AdVersarial: Perceptual Ad Blocking meets Adversarial Machine Learning

Florian Tramèr
tramer@cs.stanford.edu
Stanford University

Pascal Dupré
s9padupr@stud.uni-saarland.de
CISPA Helmholtz Center for Information Security

Gili Rusak
gili@stanford.edu
Stanford University

Giancarlo Pellegrino
gpellegrino@cispa.saarland
Stanford University, CISPA Helmholtz Center for Information Security

Dan Boneh
dabo@cs.stanford.edu
Stanford University

ABSTRACT

Perceptual ad-blocking is a novel approach that detects online advertisements based on their visual content. Compared to traditional filter lists, the use of perceptual signals is believed to be less prone to an arms race with web publishers and ad networks. We demonstrate that this may not be the case. We describe attacks on multiple perceptual ad-blocking techniques, and unveil a new arms race that likely disfavors ad-blockers. Unexpectedly, perceptual ad-blocking can also introduce new vulnerabilities that let an attacker bypass web security boundaries and mount DDoS attacks.

We first analyze the design space of perceptual ad-blockers and present a unified architecture that incorporates prior academic and commercial work. We then explore a variety of attacks on the ad-blocker's detection pipeline, that enable publishers or ad networks to evade or detect ad-blocking, and at times even abuse its high privilege level to bypass web security boundaries.

On one hand, we show that perceptual ad-blocking must visually classify rendered web content to escape an arms race centered on obfuscation of page markup. On the other, we present a concrete set of attacks on visual ad-blockers by constructing *adversarial examples* in a real web page context. For seven ad-detectors, we create perturbed ads, ad-disclosure logos, and native web content that misleads perceptual ad-blocking with 100% success rates. In one of our attacks, we demonstrate how a malicious user can upload adversarial content, such as a perturbed image in a Facebook post, that fools the ad-blocker into removing another users' non-ad content.

Moving beyond the Web and visual domain, we also build adversarial examples for AdblockRadio, an open source radio client that uses machine learning to detects ads in raw audio streams.

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11…$15.00
https://doi.org/10.1145/3319535.3354222

CCS CONCEPTS

• **Security and privacy** → **Web application security**; • **Computing methodologies** → *Machine learning approaches.*

KEYWORDS

Ad Blocking; Machine Learning; Adversarial Examples

ACM Reference Format:
Florian Tramèr, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, and Dan Boneh. 2019. AdVersarial: Perceptual Ad Blocking meets Adversarial Machine Learning. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3319535.3354222

1 INTRODUCTION

Online advertising is a contentious facet of the Web. Online ads generate over $200 billion in value [90], but many Internet users perceive them as intrusive or malicious [46, 51, 68, 93]. The growing use of ad-blockers such as Adblock Plus and uBlock [1, 7] has sparked a fierce arms race with publishers and advertising networks. Current ad-blockers maintain large crowdsourced lists of ad *metadata*—such as page markup and URLs. In turn, publishers and ad networks (including Facebook [9, 88] and 30% of the Alexa top-10K list [95]) continuously adapt and deploy small changes to web page code in an effort to evade, or detect ad-blocking.

Towards visual ad-blocking. This arms race has prompted ad-blockers to search for more robust signals within ads' visual *content*, as altering these would affect user experience. One such signal is the presence of ad-disclosures such as a "Sponsored" caption or the AdChoices logo [24]), which many ad-networks add for transparency [24]. Storey et al. [81] proposed Ad-Highlighter [82], the first *perceptual ad-blocker* that detects ad-disclosures by combining web-filtering rules and computer vision techniques. Motivated by the alleged superior robustness of perceptual techniques [81], popular ad-blockers now incorporate similar ideas. For example, Adblock Plus supports image-matching filters [1], while uBlock crawls Facebook posts in search for "Sponsored" captions [7].

However, as proposed perceptual ad-blockers still partially use markup as a proxy for ads' visual content, they appear insufficient to end the ad-blocking arms race. For example, Facebook routinely evades uBlock Origin using increasingly complex *HTML obfuscation* for the "Sponsored" captions (see [88]). Ad-Highlighter's computer

vision pipeline is also vulnerable to markup tricks such as image fragmentation or spriting (see Figure 10). Escaping the arms race over markup obfuscation requires perceptual ad-blockers to move towards operating on *rendered* web content. This is exemplified in Adblock Plus' Sentinel project [10], that uses deep learning to detect ads directly in web page screenshots. On a similar note, Percival [84] is a recently proposed ad-blocker that adds a deep learning ad-classifier into the rendering pipeline of the Chromium and Brave browsers. While these approaches might bring an end to the current markup-level arms race, our paper shows that visual ad-blocking merely replaces this arms race with a new one, involving powerful attacks that directly target the ad-blockers' visual classifier.

A security analysis of perceptual ad-blocking. In this paper, we present the first comprehensive security analysis of perceptual ad-blockers, and challenge the belief that perceptual signals will end the ad-blocking arms race. To provide a principled analysis of the design space of these nascent ad-blocking techniques, we first propose a general architecture that incorporates and extends existing approaches, e.g., Ad-Highlighter [81, 82], Sentinel [10, 61] and Percival [84]. We view perceptual ad-blocking as a classification pipeline, where segmented web data is fed into one of a variety of possible ad (or ad-disclosure) detectors.

Given this unified view of the design space, we identify and analyze a variety of attacks that affect each step of the ad-classification pipeline. Multiple adversaries—publishers, ad networks, advertisers or content creators—can exploit these vulnerabilities to evade, detect and abuse ad-blockers. Our attacks combine techniques from Web security and from adversarial machine learning [65]. In particular, we leverage visual *adversarial examples* [83], slightly perturbed images that fool state-of-the-art classifiers.

Web attacks on perceptual ad-blockers. First, we show that ad-blocking approaches that combine traditional markup filters and visual signals remain vulnerable to the same attacks as current filter-lists. HTML obfuscation of ad-disclosures is already observed today [88], and we demonstrate similar attacks against Ad-Highlighter's image-matching pipeline (e.g., by fragmenting images). Thus, unless ad-blockers move towards relying on rendered web content (as in Sentinel [10]), perceptual signals will not end the ongoing markup arms race with ad-networks and publishers.

In addition to visual signals, Storey et al. [81] suggest to detect ad-disclosures using *behavioral signals* such as the presence of a link to an ad-policy page. We demonstrate that such signals can lead to serious vulnerabilities (e.g., CSRF, DDoS or click-fraud). Specifically, we show how a Facebook user can trick Ad-Highlighter into making arbitrary web requests in other ad-block users' browsers.

Adversarial examples for ad-classifiers. Ad-blockers can counter the above attacks by operating on *rendered* web content. The main threat to visual ad-blockers are then adversarial examples, which challenge the core assumption that ML can emulate humans' visual ad-detection. To our knowledge, our attacks are the first application of adversarial examples to a real-world web-security problem. [1]

[1]Gilmer et al. [31] argue that the threat model of adversarial examples—in particular the fact that the adversary is restricted to imperceptible perturbations of a given input—is often unrepresentative of real security threats. Perceptual ad-blocking is a perfect example where this threat model is entirely appropriate. The ad-blocker's adversaries—who have white-box access to its classifier—want to evade it on specific inputs (e.g., an ad-network cannot "sample" new ads until it finds one that evades the ad-blocker), with attacks that the user should be oblivious to.

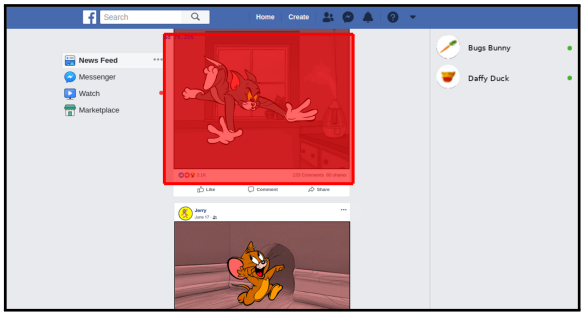


Figure 1: Ad-Blocker Privilege Hijacking. A malicious user, Jerry, posts adversarial content to Facebook that fools a perceptual ad-blocker similar to Sentinel [10] into marking Tom's benign content as an ad (red box) and blocking it in every user's browser.

We rigorously assess the threat of adversarial examples on *seven* visual ad-classifiers: Two computer-vision algorithms (perceptual hashing and OCR) used in Ad-Highlighter [81]; the ad-classification neural networks used by Percival [84] and [40]; a canonical feature matching model based on the *Scale-Invariant Feature Transform* (SIFT) [52]; and two object detector networks emulating Sentinel [10]. For each model, we create imperceptibly perturbed ads, ad-disclosure or native content, that either evade the model's detection or falsely trigger it (as a means of detecting ad-blocking).

Among our contributions is a new evasion attack [41, 73] on SIFT [52] that is conceptually simpler than prior work [39].

Attacking perceptual ad-blockers such as Sentinel [10] presents the most interesting challenges. For these, the classifier's input is a screenshot of a web page with contents controlled by different entities (e.g, publishers and ad networks). Adversarial perturbations must thus be encoded into HTML elements that the adversary controls, be robust to content changes from other parties, and scale to thousands of pages and ads. We tackle the adversary's uncertainty about other parties' page contents by adapting techniques used for creating physical adversarial examples [12, 75]. We also propose a novel application of *universal adversarial examples* [56] to create a *single* perturbation that can be applied at scale to all combinations of websites and ads with near 100% success probability.

We further show that adversarial examples enable new attacks, wherein malicious content from one user can hijack the ad-blocker's high privilege to incorrectly block another user's content. An example is shown in Figure 1. Here Jerry, the adversary, uploads a perturbed image to Facebook. That image is placed next to Tom's post, and confuses the ad-blocker into classifying Tom's benign post as an ad, and incorrectly blocking it. Hence, a malicious post by one user can cause another user's post to get blocked.

Moving beyond the Web and visual domain, we build imperceptible audio adversarial examples for AdblockRadio [2], a radio ad-blocker that uses ML to detect ads in raw audio streams.

Outlook. While visual ad-classification of rendered web content is both sufficient and necessary to bring an end to the arms race around page markup obfuscation, we show that this merely replaces

(a) (b) (c)

Figure 2: The AdChoices Logo. AdChoices is a standard for disclosure of behavioral advertising [24]. Ads are marked by the icon (a), with optional text (b). Despite creative guidelines [25], many variants of the logo are in use (c).

one arms race with a new one centered on adversarial examples. Our attacks are not just a first step in this new arms race, where ad-blockers can easily regain the upper hand. Instead, they describe an inherent difficulty with the perceptual ad-blocking approach, as ad-blockers operate in essentially the *worst threat model for visual classifiers*. Their adversaries prepare (offline) digital attacks to evade or falsely trigger a known *white-box* visual classifier running inside the ad-blocker. In contrast, the ad-blocker must resist these attacks while operating under strict real-time constraints.

Our study's goal is not to downplay the merits of ad-blocking, nor discredit the perceptual ad-blocking philosophy. Indeed, ML might one day achieve human-level perception. Instead, we highlight and raise awareness of the inherent vulnerabilities that arise from instantiating perceptual ad-blockers with existing ML techniques.

Contributions. This paper makes the following contributions:

- We conduct a detailed security analysis of perceptual ad-blocking;
- We present nine general classes of attacks against the various components of the perceptual ad-blocking pipeline;
- We evaluate adversarial examples for eight ad classifiers (seven visual, one audio). We make novel use of transformation-robust [12] and universal adversarial examples [56] to create scalable attacks robust to arbitrary changes in web content.
- We release all our data and classifiers, including a new neural network that locates ads in web page screenshots, that may prove useful *in non-adversarial settings*: https://github.com/ftramer/ad-versarial

2 PRELIMINARIES AND BACKGROUND

2.1 The Online Advertising Ecosystem

Online advertising comprises four actors: users, publishers, ad networks, and advertisers. Users browse websites owned or curated by a publisher. Publishers assigns parts of the site's layout to advertisements. Control of these spaces is often outsourced to an ad network that populates them with advertisers' contents.

To protect users from deceptive ads, the Federal Trade Commission and similar non-US agencies require ads to be clearly recognizable [26]. These provisions have also spawned industry self-regulation, such as the *AdChoices* standard [24] (see Figure 2).

2.2 Perceptual Ad-Blocking

Perceptual ad-blocking aims at identifying ads from their content, rather than from ad *metadata* such as URLs and markup. The insight of Storey et al. [81] is that many ads are explicitly marked—e.g., via a "Sponsored" link or the AdChoices logo—to comply with regulations on deceptive advertising. They developed Ad-Highlighter [82], an ad-blocker that detects ad-disclosures using different perceptual techniques: (i) textual searches for "Sponsored" tags, (ii) fuzzy image search and OCR to detect the AdChoices logo, and (iii) "behavioral" detection of ad-disclosures by identifying links to ad-policy pages. Ad-blockers that rely on perceptual signals are presumed to be less prone to an arms race, as altering these signals would affect user experience or violate ad-disclosure regulations [81].

Perceptual ad-blocking has drawn the attention of major ad-blockers, that have integrated visual signals into their pipelines. For example, uBlock blocks Facebook ads by detecting the "Sponsored" caption. Adblock Plus has added support for image-matching rules, which are easily extended to fuzzy image search [8].

The above perceptual ad-blocking approaches still rely on some markup data as a proxy for ads' visual content. This has prompted an ongoing arms race between Facebook and uBlock (see [88]) where the former continuously obfuscates the HTML tags that render its "Sponsored" tag—a process that is invisible to the user. This weakness is fundamental to perceptual approaches that rely on signals with an indirect correspondence to ads' rendered content. This insight led Adblock Plus to announce the ambitious goal of detecting ads directly from rendered web pages—with no reliance on markup—by leveraging advances in image classification. Their Sentinel [10] project uses an object-detection neural network to locate ads in raw Facebook screenshots. The recently released Percival project [84] targets a similar goal, by embedding a deep-learning based ad-blocker directly into Chromium's rendering engine.

2.2.1 Design and Goals. Ad-blockers are client-side programs running in browsers at a high privilege level. They can be implemented as browser extensions or integrated in the browser. We ignore DNS ad-blockers (e.g., Pi-hole) as these cannot use perceptual signals. [2]

The goal of ad-blockers is to identify and hide ads, while guarding against website breakage [4] resulting from the removal of functional content. As opposed to network-level filters, perceptual signals only apply to downloaded Web content and are thus unsuitable for some secondary goals of ad-blockers, such as bandwidth saving or blocking of user tracking and malvertising [46, 51, 68, 93].

Ad-blockers may strive to remove ads without being detected by the publisher. For example, many websites try to detect ad-blockers [58] and take according action (e.g., by asking users to disable ad-blockers). As perceptual ad-blockers do not interfere with web requests, they are undetectable by the web-server [81]. However, the publisher's JavaScript code can try to detect ad-blockers by observing changes in the DOM page when hiding ads.

Finally, perceptual ad-blockers have strict timing constraints, and should process a web page and detect ads in close to real-time.

2.2.2 Algorithms for Visual Ad Classification. The identification of ads or ad-disclosures can be achieved using a variety of computer vision techniques. Below, we describe existing approaches.

- *Template matching.* Ad-Highlighter detects the AdChoices logo by comparing each image in a page to a template using *average*

[2] While our work focuses on the desktop browser setting, perceptual ad-blocking might also prove useful in the mobile domain. Current mobile ad-blockers are often part of a custom browser, or act as web proxies—an insufficient approach for native apps that prevent proxying using certificate pinning. Instead, a perceptual ad-blocker (potentially with root access) could detects ads directly from app screenshots

hashing: for each image, a hash is produced by resizing the image to a fixed size and setting the i^{th} bit in the hash to 1 if the i^{th} pixel is above the mean pixel value. An image matches the template if their hashes have a small Hamming distance. A more robust template matching algorithm is the Scale Invariant Feature Transform (SIFT) [52], which creates an image hash from detected "keypoints" (e.g., edges and corners).

- *Optical Character Recognition.* To detect the rendered text inside the AdChoices logo, Ad-Highlighter uses the open-source Tesseract OCR system [6]. Tesseract splits an image into overlapping frames and transcribes this sequence using a neural network. Ad-Highlighter matches images for which the OCR output has an edit-distance with "AdChoices" below 5.
- *Image Classification.* Albeit not in an ad-blocking context, Hussain et al. [40] have demonstrated that neural networks could be trained to distinguish images of ads from non-ads (without the presence of any explicit ad-disclosures). The Percival project trained a similar neural network to classify image frames in real-time within Chromium's rendering pipeline [84].
- *Object Detection.* Sentinel [10] detects ads in rendered web pages using an object detector network based on YOLOv3 [72]. The network's output encodes locations of ads in an image. The YOLOv3 [72] model outputs bounding box coordinates and confidence scores for $B = 10{,}647$ object predictions, and retains those with confidence above a threshold $\tau = 0.5$.

2.3 Threat Model and Adversaries

We adopt the terminology of adversarial ML [65], where the defenders are users of a classifier (the ad-blocker) that its adversaries (e.g., ad networks or publishers) are trying to subvert.

Publishers, ad networks, and advertisers have financial incentives to evade or detect ad-blockers. We assume that publishers and ad networks are rational attackers that abide by regulations on online advertising, and also have incentives to avoid actively harming users or disrupting their browsing experience. As shown in prior work [67, 93], this assumption fails to hold for advertisers, as some have abused ad-networks for distributing malware. We assume that advertisers and content creators (e.g., a Facebook user) may try to actively attack ad-block users or other parties.

As ad-blockers are client-side software, adversaries can download and inspect their code offline. However, we assume that adversaries do not know *a priori* whether a user is running an ad-blocker.

Attacking ad-blockers. The primary adversarial goal of publishers, ad-networks and advertisers is to evade the ad-blocker's detection and display ads to users. These adversaries may modify the structure and content of web pages or ads to fool the ad-detector.

Alternatively, the ad-blocker's adversaries may try to detect its presence, to display warnings or deny access to the user. A common strategy (used by 30% of publishers in the Alexa top-10k) adds fake ad-content (honeypots) to a page and uses JavaScript to check if the ads were blocked [95]. This practice leads to an orthogonal arms race on ad-block detection [57, 58, 60] (see Appendix A).

Adversaries may also try to abuse ad-blockers' behaviors to degrade their usability (e.g., by intentionally causing site-breakage or slow performance). The viability of such attacks depends on the adversary's incentives to avoid disrupting ad-block users' browsing experience (e.g., Facebook adds honeypots to regular user posts to cause site-breakage for ad-block users [88]).

Finally, attackers with no ties to the online advertisement ecosystem may try to hijack an ad-blocker's high privilege-level in other users' machines. Such attackers can act as advertisers or content creators to upload malicious content that exploits an ad-blocker's vulnerabilities. Figure 1 shows one example of such an attack, where a malicious Facebook user uploads content that tricks the ad-blocker into hiding an honest user's posts. We will also show how Facebook users can exploit Ad-Highlighter's behavioral ad-blocking to trigger arbitrary web requests in other users' browsers.

2.4 Adversarial Examples

The attacks presented in this paper combines techniques from Web security and from adversarial machine learning. In particular, we leverage adversarial examples [83] to fool perceptual ad-classifiers.

An adversarial example for an input x of a model f is an input $\hat{x} = x + \delta$, where δ is a "small" perturbation such that $\hat{x}$ is misclassified with high confidence. We will consider perturbations with small ℓ_2 norm (Euclidean) or ℓ_∞ norm (maximum per-pixel change). To cause a model f to misclassify $x + \delta$ we would minimize the confidence of $f(x + \delta)$ in the true class, while also keeping δ small. This is often achieved by minimizing a differentiable *loss function* $\mathcal{L}(x + \delta)$ that acts as a proxy for the adversarial goal.

An effective algorithm for finding adversarial examples is *Projected Gradient Descent* (PGD) [48, 53]. Given an allowable perturbation set (e.g., $||\delta||_\infty \leq \epsilon$), we repeatedly update δ in the gradient direction $-\nabla_\delta \mathcal{L}(x + \delta)$ and project back onto the allowable set.

In some cases, we will want perturbations to be re-usable so that an attack can scale to a large number of websites or ads. A perturbation that can be re-used for many different inputs is called a *universal* adversarial example [56]. It is usually created by jointly minimizing $\sum_i \mathcal{L}(x^{(i)} + \delta)$ over many inputs $x^{(i)}$, for a common δ.

3 DESIGNING PERCEPTUAL AD-BLOCKERS

To analyze the security of perceptual ad-blockers, we first propose a unified architecture that incorporates and extends prior and concurrent work (e.g., Ad-Highlighter [81], visual filter-lists [8], Sentinel [10], and the recent Percival patch for Chromium's rendering engine). We explore different ways in which ad-blockers can integrate perceptual signals, and identify a variety of computer vision and ML techniques that can be used to visually identify ads.

To simplify exposition, we restrict our analysis to ad-blockers that only rely on perceptual signals. In practice, these signals are likely to be combined with existing filter lists (as in uBlock [88] or Adblock Plus [8]) but the details of such integrations are orthogonal to our work. We note that an ad-blocker that combines perceptual signals with filter lists inherits the vulnerabilities of both, so our security analysis applies to these hybrid approaches as well.

3.1 General Architecture

A perceptual ad-blocker is defined by a collection of offline and online steps, with the goal of creating, maintaining and using a classifier to detect ads. Figure 3 shows our unified architecture for perceptual ad-blockers. The ad-blocker's core visual classifier can

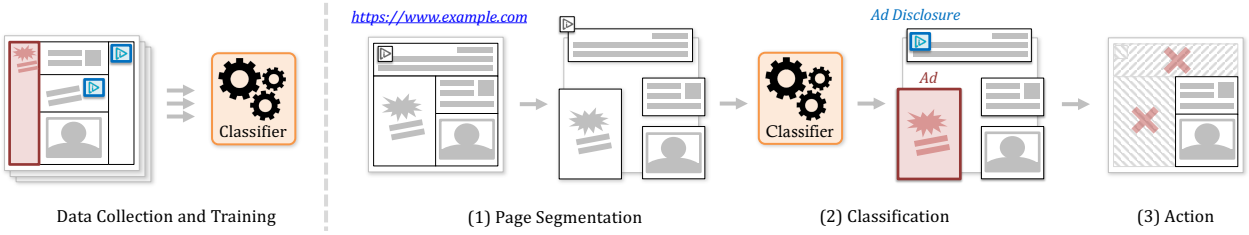


Figure 3: The Architecture of a Perceptual Ad-Blocker. In the offline phase, an ad-classifier is trained on web data. In the online phase, the ad-blocker segments visited pages (1), classifies individual elements (2), and renders the user's ad-free viewport (3).

range from classical computer vision as in Ad-Highlighter [82] to large ML models as in Sentinel [10].

The classifier may be trained using labeled web data, the type and amount of which varies by classifier. Due to continuous changes in web markup, ad-blockers may need regular updates, which can range from extending existing rules (e.g., for Ad-Highlighter [81, 82]) to re-training complex ML models such as Sentinel [10].

When deployed by a user, the ad-blocker analyzes data from visited pages to detect and block ads in real-time. Ad detection consists of three main steps. (1) The ad-blocker optionally segments the web page into smaller chunks. (2) A classifier labels each chunk as ad or non-ad content. (3) The ad-blocker acts on the underlying web page based on these predictions (e.g., to remove HTML elements labeled as ads). For some ad-classifiers, the segmentation step may be skipped. For example, Sentinel [10] uses an object-detection network that directly processes full web page screenshots.

Ad-Highlighter's use of behavioral signals (i.e., recognizing ad-disclosures by the presence of a link to an ad-policy page) can be seen as a special type of classifier that may interact with segmented Web elements (e.g., by clicking and following a link).

3.2 Approaches to Ad Detection

When online, a perceptual ad-blocker's first action is the "Page Segmentation" step that prepares inputs for the classifier. Figure 4 illustrates different possible segmentations. A cross-origin `iframe` (red box 3) displays an ad and an AdChoices icon (purple box 2). An additional textual ad-disclosure is added by the publisher outside the `iframe` (purple box 1). Publishers may use `iframes` to display native content such as videos (e.g., red box 4).

We distinguish three main perceptual ad-blocking designs that vary in the granularity of their segmentation step, and in turn in the choice of classifier and actions taken to block ads.

- *Element-based perceptual ad-blockers*, such as Ad-Highlighter, search a page's DOM tree for HTML elements that identify ads, e.g., the AdChoices logo or other ad-disclosures.
- *Page-based perceptual ad-blockers*, e.g., Sentinel [10], ignore the DOM and classify images of rendered web pages.
- *Frame-based perceptual ad-blockers*, e.g., Percival [84], classify rendered content but pre-segment pages into smaller frames.

3.2.1 Element-based Perceptual Ad-blocking. These ad-blockers segment pages into HTML elements that are likely to contain ad-disclosures. The segmentation can be coarse (e.g., Ad-Highlighter extracts all `img` tags from a page) or use custom filters as in Adblock Plus' image search [8] or Ublock's Facebook filters [88].

For textual ad-disclosures (e.g., Facebook's "Sponsored" tag) the classification step involves trivial string matching. Facebook is thus deploying HTML obfuscation that targets an ad-blocker's ability to find these tags [88]. This ongoing arms race calls for the use of visual (markup-less) detection techniques. Ad-disclosure logos (e.g., the AdChoices icon) can be visually classified using *template matching*. Yet, due to many small variations in ad-disclosures in use, *exact* matching (as in Adblock Plus [8]) is likely insufficient [81]. Instead, Ad-Highlighter uses *perceptual hashing* to match all `img` elements against the AdChoices logo. Ad-Highlighter also uses supervised ML—namely Optical Character Recognition (OCR)—to detect the "AdChoices" text [82]. Once an ad-disclosure is identified, the associated ad is found using custom rules (e.g., when Ad-Highlighter finds an AdChoices logo, it blocks the parent `iframe`).

Storey et al. [81] further suggest to detect ads through *behavioral signals* that capture the ways in which users can interact with them, e.g., the presence of a link to an ad-policy page.

3.2.2 Frame-based Perceptual Ad-blocking. The above element-based approaches require mapping elements in the DOM to rendered content (to ensure that elements are visible, and to map detected ad-identifiers to ads). As we show in Section 4.4, this step is non-trivial and exploitable if ad-blockers do not closely emulate the browser's DOM rendering, a complex process that varies across browsers. For instance, image fragmentation or spriting (see Figure 10) are simple obfuscation techniques that fool Ad-Highlighter, and would engender another cat and mouse game. To avoid this, ad-blockers can directly operate on rendered images of a page, which many browsers (e.g., Chrome and Firefox) make available to extensions. Instead of operating on an entire rendered web page (see page-based ad-blockers below), DOM features can still be used to segment a page into regions likely to contain ads. For example, segmenting a page into screenshots of each `iframe` is a good starting point for detecting ads from external ad networks. The approach of Percival is also frame-based but directly relies on image frames produced during the browser's rendering process [84].

We consider two ways to classify frames. The first searches for ad-disclosures in rendered ads. Template-matching is insufficient due to the variability of backgrounds that ad-disclosures are overlaid on. Instead, we view this as an object-detection problem and address it with supervised ML. The second approach is to train a visual classifier to directly detect ad content. Hussain et al. [40] report promising results for this task. Percival also relies on a lightweight deep learning model to classify frames as ad content [84].

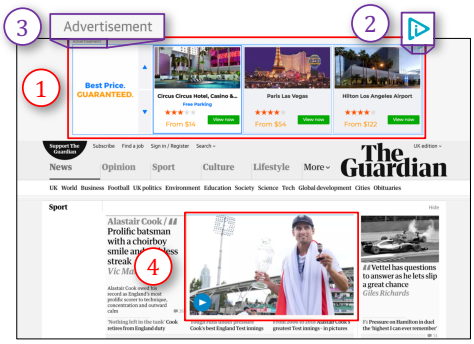


Figure 4: Perceptual Ad-Blocking Elements. An ad (box #1) is displayed in an `iframe`, that contains an AdChoices icon (box #2). A custom ad-disclosure from the publisher is outside the `iframe` (box #3). Publishers can use `iframes` to display non-ad content such as videos (box #4).

3.2.3 Page-based Perceptual Ad-blocking. The core idea of perceptual ad-blocking is to emulate the way humans detect ads. Element- and frame-based approaches embrace this goal to some extent, but still rely on DOM information that humans are oblivious to. Recently, Adblock Plus proposed an approach that fully emulates visual detection of online ads from rendered web content alone [10].

In a page-based ad-blocker, segmentation is integrated into the classifier. Its core task is best viewed as an object-detection problem: given a web page screenshot, identify the location and dimension of ads. Adblock Plus trained the YOLOv3 object-detector [72] on screenshots of Facebook with ads labeled using standard filter-lists.

Once ad locations are predicted, the ad-blocker can overlay them to hide ads, or remove the underlying HTML elements (e.g., by using the `document.elementFromPoint` browser API to get the HTML element rendered at some coordinate).

4 ATTACKS ON PERCEPTUAL AD-BLOCKING

Given the unified architecture from Section 3, we now perform a comprehensive security analysis of the perceptual ad-blocking pipeline and describe multiple attacks targeting concrete instantiations of each of the ad-blocker's components. The primary focus of our analysis is to evaluate the robustness of the ad-blocker's core visual classifier, by instantiating adversarial examples for seven different and varied approaches to ad-detection (Section 4.2). We further demonstrate powerful attacks that exploit the ad-blocker's high-privilege actions (Section 4.3). We conclude by describing more classical attacks that affect the segmentation step of current perceptual ad-blockers (Section 4.4), as well as potential attacks on an ad-blocker's offline data collection and training phase (Section 4.5).

Our attacks can be mounted by different adversaries (e.g., publishers, ad-networks, or malicious third parties) to evade or detect ad-blocking and, at times, abuse the ad-blocker's high privilege level to bypass web security boundaries. These attacks, summarized in Table 2, challenge the belief that perceptual signals can tilt the arms race with publishers and ad-networks in favor of ad-blockers. All our data, pre-trained models, and attack code are available at https://github.com/ftramer/ad-versarial.

The attacks described in this section do not violate existing laws or regulations on deceptive advertising, as the changes to the visual content of a page are imperceptible to human users.

4.1 Evaluation Setup

4.1.1 Evaluated Approaches. We analyze a variety of techniques to instantiate the different stages of the perceptual ad-blocking pipeline. In particular, we evaluate *seven* distinct approaches to the ad-blocker's core visual ad-classification step (see Table 1). Three are element-based, three frame-based, and one page-based. These seven classifiers are taken from or inspired by prior work. They are: Two computer vision algorithms used in Ad-Highlighter [81, 82] (average hashing and OCR); two ad classifiers, one from Hussain et al. [40] and one used in Percival [84]; a robust feature matcher, SIFT [52]; and two object detector networks—with the same YOLOv3 model [72] as Sentinel [10, 61]—which we trained to detect either ad-disclosures in frames, or ads in a full web page.

For the two object detector models we built, we explicitly separated (i.e., assigned to non-communicating authors) the tasks of (1) data-collection, design and training; and (2) development of attacks, to ensure fair evaluation results. Our first (frame-based) model was trained to detect AdChoices logos that we overlaid in a dataset of 6,320 ads collected by Hussain et al. [40]. We then classify an `iframe` as an ad, if the model detects the AdChoices logo in it.

Our second model emulates the approach of the unreleased Sentinel [10, 61] and was trained to detect ads in arbitrary news websites. This broadens Sentinel's original scope (which was limited to Facebook)—a decision we made due to difficulties in collecting sufficient training data [61]. One author trained YOLOv3 to locate ads in screenshots of news websites from all G20 nations. To collect a diverse dataset of labeled ads in web screenshots, we first locates ads using a web-proxy based on filter lists, and then randomly replace ads with a larger variety of examples. More details about this process, of independent interest, are in Appendix B. A video of our model in action on five websites not seen during training is available at https://github.com/ftramer/ad-versarial/blob/master/videos.

4.1.2 Evaluation Data. We use real website data to evaluate the accuracy and robustness of the above seven ad-classifiers. We built an evaluation set from the top ten news websites in the Alexa ranking (see Table 3). For each website, we extract the following data:

(1) All images smaller than 50KB in the DOM. This data is used to evaluate element-based techniques. We collect 864 images, 41 of which are AdChoices logos (17/41 logos contain the "AdChoices" text in addition to the icon).

(2) A screenshot of each `iframe` in the DOM tree, to evaluate frame-based models. We collect 59 frames. Of these, 39 are ads and 29 contain an AdChoices logo. Percival [84] only considers images of dimension at least 100×100 px so we limit it to these.[3]

[3]Taking a screenshot of an `iframe` is an approximation of how Chromium's rendering engine segments frames for Percival's classifier. We verified that our attacks on Percival's network work when deployed inside the Chromium browser.

Table 1: Evaluation of Ad-Classifiers. For each classifier, we first evaluate on "benign" data collected from websites. We report false-positives (FP)—mis-classified non-ad content—and false negatives (FN)—ad-content that the classifier missed. We then give the the attack model(s) considered when *evading* the classifier, the success rate, and the corresponding section.

			Benign Eval.		Adversarial Eval.	
Category	**Method**	**Targets**	**FP**	**FN**	**Attack Model for Evasion**	**Success**
Element-based	Blacklist	AdChoices logos	0/824	33/41	N.A.	-
	Avg. hash [81]	AdChoices logos	3/824	3/41	Add ≤ 3 empty rows/cols	100%
	SIFT	textual AdChoices	2/824	0/17	$\ell_2 \leq 1.5$	100%
	OCR [81]	textual AdChoices	0/824	1/17	$\ell_2 \leq 2.0$	100%
Frame-based	YOLOv3	AdChoices in `iframe`	0/20	5/29	$\ell_\infty \leq 4/255$	100%
	ResNet [40]	ad in `iframe`	0/20	21/39	$\ell_\infty \leq 2/255$	100%
	Percival [84]	large ads in `iframe`	2/7	3/33	$\ell_\infty \leq 2/255$	100%
Page-based	YOLOv3	ads visible in page screenshot	2	6/30	Publisher: universal full-page mask (99% transparency)	100%
					Publisher: adv. content below ads on BBC.com, $\ell_\infty \leq 3/255$	100%
					Ad network: universal mask for ads on BBC.com, $\ell_\infty \leq 4/255$	95%

(3) Two screenshots per website (the front-page and an article) taken in Google Chrome on a 1920 × 1080 display.[4] These are used to evaluate page-based models. Each screenshot contains 1 or 2 fully visible ads, with 30 ads in total.

For template-matching approaches (perceptual hashing and SIFT) we use the same 12 AdChoices templates as Ad-Highlighter [82].

When describing an ad-blocker's page segmentation and the corresponding markup obfuscation attacks in Section 4.4, we use some data collected on Facebook.com in November 2018. As Facebook continuously and aggressively adapts the obfuscation techniques it uses to target ad-blockers [88], the specific attacks we describe may have changed, which only goes to illustrate the ongoing arms race and need for more robust markup-less ad-blocking techniques.

4.1.3 Accuracy and Performance of ML Classifiers. Table 1 reports the accuracy of the seven ad-classifiers on our evaluation data. For completeness, we include a blacklist that marks any image that exactly matches one of the 12 AdChoices logos used in Ad-Highlighter. As posited by Storey et al. [81], this approach is insufficient.

Note that the datasets described above are incomparable. Some ads are not in `iframes`, or have no ad-disclosure, ans screenshots only contain images within the current view. Thus, the accuracy of the classifiers is also incomparable. This does not matter, as our aim is not to find the best classifier, but to show that *all* of them are insecure in the stringent attack model of visual ad-blockers.

Overall, element-based approaches have high accuracy but may suffer from some false-positives (i.e., non-ad content classified as ads) that can lead to site-breakage. The frame-based approaches are less accurate but have no false-positives. Finally, our Sentinel-like detector shows promising (albeit imperfect) results that demonstrate the possibility of ad-detection on arbitrary websites.

We measure performance of each classifier on an Intel Core i7-6700 Skylake Quad-Core 3.40GHz. While average hashing and SIFT process all images in a page in less than 4 seconds, OCR is much slower (Ad-Highlighter disables it by default). Our OCR model parses an image in 100 ms, a 14 second delay on some websites. The frame-based classifiers process all `iframes` in 1-7 seconds. Our page-based model processes pages downsized to 416 × 416px at 1.5 frames-per-second (on CPU), which may suffice for ad-blocking. The authors of Percival recently demonstrated that an optimized deployment of perceptual ad-blocking with a deep learning classifier incurs only minimal overhead on page rendering (< 200 ms).

[4]We experimentally verified that our attacks on page-based ad-blockers are robust to changes in the user's viewport. An attacker could also explicitly incorporate multiple browsers and display sizes into its training set to create more robust attacks. Alternatively, the adversary could first detect the type of browser and viewport (properties that are easily and routinely accessed in JavaScript) and then deploy "responsive" attacks tailored to the user's setting.

4.2 Attacks against Classification with Adversarial Examples

For perceptual ad-blockers that operate over images (whether on segmented elements as in Ad-Highlighter [82], or rendered content as in Sentinel [10] or Percival [84]), security is contingent on the robustness of the ad-blocker's visual classifier. *False negatives* result in ads being shown, and *false positives* cause non-ads to be blocked.

Both error types are exploitable using *adversarial examples* [33, 83]—small input perturbations that fool a classifier. Adversarial examples can be used to generate web content that fools the ad-blocker's classifier, without affecting a user's browsing experience.

In this section, we describe and evaluate four concrete types of attacks on the seven visual classifiers we consider: (C1) *adversarial ad-disclosures* that evade detection; (C2) *adversarial ads* that evade detection; (C3) *adversarial non-ad content* that alters the classifier's output on nearby ads; (C4) *adversarial honeypots* (misclassified non-ad elements, to detect ad-blocking). Our attacks allow adversaries to evade or detect ad-blocking with (near)-100% probability.

Attack Model. We consider adversaries that perturb web content to produce *false-negatives* (to evade ad-blocking) or *false-positives* (honeypots to detect ad-blocking). Each attack targets a single classifier—but is easily extended to multiple models (see Section 5).

- *False negative.* To evade ad-blocking, publishers, ad networks or advertisers can perturb any web content they control, but aim to make their attacks imperceptible. We consider perturbations with small ℓ_2 or ℓ_∞ norm (for images with pixels normalized to $[0, 1]$)—a sufficient condition for imperceptibility. An exception to the above are our attacks on average hashing, which is by design invariant to small ℓ_p changes but highly vulnerable to other imperceptible variations. The attack model used for all evasion attacks are summarized in Table 1.
- *False positive.* The space of non-disruptive false positive attacks is vast. We focus on one easy-to-deploy attack, that generates

Table 2: Attack Strategies on Perceptual Ad-Blockers. Strategies are grouped by the component that they exploit—(D)ata collection, (S)egmentation, (C)lassification, (A)ction. For each strategy, we specify which goals it can achieve, which adversaries can execute it, and which ad-blockers it applies to (fully: ● or partially: ◐).

	Goals			Actors				Target		
Strategy	Evasion	Detection	Abuse	Publisher	Ad Network	Advertiser	Content creator	Element-based	Frame-based	Page-based
D1: Data Training Poisoning	●	●	●	●	●	●	●	●	●	●
S1: DOM Obfuscation	●	●	○	●	●	○	○	●	◐	○
S2: Resource Exhaustion (over-Segmentation)	●	○	○	●	●	◐	◐	●	◐	○
C1: Evasion with Adversarial Ad-Disclosures	●	○	○	○	●	○	○	●	○	○
C2: Evasion with Adversarial Ads	●	○	○	○	●	●	○	○	●	●
C3: Evasion with Adversarial Content	●	○	○	●	○	○	●	○	○	●
C4: Detection with Adversarial Honeypots	○	●	○	●	○	○	○	●	●	●
A1: Cross-Boundary Blocking	○	○	●	○	○	●	●	○	○	●
A2: Cross-Origin Web Requests	○	○	●	○	○	●	●	●	○	○

near-uniform rectangular blocks that blend into the page's background yet falsely trigger the ad-detector.

We assume the publisher controls the page's HTML and CSS, but cannot access the content of ad frames. This content, including the AdChoices logo, is added by the ad network.

Gilmer et al. [31] argue that the typical setting of adversarial examples, where the adversary is restricted to finding imperceptible perturbations for given inputs, is often unrepresentative of actual security threats. Interestingly, the threat model for visual ad classifiers does align perfectly with this setting. The ad-blocker's adversaries want to evade its classifier for a specific input (e.g., the publisher's current web page and an advertiser's latest ad campaign), while ensuring that the users' browsing experience is unaffected.

4.2.1 Overview of Attack Techniques and Results. For all seven ad-classifiers, we craft imperceptible adversarial perturbations for ad-disclosures, ads and other web content, which can be used by publishers, ad-networks, or advertisers to evade or detect ad-blocking.

Some of our classifiers can be attacked using existing techniques. For example, we show that ad-networks and publishers can use standard gradient-based attacks (see Section 2.4) to create imperceptibly perturbed ads or background content that fool our two frame-based classifiers with 100% success rates (see Figure 6). We verify that similar attacks bypass the model used in Percival [84].

Attacking element-based classifiers is less straightforward, as they operate on small images (adversarial examples are presumed to be a consequence of high dimensional data [32]), and some rely on traditional computer vision algorithms (e.g., average hashing or SIFT) for which gradient-based attacks do not apply. Nevertheless, we succeed in creating virtually invisible perturbations for the AdChoices logo, or background honeypot elements, that fool these classifiers (see Figure 5). Our attacks on Ad-Highlighter's OCR network build upon prior work by Song and Shmatikov [79]. For non-parametric algorithms such as SIFT, we propose a new generic attack using black-box optimization [41, 73] (see Section 4.2.2), that is conceptually simpler than previous attacks [39].

Our most interesting attacks are those that target page-based ad-blockers such as Sentinel [10] (see Figure 8, as well as Figure 14). Our attacks let publishers create perturbed web content to evade or detect ad-blocking, and let ad-networks perturb ads that evade ad-blocking on the multitude of websites that they are deployed in. These attacks overcome a series of novel constraints.

First, attacks on visual ML classifiers often assume that the adversary controls the full digital image fed to the classifier. This is not the case for page-based ad-blockers, whose input is a screenshot of a web document with content controlled by different actors (e.g., ad networks only control the content of ad frames, while publishers can make arbitrary website changes but cannot alter ads loaded in cross-origin `iframes`). Moreover, neither actor precisely knows what content the other actors will provide. Adversarial examples for page-based ad-blockers thus need to be encoded into the HTML elements that the adversary controls, and must be robust to variations in other page content. We solve this constraint with techniques similar to those used to make physical-world adversarial examples robust to random transformations [28, 48, 75]. We consider multiple tricks to encode a publisher's perturbations into valid HTML One attack uses CSS rules to overlay a near-transparent perturbed mask over the full page (Figure 8 (b)). To detect ad-blocking, we craft an innocuous page-footer that triggers the ad-blocker (Figure 8 (d)). Details on our attacks are in Section 4.2.2.

A further challenge is the deployment of these attacks at scale, as creating perturbations for every ad and website is intractable. This challenge is exactly addressed by attacks that create *universal adversarial examples* [56]—*single* perturbations that are crafted so as to be effective when applied to most classifier inputs. Universal perturbations were originally presented as a curious consequence of the geometry of ML classifiers [56], and their usefulness for the scalability of attacks had not yet been suggested.

Attacks on page-based ad-blockers have unique constraints, but also enable unique exploits. Indeed, as a page-based classifier produces outputs based on a single full-page input, perturbing content controlled by the attacker can also affect the classifier's outputs on unperturbed page regions. The effectiveness of such attacks depends on the classifier. For the YOLOv3 [72] architecture, we show that publishers can perturb website content near ad `iframes` so as to fool the classifier into missing the actual ads (see Figure 14).

4.2.2 Algorithms for Adversarial Examples. For some of the considered classifiers, adversarial examples for each of the attack strategies C1-C4 in Table 2 can be constructed using existing and well-known techniques (see Section 2.4). Below, we provide more details on the attack we use to target SIFT, and on the techniques we use to create robust and scalable attacks for page-based classifiers [10].

Black-box optimization attacks for non-parametric classifiers. SIFT is a non-parametric algorithm (i.e., with no learned parameters). As such, the standard approach for generating adversarial examples by minimizing the model's training-loss function does not apply [83]. To remedy this, we first formulate a near-continuous loss function $\mathcal{L}_{\text{SIFT}}(x + \delta)$ that acts as a proxy for SIFT's similarity measure between the perturbed image $x + \delta$ and some fixed template. The next difficulty is that this loss function is hard to differentiate, so we use black-box optimization techniques [41, 73] to minimize $\mathcal{L}_{\text{SIFT}}$.

SIFT's output is a variable-sized set of keypoints, where each keypoint is a vector $v \in \mathbb{R}^{132}$—four positional values, and a 128-dimensional *descriptor* [52]. Let t be a template with keypoint descriptors T. To match an image x against t, SIFT computes descriptor vectors for x, denoted $\{v_1, \ldots, v_m\}$. Then, for each v_i it finds the distances $d_{i,1}, d_{i,2}$ to its two nearest neighbors in T. The keypoint v_i is a match if the *ratio test* $d_{i,1}/d_{i,2} < \tau$ holds (where $\tau = 0.6$). Let $M(x,t)$ be the keypoints of x that match with t. To evade detection, we minimize the size of M via the following proxy loss:

$$\mathcal{L}_{\text{SIFT}}(x+\delta) := \sum_{v_i \in M_\tau(x,t)} d_{i,2}/d_{i,1} \ . \tag{1}$$

Minimizing $\mathcal{L}$ increases $d_{\cdot,1}/d_{\cdot,2}$ for matched keypoints until they fall below the ratio test. To create false positives, we minimize an analogous loss that sums over $v_i \notin M_\tau(x,t)$ and decreases the ratio.

Scalable attacks with partial input control. When attacking page-based classifiers, we need to overcome two challenges: (1) the attacker only controls part of the page content and does not know which content other actors will add; (2) the attacks should be deployable at scale for a variety of web pages and ads. To create adversarial examples under these novel constraints, we combine universal [56] and transformation-robust [29, 48, 75] attacks.

To create universal perturbations, we collect additional website screenshots: D^{train} is a set of 200 screenshots of news websites, and D^{eval} contains the 20 screenshots collected in Section 4.1 (no website or ad appears in both sets). We also collect $D^{\text{train}}_{\text{BBC}}$ and $D^{\text{eval}}_{\text{BBC}}$ with 180 and 20 screenshots from bbc.com/sports. The training sets are used to create perturbations that work for arbitrary websites or ads. We measure attacks' success rates on the evaluation sets.

We craft a perturbation δ by minimizing $\sum_{x \in D^{\text{train}}_*} \mathcal{L}(x \odot \delta)$, where $x \odot \delta$ means applying the perturbation δ to a page x. Depending on the attack, the perturbation is added pixel-wise to a page region that the adversary controls, or replaces that region with δ. All that remains is the design of a suitable loss function $\mathcal{L}$.

The YOLOv3 model we trained outputs multiple $B = 10{,}647$ boxes for detected ads, and retains a box b if its confidence—denoted $\mathsf{conf}(f(x), b)$—is larger than a threshold τ. To cause ads to be undetected, we thus minimize the following loss which causes all B boxes to have confidence below $\tau - \kappa$, for some slack $\kappa > 0$:

$$\mathcal{L}^{\text{FN}}_{\text{YOLO}}(x \odot \delta) := \sum_{1 \le b \le B} \max\left(\mathsf{conf}(f(x \odot \delta), b) - (\tau - \kappa), 0\right) \ , \tag{2}$$

For false-positives, i.e., a fake object prediction, we instead increase all boxes' confidence up to $\tau + \kappa$ by minimizing:

$$\mathcal{L}^{\text{FP}}_{\text{YOLO}}(x \odot \delta) := \sum_{1 \le b \le B} \max\left(\tau + \kappa - \mathsf{conf}(f(x \odot \delta), b), 0\right) \ . \tag{3}$$

4.2.3 Evaluation of Attacks. We now instantiate and evaluate the attack strategies C1-C4 from Table 2 on our seven ad-classifiers

Attack C1: Evasion with adversarial ad-disclosures. Figure 5 shows examples of perturbed AdChoices logos that fool all element-based classifiers. An ad-network can use these to evade ad-blocking.

Average hashing is invariant to small ℓ_p noise, but this comes at the cost of high sensitivity to other perturbations: we evade it by adding up to 3 transparent rows and columns to the logo. When overlaid on an ad, the rendered content is identical.

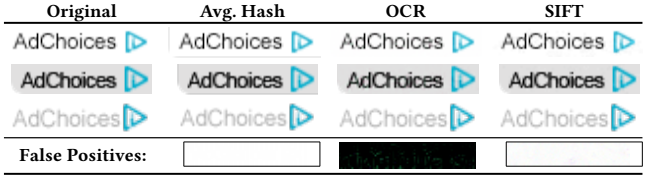


Figure 5: Adversarial Examples for Element-Based Classifiers. These correspond to attacks (C1) and (C4) in Table 2.

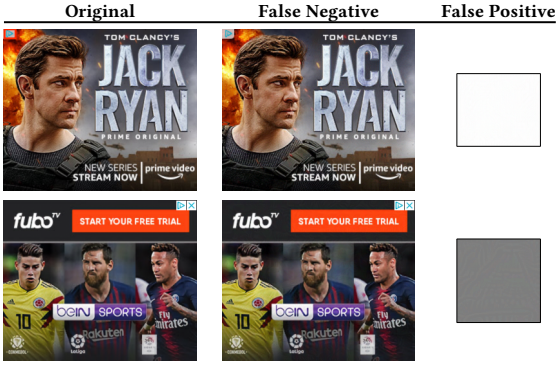
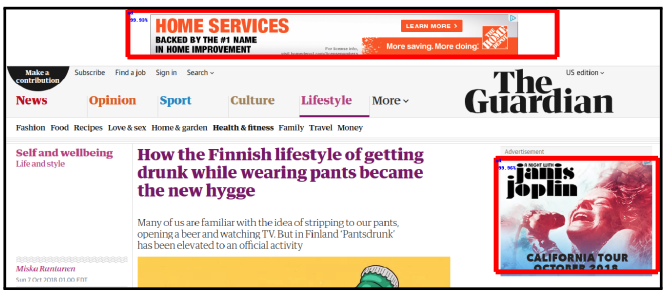


Figure 6: Adversarial Examples for Frame-based Classifiers. These are attacks (C2) and (C4) in Table 2. Top: Attacks on our YOLOv3 model that detects the AdChoices logo. Bottom: attacks on the ad-classifier from [40] (we crafted similar adversarial examples for the classifier used in Percival [84])

Adversarial examples for OCR bear similarities to CAPTCHAs. As ML models can solve CAPTCHAs [15, 94], one may wonder why transcribing ad disclosures is harder. The difference lies in the stronger threat model that ad-blockers face. Indeed, CAPTCHA creators have no access to the ML models they aim to fool, and must thus craft universally hard perturbations. Attacking an ad-blocker is much easier as its internal model must be public. Moreover the ad-blocker must also prevent false positives—which CAPTCHA solvers do not need to consider—and operate under stricter real-time constraints on consumer hardware.

Attack C2: Evasion with adversarial ads. Ad networks can directly perturb the ads they server to evade frame or page-based ad-blockers. For frame-based classifiers, the attacks are very simple and succeed with 100% probability (see Figure 6). We verified that the ad-classifier used by Percival [84] is vulnerable to similar attacks. Specifically, we create a valid HTML page containing two images—an ad and an opaque white box—which are both misclassified when the page is rendered in Percival's modified Chromium browser (see Figure 13).

For our page-based model, crafting a "doubly-universal" perturbation that works for all ads on all websites is hard (this is due to the model's reliance on page layout for detecting ads, see Appendix B for details). Instead, we show that an ad-network can create a universal perturbation that works with 100% success rate for all ads that it serves on a specific domain (see Figure 14). For this attack, we minimized the $\mathcal{L}^{\text{FN}}_{\text{YOLO}}$ loss over the collected screenshots in $D^{\text{train}}_{\text{BBC}}$, by applying the same perturbation δ over all ad frames.



(a) Original Page: two ads are detected.

(b) Attack C3 (Universal): The publisher overlays a transparent mask over the full page to evade the ad-blocker.

(c) Attack C3 (Universal): The publisher overlays a mask on the page to generate unreasonably large boxes and disable the ad-blocker.

(d) Attack C4 (Universal): The publisher adds an opaque footer to detect an ad-blockers that blocks the honeypot element (bottom-left).

Figure 8: Universal Adversarial Examples for Page-Based Ad-Blockers. Displays examples of universal evasion attacks (C3) and detection attacks (C4) on a page from theguardian.com. Best viewed with 2x zoom in.

```
<div id="overlay"></div>
```

```
#overlay {
background-image: url("...");
width: 100%; height: 100%; top: 0; left: 0;
position: fixed; z-index: 10000; pointer-events: none;
opacity: 0.01; }
```

Figure 7: Code for Attack C4-U. An adversarial mask is tiled over the full page with a small opacity factor.

Attack C3: Evasion with adversarial content. These attacks apply to page-based ad-blockers and allow publishers to evade ad-blocking while only perturbing HTML elements that they control (which crucially does not include the content of ad-frames). We show that a publisher can actually perturb the full screenshot image fed into the classifier using CSS techniques. The HTML perturbation is a *near-transparent mask*, that is overlaid on the entire web page (see Figure 7). The CSS properties `z-index` and `pointer-events` are used to display the mask over all other web content, but allow underlying elements to still be accessed and clicked normally.

Adding a mask over the full image is prohibitive, as the mask would be large and tied to a fixed resolution. We thus build a smaller mask and *tile* it over the full page. We generate a universal adversarial mask δ of 20KB by minimizing $\mathcal{L}^{\text{FN}}_{\text{YOLO}}$ over D^{train}. The overlaid mask evades detection of all ads in our evaluation set (see Figure 8, (b)). This attack can be deployed by *any* publisher, to evade all ads. The perturbation mask is robust to scrolling and viewport changes when tested in Google Chrome.

Figure 8 (c) shows a similar attack that *overloads* the ad-blocker. The transparent mask is crafted to minimize $\mathcal{L}^{\text{FN}}_{\text{YOLO}}$ over D^{train}, and creates many incorrect predictions that coerce the ad-blocker into abdicating or breaking the site. On all websites, the mask causes the model to detect abnormally large ads or fail to detect real ads.

These attacks are powerful and can be re-used by any publisher. Yet, ad-blockers might try to detect certain CSS tricks and disable them. We thus also propose stealthier attacks tuned to a single domain. For pages on BBC.com, we create a small perturbation (40×1020 px) that is applied to the white background right below an ad frame (see Figure 14(b)) and that universally applies to all pages from that publisher that use a similar layout.

Attack C4: Detection with adversarial honeypots. To detect ad-blocking, publishers can use honeypots that falsely trigger ad-blockers [95]. The false positives in Figures 5-6 are innocuous elements that are falsely classified as ads or ad-disclosures. For OCR and the model of Hussain et al. [40], generating near-opaque black elements worked best. As average hashing is invariant to changes in image intensity, creating false positives for it is trivial.

For page-based ad-blockers, our first attack embeds a perturbation into a small page footer (see Figure 8 (d)). The footer causes false predictions for 19/20 pages in our evaluation set, and is robust to a user scrolling over the page. Figure 14 (c) shows a stealthier attack—tailored to bbc.com—that hides a honeypot in the page header and has 100% success rate across pages from that publisher.

4.3 Attacks against Ad-Blocker Actions

Ad-blockers usually run at a higher privilege level than any web page. They are generally not affected by the same-origin policy and can read and write any part of any web page that the user visits.

The main privileged action taken by an ad-blocker is altering of web content. Attackers exploit this action when using honeypots

to detect ad-blockers. But triggering ad-blocker actions can have more pernicious effects. Below, we describe two attacks that can be deployed by *arbitrary content creators* (e.g., a Facebook user) to trigger malicious ad-blocker actions in other users' browsers.

Attack A1: Cross-boundary blocking. In this attack (see Figure 1) a malicious user (Jerry) uploads adversarial content that triggers a Sentinel-like ad-blocker into marking content of another user (Tom) as and ad. This "cross-boundary blocking attack" hijacks the ad-blocker's elevated privilege to bypass web security boundaries.

To mount the attack, we optimally perturb Jerry's content so as to maximize the model's confidence in a box that covers Tom's content. The attack works because object-detector models such as YOLOv3 [72] predict bounding boxes by taking into account the *full* input image—a *design feature* which increases accuracy and speed [70]. As a result, adversarial content can affect bounding boxes in arbitrary image regions. Our attack reveals an inherent vulnerability of *any* object detector applied to web content—wherein the model's segmentation misaligns with web-security boundaries.

Attack A2: Cross-origin web requests. In addition to searching for the "Sponsored" text on Facebook, Ad-Highlighter [82] uses the fact that the ad-disclosure contains a link to Facebook's ad-policy page as an additional signal. Specifically, Ad-Highlighter parses the DOM in search for links containing the text "Sponsored" and determines whether the link leads to Facebook's ad statement page by simulating a user-click on the link and following any redirects. [5]

These techniques are dangerous and enable serious vulnerabilities (e.g., CSRF [66], DDoS [67] or click-fraud [22]) with consequences extending beyond ad-blocking. Clicking links on a user's behalf is a highly privileged action, which can thus be exploited by *any party that can add links in a page*, which can include arbitrary website users. To illustrate the dangers of behavioral ad-blocking, we create a regular Facebook post with an URL to a web page with title "Sponsored". Facebook converts this URL into a link which Ad-Highlighter clicks on. *Albeit sound, this attack luckily and coincidentally fails due to Facebook's Link Shim*, that inspects clicked links before redirecting the user. Ad-Highlighter fails to follow this particular redirection thus inadvertently preventing the attack. Yet, this also means that Facebook could use the same layer of indirection for their "Sponsored" link. If the behavioral ad-blocking idea were to be extended to disclosure cues on other websites (e.g., the AdChoices logo), such attacks would also be easily mounted. Prefiltering inputs passed to a behavioral layer does not help. Either the filter is perfect, in which case no extra step is required—or its false positives can be exploited to trigger the behavioral component.

4.4 Attacks against Page Segmentation

In this Section, we describe attacks targeting the ad-blocker's page segmentation logic, in an effort to evade the ad-blocker or exhaust its resources. These attacks use standard Web techniques (e.g., HTML obfuscation) and are already applied in an ongoing arms race between Facebook and uBlock [88]. We argue that to escape the arms race caused by these segmentation attacks, perceptual ad-blockers have to operate over *rendered* web-content (i.e., frame or page-based approaches), which in turn increases the attack surface for adversarial examples on the ad-blocker's visual classifier.

[5] Ad-Highlighter simulates clicks because Facebook used to resolve links server-side (the ad-disclosure used to link to www.facebook.com/#). Facebook recently changed its obfuscation of the link in post captions. It now uses an empty <a> tag that is populated using JavaScript during the click event. This change fools Ad-Highlighter and still requires an ad-blocker to simulate a potentially dangerous click to uncover the link.

```
<a><span>
<span class="c1">Sp</span>
<span class="c2">S</span>
<span class="c1">on</span>
<span class="c2">S</span>
<span class="c1">so</span>
<span class="c2">S</span>
<span class="c1">red</span>
<span class="c2">S</span>
</span></a>
```

```
.c2 { font-size: 0; }
```

Figure 9: CSS Obfuscation on Facebook. (Left) HTML and CSS that render Facebook's "Sponsored" caption. (Right) A proof-of-concept where the ad-disclosure is an adversarial image that Ad-Highlighter's OCR decodes as "8parisared".

Figure 10: Image Sprites of the AdChoices Logo. *Image-sprites* are sets of images stored in a single file, and segmented using CSS rules. For example, the left sprite allows to smoothly switch from the icon to the full logo on hover. The right sprite is used by cnn.com to load a variety of logos used on the page in a single request.

Attack S1: DOM obfuscation. These attacks aim to fool the ad-blocker into feeding ambiguous inputs to its classifier. They exploit some of the same limitations that affect traditional filter lists, and can also be applied to element-based ad-blockers that rely on computer-vision classifiers, such as Ad-Highlighter.

DOM obfuscation is exemplified by Facebook's continuous efforts to regularly alter the HTML code of its "Sponsored" caption (see Figure 9). Facebook deploys a variety of CSS tricks to obfuscate the caption, and simultaneously embeds hidden ad-disclosure honeypots within regular user posts in an effort to deliberately cause site-breakage for ad-block users. Facebook's obfuscation attempts routinely fool uBlock [88] as well as Ad-Highlighter.

If ad-blockers adopt computer-vision techniques as in Ad-Highlighter, DOM obfuscation attacks still apply if ad-blockers assume a direct correspondence between elements in the DOM and their visual representation when rendered. For example, Ad-Highlighter assumes that all `img` tags in the DOM are shown as is, thereby ignoring potentially complex CSS transformations applied when rendering HTML. This can cause the downstream classifier to process images with unexpected properties.

Ad networks already use CSS rules that significantly alter rendered ad-disclosures. Figure 10 shows two AdChoices logos found on cnn.com. These are image-sprites—multiple images included in a single file to minimize HTTP requests—that are cropped using CSS to display only a single logo at a time. Image-sprites highlight an exploitable blind-spot in element-based perceptual ad-blockers—e.g., the logos in Figure 10 fool Ad-Highlighter [82]. Images can also be fragmented into multiple elements. The ad-blocker then

has to stitch them together to correctly recognize the image (e.g., Google's AdChoices logo consists of two separate SVG tags).

Finally, the rules used by ad-blockers to link ad-disclosures back to the corresponding ad frame can also be targeted. For example, on pages with an integrated ad network, such as Facebook, the publisher could place ad-disclosures (i.e., "Sponsored" links) and ads at arbitrary places in the DOM and re-position them using CSS.

Frame-based and page-based ad-blockers bypass all these issues by operating on already-rendered content.

Attack S2: Over-segmentation. Here the publisher injects a large number of elements into the DOM (say, by generating dummy images in JavaScript) to overwhelm an ad-blocker's classifier with inputs and exhaust its resources. In response, ad-blockers would have to aggressively filter DOM elements—with the risk of these filters' blind spots being exploited to evade or detect ad-blocking. The viability of this attack may seem unclear, as users might blame publishers for high page-load latency resulting from an overloaded ad-blocker. Yet, Facebook's efforts to cause site-breakage by embedding ad-disclosure honeypots within all regular user posts demonstrates that some ad networks may result to such tactics.

4.5 Attacks against Training

For classifiers that are trained on labeled images, the data collection and training phase can be vulnerable to *data poisoning attacks* (D1)—especially when crowdsourced as with Sentinel [10]. We describe these attacks for completeness, but refrain from a detailed evaluation as the test-time attacks described in Sections 4.2 through 4.4 are conceptually more interesting and more broadly applicable.

In these attacks, the adversary joins the crowdsourced data collection to submit maliciously crafted images that adversely influence the training process. For example, malicious training data can contain *visual backdoors* [20], which are later used to evade the ad-blocker. The ad-blocker developer cannot tell if a client is contributing real data for training or malicious samples. Similar attacks against crowdsourced filter lists such as Easylist are theoretically possible. A malicious user could propose changes to filter lists that degrade their utility. However, new filters are easily interpreted and vetted before inclusion—a property not shared by visual classifiers.

Sentinel's crowdsourced data collection of users' Facebook feeds also raises serious privacy concerns, as a deployed model might leak parts of its training data [30, 77].

5 DISCUSSION

We have presented multiple attacks to evade, detect and abuse recently proposed and deployed perceptual ad-blockers. We now provide an in-depth analysis of our results.

5.1 A New Arms Race

Our results indicate that perceptual ad-blocking will either perpetuate the arms race of filter lists, or replace it with an arms race around adversarial examples. Where perceptual ad-blockers that rely heavily on page markup (e.g., as in uBlock [7] or Ad-Highlighter [82]) remain vulnerable to continuous markup obfuscation [88], visual classification of rendered web content (as in Sentinel [10] or Percival [84]) inherits a crucial weakness of current visual classifiers—adversarial examples [33, 83].

The past years have seen considerable work towards mitigating the threat of adversarial examples. Yet, defenses are either broken by improved attacks [11, 17], or limited to restricted adversaries [21, 45, 53, 69, 86]. Even if ad-block developers proactively detect adversarial perturbations and blacklist them (e.g., using adversarial training [53, 83] to fine-tune their classifier), adversaries can simply regenerate new attacks (or use slightly different perturbations [76]).

5.2 Strategic Advantage of Adversaries and Lack of Defenses

Our attacks with adversarial examples are not a *quid pro quo* step in this new arms race, but indicate a pessimistic outcome for perceptual ad-blocking. Indeed, these ad-blockers operate in essentially *the worst threat model for visual classifiers*. Their adversaries have access to the ad-blockers' code and prepare offline digital adversarial examples to trigger both false-negatives and false-positives in the ad-blocker's online (and time constrained) decision making.

Even if ad-blockers obfuscate their code, black-box attacks [41] or model stealing [63, 87] still apply. Randomizing predictions or deploying multiple classifiers is also ineffective [11, 37]. For example, some of the adversarial examples in Figure 5 work for both OCR and SIFT despite being targeted at a single one of these classifiers.

The severity of the above threat model is apparent when considering existing defenses to adversarial examples. For instance, adversarial training [53, 83] assumes restricted adversaries (e.g., limited to ℓ_∞ perturbations), and breaks under other attacks [27, 76, 85]. Robustness to adversarial false positives (or "garbage examples" [33]) is even harder. Even if ad-blockers proactively re-train on adversarial examples deployed by publishers and ad-networks, training has a much higher cost than the attack generation and is unlikely to generalize well to new perturbations [74]. *Detecting* adversarial examples [34, 55] (also an unsolved problem [16]) is insufficient as Ad-blockers face both adversarial false-positives and false-negatives, so merely detecting an attack does not aid in decision-making. A few recently proposed defenses achieve promising results in some restricted threat models, e.g., black-box attacks [19] or physically-realizable attacks [21]. These defenses are currently inapplicable in the threat model of perceptual ad-blocking, but might ultimately reveal new insights for building more robust models.

Our attacks also apply if perceptual ad-blocking is used as a complement to filter lists rather than as a standalone approach. Ad-blockers that combine both types of techniques are vulnerable to attacks targeting either. If perceptual ad-blocking is only used passively (e.g., to aid in the maintenance of filter lists, by logging potential ads that filter lists miss), the ad-blocker's adversaries still have incentive to attack to delay the detection of new ads.

This stringent threat model above also applies to ML-based ad-blockers that use URL and DOM features [14, 36, 43], which have not been evaluated against *adaptive white-box* attacks.

5.3 Beyond the Web and Vision.

The use of sensory signals for ad-blocking has been considered outside the Web, e.g., AdblockRadio detects ads in radio streams using neural networks [2]. Emerging technologies such as virtual reality [62], voice assistants [44] and smart TVs [59] are posited to

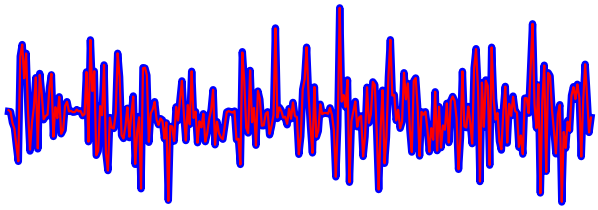


Figure 11: Original and Adversarial Audio Waveforms. Shows a ten second segment of an ad audio waveform (thick blue) overlaid with its adversarial perturbation (thin red).

become platforms for large-scale targeted advertising, and perceptual ad-blockers might emerge in those domains as well.

The threats described in this paper—and adversarial examples in particular—are likely to also affect perceptual ad-blockers that operate outside the vision domain. To illustrate, we take a closer look at AdblockRadio, a radio client that continuously classifies short audio segments as speech, music or ads based on spectral characteristics. When ads are detected, the radio lowers the volume or switches stations. Radio ad-blockers face a different threat model than on the Web. All content, including ads, is served as raw audio from a single origin, so filter lists are useless. The publisher cannot run any client-side code, so ad-block detection is also impossible. Yet, the threat of adversarial examples does apply. Indeed, we show that by adding near-inaudible[6] noise to the ad content in AdblockRadio's demo podcast, the perturbed audio stream evades ad detection.

Concretely, AdblockRadio takes as input a raw audio stream, computes the Mel-frequency cepstral coefficients (MFCCs), and splits them into non-overlapping windows of 4 seconds. Each segment is fed into a standard feed-forward classifier that predicts whether the segment corresponds to music, speech, or an ad. A post-processing phase merges all consecutive segments of a same class, and removes ad-segments. As the whole prediction pipeline is differentiable, crafting adversarial examples is straightforward: we use projected gradient descent (in the l_∞-norm) to modify the raw ad audio segments so as to minimize the classifier's confidence in the ad class. The resulting audio stream fully bypasses AdblockRadio's ad detection. An ad segment in the original and adversarial audio waveforms is displayed in Figure 11.

6 RELATED WORK

Our work bridges two areas of computer security research—studies of the online ad-ecosystem and associated ad-blocking arms race, and adversarial examples for ML models.

Behavioral advertising. A 2015 study found that 22% of web users use ad-blockers, mainly due to intrusive behavior [46, 68, 78, 89]. The use of ad-disclosures—which some perceptual ad-blockers rely on—is rising. On the Alexa top 500, the fraction of ads with an AdChoices logo has grown from 10% to 60% in five years [38, 81]. Yet, less than 27% of users understand the logo's meaning [50, 89].

Ad-blocking. Limitations of filter lists are well-studied [54, 91, 92]. Many new ad-blocker designs (e.g., [14, 36, 43]) replace hard-coded rules with ML models trained on similar features (e.g., markup [23] or URLs [47]). Many of these works limit their security analysis to *non-adaptive* attacks. Ours is the first to rigorously evaluate ML-based ad-blockers.

Ad-block detection has spawned an arms race around anti-ad-blocking scripts [57, 58, 60]. Iqbal et al. [42] and Zhu et al. [95] detect anti-ad-blocking using code analysis and differential-testing. Storey et al. [81] build *stealthy* ad-blockers that aim to hide from client-side scripts, a challenging task in current browsers (see Appendix A).

Adversarial examples. Our work is the first to apply adversarial examples in a real-world web-security context. Prior work attacked image classifiers [17, 33, 64, 83], malware [35], speech recognition [18] and others. We make use of white-box attacks on visual classifiers [17, 53], sequential models [18, 79] and object detectors [28]. We show that *black-box* attacks [41] are a generic alternative to prior attacks on SIFT [39].

Attacking page-based ad-blockers introduce novel challenges. Perturbing HTML bares similarities to discrete domain attacks, e.g., PDF malware detection [80]. The ad-blocker's inputs can also be controlled by multiple entities, a constraint reminiscent of those that arise in *physical-world* attacks [12, 28, 29, 49, 75].

Preventing adversarial examples is an open problem. Adversarial training is a viable strategy [33, 49, 53, 86], but considers a less stringent threat model than perceptual ad-blockers.

7 CONCLUSION

We have presented a comprehensive security evaluation of perceptual ad-blocking. To understand the design space of these recently deployed systems, we have derived a unified architecture that incorporates and extends prior work. Our analysis of this architecture has revealed multiple vulnerabilities at every stage of the visual ad-classification pipeline. We have shown that unless perceptual ad-blockers operate over rendered web content, the arms race around page markup obfuscation will likely carry on. Conversely, we have demonstrated that current visual ad-classifiers are inherently vulnerable to adversarial examples—the first application of these attacks to web-security. We have shown how to craft near-imperceptible perturbation for ads, ad-disclosures, and native content, in order to evade or detect ad-blocking with seven different classifiers. Finally, we have discovered a powerful attack on page-based ad-blockers, wherein a malicious user fools the model into blocking content supposedly protected by web-security boundaries.

Our aim was to highlight the fundamental vulnerabilities that perceptual ad-blockers inherit from existing image classifiers. As long as defenses to adversarial examples are elusive, perceptual ad-blockers will be dragged into a new arms race in which they start from a precariously disadvantaged position—given the stringent threat model that they must survive.

ACKNOWLEDGMENTS

This work was partially supported by NSF, ONR, the Simons Foundation, a Google faculty fellowship, the Swiss National Science Foundation (SNSF project P1SKP2_178149), and the German Federal Ministry of Education and Research (BMBF) through funding for the CISPA-Stanford Center for Cybersecurity (FKZ: 13N1S0762).

[6] The perturbed audio stream has a signal-to-noise ratio of 37 dB.

REFERENCES

[1] Adblock Plus. https://adblockplus.org/
[2] AdblockRadio. https://www.adblockradio.com.
[3] EasyList. https://easylist.to/.
[4] Easylist Forum: Report incorrectly removed content. https://forums.lanik.us/viewforum.php?f=64&sid=ba948dbdbad9334b72c143f26db58ff0.
[5] Ghostery. https://www.ghostery.com/.
[6] Tesseract. https://github.com/tesseract-ocr/.
[7] uBlock. https://www.ublock.org/.
[8] Adblock Plus. Issue 7088: Implement hide-if-contains-image snippet. https://issues.adblockplus.org/ticket/7088.
[9] Adblock Plus. 2018. Customize Facebook with Adblock Plus. https://facebook.adblockplus.me/.
[10] Adblock Plus. 2018. Sentinel. https://adblock.ai/.
[11] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning (ICML)*.
[12] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2018. Synthesizing robust adversarial examples. In *International Conference on Machine Learning (ICML)*.
[13] Shai Avidan and Ariel Shamir. 2007. Seam carving for content-aware image resizing. In *ACM Transactions on graphics*, Vol. 26. 10.
[14] Sruti Bhagavatula, Christopher Dunn, Chris Kanich, Minaxi Gupta, and Brian Ziebart. 2014. Leveraging machine learning to improve unwanted resource filtering. In *ACM Workshop on Artificial Intelligence and Security*. ACM.
[15] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C Mitchell. 2014. The End is Nigh: Generic Solving of Text-based CAPTCHAs.. In *USENIX Workshop on Offensive Technologies*. USENIX.
[16] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*. ACM, 3–14.
[17] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*. IEEE.
[18] Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *DLS*.
[19] Steven Chen, Nicholas Carlini, and David Wagner. 2019. Stateful Detection of Black-Box Adversarial Attacks. *arXiv preprint arXiv:1907.05587* (2019).
[20] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).
[21] Edward Chou, Florian Tramèr, Giancarlo Pellegrino, and Dan Boneh. 2018. Sentinet: Detecting physical attacks against deep learning systems. *arXiv preprint arXiv:1812.00292* (2018).
[22] Nicolas Christin, Sally S Yanagihara, and Keisuke Kamataki. 2010. Dissecting one click frauds. In *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 15–26.
[23] Justin Crites and Mathias Ricken. 2004. Automatic ad blocking: Improving AdBlock for the Mozilla platform. (2004).
[24] Digital Advertising Alliance (DAA). 2009. Self Regulatory Principles for Online Behavioral Advertising. https://digitaladvertisingalliance.org/sites/aboutads/files/DAA_files/seven-principles-07-01-09.pdf.
[25] Digital Advertising Alliance (DAA). 2013. DAA Icon Ad Marker Creative Guidelines. https://digitaladvertisingalliance.org/sites/aboutads/files/DAA_files/DAA_Icon_Ad_Creative_Guidelines.pdf.
[26] Benjamin Edelman. 2009. False and Deceptive Display Ads at Yahoo's Right Media. http://www.benedelman.org/rightmedia-deception.
[27] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. 2017. A rotation and a translation suffice: Fooling CNNs with simple transformations. *arXiv preprint arXiv:1712.02779* (2017).
[28] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Physical Adversarial Examples for Object Detectors. In *USENIX Workshop on Offensive Technologies*. USENIX.
[29] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1625–1634.
[30] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM SIGSAC Conference on Computer and Communications Security*. ACM.
[31] Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. 2018. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732* (2018).
[32] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. 2018. Adversarial spheres. *arXiv preprint arXiv:1801.02774* (2018).
[33] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*.
[34] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).
[35] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2017. Adversarial perturbations against deep neural networks for malware classification, In ESORICS. *arXiv preprint arXiv:1606.04435*.
[36] David Gugelmann, Markus Happe, Bernhard Ager, and Vincent Lenders. 2015. An automated approach for complementing ad blockers' blacklists. *Privacy Enhancing Technologies Symposium* 2 (2015), 282–298.
[37] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. 2017. Adversarial Example Defenses: Ensembles of Weak Defenses are not Strong. *arXiv preprint arXiv:1706.04701* (2017).
[38] Jovanni Hernandez, Akshay Jagadeesh, and Jonathan Mayer. 2011. Tracking the trackers: The AdChoices icon. http://cyberlaw.stanford.edu/blog/2011/08/tracking-trackers-adchoices-icon.
[39] Chao-Yung Hsu, Chun-Shien Lu, and Soo-Chang Pei. 2009. ACM International conference on Multimedia. In *ICM*. ACM, 637–640.
[40] Zaeem Hussain, Mingda Zhang, Xiaozhong Zhang, Keren Ye, Christopher Thomas, Zuha Agha, Nathan Ong, and Adriana Kovashka. 2017. Automatic understanding of image and video advertisements. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1100–1110.
[41] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box Adversarial Attacks with Limited Queries and Information. In *International Conference on Machine Learning (ICML)*.
[42] Umar Iqbal, Zubair Shafiq, and Zhiyun Qian. 2017. The ad wars: retrospective measurement and analysis of anti-adblock filter lists. In *Internet Measurement Conference*. ACM, 171–183.
[43] Umar Iqbal, Zubair Shafiq, Peter Snyder, Shitong Zhu, Zhiyun Qian, and Benjamin Livshits. 2018. AdGraph: A Machine Learning Approach to Automatic and Effective Adblocking. *arXiv preprint arXiv:1805.09155* (2018).
[44] Ilker Koksal. 2018. How Alexa Is Changing The Future Of Advertising. https://www.forbes.com/sites/ilkerkoksal/2018/12/11/how-alexa-is-changing-the-future-of-advertising.
[45] Zico Kolter and Eric Wong. 2017. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*.
[46] Georgios Kontaxis and Monica Chew. 2015. Tracking protection in Firefox for privacy and performance. *arXiv preprint arXiv:1506.04104* (2015).
[47] Viktor Krammer. 2008. An effective defense against intrusive web advertising. In *Conference on Privacy, Security and Trust*. IEEE, 3–14.
[48] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *International Conference on Learning Representations (ICLR)*.
[49] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial Machine Learning at Scale. In *International Conference on Learning Representations (ICLR)*.
[50] Pedro Giovanni Leon, Justin Cranshaw, Lorrie Faith Cranor, Jim Graves, Manoj Hastak, Blase Ur, and Guzi Xu. 2012. What do online behavioral advertising privacy disclosures communicate to users?. In *Workshop on Privacy in the electronic society*. ACM, 19–30.
[51] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. 2012. Knowing your enemy: understanding and detecting malicious web advertising. In *ACM SIGSAC Conference on Computer and Communications Security*. ACM.
[52] David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 2 (2004), 91–110.
[53] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*.
[54] Matthew Malloy, Mark McNamara, Aaron Cahn, and Paul Barford. 2016. Ad blockers: Global prevalence and impact. In *Internet Measurement Conference*. ACM, 119–125.
[55] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. In *International Conference on Learning Representations (ICLR)*.
[56] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal Adversarial Perturbations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1765–1773.
[57] Muhammad Haris Mughees, Zhiyun Qian, and Zubair Shafiq. 2017. Detecting anti ad-blockers in the wild. In *Privacy Enhancing Technologies Symposium*, Vol. 2017.
[58] Muhammad Haris Mughees, Zhiyun Qian, Zubair Shafiq, Karishma Dash, and Pan Hui. 2016. A first look at ad-block detection: A new arms race on the web. *arXiv preprint arXiv:1605.05841* (2016).
[59] Meghan Neal. 2016. You're Going to Need an Ad Blocker for Your Next TV. https://motherboard.vice.com/en_us/article/mg7ek8/youre-going-to-need-an-ad-blocker-for-your-next-tv.
[60] Rishab Nithyanand, Sheharbano Khattak, Mobin Javed, Narseo Vallina-Rodriguez, Marjan Falahrastegar, Julia E Powles, ED Cristofaro, Hamed Haddadi, and Steven J

Murdoch. 2016. Adblocking and counter blocking: A slice of the arms race. In *USENIX Workshop on Free and Open Communications on the Internet*.
[61] Paraska Oleksandr. 2018. Towards more intelligent ad blocking on the web. https://medium.com/@shoniko/towards-more-intelligent-ad-blocking-on-the-web-9f67bf2a12b5.
[62] George Paliy. 2018. The Future Of Advertising In Virtual Reality. https://stopad.io/blog/future-virtual-reality-advertising.
[63] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine Learning. In *ACM ASIA Conference on Computer and Communications Security*. ACM, 506–519.
[64] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy*.
[65] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. 2016. Towards the Science of Security and Privacy in Machine Learning. *arXiv preprint arXiv:1611.03814* (2016).
[66] Giancarlo Pellegrino, Martin Johns, Simon Koch, Michael Backes, and Christian Rossow. 2017. Deemon: Detecting CSRF with Dynamic Analysis and Property Graphs. In *ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1757–1771.
[67] Giancarlo Pellegrino, Christian Rossow, Fabrice J Ryba, Thomas C Schmidt, and Matthias Wählisch. 2015. Cashing Out the Great Cannon? On Browser-Based DDoS Attacks and Economics.. In *USENIX Workshop on Offensive Technologies*.
[68] Enric Pujol, Oliver Hohlfeld, and Anja Feldmann. 2015. Annoyed Users: Ads and Ad-Block Usage in the Wild. In *Internet Measurement Conference*. ACM, 93–106.
[69] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. 2018. Certified defenses against adversarial examples. In *International Conference on Learning Representations (ICLR)*.
[70] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 779–788.
[71] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE (Ed.).
[72] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767* (2018).
[73] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* (2017).
[74] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. 2018. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*. 5014–5026.
[75] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *ACM SIGSAC Conference on Computer and Communications Security*. ACM.
[76] Yash Sharma and Pin-Yu Chen. 2017. Attacking the Madry Defense Model with L_1-based Adversarial Examples. *arXiv preprint arXiv:1710.10733* (2017).
[77] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy*. IEEE, 3–18.
[78] Ashish Kumar Singh and Vidyasagar Potdar. 2009. Blocking online advertising-A state of the art. In *IEEE International Conference on Industrial Technology*. IEEE.
[79] Congzheng Song and Vitaly Shmatikov. 2018. Fooling OCR Systems with Adversarial Text Images. *arXiv preprint arXiv:1802.05385* (2018).
[80] Nedim Srndic and Pavel Laskov. 2014. Practical evasion of a learning-based classifier: A case study. In *IEEE Symposium on Security and Privacy*.
[81] Grant Storey, Dillon Reisman, Jonathan Mayer, and Arvind Narayanan. 2017. The Future of Ad Blocking: An Analytical Framework and New Techniques. *arXiv preprint arXiv:1705.08568* (2017).
[82] Grant Storey, Dillon Reisman, Jonathan Mayer, and Arvind Narayanan. 2017. Perceptual Ad Highlighter. Chrome Extension: https://chrome.google.com/webstore/detail/perceptual-ad-highlighter/mahgiflleahghaapkboihnbhdplhnchp; Source code: https://github.com/citp/ad-blocking.
[83] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*.
[84] Panagiotis Tigas, Samuel T King, Benjamin Livshits, et al. 2019. Percival: Making In-Browser Perceptual Ad Blocking Practical With Deep Learning. *arXiv preprint arXiv:1905.07444* (2019).
[85] Florian Tramèr and Dan Boneh. 2019. Adversarial Training and Robustness for Multiple Perturbations. *arXiv preprint arXiv:1904.13000* (2019).
[86] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations (ICLR)*.
[87] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *USENIX Security Symposium*.
[88] uBlockOrigin. Issue 3367: Facebook. https://github.com/uBlockOrigin/uAssets/issues/3367.
[89] Blase Ur, Pedro Giovanni Leon, Lorrie Faith Cranor, Richard Shay, and Yang Wang. 2012. Smart, useful, scary, creepy: perceptions of online behavioral advertising. In *Symposium on Usable Privacy and Security*. ACM, 4.
[90] Ennèl van Eeden and Wilson Chow. 2018. Perspectives from the Global Entertainment & Media Outlook 2018–2022. https://www.statista.com/topics/1176/online-advertising/.
[91] Antoine Vastel, Peter Snyder, and Benjamin Livshits. 2018. Who Filters the Filters: Understanding the Growth, Usefulness and Efficiency of Crowdsourced Ad Blocking. *arXiv preprint arXiv:1810.09160* (2018).
[92] Craig E Wills and Doruk C Uzunoglu. 2016. What ad blockers are (and are not) doing. In *IEEE Workshop on Hot Topics in Web Systems and Technologies*. IEEE.
[93] Xinyu Xing, Wei Meng, Byoungyoung Lee, Udi Weinsberg, Anmol Sheth, Roberto Perdisci, and Wenke Lee. 2015. Understanding Malvertising Through Ad-Injecting Browser Extensions. In *International Conference on World Wide Web*.
[94] Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. 2018. Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach. In *ACM SIGSAC Conference on Computer and Communications Security*. ACM.
[95] Shitong Zhu, Xunchao Hu, Zhiyun Qian, Zubair Shafiq, and Heng Yin. 2018. Measuring and Disrupting Anti-Adblockers Using Differential Execution Analysis. In *Network and Distributed System Security Symposium (NDSS)*.

A THE AD-BLOCK DETECTION ARMS RACE

Many publishers actively detect the presence of ad-blockers [42, 58, 60] and take actions ranging from user warnings to service disabling for ad-block users. Ad-block detection operates among three main axes [81]: (1) detecting absence of known ads; (2) injecting "honeypots" and detecting that they are mistakenly blocked, and (3) detecting ad-blocking code through side-channels (e.g., timing).

Perceptual ad-blockers cannot be detected server-side as they do not alter any web requests. To remain stealthy, a perceptual ad-blocker thus only needs to fool publisher JavaScript code into observing an unmodified DOM [81]. This challenge is surmountable for native in-browser ad-blockers, as these can simply modify the user's view without affecting the DOM. Yet, the main ad-blockers today are browser extensions, which do not have such high privilege levels and share the same JavaScript API as client scripts. Storey et al. [81] suggest the following arms race for a stealthy ad-blocker:

(1) The ad-blocker modifies the DOM to block or mask detected ads and honeypots. It then *overwrites the JavaScript DOM traversal API* (e.g., with JavaScript *proxies*) so that the publisher's code sees the original DOM.

(2) The publisher inspects changes to global APIs by using the `toString()` method to unveil changes on the function.[7]

(3) The ad-blocker overwrites the universal `toString()` method used by all JavaScript functions, so that it always returns the same value as for a non-blocked website.

We argue that this is not the end of the arms race. We sketch three strategies to detect or reverse the above ad-blocker modifications. Preventing the attacks below requires the ad-blocker to emulate a much larger set of JavaScript APIs, parts-of-which appear inaccessible to browser extensions.

(1) **Borrowing native functions.** A publisher creates an `iframe`, which gets a new JavaScript environment, and extracts a "fresh" native function (e.g., `toString`) from it to unveil changes. In

[7] Even proxied functions can be distinguished from their native counterparts: https://bugs.chromium.org/p/v8/issues/detail?id=7484.

turn, the ad-blocker has to intercept all `iframe` creations and re-apply the same changes.

(2) **Detecting non-native functions.** The `toString` method is native (i.e., implemented by the browser). Some properties differ between native and non-native functions and do not appear to be mockable (e.g., setting a native function's `arguments` property raises an error whereas this property can be set for JavaScript functions).[8]

(3) **Timing.** If the above attacks are solved by emulating large portions of native JavaScript, the performance overhead may lead to a strong timing side-channel.

B TRAINING A PAGE-BASED AD-BLOCKER

As the trained neural network of Sentinel [10] is not available for an evaluation, we trained one for the analysis of Section 4. We used the same architecture as Sentinel, i.e., YOLO (v3) [70–72].

B.1 Data Collection

YOLO is an object detection network. Given an image, it returns a set of bounding boxes for each detected object. To train and evaluate YOLO, we created a dataset of labeled web page screenshots where each label encodes coordinates and dimensions of an ad on the page. We created the dataset with an ad-hoc automated system that operates in two steps. First, given a URL, it retrieves the web page and identifies the position of ads in the page using filter lists of traditional ad-blockers. Then, our system generates a web page template where ads are replaced with placeholder boxes. The concept of web page templates is convenient as it enables us to create multiple screenshots from the same web page with different ads, a form of data-augmentation. Second, from each web page template, we derive a number of images by placing ads on the template.

Web pages. We acquired web pages by retrieving the URLs of the top 30 news websites of each of the G20 nations listed in allyoucanread.com. For each news site, we searched for the RSS feed URLs and discarded sites with no RSS feeds. The total number of RSS feed URLs is 143. We visited each RSS feed URL daily and fetched the URLs to the daily news.

Template generation. Given a URL of a news article, we generate a page template using a modified HTTP proxy that matches incoming HTTP requests against traditional ad-blocker filter lists, i.e., Easylist [3] and Ghostery [5]. The proxy replaces ad contents with monochrome boxes using a unique color for each ad. These boxes are placeholders that we use to insert new ads. We manually inspected all templates generated during this step to remove pages with a broken layout (caused by filter lists' false positives) or pages whose ads are still visible (caused by filter lists' false negatives).

Image generation. From each page template, we generate multiple images by replacing placeholder boxes with ads. We select ads from the dataset of Hussain et al. [40]. This dataset contains about 64K images of ads of variable sizes and ratios. We complemented the dataset with 136 ads we retrieved online. To insert pictures inside a template, we follow four strategies:

(1) We directly replace the placeholder with an ad;

(2) We replace the placeholder with an ad, and we also include an AdChoices logo in the top right corner of the ad;

(3) We augment templates without placeholders by adding a large ad popup in the page. The page is darkened to highlight the ad;

(4) We insert ads as background of the website, that fully cover the left- and right-hand margins of the page.

When inserting an ad, we select an image with a similar aspect ratio. When we cannot find an exact match, we resize the image using Seam Carving [13], a content-aware image resizing algorithm that minimizes image distortion. To avoid overfitting during training, we limited the number of times each ad image can be used to 20.

B.2 Evaluation and Results

Datasets. The training set contains 2,901 images, of which 2,600 have ads. 1,600 images with ads were obtained with placeholder replacement, 800 with placeholder replacements with AdChoices logos, 100 with background ads, and 100 with interstitials.

The evaluation set contains a total of 2,684 images—2,585 with ads and 99 without ads. These are 1,595 images with placeholder replacement, 790 images with placeholder replacement with AdChoices logos, 100 images with background ads, and 100 images with interstitials. We also compiled a second evaluation set from 10 domains that were not used for training (this set is different from the one used to evaluate attacks in Section 4). For each domain, we took a screenshot of the front page and four screenshots of different subpages, resulting in 50 screenshots overall with a total of 75 advertisements. We trained using the default configuration of YOLOv3 [72], adapted for a unary classification task.

Accuracy and performance. We tested our model against both evaluation sets. The model achieved the best results after 3,600 training iterations. In the first set, our model achieved a mean average precision of 90.88%, an average intersect of union of 84.23% and an F1-score of 0.96. On the second set, our model achieved a mean average precision of 87.28%, an average intersect of union of 77.37% and an F1-score of 0.85. A video demonstrating our model detecting ads on five never seen websites is available at https://github.com/ftramer/ad-versarial/blob/master/videos.

We evaluate performance of the model in TensorFlow 1.8.0 with Intel AVX support. On an Intel Core i7-6700 CPU the prediction for a single image took 650ms.

Inspecting our model. We conduct a preliminary study of the inner-workings of our neural network. By inspecting the model's *activation map* on different inputs (see Figure 12), we find that the model mainly focuses on the layout of ads in a page, rather than their visual content. This shows that our ad-blocker detects ads using very different visual signals than humans. This raises an intriguing question about the Sentinel model of Adblock Plus [10], which was trained solely on Facebook data, where ads are visually

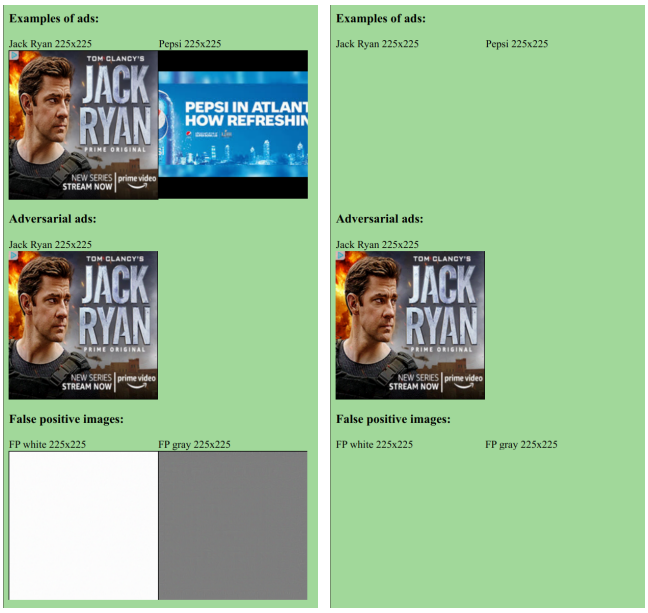
[8] It might be possible to circumvent this issue with a Proxy. Yet, we found that function Proxies can be distinguished from native functions in Google Chrome via the error message of a `postMessage` call—this might be mockable too, but vastly expands the portion of the JavaScript API to cover.

Figure 12: Activation Maps of our Ad Detection Model. The most salient features appear to be the surroundings of ads rather than their visual content.

close to the website's native content. Thus, it seems less likely that Sentinel would have learned to detect ads using layout information.

To generate the map in Figure 12, we compute the absolute value of the gradient of the network's output with respect to every input pixel, and apply a smoothing Gaussian kernel over the resulting image. The gradient map is then overlaid on the original input.

C EXTRA TABLES AND FIGURES



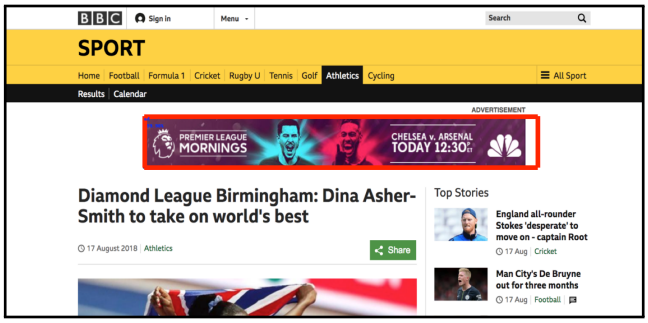
(a) Page displayed in Chromium. **(b) Page displayed in Percival.**

Figure 13: Attack on the Percival Browser from [84]. On the left, a dummy web page is displayed in the standard Chromium browser with two ads (top), an adversarially perturbed ad (middle) and two adversarial opaque boxes (bottom). On the right, the same page is displayed in the Percival browser. The two unperturbed ads on top are correctly blocked, but the adversarial ad evades detection, and the adversarial opaque boxes are mistakenly blocked.

Table 3: Evaluation Data for Adversarial Examples. We collect images, frames and screenshots from the Alexa top ten news websites that use the AdChoices standard (we exclude news.google.com and shutterstock.com which contain no ads on their front-page). For each page, we extract all images below 50 KB, all iframes, and take two screenshots (the front page and an article) of the user's viewport, and report the number of visible ads in these.

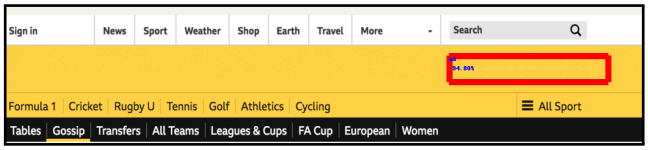
	Images		Iframes			Visible
Website	Total	AdChoices	Total	Ads	AdChoices	Ads
reddit.com	70	2	2	2	2	2
cnn.com	36	7	7	5	2	3
nytimes.com	89	4	3	3	3	2
theguardian.com	75	4	8	3	3	3
indiatimes.com	125	4	5	5	4	3
weather.com	144	5	11	7	3	3
news.yahoo.com	100	5	3	3	2	3
washingtonpost.com	40	1	5	2	1	3
foxnews.com	96	5	6	5	4	4
huffingtonpost.com	90	4	9	4	5†	4
Total	865	41	59	39	29	30

† One AdChoices logo appears in two rendered `iframes` laid on top of each other.



(a) Original Page: The ad banner is correctly detected.

(b) Attack C3-C4: The publisher perturbs the white background beneath the ad to evade ad-blocking (C4). Alternatively, an ad network adds a universal mask on the ad (C3, not displayed here for brevity). In both cases, the perturbation is invisible to the user.



(c) Attack C1: The publisher adds a honeypot element to the page header (top-right) to detect an ad-blocker.

Figure 14: Universal Adversarial Examples for Page-Based Ad-Blockers on BBC.com. Examples of evasion attacks C3-C4 and detection attack C1 (see Section 4.2).