonstration purposes, we first develop a non-targeted white-box attack on a single static image. This illustrates the easier end of the spectrum, and helps understand visually how little modification can be needed to perform an adversarial attack in favorable conditions.

On the other end of that spectrum lies the targeted black-box attack in dynamic conditions, which will be demonstrated in a real-world setup by actually applying the perturbation to a physical stop sign and presented to the classifier through a live video feed.

In order to make this physical modification easier to apply to our sign in this second attack, we define a custom metric which constrains the perturbation to square patches of uniform color whose area represents 0.3% of the sign's area each. This is similar to using the l_0 norm with additional locality constraints. Working with discrete patches also makes it easier to use genetic algorithms to generate the perturbation.

4 Implementation of the white-box attack

Fig. 4. In left-to-right order: Base image, attack on 4 pixels, attack on 9 pixels, zoomed detail of the 9-pixel attack.

We use an iterative variant of the adversarial attack described in [13] called JSMA (Jacobian Saliency Map Attack).

Each iteration computes a new target pixel by determining the Jacobian matrix of the model near the image, and choosing the pixel with the highest partial derivative. The target pixel is then modified to maximize the error, and the next iteration begins.

After each iteration, we evaluate the classification probabilities and stop once a satisfying result is reached. The perturbation is nearly invisible to the naked eye:

- If the objective is to change the predicted class (i.e. $argmax(f(img + atk)) \neq src$), we only need to change 4 pixels for the attack to work.

- If we want the prediction score of the source class ("stop sign") to be lower than 1%, then only 9 pixels are needed.

This attack runs in less than 1 second on a standard laptop.

5 Implementation of the black-box attack

As mentioned earlier, the static attack is not robust enough to work in realistic conditions. Even the noise induced by a camera sensor is enough to make the prediction of the correct class go back to more than 99%.

As an additional challenge, we are not allowed to use model introspection to compute gradients or internal values (the second-to-last layer of logits before softmax normalization is sometimes used [14]). We set to work on a black-box environment: our only information comes from the prediction values provided by the classifier on the chosen input images.

We set an arbitrary target class "speed limit 50 km/h", which would represent a realistic attack scenario where the targeted vehicle speeds past the crossing.

5.1 Genetic algorithm

Genetic algorithms [15] are a metaheuristic method used to find solutions to optimization problems. A genetic algorithm maintains a "population" of candidate solutions to the problem, and simulates their evolution in a lifelike manner them through crossover, mutation and selection. After several generations, the highest-scoring individual across all generations represents the algorithm's proposed solution.

1: **Initialize population:** Generate random population of individuals
2: **Evaluate:** Evaluate the fitness of each individual in the population
3: **while** termination condition not met **do**
4: **Selection:** Select parents for crossover based on their fitness
5: **Crossover:** Create new individuals by combining the genetic material of parents
6: **Mutation:** Introduce random changes in the offspring's genetic material
7: **Evaluation:** Evaluate the fitness of new individuals
8: **Replacement:** Replace the old population with the new population
9: **Return:** Best individual found during the run

Algorithm 1. Pseudocode of a genetic algorithm

In the case of our adversarial attack, each individual in the population is a list of K patches represented each by their center coordinate and color.

After experimenting with several parameters, we reach two conclusions:

- Working with an initial population of K patches from the start leads to a slow and difficult convergence of the algorithm.

- Crossover almost never results in useful changes. Exceptions occur only when the whole population is made of near-identical copies of the same individual, a scenario generally considered undesirable.

Considering these observations, we change the algorithm to gradually increase the perturbation from 1 up to K colored patches, and get rid of the concept of genetic population to focus on a single individual instead. In each iteration, we alternate between an exploration phase to find a good candidate for the new patch, followed by an optimization phase that adjusts the position and color of all existing patches.

Initialize individual: $atk \leftarrow []$
for $N_{patches} \leftarrow 1$ to K **do**

    $candidates \leftarrow []$
    for $i \leftarrow 1$ to N_{expl} **do** ▷ EXPLORATION PHASE
        Exploration: $patch_i \leftarrow$ a new random patch
        Append $atk + patch$ to $candidates$
    Selection: $atk \leftarrow \underset{x \in candidates}{\arg\max}\ (\text{SCORE}(x))$ ▷ Keep the best candidate

    for $i \leftarrow 1$ to N_{optim} **do** ▷ OPTIMIZATION PHASE
        Mutation: $atk\_new \leftarrow \text{MUTATE}(atk)$
        if $\text{SCORE}(atk\_new) > \text{SCORE}(atk)$ **then**
            $atk \leftarrow atk\_new$

Return: atk

function MUTATE(atk)
    Select a random index i from atk
    $(patch\_position, patch\_color) \leftarrow atk[i]$
    $patch\_position\_new \leftarrow$ random position close to $patch\_position$
    $patch\_color\_new \leftarrow$ color of $patch\_color$ flipped with probability p_{flip}
    return $(patch\_position\_new, patch\_color\_new)$

function SCORE(atk)
    $probabilities \leftarrow f(img + atk)$
    return $probabilities[target] - probabilities[src]$

Algorithm 3. Pseudocode of our modified genetic algorithm

5.2 Simulated graphics

In static conditions, the genetic algorithm performs very well : working with a single patch and after running 2500 iterations of optimization and exploration each, we are successfully able to shift the classification from "stop sign (99.9% confidence)" to "speed limit 50 (74.2% confidence)".

Fig. 5. Successful black-box perturbation (left), and the same image displayed on a smartphone and photographed through a laptop camera, rendering the attack ineffective (right).

However, this attack is very sensitive and does not hold well when a small amount of noise is introduced: the two images in figure 5 look nearly identical, but the image recaptured through a camera does not introduce a misclassification and the stop sign probability is back to 99.9%. Similarly, applying the same perturbation to a different image does not result in a successful adversarial attack.

Both of these issues are due to the fact that vulnerable regions are common in the input space, but most of them are very localized and tend to revert back to the original predicted class with a small nudge in any direction.

In order to produce a robust adversarial attack, we need to find a larger region of perturbated inputs that consistently produce misclassification. This will result in more stability against channel noise and external variations, such as the orientation of the sign and background of the image. We achieve this by simulating the same real-world perturbation on a dataset of 50 pictures of the stop sign taken in various positions. Another 10 pictures are used for validation.

For each image in our dataset, we manually annotate the bounding box of the sign in the image. This bounding quadrilateral is then mapped to $[0, 1] \times [0, 1]$ space using perspective projection, which gives us a bijective mapping between picture-specific coordinates and a common coordinate

system. This allows us to draw shapes on each sign independently of its position in the image, as if all images had the same perturbation applied to the physical sign.

Fig. 6. Example of images from our simulation dataset (top row) with the same octagon mapped on all three images using perspective projection (bottom row).

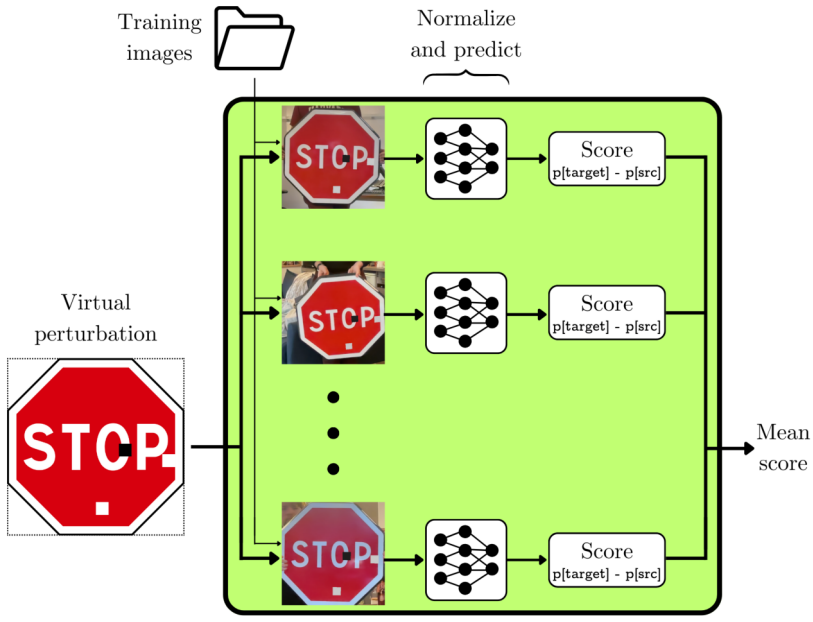

Fig. 7. Multi-image scoring process for a perturbation.

In the new version of our genetic algorithm, the fitness function (illustrated in figure 7) starts by virtually applying its patches to all signs in the training dataset, then averages the misclassification score over all generated images.

5.3 Results

Our genetic algorithm depends on three main hyperparameters: the number of patches K, and the iteration count in each phase N_{expl} and N_{optim}. The time complexity of the attack is $\mathcal{O}(K \cdot (N_{expl} + N_{optim}))$. We now explore several configurations to find the best tradeoff between speed and attack success.

For a better accuracy in our results, each attack presented below is run 5 times and the mean score is shown.

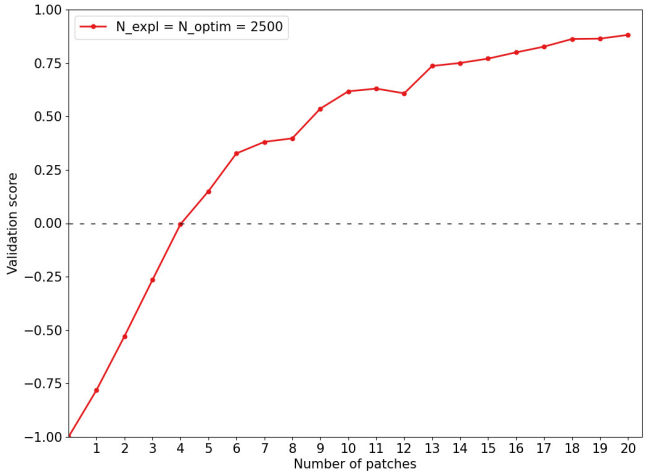

Fig. 8. Validation score of the perturbation after each new patch is added up to $K = 20$, with $N_{expl} = N_{optim} = 2500$. A validation score of -1 means that the classifier always predicts the source class, and +1 means that it always predicts the targeted class.

As shown in figure 8, increasing K leads to a more successful attack. This is easy to understand, since a larger number of patches means a greater area is masked and the sign is less similar to the original. For the rest of this experiment, we set the perturbation constraint to $K = 10$ in

order to remain visually close to the original sign: this represents 3% of its surface.

Next, we try to determine the optimal values of N_{expl} and N_{optim}. We can intuitively guess that spending more time searching for the best configuration results in a better attack, but we also want to make our algorithm reasonably fast for practical reasons.

Results shown in figure 9 confirm this hypothesis: when running progressively more iterations, the attack performance increases up to a certain point near $N_{expl} = N_{optim} = 2500$ after which the score remains stable.

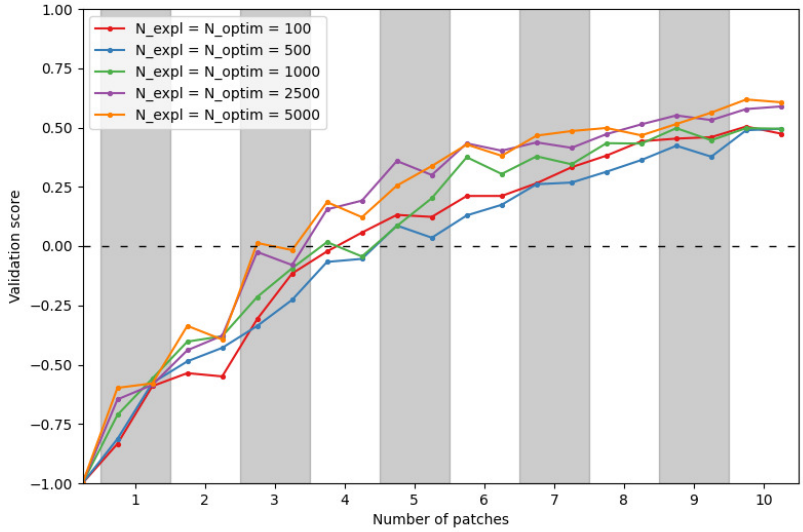

Fig. 9. Validation score of 5 different attack configurations with $K = 10$. In each stripe, the first point is the score after running the exploration phase, and the other is the score after optimization.

Figure 9 reveals that the optimization phase often decreases the validation score, which could indicate potential overfitting on the training data. In order to find the optimal split between exploration and optimization, we fix a computing budget $N_{expl} + N_{optim} = 5000$. We can then explore multiple configurations of (N_{expl}, N_{optim}) ranging from $(5000, 0)$ to the other extreme $(1, 4999)$ (we always need at least one exploration iteration to create new patches). The results are shown in figure 10.

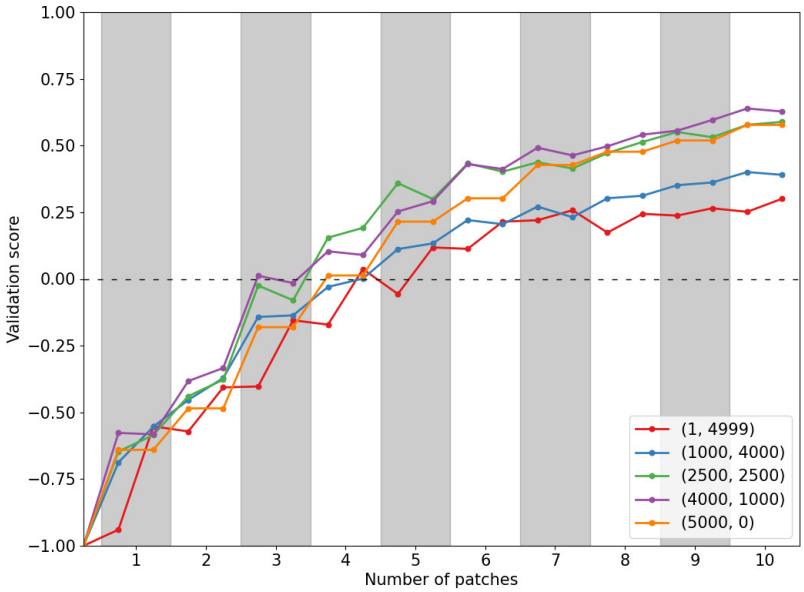

Fig. 10. Validation score of 5 different attack configurations ((N_{expl}, N_{optim})) of identical budget, with $K = 10$.

The results show that despite occasionally lowering the score, maintaining some balance between exploration and optimization is still beneficial compared to heavily favoring one phase over the other.

6 Conclusion

Our custom genetic algorithm is able to generate efficient and robust black-box adversarial attacks in less than an hour. Such attacks pose a very serious risk on critical machine learning systems, since there currently exists no reliable protection against them [5, 16, 17].

Security by obscurity is not even a valid defense strategy, as black-box algorithms only require normal user access to the model. For the case of autonomous vehicles, real-world proofs of concept have already been demonstrated on self-driving cars [18], and the need to devise and implement defensive measures is becoming critical as these systems get deployed into more and more aspects of our lives [19]. However, there is no clear indication that such a protection will ever be discovered.

Config		Time	Prediction[source]	Prediction[target]	Score
N_{expl}	N_{optim}				
100	100	1m38s	13.02%	60.41%	47.39%
500	500	8m3s	14.67%	64.14%	49.47%
1000	1000	15m18s	13.60%	63.09%	49.49%
2500	2500	30m21s	10.14%	69.01%	58.88%
5000	5000	63m13s	11.57%	72.20%	60.63%
1	5000	30m9s	20.69%	50.77%	30.08%
1000	4000	30m16s	19.78%	58.80%	39.02%
2500	2500	30m21s	10.14%	69.01%	58.88%
4000	1000	30m11s	10.86%	73.64%	62.78%
5000	0	29m48s	7.93%	65.67%	57.74%

Table 1. Full results of the experiments with $K = 10$. Execution time is measured on a MacBook Pro M2. Configuration (2500, 2500) is shown twice for readability.

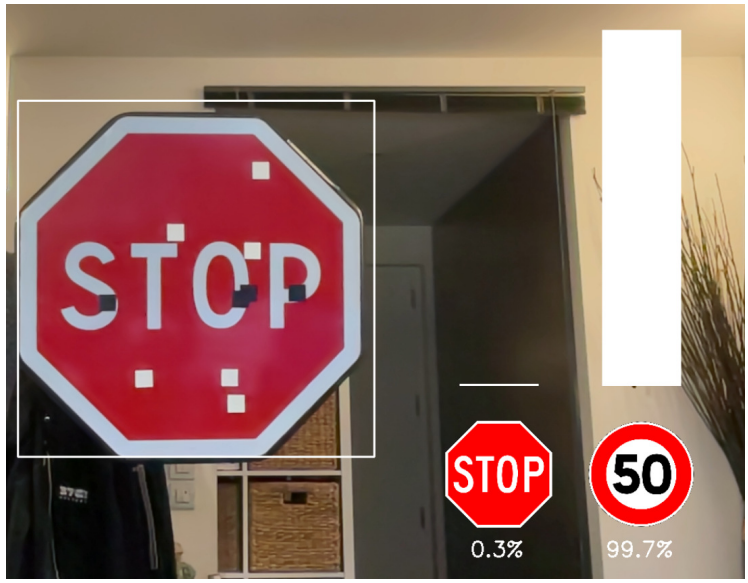

Fig. 11. Demonstration of the attack on our live video classifier, the perturbation is made from cardboard squares. The video classifier uses a sliding window to increase its stability, which also tends to make the attack more efficient.

References

1. Shu-wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y Lin, Andy T Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, et al. Superb: Speech processing universal performance benchmark. *arXiv preprint arXiv:2105.01051*, 2021.
2. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
3. Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
4. Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
5. Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.
6. Raz Lapid and Moshe Sipper. I see dead people: Gray-box adversarial attack on image-to-text models. *arXiv preprint arXiv:2306.07591*, 2023.
7. Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.
8. Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.
9. Shawn Shan, Jenna Cryan, Emily Wenger, Haitao Zheng, Rana Hanocka, and Ben Y Zhao. Glaze: Protecting artists from style mimicry by text-to-image models. *arXiv preprint arXiv:2302.04222*, 2023.
10. Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.
11. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
12. Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013.
13. Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

14. Juanjuan Weng, Zhiming Luo, Shaozi Li, Nicu Sebe, and Zhun Zhong. Logit margin matters: Improving transferable targeted adversarial attack by logit calibration. *IEEE Transactions on Information Forensics and Security*, 2023.
15. N.A. Barricelli. *Symbiogenetic Evolution Processes Realized by Artificial Methods*. 1957.
16. Austin Short, Trevor La Pay, and Apurva Gandhi. Defending against adversarial examples. 9 2019.
17. Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017.
18. Steve Povolny. Model hacking adas to pave safer roads for autonomous vehicles. 2020.
19. Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 69–75. IEEE, 2020.