# Poisoning Web-Scale Training Datasets is Practical

*Nicholas Carlini*[1] *Matthew Jagielski*[1] *Christopher A. Choquette-Choo*[1] *Daniel Paleka*[2]
*Will Pearce*[3] *Hyrum Anderson*[4] *Andreas Terzis*[1] *Kurt Thomas*[1] *Florian Tramèr*[2]
[1]Google [2]ETH Zurich [3]NVIDIA [4]Robust Intelligence

## Abstract

Deep learning models are often trained on distributed, web-scale datasets crawled from the internet. In this paper, we introduce two new dataset *poisoning attacks* that intentionally introduce malicious examples to a model's performance. Our attacks are immediately practical and could, today, poison 10 popular datasets. Our first attack, *split-view poisoning*, exploits the mutable nature of internet content to ensure a dataset annotator's initial view of the dataset differs from the view downloaded by subsequent clients. By exploiting specific invalid trust assumptions, we show how we could have poisoned 0.01% of the LAION-400M or COYO-700M datasets for just $60 USD. Our second attack, *frontrunning poisoning*, targets web-scale datasets that periodically snapshot crowd-sourced content—such as Wikipedia—where an attacker only needs a time-limited window to inject malicious examples. In light of both attacks, we notify the maintainers of each affected dataset and recommended several low-overhead defenses.

## 1 Introduction

Datasets used to train deep learning models have grown from thousands of carefully-curated examples [20, 33, 41] to *web-scale datasets* with billions of samples automatically crawled from the internet [10, 48, 53, 57]. At this scale, it is infeasible to manually curate and ensure the quality of each example. This quantity-over-quality tradeoff has so far been deemed acceptable, both because modern neural networks are extremely resilient to large amounts of label noise [55, 83], and because training on noisy data can even improve model utility on out-of-distribution data [50, 51].

While large deep learning models are resilient to random noise, even minuscule amounts of *adversarial* noise in training sets (i.e., a *poisoning attack* [6]) suffices to introduce targeted mistakes in model behavior [14, 15, 60, 76]. These prior works argued that poisoning attacks on modern deep learning models are practical due to the lack of human curation. Yet, despite the potential threat, to our knowledge no real-world attacks involving poisoning of web-scale datasets have occurred. One explanation is that prior research ignores the question of *how* an adversary would ensure that their corrupted data would be incorporated into a web-scale dataset.

In this paper, we introduce two novel poisoning attacks that *guarantee* malicious examples will appear in web-scale datasets used for training the largest machine learning models in production today. Our attacks exploit critical weaknesses in the current trust assumptions of web-scale datasets: due to a combination of monetary, privacy, and legal restrictions, many existing datasets are not published as static, standalone artifacts. Instead, datasets either consist of an *index* of web content that individual clients must crawl; or a periodic *snapshot* of web content that clients download. This allows an attacker to know with certainty *what* web content to poison (and, as we will show, even *when* to poison this content).

Our two attacks work as follows:

- **Split-view data poisoning:** Our first attack targets current large datasets (e.g., LAION-400M) and exploits the fact that the data seen by the dataset curator at collection time might differ (significantly and arbitrarily) from the data seen by the end-user at training time. This attack is feasible due to a lack of (cryptographic) integrity protections: there is no guarantee that clients observe the same data when they crawl a page as when the dataset maintainer added it to the index.
- **Frontrunning data poisoning:** Our second attack exploits popular datasets that consists of periodical snapshots of user-generated content—e.g., Wikipedia snapshots. Here, if an attacker can precisely time malicious modifications just prior to a snapshot for inclusion in a web-scale dataset, they can *front-run* the collection procedure. This attack is feasible due to predictable snapshot schedules, latency in content moderation, and snapshot immutability: even if a content moderator detects and reverts malicious modifications after-the-fact, the attacker's malicious content will persist in the snapshot used for training deep learning models.

We explore the feasibility of both of these attacks in practice on 10 popular web-scale datasets. We show these attacks are practical and realistic even for a low-resourced attacker: for just $60 USD, we could have poisoned 0.01% of the LAION-400M or COYO-700M datasets in 2022.

To counteract these attacks, we propose two defenses:

- **Integrity verification** prevents split-view poisoning by distributing cryptographic hashes for all indexed content, thus ensuring that clients observe the same data as when maintainers first indexed and annotated it.
- **Timing-based defenses** prevent frontrunning poisoning by either randomizing the order in which data is snapshotted and introduced into web-scale datasets; or delaying content prior to its inclusion into a snapshot and applying reversions from trusted moderators.

We discuss limitations of these defenses (e.g., in the case of integrity checks, preventing *benign* modifications such as re-encoding, re-sizing, or cropping images) and more robust, future-looking solutions with fewer trust assumptions.

**Responsible disclosure.** We disclosed our results to the maintainers of each of the 10 datasets appearing in this study. Six of these datasets now follow our recommended implementation for integrity checks. Additionally, we provided patches to the most popular web-scale dataset downloader to support integrity checks. Finally, we have notified Wikipedia about the frontrunning vulnerability in their data collection process.

## 2 Background & Related Work

Our work builds on existing knowledge of the risk of poisoning web-scale datasets, but focuses on the practical exploit vectors to launch such an attack. We outline why web-scale datasets have become of critical importance, known security risks of web-scale datasets, as well as auxiliary dataset quality issues that stem from ingesting uncurated data into models.

**Momentum towards uncurated datasets.** Deep learning is most effective when applied to large datasets [10, 30]. But curating such datasets is expensive. Even without manual labeling, the availability of training data has become a limiting factor for further improving model utility. For example, the scaling laws observed in the recent Chinchilla [27] language model indicate that training a compute-optimal 500 billion parameter model would require 11 *trillion* tokens of training data—over 10× more data than is currently used to train models of this size [19]. To drastically scale dataset sizes, it has become common to scrape data from a wider and wider range of untrusted and uncurated web sources.

**Security risk of poisoning attacks.** Uncurated training datasets are prime targets for *poisoning attacks* [6]. For example, an adversary could modify the training dataset ("poisoning" it) so that some targeted example will be misclassified by models trained on this dataset. Early poisoning attacks were designed to target fully-supervised classifiers [18,24,64] trained on curated datasets. These attacks often aim to be "stealthy", by altering data points in a manner indiscernible to human annotators [74]. Attacks on uncurated datasets do not require this strong property. Recent work [14, 15] shows that arbitrarily poisoning only 0.001% of uncurated web-scale training datasets is sufficient to induce targeted model mistakes, or plant model "backdoors" [18, 24].

It is thus known that *if* an adversary were somehow able to poison a fraction of a web-scale dataset, then they could cause significant harm. However, it is not well understood *how* an adversary could place their poisoned samples in any common training dataset *without guessing beforehand which parts of the web will be collected*. This paper answers that question.

**Auxiliary risks related to data quality.** Spending time and effort to curate datasets has benefits besides security. Possibly most important among these is that uncurated data has serious implications for fairness, bias, and ethics [8, 50, 77]. For example, Birhane *et al.* [7] note that LAION-400M has "troublesome and explicit images and text pairs of rape, pornography, malign stereotypes, racist and ethnic slurs, and other extremely problematic content". Many language datasets also contain similarly harmful "hate speech and sexually explicit content, even after filtering" [40].

Dataset curation is not a perfect solution to these problems. Birhane *et al.* [7] note, "without careful contextual analysis, filtering mechanisms are likely to censor and erase marginalized experiences". Any filtering approaches that selectively remove some data sources over others should carefully consider not only the security implications of doing this, but also other more general data quality metrics.

## 3 Threat Model & Attack Scenarios

Before presenting the implementation details of our attacks, we introduce key terminology, our threat model, and a high-level description of the intuition behind our two attacks.

### 3.1 Terminology

Because it is infeasible to distribute web-scale datasets as standalone artifacts (due to the dataset size or regulatory concerns), current training datasets fall into one of two categories.

In the first category, a *maintainer* generates a set of $N$ tuples $\{(url_i, c_i)\}_{i=1}^{N}$ consisting of a resource identifier $url_i$ and auxiliary data $c_i$ (typically a label). We let $t_i$ denote the time at which the $i$-th sample was originally collected. Critically, the maintainer does not provide a snapshot of the data associated with $url_i$, due either to untenable storage costs [3,4,17,31], privacy concerns [13, 72], or copyright limitations [16]. As such, we refer to these as *distributed datasets*. One example is the LAION-5B dataset [57] which consists of five billion tuples of

image URLs and corresponding text captions—corresponding to several hundred terabytes of data.

In the second category of datasets, a *curator* produces a snapshot of a dataset $\{x_i\}_{i=1}^N$, where each sample $x_i$ is drawn from a set of URLs $\{url_i\}_{i=1}^N$ at time $t_i$, and then makes this snapshot publicly available. Because data served by these URLs changes over time, the curator will frequently (e.g., monthly) re-collect a dataset snapshot so that users have an up-to-date view of the data. We refer to these as *centralized datasets*. For example, both Wikipedia and Common Crawl regularly produce snapshots of their entire database. This simplifies access for people training large models, while also discouraging researchers from re-scraping the database directly.

Once one of these two types of datasets has been published, a *client* (e.g., a researcher or applied practitioner) downloads a local copy of the training dataset $\mathcal{D}$, either by crawling each URL for decentralized datasets $\{(url_i, c_i)\}_{i=1}^N$ at a future time $t_i' > t_i$, or by downloading the centralized dataset $\{x_i\}_{i=1}^N$. In practice, clients often use a *downloader* tool developed and maintained by a third party.

## 3.2 Threat Model

We assume the existence of a relatively unskilled, low-resource adversary who can tamper with the contents of a small number of URLs $\{url_i\}_{i=1}^N$ at some point in time $\hat{t}_i$, such that when a client or curator accesses resource $i$ at a future time $t_i' > \hat{t}_i$, they receive a modified (poisoned) dataset $\mathcal{D}' \neq \mathcal{D}$. The difference between the poisoned and intended datasets must be sufficiently large such that a model $f$ trained on $\mathcal{D}'$ will produce poisoned results for some desired input. We let $S_{adv} \subset \{url_i\}_{i=1}^N$ denote the set of URLs an adversary can modify.

We assume the adversary has no specialized or insider knowledge about the curator, downloader, or maintainer other than knowledge of the set of URLs $\{url_i\}_{i=1}^N$ used to generate $\mathcal{D}$ (this information is published by the dataset maintainer or curator). We further assume all maintainers, curators and downloaders behave honestly and do not assist the adversary in any way. As such, the adversary has no control over the auxiliary data $c_i$ (e.g., supervised labels or text descriptions), nor can they add or remove any URLs from the training data that will be crawled by a client or curator.

We make two critical (yet realistic) assumptions that enable our attacks. For distributed datasets, we assume that clients do not compare the cryptographic *integrity* of the local dataset copy $\mathcal{D}'$ that they downloaded, with the original dataset $\mathcal{D}$ indexed by the maintainer. For centralized datasets, we assume that it takes the curator at least some time $\Delta$ to detect malicious changes to the content hosted at any URL $url_i$ in the dataset (e.g., for Wikipedia, $\Delta$ is the time it takes to revert a malicious edit). Thus, the curator cannot detect that $url_i$ hosts poisoned content if the attacker poisoned the content at any time $t_i - \Delta \leq \hat{t}_i \leq t_i$, where $t_i$ is the time at which the content of $url_i$ is included in the dataset snapshot. As we will show, these assumptions holds true for nearly *all* modern web-scale datasets. We discuss (in Section 6) how invalidating these assumptions—via cryptographic integrity checks and randomized crawling—can mitigate the attacks we describe.

## 3.3 Attack Scenarios

We propose two attack strategies for poisoning recent web-scale datasets. In the subsequent sections we demonstrate the efficacy of these attacks in practice on real-world datasets, and describe the ethical safeguards we followed to minimize harm. We focus our attacks on mechanisms that are unique to our study of dataset poisoning. Other potential security vulnerabilities (e.g., the ability of an adversary to interfere with unencrypted network requests from clients, or to exploit website vulnerabilities to inject new content) are out of scope and would only improve our attack success rates.

**Split-view poisoning.** Our first attack exploits the fact that while the index of a distributed dataset published by a maintainer cannot be modified, the content of URLs in the dataset can.[1] This allows an adversary who can exert sustained control over a web resource indexed by the dataset to poison the resulting collected dataset collected by the end-user.

The specific vulnerability we exploit in our attack results from a fairly simple observation: just because a web page hosted benign content when the dataset was initially collected, this does not mean the same page is *currently* hosting benign content. In particular, domain names occasionally expire—and when they do, *anyone can buy them*. We show that domain name expiration is exceptionally common in large datasets. The adversary does not need to know the exact time at which clients will download the resource in the future: by owning the domain the adversary guarantees that *any* future download will collect poisoned data.

We note that attackers already routinely buy expired domains to hijack the residual trust attached with these domains [35, 39, 67]. Attackers have in the past targeted residual trust to defunct banking domains [44] and imported JavaScript libraries [47] to serve malware or steal user data, to take over email addresses associated with the domain [56], to control authoritative nameservers [39], or simply to serve ads [36]. Here, we abuse residual trust to poison distributed datasets. While more sophisticated attacks may accomplish the same goal—such as exploiting a website, coercing a website's owner to modify content, or modifying unencrypted network traffic in flight—we focus on the natural phenomenon of domain expiration in this work.

To select domains to purchase, the adversary can either prioritize cheap domains that host multiple URLs in the dataset

[1] Put differently, there is an important difference between the C types "`int const *`" (how practitioners often treat these URL-based datasets) and "`int * const`" (what the dataset actually provides).

(minimizing the cost per poisoned URL), or domains that host content with specific auxiliary data $c_i$ (recall that the adversary *cannot* modify the auxiliary data contained in the distributed dataset index). We show that split-view poisoning is effective in practice, as the index of most web-scale datasets remain unchanged long after their first publication, even after a significant fraction of the data goes stale. And critically, very few (and no modern) datasets include any form of cryptographic *integrity* check of the downloaded content.

**Frontrunning poisoning.** Our second attack extends the scope of split-view poisoning to the settings where an adversary does *not* have sustained control over web resources indexed by the dataset. Instead, the adversary can only modify web content for a short time period (e.g., a few minutes) before the malicious modification is detected.

This setting is common for datasets that aggregate content published on crowdsourced web pages, such as on Wikipedia. Indeed, it is easy to *temporarily* edit Wikipedia to vandalize its contents [63, 69]. A naive adversary might thus poison some Wikipedia content at arbitrary times and hope that a dataset curator will scrape the poisoned pages before the malicious edits are reverted. However, Wikipedia vandalism is reverted in a few minutes on average [80], so any randomly-timed malicious edits are unlikely to affect a dataset collection.

Our frontrunning attack relies on the fact that an adversary can, in some cases, predict *exactly* when a web resource will be accessed for inclusion in a dataset snapshot. As a result, the adversary can poison dataset contents just prior to a curator collecting a snapshot, thereby *frontrunning* content moderators who will later revert the malicious edits. We will show that frontrunning attacks are particularly effective for Wikipedia datasets, because the official Wikipedia snapshot procedure accesses articles in a predictable—and well documented[2]—linear sequence. An attacker can thus predict the snapshot time $t_i$ of any Wikipedia article down to the minute.

# 4 Split-View Data Poisoning

We now begin our evaluation starting with split-view data poisoning attacks, where an attacker poisons a distributed dataset by purchasing and modifying expired domains.

## 4.1 Our Attack: Purchasing Expired Domains

While split-view poisoning is applicable to any distributed dataset, we focus on multimodal image-text datasets. In such datasets, each URL points to an image hosted by some data provider, and the auxiliary data contains a textual description of the image, e.g., an annotated class label or a caption extracted from the web page.

[2] https://en.wikipedia.org/w/index.php?title=Wikipedia:Database_download&oldid=1138465486

Our attack exploits the fact that the Domain Name System (DNS) does not assign permanent ownership of any domain to a particular person or organization, but rather grants short (yearly) "leases" that must be frequently renewed. Thus, domain names continuously expire over time—intentionally or not—when the re-registration fees are not paid.

When the domain that hosts an image in a distributed dataset expires, *anyone* can pay to take ownership over this domain and thereby gain the ability to return arbitrary content when the indexed image is later downloaded by a victim client. Split-view poisoning abuses the residual trust inherent in an expired domain, as in traditional domain hijacking attacks [39]. We find that for many popular distributed datasets, domains are included with relatively lax quality-assurance measures (if any), and thus domains with no special status that have been expired for months can freely be acquired to control a modest fraction of the entire dataset.

In this section we study to what extent it is possible to poison datasets by purchasing expired domains. We first quantify the fraction of domains that are expired in popular distributed datasets (§ 4.2), then measure the frequency at which these datasets are scraped (§ 4.3), verify this attack is not currently exploited in the wild (§ 4.4) and finally study the attack's potential down-stream impact (§ 4.5).

## 4.2 Quantifying the Attack Surface

Table 1 lists ten recent datasets we study in this paper that are vulnerable to split-view poisoning. The three oldest datasets (PubFig, FaceScrub, and VGG Face) are datasets of faces and associate each image with the identity of a single person (most often a popular celebrity). The remaining seven datasets are multimodal datasets containing URLs that point to images, along with textual captions automatically extracted from the HTML of the webpage. As such, for these datasets, the image can be modified by the owner of the corresponding domain, but the image's caption is fixed in the dataset index.

To measure the fraction of images that could be potentially poisoned, we count the number of images hosted on domains that are *expired* and *buyable*. We say that a domain is **expired** if the DNS record for that domain does not exist. To measure this, we perform an `nslookup` on every domain name in the dataset from two geographically distinct data-centers in May 2022 and August 2022 and report the domain as expired if all four lookups result in a `NXDOMAIN` response.

We further call a domain **buyable** if it is expired, and if at least one domain name registrar listed the domain as explicitly for sale by the registrar[3] in August 2022. Instead of counting the *total* fraction of data that is buyable (which would represent a financially unconstrained adversary), Table 1 reports the fraction of images in the dataset from domains that can be

[3] Some registrars list domains as for sale even if they are actually owned by a squatter. We exclude these domains from our set of buyable domains because purchasing these domains is often an expensive and lengthy process.

| Dataset name | Size ($\times 10^6$) | Release date | Cryptographic hash? | Data from expired domains | Data buyable for $10K USD | Downloads per month |
|---|---|---|---|---|---|---|
| LAION-2B-en [57] | 2323 | 2022 | ✗† | 0.29% | $\geq$ 0.02% | $\geq$7 |
| LAION-2B-multi [57] | 2266 | 2022 | ✗† | 0.55% | $\geq$ 0.03% | $\geq$4 |
| LAION-1B-nolang [57] | 1272 | 2022 | ✗† | 0.37% | $\geq$ 0.03% | $\geq$2 |
| COYO-700M [11] | 747 | 2022 | ✗‡ | 1.51% | $\geq$ 0.15% | $\geq$5 |
| LAION-400M [58] | 408 | 2021 | ✗ | 0.71% | $\geq$ 0.06% | $\geq$10 |
| Conceptual 12M [16] | 12 | 2021 | ✗ | 1.19% | $\geq$ 0.15% | $\geq$33 |
| CC-3M [65] | 3 | 2018 | ✗ | 1.04% | $\geq$ 0.11% | $\geq$29 |
| VGG Face [49] | 2.6 | 2015 | ✗ | 3.70% | $\geq$ 0.23% | $\geq$3 |
| FaceScrub [46] | 0.10 | 2014 | ✓§ | 4.51% | $\geq$ 0.79% | $\geq$7 |
| PubFig [34] | 0.06 | 2010 | ✓§* | 6.48% | $\geq$ 0.48% | $\geq$15 |

Table 1: **All recently-published large datasets are vulnerable to *split-view poisoning* attacks.** We have disclosed this vulnerability to the maintainers of affected datasets. All datasets have $> 0.01\%$ of data purchaseable (in 2022), far exceeding the poisoning thresholds required in prior work [14]. Each of these datasets is regularly downloaded, with each download prior to our disclosure being vulnerable.

† LAION-5B released a "joined" version of this dataset with a cryptographic hash over the text of the URL and Caption (not the *contents* of the URL), and as such does not protect the integrity of the actual image.
‡ COYO-700M images are distributed with pHash [32] which validates benign image changes, but is not adversarially robust [29].
§ FaceScrub and PubFig contain cryptographic hashes, but the dataset maintainers do not provide an official downloader client that verifies these. We find that nearly all third-party downloaders for these datasets ignore hashes.
* PubFig was initially released without hashes, but hashes were later added in Version 1.2 of the dataset.

purchased for a total cost of $10,000 USD. (Figure 1 plots the fraction of datasets that can be purchased as a function of total cost.) To compute this, we sort domains in decreasing order of "images per dollar": the number of images the domain hosts divided by the cost to purchase this domain.

Overall, we see that an adversary with a modest budget could purchase control over at least 0.02%–0.79% of the images for each of the 10 datasets we study. This is sufficient to launch existing poisoning attacks on uncurated datasets, which often require poisoning just 0.01% of the data [15].[4]

We also find that there is a direct relationship between the age of a dataset and how easy it is to poison. Older datasets are more likely to contain expired domains, and therefore an adversary can purchase a larger fraction of the dataset.

**Datasets are vulnerable from day zero.** One limitation of our above analysis is that we have measured the fraction of datasets vulnerable to poisoning in August 2022, but many of these datasets were constructed years earlier. It is therefore likely that many people who use these datasets would have already downloaded them at an earlier date, and thus may not have been vulnerable to our poisoning attack (although we will show in the following section that fresh downloads of each of these datasets remain frequent today).

Fortunately, the COYO-700M dataset was released during the writing of this research paper, on 30 August 2022. On that same day, we computed the fraction of the dataset that was vulnerable to our poisoning attack and found that already 0.15% of the images were hosted on expired domains that cost fewer than $10,000 USD to purchase. The reason that the number of expired domains is not zero on release is that building these large datasets is a time-consuming and expensive procedure. And so even though the COYO-700M index was released in August 2022, it took nearly a year to collect the dataset [11], giving ample time for the earliest scraped domains to have expired before the dataset was released.

**Measuring the attack cost.** The most immediate question is if this attack can be realized in practice. The primary constraint of our attack is the monetary cost of purchasing domains, which we measure using the costs reported by Google Domains in August 2022. In Figure 1 we show the fraction of images in a dataset that can be controlled by the attacker as a function of their budget. We find that at least 0.01% of each dataset can be controlled for less that $60 USD per year.

## 4.3 Measuring the Attack Impact

Our attacks are "retroactive" in the sense that we can poison a dataset after it has been initially constructed by the curators—but the impact is limited to those who download it *after* we take over the domains. And given that it has been several years since many of these datasets were initially constructed, it is not obvious that anyone would still download them by scrapping URLs instead of reusing a previously downloaded version of the dataset. As a result, in order to measure the potential impact of a split-view poisoning attack it is necessary

[4] Carlini & Terzis [15] assume the adversary can modify images *and* their captions, whereas our adversary only controls images. In § 4.5, we show modifying captions is unnecessary for a successful poisoning attack.

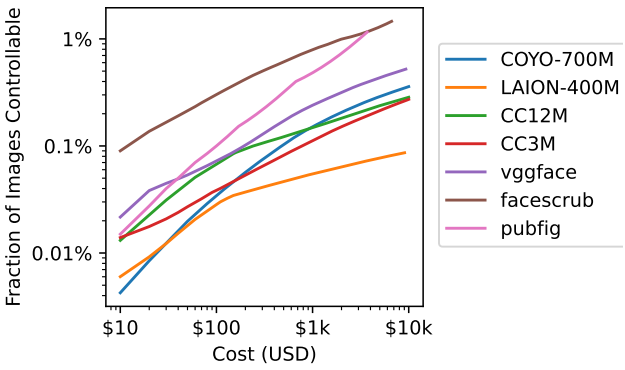


Figure 1: **It often costs $\leq$ \$60 USD to control at least 0.01% of the data**. Costs are measured by purchasing domains in order of lowest cost per image first.

to study the rate at which these datasets are actually still being actively downloaded by researchers and practitioners today.

**Methodology.** We measure the rate at which each of these distributed datasets are downloaded from the internet by purchasing multiple expired domains from each of the 10 listed datasets, and passively monitoring requests to measure the rate at which URLs corresponding to images from the distributed datasets are being downloaded.

For each dataset, we purchase the three most popular expired and buyable domains (that is, the domains available for purchase that hosted the most images), and three randomly selected domains that were available for purchase. We wrote a simple server to log all incoming HTTP and HTTPS[5] requests, including the access time, a hash of the IP address, the full URL being requested, and any additional headers provided. This allowed us to count the frequency at which these datasets are still downloaded. We ran this server for six months beginning in August 2022.

**Analysis approach.** Our server received approximately 15 million requests per month during our study, a rate of 6 requests every second. However from here it becomes necessary to separate the requests that were actually intending to download images from one of these datasets, from requests that come from other internet users or web crawlers.

To begin our analysis, we make a (significant) simplifying assumption that anyone downloading one of these datasets does so from a single IP address. We may thus underestimate the true rate at which each dataset is downloaded. We then say that a particular IP address X downloads a dataset D at time $[T_0, T_1]$ if we meet both a precision and recall requirement:

1. **Recall:** Within the time range $[T_0, T_1]$, the IP address X downloads at least 90% of the URLs contained in $D$ under our control, including at least one URL from each of the 6 domains we own for that dataset.

2. **Precision:** At least 50% of the requests issued by X within this time range to the domains we control are to URLs in $D$.

These conditions are conservative, but ensure we filter out web crawlers and other mass internet scrapers because these are likely to crawl other URLs from this domain (violating the precision constraint) or not crawl the majority of the URLs from the dataset (violating the recall constraint). Additionally, because we own six domains per dataset it is exceptionally unlikely that, by random chance alone, one particular IP will request URLs from each of these six otherwise-unrelated domains. (In fact, we find that even checking 3 of these URLs would give almost identical precision.) As a point of reference, for the CC-3M dataset, we received 51,000 image requests per month from a total of 2401 unique IPs. By applying the precision constraint alone, we reduce this to 2007 unique IPs and 43,000 image requests; by applying the recall constraint alone we get 70 unique IPs and 32,000 image requests; and both together yields a further reduction of 64 unique IPs and 28,000 image requests (per month).

#### 4.3.1 Results

We report our results in the rightmost column of Table 1. Even the oldest and least frequently accessed datasets still had at least 3 downloads per month. Thus, over the six months we tracked data, there were over 800 downloads that we could have poisoned with our attack. Unsurprisingly, we found that newer datasets are requested more often than older datasets. Different datasets thus offer different tradeoffs for attackers: newer datasets have a smaller fraction of purchasable images, but an attack can reach many more vulnerable clients.

We observe that the largest billion-image datasets are downloaded significantly less often than smaller recent datasets. We found that the reason for this is that these datasets are rarely downloaded in their entirety; instead, they serve as an upstream source for smaller subsets. For example, the Public Multimodal Dataset (PMD) [66] and LAION-Aesthetics [59] datasets consist almost entirely of images drawn from LAION-2B-en. Sub-datasets like this explain why sometimes we see IP addresses with very high precision but low recall.

**Visualizing dataset crawlers.** Using our log files, we can visualize the ways in which dataset downloaders access these datasets by plotting URL requests as a function of time. We order the set of URLs we bought for each dataset according to their order in the original dataset index. In this way, crawlers that process the dataset index linearly should appear (roughly) as a linear line in our plot of URL requests over time. To improve clarity, we assign each unique IP that accesses our server a separate random color.

[5] To also monitor HTTPS traffic we obtained certificates from LetsEncrypt for each of our domains.

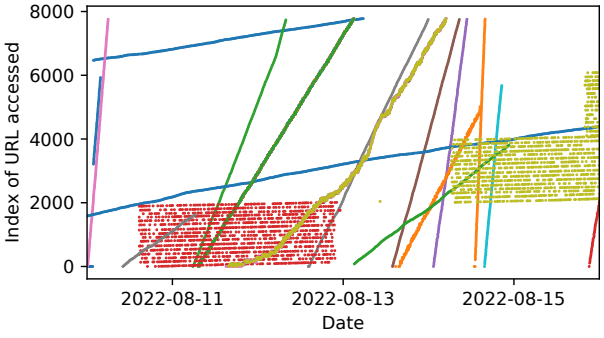


Figure 2: Visualization of users downloading Conceptual 12M. By monitoring which URLs are requested from the domains we purchased, we plot every time a URL is requested over time, color coded by the source IP, and can directly read off several dozen users crawling Conceptual 12M. Appendix Figure 8 compares various filtering approaches.

We show this plot for the Conceptual 12M dataset in Figure 2. We can find several trends in this data. First, most users who download this dataset do so in a linear order from the first to the last URL. However, the *rate* at which URLs are accessed is highly variable: some downloaders crawl the entire dataset in a few hours, while others take several weeks to download the same dataset. We also observe some users that batch the data into chunks and download each chunk in parallel, as well as users that pause their download momentarily and then resume a few hours later (on a different IP).

While our strict precision and recall requirements already give strong evidence that the IP addresses we logged were indeed downloading the dataset, the linear ordering of URL requests from these addresses all but confirms this. Indeed, because the ordering of URLs in the dataset index is *random* (instead of, say, alphabetical), a dataset download appears to be the only explanation for why the URLs would be linearly accessed in this particular order.

**User-agent verification.** The most popular user agent[6] is responsible for 77% of the traffic to our domains. This user agent is hardcoded[7] in `img2dataset` tool [5], a popular dataset crawler. Given the browser involved is Firefox 72–which was superseded in February 2020–it is highly likely that most of the requests indeed originate from `img2dataset`.

**Ethical considerations.** We do not actually poison any datasets. For all URLs we own, we return a 404 Not Found response; so from the perspective of a dataset downloader our purchasing of the domain is completely transparent. We further place a `robots.txt` file to prevent typical web-scrapers from crawling our domains—this is unlikely to impact dataset downloads since dataset crawlers ignore this file. Finally, a request to the root domain returns a 403 Forbidden with a response body explaining that this domain is part of a research study. In this response we list a contact email address and also offer to return this domain to the original owner in case it was allowed to expire accidentally. We have not received any contact on this address. Appendix E contains the text of our landing web page.

Our data collection is minimally invasive. When searching for expired domains, we limit our DNS requests to 500/second, we only ask for the cost of purchasing the top-10,000 domains in each dataset, and only eventually purchase six.

This research study was deemed "exempt" by the ETH Zurich IRB. Google does not have an IRB, but the research plan was reviewed by experts at Google in areas including ethics, human subjects research, policy, legal, security, privacy, and anti-abuse.

### 4.4 Is This Attack Exploited in the Wild?

Given that this attack vector has existed for years against many datasets, and is easy to execute, it is not implausible that someone would have carried out such a poisoning attack in the past. Yet, we could not find any evidence of this.

We search for a signature of a domain-purchasing attack by looking for domains that (1) host images that have been modified since the initial dataset release and (2) have changed ownership since the initial dataset release. To detect image changes, we can either check if images are perceptually similar to the originals (via, e.g., CLIP embeddings [52]) or exactly identical (via a cryptographic hash). Comparing images with cryptographic hashes has no false-negatives (i.e., any change is detected) but gives false-positives for "benign" changes, e.g., if a domain re-encodes or resizes its images. In contrast, a perceptual hash has fewer false-positives but can have false-negatives if an adversary buys a domain and modifies images while preserving the perceptual hash (which is not collision-resistant). Finally, to detect if a domain changed ownership, we request the `whois` record and check the last purchase date.

**Results.** We perform our initial analysis on CC3M, a dataset where we have the original raw images as ground truth. Among all domains hosting more than 10 images, we find *just one* domain has our attack signature when comparing with perceptual image similarity. Upon further investigation, we find that this domain has been purchased by a domain squatter and any request to an image file on the domain return an advertisement. If we instead compare cryptographic hashes of images, we find the same domain squatter and also two other domains that have been repurchased. However, further investigation reveals the ownership of these domains has not changed and the DNS record simply lapsed, and images were re-encoded.

[6] `Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:72.0) Gecko/20100101 Firefox/72.0`

[7] https://github.com/rom1504/img2dataset/blob/fc3fb2e/img2dataset/downloader.py#L41

We also conduct this same analysis on LAION-400M. Here we study three versions of the data: at original release (2021-11), and our own downloads at two later dates (2022-04 and 2022-08). We find no domain with such a signature in LAION-400M, and thus have no evidence at present that this attack would have been exploited against this dataset. Because we only have the original CLIP embeddings for this dataset (the original raw bytes were not saved), we only perform this comparison. We find that by the first and second snapshot respectively, there are 4.1M and 4.2M unique domains (of 5.6M) that host at least one modified image—in total, we found 175M and 183M modified images, respectively. We measured this using a CLIP cosine similarity $< 0.99$. We randomly sample a few thousand domains including the 700 with the most modified images. We find many cases where domains are still owned by the original owner, are currently for sale, or have been redacted, but none appear malicious.

### 4.5 Putting It All Together

Prior work [15] has successfully poisoned multimodal models contrastively trained on datasets of 3 million images, under the assumption the adversary can *arbitrarily* control the label of manipulated images. However, in this paper we study datasets over $100\times$ larger, and assume an adversary with *no* control over the text captions. Are poisoning attacks effective with these two changes?

We find they are. We consider two poisoning attack objectives: (1) cause a particular image to be misclassified as some target label from ImageNet, and (2) cause a particular image to be classified as NSFW by the Stable Diffusion Safety Filter [54]. For each of these attack objectives, we first identify appropriate text captions in LAION 400M for which the corresponding image domains can be purchased for a total of $1,000 USD. Then, we locally replace these images with poisoned samples to simulate the effect of an attack, without any potential to cause harm to others.

Specifically, we train one OpenCLIP [28] model using a ViT-B-32 architecture for 32 epochs, at a batch size of 3072 on 16 A100 GPUs. For our object-misclassification objective, we choose ten ImageNet classes that appeared in multiple text captions of images we can control. When we poison 1,000 images (0.00025% of the total dataset) our attack has a success rate of 60% in flipping the model's zero-shot classification of the targeted image. For our NSFW objective, we find captions corresponding to images (from buyable domains) labeled as UNSAFE in the LAION 400M dataset index. Again at 1,000 poisoned samples, our attack has a success rate of above 90%. More details about this experiment are in Appendix D.

## 5 Frontrunning Poisoning

Our second attack removes the assumption that the adversary has *sustained* control over the web data in a training set. To do this we make a new assumption: that we can predict precisely *when* the web content will be downloaded. We will investigate this attack on Wikipedia-derived datasets, but also discuss how similar vulnerabilities may exist in the Common Crawl dataset in Appendix B.

### 5.1 Our Attack: Editing Wikipedia

Wikipedia is a crowdsourced encyclopedia. This makes it one of the most comprehensive and reliable datasets available on the internet [79]. As a result of its quality and diversity, Wikipedia is frequently sourced for ML training data. Indeed, many language modelling datasets heavily rely on the English Wikipedia, e.g., it formed over 75% of the words in the BERT training set [21], 1.5% of the Pile dataset [23], and the entirety of the WikiText dataset [43]. Many task-specific datasets also rely on the English Wikipedia: e.g., the WikiQA [81] question answering dataset (30,000+ downloads), and the WikiBio [38] biography writing dataset (19,000+ downloads). Finally, some of the distributed datasets discussed in Section 4 index many images from Wikipedia articles.

Because Wikipedia is a *live* resource that anyone can edit, an attacker can poison a training set sourced from Wikipedia by making malicious edits. Deliberate malicious edits (or "vandalism") are not uncommon on Wikipedia, but are often manually reverted within a few minutes [80]. As a result, actually poisoning Wikipedia appears challenging: unlike the attacks in our prior section, an adversary cannot exert sustained control of any particular page and would thus have to hope that their malicious edit is timed just perfectly to affect a dataset download before being reverted.

However, we make one key observation that will guarantee the success of our poisoning attack: Wikipedia-derived datasets are not themselves live, but rather a collection of static snapshots. This is because Wikipedia forbids using web crawlers to scrape the live website. Instead, Wikipedia makes available regular "dumps" (or snapshots) of the entire encyclopedia. Thus, training datasets sourced from Wikipedia use these snapshots instead of data crawled directly from the site. For example, the authors of the BERT model [21] explicitly recommend "to download the latest [Wikipedia] dump" to reproduce their results.

This makes it possible to mount what we call a **frontrunning attack**. An attacker who can predict *when* a Wikipedia page will be scraped for inclusion in the next snapshot can perform poisoning immediately prior to scraping. Even if the edit is quickly reverted on the live page, the snapshot will contain the malicious content—*forever*. The attentive reader may argue we have not gained much: instead of having to predict the time at which Wikipedia is crawled by the end-user to produce a training set, the attacker has to predict the time at which Wikipedia is crawled to produce an official snapshot. But as we will see, the latter is actually easy.

In this section, we explore how an adversary can time mali-

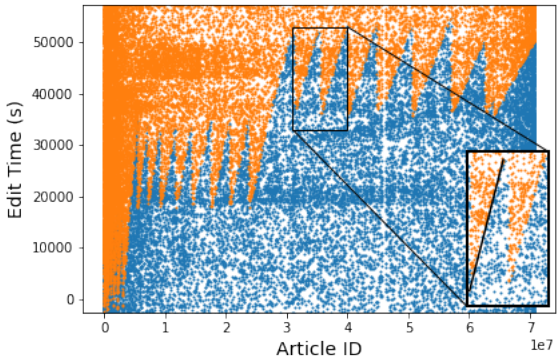


Figure 3: **An adversary can easily predict when any given Wikipedia article will be snapshot for inclusion in the bi-monthly dump.** We visualize edits around the June 1st, 2022 Wikipedia snapshot. Each point corresponds to an edit made to a Wikipedia article, with the article ID on the X axis and time (in seconds) that the edit was made on the Y axis. Edit points colored blue were *included* in the snapshot, and edits colored orange were *not* included. The "sawtooth" pattern exhibited in the plot indicates a trend where multiple parallel jobs crawl Wikipedia articles sequentially to construct the snapshot. Furthermore, these parallel jobs run almost perfectly linearly through their allocated pages.

cious edits to guarantee successful poisoning of a Wikipedia snapshot. To this end, we need to answer two questions:

1. How accurately can we predict the time at which a page is scraped for inclusion in a Wikipedia snapshot?

2. How quickly do malicious edits get reverted?

## 5.2 Predicting Checkpoint Times

Wikipedia produces snapshots using a deterministic, well-documented protocol (with details that are easy to reverse-engineer through inspection). This makes it possible to predict snapshot times of individual articles with high accuracy.

### 5.2.1 How Wikipedia Snapshots Work

English Wikipedia is archived on the 1st and 20th of each month. The snapshot is produced by $n$ parallel workers; all Wikipedia articles are ordered sequentially by their ID and split into $n$ chunks, and each worker independently and linearly scrapes all articles in their chunk.

Due to Wikipedia's size, the whole process takes nearly a day to complete. Different articles thus get scraped at significantly different wall-clock times. As a result an edit at time $t_i$ for one article may be excluded from the snapshot, while an edit at time $t_j > t_i$ for a different article might be included. Figure 3 visualizes this "sawtooth" effect: there are many edits (in blue) that are included in the June 1st dump that were made *before* (i.e., below) a different edit that was **not** included (in orange).

For a frontrunning attack to succeed, it is thus not sufficient to just predict the time at which the snapshot procedure begins. The attacker also needs to predict the precise time at which each individual page is scraped.

### 5.2.2 Exploiting Rolling Snapshots

To precisely predict each article's scrape time, the attacker can exploit consistencies in Wikipedia's snapshot process.

First, the adversary knows precisely when each dump starts, because Wikimedia *explicitly makes this information available* by publishing live statistics on ongoing snapshots.[8]

Second, the *rate* at which articles are crawled in a dump remains nearly consistent across dumps, and can thus be approximated from prior dumps (interestingly, crawls tend to speed up slightly over time).

With these two pieces of information, the attacker can precisely predict when any given article will be crawled. For an article $i$, we denote the time at which it is crawled for the current snapshot as $t_i$ and its crawl time in the previous snapshot as $t_{i,\text{prev}}$. We denote the start time of the current and previous snapshots (as reported by Wikimedia) as $t_0$ and $t_{0,\text{prev}}$ respectively. Due to our first observation above, the attacker knows $t_0$ and $t_{0,\text{prev}}$. Due to our second observation, we have that $t_i - t_0 \approx t_{i,\text{prev}} - t_{0,\text{prev}}$. This allows us to estimate the snapshot time of the $i$-th article as $t_i \approx t_0 + (t_{i,\text{prev}} - t_{0,\text{prev}})$. But calculating this requires knowledge of $t_{i,\text{prev}}$—the time at which the $i$-th article was crawled in the previous snapshot. We now discuss how to retroactively estimate this.

### 5.2.3 Determining the Article Snapshot Time

Wikipedia snapshots do not explicitly list the snapshot time for each article. But Wikipedia does give some auxiliary information: a complete list of edits with the precise time every edit is made. We show how this information can be used to retroactively estimate an article's snapshot time.

Recall from Figure 3 that for each article we can find the list of edits that were included in the current snapshot (blue points), with later edits appearing in the next snapshot (orange points). For each article, we thus know that the snapshot time $t_i$ was somewhere between the times of the article's last included edit (top-most blue point) and the first non-included edit (bottom-most orange point). However, this interval is loose: the time between these edits is often several days.

To refine our estimate of the snapshot time for each article, we can again exploit the consistency present in the Wikipedia crawling process. We observe that articles are processed sequentially: by zooming in to just a single crawling job as shown in Figure 3, we see that articles are crawled sequentially, and a clear line separates the last-included and first-

[8] On https://dumps.wikimedia.org/backup-index.html

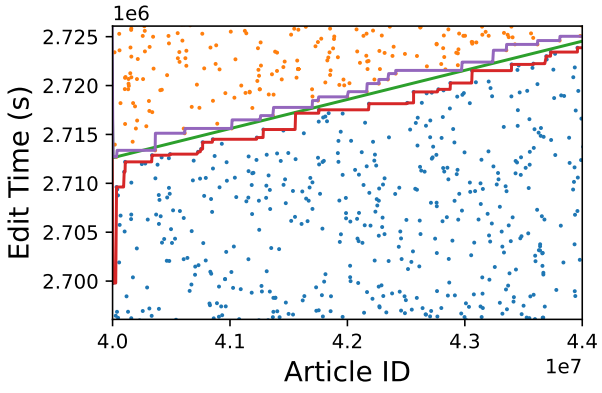


Figure 4: We can obtain tight estimates on the time at which each article is snapshot. The green and orange lines show the interval $[t^{\text{low}}_{i,\text{prev}}, t^{\text{high}}_{i,\text{prev}}]$ for a range of articles from the English Wikipedia. On average, our predictions (blue line) are 27 minutes from the furthest interval boundary.

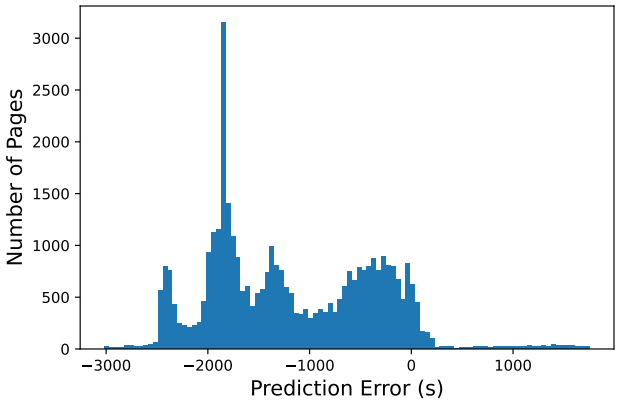


Figure 5: Distribution of Wikipedia checkpoint time prediction errors. Most predicted checkpoint times are within 30 minutes of our constructed ground truth. In general, we predict edits too early, so it is important to later adjust for this bias, as we will discuss in Section 5.4.

not-included edits of each article. That is, for articles $i$ and $j$ processed in the same job, we have that $t_i < t_j$ if $i < j$. We can thus tighten our interval around each article's edit time by continually tracking the most recent edit made before the snapshot (for each article, this is the top-most blue edit made on an earlier article in that job), as well as the earliest edit among all subsequent articles in the job that was not included in the snapshot. We visualize this in Figure 4. For each article in the previous snapshot, we can thus obtain a time interval $[t^{\text{low}}_{i,\text{prev}}, t^{\text{high}}_{i,\text{prev}}]$ that contains the true (but unknown) snapshot time $t_{i,\text{prev}}$. By our construction outlined above, we guarantee that the lower and upper limits of these intervals monotonically increase for all articles in a job (see Figure 4).

To produce an estimate $\hat{t}_{i,\text{prev}}$ for each article's previous snapshot time, we compute a best linear fit for the snapshot intervals of all articles processed by a single thread, as illustrated in Figure 4. This lets us predict the article's next snapshot time as $\hat{t}_i \approx t_0 + (\hat{t}_{i,\text{prev}} - t_{0,\text{prev}})$.

#### 5.2.4 Evaluating our Predictions

We now evaluate our procedure for estimating article snapshot times. Ideally, we would directly compare our predicted snapshot times $\hat{t}_i$ with the true snapshot time of the $i$-th article. But we do not know the ground truth, except up to some interval $[t^{\text{low}}_i, t^{\text{high}}_i]$ which we can compute a posteriori as described above. We thus proceed in two steps.

First, we show that our linear fit to estimate the previous snapshot time $\hat{t}_{i,\text{prev}}$ is accurate. For this we measure the maximum absolute error between the predicted time $\hat{t}_{i,\text{prev}}$, and the unknown ground truth in the interval $[t^{\text{low}}_{i,\text{prev}}, t^{\text{high}}_{i,\text{prev}}]$. This provides an upper bound on the true estimation error. We find that the estimation error is bounded by 27 minutes on average.

Second, we evaluate the accuracy of the *extrapolation* of our predictions from one snapshot to the next. That is, we evaluate how close our *a priori* predicted snapshot time $\hat{t}_i := t_0 + (\hat{t}_{i,\text{prev}} - t_{0,\text{prev}})$ is to the snapshot time that we could have estimated *a posteriori* using the linear fit described above. Figure 5 shows the distribution of the error estimates. Our predictions are correct to within roughly 30 minutes in most cases. We notice, however, that our extrapolation errors are biased towards negative. We find that this is because snapshots slightly speed up over time, so we typically overestimate the next snapshot time of an article. We will correct for this in Section 5.4, when we produce a conservative estimate of our attack's success in poisoning Wikipedia snapshots.

### 5.3 Estimating Revision Speed

Now that we have measured how accurately we can predict when a future snapshot will happen, we turn our attention to measuring the size of the opportunity window to make a malicious edit before it is reverted.

We remark that while the most accurate methodology would be to inject malicious edits and measure the distribution of reversion times, we believe this would be unethical. Instead, we take an entirely *passive*—albeit less accurate—approach as discussed in Section 5.6.

To measure the speed of revisions, we construct a dataset of all edits made to Wikipedia from January 2021 to June 2022 (for a total of 18 months), and classify every edit as either an addition or as a reversion if they contain one of a fixed set of strings[9] which are frequently used in reversion comments. We then *conservatively* assume that the edit being

[9]This set of strings is produced by manual analysis of a sample of comments from each Wikipedia; details are given in Appendix B.1.

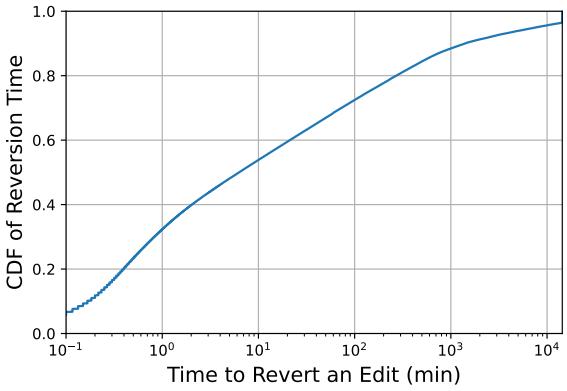


Figure 6: A CDF of revision times for English Wikipedia. Roughly 35% of revisions take more than 30 minutes.

reverted was the immediately preceding edit, and so measure the reversion time as the elapsed interval between these two edits.[10] Figure 6 plots this distribution. When we combine the roughly 30 minutes of error in predicting future snapshot times (c.f. Figure 5), with another roughly 30 minutes for the average uncertainty in our estimate of the true snapshot time (c.f. Figure 4), we can conservatively estimate that the attacker can time their edit so as to be within one hour, on average, of the true snapshot time. Approximately 32% of reversions take more than an to be reverted and so the attack is likely to succeed often. In the next section, we refine this estimate to determine more precisely how many articles we could have poisoned.

## 5.4 Putting It All Together

Using our predictions of relative article snapshot times, our interval bound on the true snapshot time, and the distribution of reversion times, we can now (conservatively) determine what fraction of Wikipedia an adversary could have poisoned. There are two potential "failure cases" where a malicious edit may not make it into the checkpoint:

- the malicious edit is applied too late: the article was already snapshot, or
- the malicious edit is applied too early: the edit gets reverted before the article is snapshot.

This induces a tradeoff: the attacker wants to make edits early enough to ensure they do not miss the snapshot time, but late enough to maximize the chance of frontrunning editors.

We therefore compute the optimal time to apply a malicious edit as follows. Recall that we use $[t_i^{\text{low}}, t_i^{\text{high}}]$ to represent the tightest interval around the true (but unknown) snapshot time $t_i$ of the $i$-th article, and $\hat{t}_i$ is the predicted snapshot time. To balance the two failure modes above, and to account for the bias in our predictions (see Section 5.2.4), we introduce an "adjustment" variable $a$ so that the adversary adds their malicious edits at time $\hat{t}_i + a$ instead of exactly at time $\hat{t}_i$.

Then the fraction of malicious Wikipedia edits that will make it into the snapshot, when they are made at time $\hat{t}_i + a$, can be lower-bounded as:

$$\mathcal{A}(a) = \frac{1}{|D|} \sum_{i \in D} (1 - \overbrace{p_{rev}(\hat{t}_i + a; t_i^{\text{high}})}^{\text{Edit applied too early}}) \cdot (1 - \underbrace{\mathbb{1}[\hat{t}_i + a > t_i^{\text{low}}]}_{\text{Edit applied too late}}),$$

where $\mathbb{1}[\hat{t}_i + a > t_i^{\text{low}}]$ is the indicator function that is one if the edit is applied after the checkpoint (here we conservatively use our lower bound $t_i^{\text{low}}$ on the true checkpoint time), and $p_{rev}(\hat{t}_i + a; t_i^{\text{high}})$ is the probability that the edit is reverted before the checkpoint (here we conservatively use the upper bound $t_i^{\text{high}}$ on the true checkpoint time).

We compute this sum using our results from Section 5.2 and Section 5.3. By taking the maximum over a sweep of potential $a$ values, we obtain $\max_a A(a) = 0.065$. That is, according to our conservative analysis, **we can poison 6.5% of Wikipedia documents** absent any other defensive measures.

In reality, of course, there are a number of factors beyond our analysis that would likely prevent us from reaching this fraction, such as rate limiting of edits or IP bans. We also "cheat" in choosing the optimal value of the adjustment value $a$, but we do not consider this a major limitation—it is likely that an adversary could use more historical data to produce better estimates $\hat{t}_i$ as well as good estimates of $a$. However, our analysis is also pessimistic in that we assume we only try *once* to poison any given article. An adversary may attempt a more targeted attack, as we discuss in Appendix B.2, where they retry edits on targeted articles, to force editors to revert multiple times and increase the likelihood of the edit making it into the dump. Ultimately, our best-effort estimate of 6.5% of poisoning success is orders-of-magnitude higher than what is required in prior poisoning attacks [15]. We thus argue that a successful frontrunning poisoning attack on Wikipedia snapshots is practical, and that finding ways to mitigate such attacks is a worthwhile research direction.

## 5.5 Multilingual Wikipedia

Wikipedia is also frequently used for non-English language modeling. For example, the multilingual version of BERT is trained entirely on the top 104 Wikipedia languages.[11]. Multilingual datasets often rely *more heavily* on Wikipedia than English datasets. Thus, poisoning Wikipedia is even more harmful for non-English language modeling tasks. To measure

[10] This under-reports the edit time because if the vandalism was from an earlier edit, we would incorrectly use the later edit's time instead.

[11] See https://github.com/google-research/bert/blob/master/multilingual.md#list-of-languages.

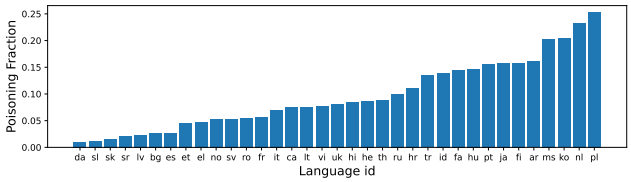


Figure 7: **Multilingual Wikipedia may be more vulnerable to frontrunning poisoning attacks.** We compute poisoning rates for 36 of the 40 languages languages contained in Wiki-40B [25] by reusing our attack from Sections 5.2 to 5.4.

this vulnerability, we investigate the Wiki-40B dataset [25] which is frequently used to train large multilingual models.

We repeat our analysis from the previous section on 35 of the 39 non-English languages contained in Wiki-40B by identifying which strings often represent a reversion in these languages.[12] Again our analysis here is loose: we identify only a subset of (often automated) reversions; however for the same reason as above we believe this represents a lower bound of the mean reversion time.

We find that 22 (63%) of the non-English Wikipedias were easier to poison than the English Wikipedia, as shown in Figure 7. Feasible poisoning rates range from 0.95% to as much as 25.3%, with a median value of 8.2%. In general, the increase in vulnerability comes from multilingual Wikipedias having more predictable checkpoints, for two reasons. First, because these Wikipedias are smaller, the entire checkpointing procedure is shorter, reducing the amount of variance in checkpoint time between different pages. Second, because the Wikipedias change less between successive checkpoints, the speed of checkpointing is more stable, improving our predictions. This may be why some of the larger Wikipedias, such as Spanish, Danish, and Italian, have comparable poisoning rates to English Wikipedia. However, our interval-based measurement is also more conservative for languages with slower edits, as the intervals will be larger, giving very small lower bounds for some small Wikipedias, such as Slovak and Slovenian.

We reiterate that such large poisoning rates are unlikely to ever happen, due to IP bans or rate limiting. The most important takeaways from our analysis here are that 1) multilingual Wikipedias are vulnerable to poisoning, and often more vulnerable than English Wikipedia, and 2) multilingual datasets tend to rely more on Wikipedia than English datasets do, compounding this risk.

### 5.6 Ethical Considerations

Our actions here are entirely passive. We make no edits to Wikipedia and, aside from downloading datasets from the official sources, never interact with Wikipedia. While this leads to limitations in our analysis, we believe this is the correct way to run such a study to avoid harming the Wikipedia editor community. We disclosed our attack analysis (and later defense recommendations) to researchers at Wikimedia who acknowledged the vulnerability before release of this paper.

[12] We were unable to access the German, Chinese, and Czech checkpoints for the checkpoint times we studied, and Tagalog did not have enough data to reliably analyze.

## 6 Defenses

In order to address the attacks that we identified, we propose an integrity-based defense for split-view poisoning and a timing-based defense for frontrunning poisoning. We also discuss potential directions to address poisoning more generally. We shared these defenses with curators, maintainers, and downloaders as part of our responsible disclosure and report on the status of their implementation of defenses.

### 6.1 Existing Trust Assumptions

Per our threat model (Section 3), our proposed defenses assume that all maintainers, curators, and downloaders are trusted and behave honestly. This means that maintainers provide the same distributed dataset index $\{(url_i, c_i)\}_{i=1}^{N}$ to any client and that the index itself has not been poisoned (e.g., due to insider risk or compromise). Downloaders for distributed datasets honestly access all $url_i$ and compute any integrity checks that we add via our defenses. Curators provide the same centralized dataset $\mathcal{D}$ to all clients, with curators controlling the time $t_i$ at which any element $url_i$ is snapshot. These assumptions mirror the existing trust that clients place in maintainers, curators, and downloaders and thus represent the quickest, short-term path to enacting defenses. We discuss limitations of these trust assumptions and more robust solutions with fewer trust assumptions in Section 6.5.

### 6.2 Preventing Split-View Poisoning

While the simplest defense to split-view poisonings attack would be to convert the distributed dataset $\{(url_i, c_i)\}_{i=1}^{N}$ into a centralized dataset (e.g., as in YFCC100M [71]), this is presently unrealistic due to the monetary, privacy, and legal challenges laid out in Section 3. Instead, maintainers—or another trusted third-party—can prevent split-view attacks by attaching a cryptographic hash $h_i = H(x_i)$ of the raw data $x_i$ obtained from $url_i$ at time $t_i$ prior to any attack. A downloader would then check whether $H(x_i') = h_i$, where $x_i'$ is the content of $url_i$ at time $t_i'$. The downloader discards any data where the client and maintainer receive distinct content. Here, $H$ should be a cryptographic hash function such as SHA-256.

**Implementation & Responsible Disclosure.** Enacting this defense requires a number of ecosystem changes. Currently, only PubFig and FaceScrub include cryptographic hashes as part of their distributed dataset (see Table 1). And because

these two datasets do not provide an official downloader, the community has relied on a number of third-party downloader scripts that for the most part do not actually verify these hashes.[13] Fortunately, for larger datasets the `img2dataset` [5] tool has become the canonical downloader used in 75% of requests to our domains.

As part of our responsible disclosure, we reached out to every maintainer (see Table 1), suggesting the addition of SHA-256 hashes as part of the dataset index. At the time of writing, CC3M, CC12M, LAION-2B-en, LAION-2b-multi, LAION-1B-nolang, and LAION-400M now release their dataset with SHA-256 hashes of the image contents. We additionally implemented an option in `img2dataset` to verify SHA-256 image hashes upon download, thus preventing our attack for anyone using this tool, and provide our own backup of hashes in a Google Cloud Bucket at `gs://gresearch/distributed-dataset-hashes/` for the datasets where we have (near-)original data.

**Limitations.** Integrity checks are viable if most benign content remains static. If content is altered in any way (e.g., by re-encoding, cropping, re-sizing, or uploading a higher-resolution image) the original hashes will no longer match. This can significantly degrade the utility of a dataset: for example, we obtain the original raw data from the first Conceptual Captions 3M dataset downloaded in 2018 and compare this to our most recent download of these same images in 2023. Of the 3.3 million original images, 2.9 million images are still hosted online, but just 1.1 million of these images have hashes that match the original—the other 1.8 million images have changed since the initial dataset construction.

This suggests that our defense, while providing perfect protection against split-view poisoning attacks, has the potential to significantly degrade utility. In Appendix C, we perform a case-study analysis on the PubFig and FaceScrub datasets, showing that modified but useful content makes up a significant fraction of the images with invalid hashes. Switching to a perceptual hash function (which aims for invariance to small image changes) would lead to higher utility, but would not meaningfully prevent our poisoning attacks because an attacker could upload poisoned images that were adversarially modified to fool the perceptual hash [26,29,68]. This suggests qualitatively new defense ideas will be necessary to defend against our attacks without a high utility cost.

## 6.3 Preventing Frontrunning Poisoning

Our frontrunning poisoning attack relies on the fact that an adversary only needs sustained control of data for a few minutes to succeed. To defend against this attack, it suffices to increase the duration $d = t_i - \hat{t}_i$ that an attacker must retain control over $url_i$ for it to be included in the snapshot at time $t_i$, where $\hat{t}_i$ indicates the time an attacker first modifies the URL's contents. If a curator can detect malicious modifications within time $\Delta$, then increasing $d > \Delta$ effectively thwarts the attack. This can be achieved in one of two ways: (1) curators can randomize the snapshot order of the $url_i$ and extend the time required to snapshot the complete corpus; or (2) curators could freeze edits to the content of $url_i$ at time $t_i$, wait for a period $T > \Delta$ for edits to go through review, and then finally release the snapshot at time $t_i + T$.

**Implementation & Responsible Disclosure.** For our first approach, Wikipedia could randomize its snapshot order of articles instead of its current sequential method based on article IDs. This thwarts an adversary's ability to predict precisely when an article will be selected for snapshotting, requiring they sustain control of articles for the entirety of the snapshot time, $t_n - t_0$, to ensure success. For the English Wikipedia, the current average review time to detect vandalism $\Delta$ is 2.5 hours (Figure 6). Increasing the snapshot time beyond $\Delta$ would protect 1 - $\Delta/(t_n - t_0)$ articles from random, malicious modification, or 89.5% of articles if snapshotting was uniformly randomized over 24 hours. This assumes an attacker is unable to use Sybil accounts to automatically reintroduce malicious edits after their first detection and reversion. If this assumption is invalid, this protection will be weaker in practice.

For our second approach which is more comprehensive, Wikipedia could create an initial snapshot of an an article, hold it for a period $T > \Delta$, and then back-apply ("cherry-pick") modifications or reversions from trusted moderators that occur within time $T$ before finalizing the snapshot. (Subsequent edits must be accepted from trusted moderators so as to avoid selective deletion or reversion by attackers.) Even a reasonable grace period of *one day* could have a significant impact on the number of malicious edits that will be caught. For example, on the English Wikipedia (Figure 6), increasing from a 5 minute to a 1 day window would increase the reversion rate from 50% to 90%, reducing vandalism by a factor of 5.

As part of our responsible disclosure, we notified Wikipedia of these attacks and our proposed defenses.

**Limitations.** In practice, these defenses make it more difficult for an attacker to operationalize frontrunning, but cannot prevent it entirely as $\Delta$ is not uniform across articles. For example, attackers might target less active articles, or languages with fewer moderators, in order to increase their frontrunning success rate. Furthermore, our defenses hinge on the existence of a trusted curator who can detect malicious edits—something that may be difficult if an attacker intentionally introduces imperceptible changes over time that impact only machine understanding, but appear valid to human review. Overcoming these risks—which exist for any "living" dataset—requires much more sophisticated solutions, which we explore next.

[13] We examine the 6 most popular downloader scripts for each dataset (gathered by searching for "`[pubfig|facescrub] dataset download github`"), and find that only one script per dataset implements hash verification. The one for FaceScrubchecks hashes by default, while the one for PubFig requires users to run a separate verification script.

### 6.4 Preventing Poisoning in General

Preventing poisoning attacks on more general web-scale datasets such as Common Crawl (a peta-byte sized dataset of web crawls) is more complex. No trusted "golden" snapshot exists here as we had for split-view poisoning. Nor is there a trusted curator who can detect malicious edits. Equally problematic, updates to a web page have no realistic bound on the delta between versions which might act as a signal for attaching trust. Ultimately, any notion of which domains to trust is ad-hoc at best.

Client could thus rely on consensus-based approaches (e.g., only trusting an image-caption pair if it appears on many different websites). An attacker would then have to poison a sufficiently larger number of similar websites to ensure success, which mirrors the same consensus challenges present in distributed systems like blockchains [45]. However, any solution in this space requires downstream knowledge of how URL content is consumed, vectorized, and deconflicted during training. We leave application-specific solutions to general poisoning to future work.

### 6.5 Transparency to Improve Trust

Web-scale datasets today hinge on implicit trust. Clients trust maintainers to distribute identical and accurate auxiliary data $c_i$, which may in fact be malicious due to a compromised maintainer. Clients trust curators to enact effective moderation to detect malicious edits to $x_i$. Clients trust downloaders to accurately retrieve $url_i$. And finally, clients trust websites to deliver the same $x_i$ for every $url_i$, even though attackers have countless mechanisms to subvert $x_i$—going beyond just purchasing expired domains or frontrunning snapshots.

We believe improving the safety of web-scale datasets requires introducing transparency into the ecosystem. Data transparency around the set of $\{(url_i, c_i, h_i)\}$ distributed to clients—akin to certificate transparency [37]—can prevent transient failures or a compromised maintainer from distributing different datasets to different clients and to assist in the detection and removal of inaccurate $c_i$ or expired $url_i$ over time. Curators could engage in a similar process to ensure all clients receive an identical corpus $\mathcal{D}$. While many downloaders are already open source, binary transparency would bolster protection to prevent selective inclusion of malicious modules [1]. Such transparency would prepare the ecosystem for a future where multiple maintainers and curators continuously update web-scale datasets, rather than the current reliance on centralized entities and static datasets.

## 7 Conclusion

Our paper demonstrates that web-scale datasets are vulnerable to low-cost and extremely practical poisoning attacks that could be carried out today. This is true even when attackers can target only a fraction of curated datasets, where corrupting 0.01% of examples is sufficient to poison a model. Those who publish and maintain datasets should consider the defenses we introduced—including integrity checks and randomized or time-gated snapshots—or alternate, application-specific defenses. In light of our findings, we argue that machine learning researchers must reassess the trust assumptions they place in web-scale data and begin exploring solutions that do not assume a single root of trust. Our findings also expose a variety of future directions for attack research: threat models where attackers can edit only raw content but not auxiliary data such as labels; assessing the practical costs of proposed attacks; and assessing the efficacy of more permissive, but potentially vulnerable near-duplicate integrity checks. As such, our work is only a starting point for the community to develop a better understanding of the risks involved in generating models from web-scale data.

## Acknowledgements

We are grateful to the dataset curators (in particular Beer Changpinyo, Saehoon Kim, Romain Beaumont, Ludwig Schmidt, and Chris Albon) for discussions around datasets and defenses. We are also grateful to Milad Nasr and Alex Kurakin for comments on early drafts of this paper.

## References

[1] Mustafa Al-Bassam and Sarah Meiklejohn. Contour: A practical system for binary transparency. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 94–110. Springer, 2018.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[3] Ankan Bansal, Carlos Castillo, Rajeev Ranjan, and Rama Chellappa. The do's and don'ts for CNN-based face verification. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 2545–2554, 2017.

[4] Ankan Bansal, Anirudh Nanduri, Carlos D Castillo, Rajeev Ranjan, and Rama Chellappa. UMDfaces: An annotated face dataset for training deep networks. In *2017 IEEE international joint conference on biometrics (IJCB)*, pages 464–473. IEEE, 2017.

[5] Romain Beaumont. img2dataset: Easily turn large sets of image urls to an image dataset. https://github.com/rom1504/img2dataset, 2021.

[6] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[7] Abeba Birhane, Vinay Uday Prabhu, and Emmanuel Kahembwe. Multimodal datasets: misogyny, pornography, and malignant stereotypes. *arXiv preprint arXiv:2110.01963*, 2021.

[8] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29, 2016.

[9] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] Minwoo Byeon, Beomhee Park, Haecheon Kim, Sungjun Lee, Woonhyuk Baek, and Saehoon Kim. COYO-700M: Image-text pair dataset. https://github.com/kakaobrain/coyo-dataset, 2022.

[12] Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 423–433, 2013.

[13] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. VGGFace2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018.

[14] Nicholas Carlini. Poisoning the unlabeled dataset of Semi-Supervised learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1577–1592, 2021.

[15] Nicholas Carlini and Andreas Terzis. Poisoning and backdooring contrastive learning. *arXiv preprint arXiv:2106.09667*, 2021.

[16] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12M: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3558–3568, 2021.

[17] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. VGGsound: A large-scale audio-visual dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 721–725. IEEE, 2020.

[18] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[19] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling language modeling with Pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[22] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and DBpedia. In *International semantic web conference*, pages 69–78. Springer, 2019.

[23] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[24] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[25] Mandy Guo, Zihang Dai, Denny Vrandečić, and Rami Al-Rfou. Wiki-40B: Multilingual language model dataset. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2440–2452, 2020.

[26] Qingying Hao, Licheng Luo, Steve TK Jan, and Gang Wang. It's not what it looks like: Manipulating perceptual hashing based applications. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[27] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[28] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. OpenCLIP, July 2021.

[29] Shubham Jain, Ana-Maria Creţu, and Yves-Alexandre de Montjoye. Adversarial detection avoidance attacks: Evaluating the robustness of perceptual hashing-based client-side scanning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2317–2334, 2022.

[30] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[31] Ira Kemelmacher-Shlizerman, Steven M Seitz, Daniel Miller, and Evan Brossard. The MegaFace benchmark: 1 million faces for recognition at scale. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4873–4882, 2016.

[32] Evan Klinger and David Starkweather. pHash: The open source perceptual hash library. https://phash.org/, 2013.

[33] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.

[34] Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. Attribute and simile classifiers for face verification. In *2009 IEEE 12th international conference on computer vision*, pages 365–372. IEEE, 2009.

[35] Tobias Lauinger, Ahmet S Buyukkayhan, Abdelberi Chaabane, William Robertson, and Engin Kirda. From deletion to re-registration in zero seconds: Domain registrar behaviour during the drop. In *Proceedings of the Internet Measurement Conference*, 2018.

[36] Tobias Lauinger, Abdelberi Chaabane, Ahmet Salih Buyukkayhan, Kaan Onarlioglu, and William Robertson. Game of registrars: An empirical analysis of post-expiration domain name takeovers. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 865–880, Vancouver, BC, August 2017. USENIX Association.

[37] Ben Laurie. Certificate transparency. *Communications of the ACM*, 57(10):40–46, 2014.

[38] Rémi Lebret, David Grangier, and Michael Auli. Generating text from structured data with application to the biography domain. *CoRR*, abs/1603.07771, 2016.

[39] Chaz Lever, Robert Walls, Yacin Nadji, David Dagon, Patrick McDaniel, and Manos Antonakakis. Domain-Z: 28 registrations later measuring the exploitation of residual trust in domains. In *2016 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2016.

[40] Alexandra Luccioni and Joseph Viviano. What's in the box? an analysis of undesirable content in the Common Crawl corpus. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 182–189, 2021.

[41] Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Using Large Corpora*, 273, 1994.

[42] Pablo Mendes, Max Jakob, and Christian Bizer. DBpedia: A multilingual cross-domain knowledge base. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1813–1817, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA).

[43] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[44] Tyler Moore and Richard Clayton. The ghosts of banking past: Empirical analysis of closed bank websites. In *International Conference on Financial Cryptography and Data Security*, pages 33–48. Springer, 2014.

[45] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

[46] Hong-Wei Ng and Stefan Winkler. A data-driven approach to cleaning large face datasets. In *2014 IEEE international conference on image processing (ICIP)*, pages 343–347. IEEE, 2014.

[47] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. You are what you include: large-scale evaluation of remote Javascript inclusions. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 736–747, 2012.

[48] OpenAI. Introducing Whisper. https://openai.com/blog/whisper/, 2022.

[49] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

[50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[51] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision.

[52] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[53] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[54] Javier Rando, Daniel Paleka, David Lindner, Lennart Heim, and Florian Tramèr. Red-teaming the Stable Diffusion safety filter. *arXiv preprint arXiv:2210.04610*, 2022.

[55] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

[56] Johann Schlamp, Josef Gustafsson, Matthias Wählisch, Thomas C Schmidt, and Georg Carle. The abandoned side of the Internet: Hijacking Internet resources when domain names expire. In *International Workshop on Traffic Monitoring and Analysis*, pages 188–201. Springer, 2015.

[57] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. LAION-5B: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[58] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. LAION-400M: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

[59] Christoph Schumann and Romain Beaumont. LAION-Aesthetics. https://web.archive.org/web/20230119181400/https://laion.ai/blog/laion-aesthetics/, 2022.

[60] Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. You autocomplete me: Poisoning vulnerabilities in neural code completion. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1559–1575, 2021.

[61] Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, and Armand Joulin. CCMatrix: Mining billions of high-quality parallel sentences on the web. *arXiv preprint arXiv:1911.04944*, 2019.

[62] Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*, 2016.

[63] Pnina Shachaf and Noriko Hara. Beyond vandalism: Wikipedia trolls. *Journal of Information Science*, 36(3):357–370, 2010.

[64] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31, 2018.

[65] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual Captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, 2018.

[66] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. Flava: A foundational language and vision alignment model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15638–15650, 2022.

[67] Johnny So, Najmeh Miramirkhani, Michael Ferdman, and Nick Nikiforakis. Domains do change their spots: Quantifying potential abuse of residual trust. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 2130–2144, 2022.

[68] Lukas Struppek, Dominik Hintersdorf, Daniel Neider, and Kristian Kersting. Learning to break deep perceptual hashing: The use case NeuralHash. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 58–69, 2022.

[69] Besiki Stvilia, Michael B Twidale, Linda C Smith, and Les Gasser. Information quality work organization in Wikipedia. *Journal of the American society for information science and technology*, 59(6):983–1001, 2008.

[70] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*, 2018.

[71] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[72] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

[73] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. LC-QuAD: A corpus for complex question answering over knowledge graphs. In *International Semantic Web Conference*, pages 210–218. Springer, 2017.

[74] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018.

[75] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers/blob/8178c840f265d4bee91fe9cf9fdd6dfef091a720/src/diffusers/pipelines/stable_diffusion/safety_checker.py, 2022. Accessed 7 Feb 2023.

[76] Eric Wallace, Tony Z Zhao, Shi Feng, and Sameer Singh. Concealed data poisoning attacks on NLP models. *arXiv preprint arXiv:2010.12563*, 2020.

[77] Angelina Wang, Alexander Liu, Ryan Zhang, Anat Kleiman, Leslie Kim, Dora Zhao, Iroha Shirai, Arvind Narayanan, and Olga Russakovsky. REVISE: A tool for measuring and mitigating bias in visual datasets. *International Journal of Computer Vision*, pages 1–21, 2022.

[78] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France, May 2020. European Language Resources Association.

[79] Wikipedia contributors. Reliability of wikipedia — Wikipedia, the free encyclopedia, 2022. [Online; accessed 21-July-2022].

[80] Wikipedia contributors. Wikipedia:Go ahead, vandalize, 2022. [Online; accessed 5-December-2022].

[81] Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2013–2018, 2015.

[82] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*, 2015.

[83] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

# A Additional Figures

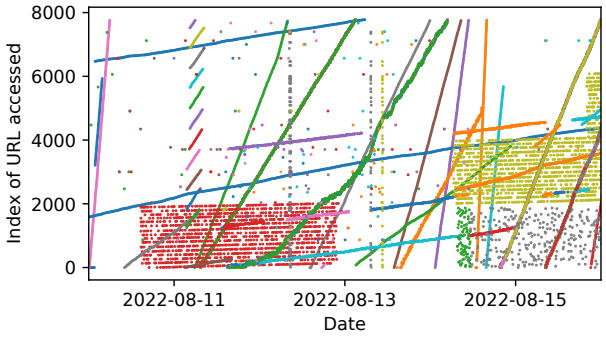


Figure 8: An unfiltered (without any precision or recall requirement) view of accesses to our server for URLs contained in Conceptual 12M. Compare to Figure 2 for the filtered view.

# B Further Discussion for Text Datasets

In this section, we further discuss vulnerabilities present in text datasets, focusing first on targeted poisoning attacks on Wikipedia, and then on the Common Crawl dataset.

## B.1 Annotating Reversions

In each language, we produce a list of words which are commonly used to denote reversions. The specific words used are emergent from each language's Wikipedia contributor community, so there is no concrete list. However, in each language, their are automated reversion tools and manual reversions which are tagged with the English word "reversion" or simply the abbreviation "rv". These form a starting point for our manual analysis: we identify words which roughly translate to "revert", "undo", or similar, which also appear in a sample of reversion comments we identify as automated reversions or manual reversions. We then sample comments with each of these newly identified words to verify that they capture new instances of reversions (that is, they are used to uniquely tag reversions in the language's Wikipedia), and do not produce too many false positives.

Overall, we don't expect this list to be perfect for any given language, as no author of this paper is an active contributor to any language's Wikipedia. However, we do believe our analysis is sufficient to validate the two trends we notice: frontrunning attacks are still possible on non-English Wikipedias, and attacks may be more powerful on non-English Wikipedias.

## B.2 Cascading Effects of Frontrunning

Our attack enables an adversary to poison any Wikipedia snapshot. An adversary can use frontrunning to harm downstream datasets that depend directly on these snapshots, as well. Rather than arbitrarily modifying large fractions of

Wikipedia, these can be achieved with very targeted frontrunning poisoning.

Knowledge base extraction databases are heavily relied upon for analytics (e.g., asking queries like "which politicians were gang members?") and machine learning (e.g., for training and evaluating question-answering models). One of the largest databases for it is DBPedia [2, 42]. Since this database is created using the monthly snapshots, they are, for all intents and purposes, a filtered view of Wikipedia. This enables adversaries to directly leverage our frontrunning attack to poison these databases as well in a more targeted manner.

As an example in analytics, a user may issue a query such as "which politicians were gang members?" A targeted poisoning attack could force individuals to be falsely included in such a list, with a small edit via a frontrunning attack. More broadly, such attacks could be designed to harm specific individuals or reduce the reliability of certain aggregate statistics.

Similarly, in question-answering tasks for machine learning, some datasets, like LC-Quad [22, 73], store data as question-query pairs. The adversary can also target these dataset by compromising the query with frontrunning (as above). Because Wikipedia snapshots are infrequent, and downstream systems like these knowledge bases may update their endpoints even more infrequently [2], these attacks can remain effective for multiple months. Another common approach is to store data directly as question-answer pairs [9, 12, 62, 70, 82]. This mitigates our attack above, assuming the specific checkpoint used to originally generate the dataset was not compromised. However, this only prevents poisoning of a model trained on these downstream tasks—a model trained on a poisoned Wikipedia snapshot and fine tuned on the downstream task will still be vulnerable. Finally, we remark that the impacts of these attacks may be exacerbated on multilingual data as we discussed in Section 5.1.

## B.3 Common Crawl

Common Crawl is a petabyte-scale corpus of web crawl data that is repeatedly captured on a roughly monthly basis. Each archive is a complete re-crawl of the internet that records the full activity, including all requests of the crawler and the host responses—with both HTTP headers and content. As such, each archive contains a static snapshot of all crawled pages at the time of visit. This may include new page content not seen during a previous crawl, and may exclude content that has become stale since the previous crawl. For example, data crawled during September 24 through October 8, 2022 contains 3.15 billion web pages with 380 TiB of uncompressed content from 34 million registered domains—1.3 billion URLs were not visited in any of the prior crawls.[14]

The Common Crawl dataset is vulnerable to an attack which is similar to both our frontrunning and split-view poisoning attacks. The adversary can purchase an expired domain which was previously contained in the Common Crawl, and it will be re-crawled with the adversary's choice of content, which will then appear in subsequent Common Crawl snapshots. Notice that, differently from the snapshot-poisoning attack on Wikipedia, there is **no** content moderation here and so the adversary simply needs to continue to control the domain to poison all future Common Crawl snapshots. Buying recently-expired domains that existed in previous Common Crawl snapshots allows a stronger form of attack where the attack can inject entirely new links into the crawl. This can be accomplished by adding links or subdomains to poisoned domains, and allowing the crawler to discover the new poisoned domains. Thus, an adversary may inject arbitrarily many pages into the Common Crawl dataset, not only from the originally expired subset. We do not implement this attack following our ethics statements outlined earlier.

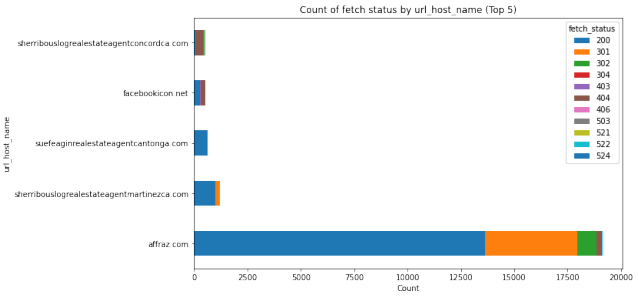


Figure 9: Top 5 domains by status count

Since Common Crawl WARC files have been hosted by Amazon on a AWS Athena (serverless service)[15], domain reconnaissance work to analyze URLs is inexpensive. Scanning through 10 years of Common Crawl data to analyze domains from popular TLDs and high number of Common Crawl entries cost us USD$ 0.84. While additional analysis might somewhat increase this cost, it remains an inexpensive way to search for vulnerable domains. Buying recently expired domains, or domains that have a dangling DNS record with an active IP address is preferred, as domains that failed to return a 200-OK status in consecutive crawls seem to be moved to a lower priority. For example, among expired domains we purchased, just one domain accounts for more than 90% of all status codes among the purchased domains, while other domains we purchased as early as 12/20/2020 have seen relatively less scraping traffic across a 3 year period.[16]

Because Common Crawl is enormous and uncurated (to accurately reflect the content of the internet) poisoning all of Common Crawl is impractical due to size. Additionally, it is

[14] https://commoncrawl.org/2022/10/sep-oct-2022-crawl-archive-now-available/

[15] https://commoncrawl.org/2018/03/index-to-warc-files-and-urls-in-columnar-format/

[16] The domains used in this study responded with a 404 HTTP status to explicitly prevent scraping of content, potentially affecting analysis of domain re-emergence. That is, domains that previously returned a 404 now return a 200.

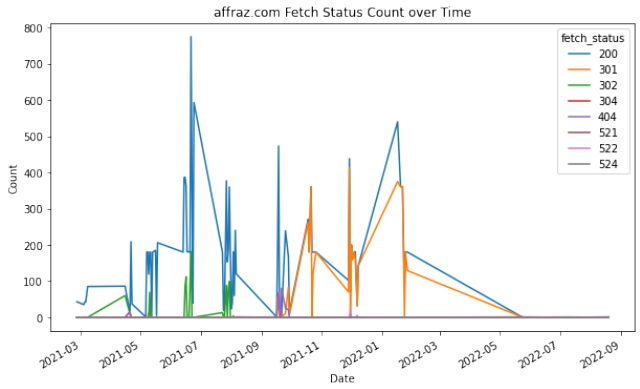


Figure 10: affraz.com status counts over time

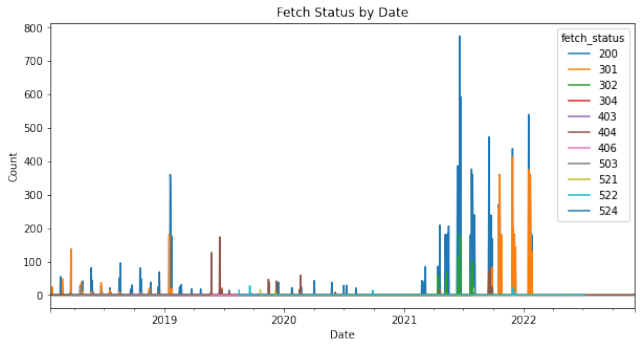


Figure 11: Aggregate status counts for all purchased domains

not always apparent how consumers of this data are processing it for downstream machine learning tasks. However, there exist many derivative datasets which are constructed by curating a relevant subset of the Common Crawl. This includes the LAION-5B image dataset [57], the text dataset known as the Pile [23], the multilingual text dataset CC-100 [78], and the CCMatrix dataset [61], a translation dataset of pairs of translated sentences. Such curation actually amplifies the power of an attack: an attack which adds 1MB of text to the Common Crawl would be poisoning a $2.5 \cdot 10^{-9}$ fraction of the Common Crawl, but if this text bypasses the curation done for the CC-100 dataset, it could instead poison a $1.2 \cdot 10^{-5}$ fraction of the English corpus, or even a full 9.1% of the Oromo corpus.

# C Limitations of Integrity Check Defenses

In Section 6.2, we outlined a natural defense against split-view poisoning that adds integrity checks to the dataset index. Specifically, upon collecting the dataset the maintainer computes a cryptographic hash of each image, and adds this hash to the index. Upon downloading a copy of the dataset, the client downloader then discards images for which the hash no longer matches. Unfortunately, this defense has a severe

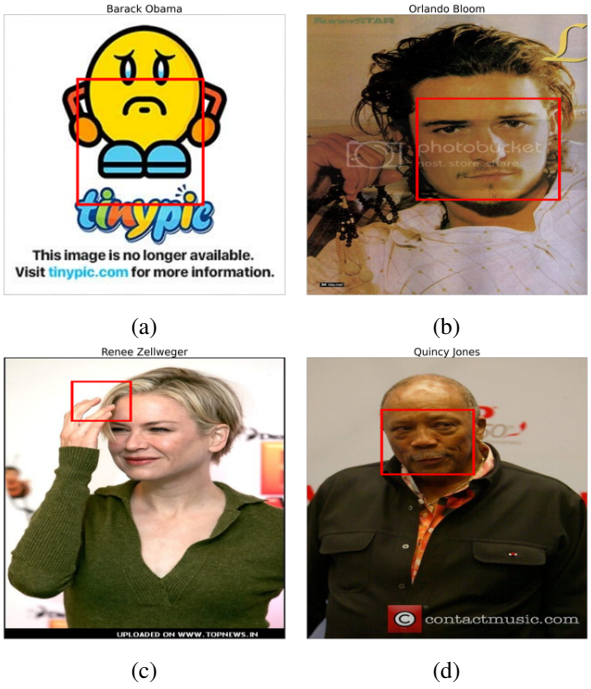


Figure 12: What happens in practice when the image no longer matches the original hash? We show 4 examples from the PubFig dataset. In Figure 12a, the image is replaced by a placeholder. In Figure 12b, a watermark partially covers the face, making that image slightly less useful for training. In Figure 12c, the image was resized and thus the original bounding box contained in the dataset index no longer matches the face. As opposed to the previous three images, Figure 12d is perfectly suitable for training: the watermark is outside the facial bounding box.

impact on utility. As we saw in Section 6.2, implementing this defense for the Conceptual Captions 3M dataset would result in discarding roughly 55% of the entire dataset—mainly because of small yet benign image changes such as re-sizing.

Here, we perform a more in-depth analysis of the impact of cryptographic integrity protections on other datasets.

## C.1 Modified Images in the Wild: A Case Study on the PubFig Dataset

PubFig [34] is a 2010 dataset that indexes close to 60,000 images of 200 different celebrities hosted across a variety of domains. PubFig is one of the oldest example of a distributed dataset. Thus, it is likely that many of the URLs in its index are no longer live (a phenomenon we refer to as "link rot"), and that many of the remaining URLs would point to images that have been slightly modified over time. Small image modifications make integrity check defenses problematic because they discard many perfectly useful training images. In Figure 12, we show four different ways in which original images have

| Dataset | # Images | Success | Download failure | Invalid image |
|---|---|---|---|---|
| pubfig | 58795 | 35.2% | 54.7% | 10.2% |
| facescrub | 106863 | 48.5% | 44.5% | 7.0% |

Table 2: Link rot in old image datasets, downloaded using `img2dataset` [5]. In older datasets, the proportion of images that are no longer available online is high.

| Dataset | Downloaded | Modified | Accuracy on modified, no crop | Accuracy on modified, crop | Accuracy on original, no crop | Accuracy on original, crop |
|---|---|---|---|---|---|---|
| pubfig | 20668 | 52.4% | 59.7% | 55.2% | 96.0% | 98.2% |
| facescrub | 51847 | 29.4% | 93.3% | 91.6% | 97.7% | 99.1% |

Table 3: Quantifying changes on successfully downloaded images. Modified images are those that have a different hash than the original image. We measure the accuracy as the proportion of modified images that have a CLIP embedding close to *any* of the possible labels, approximating the question "Is there a recognizable face in the image?". The difference between cropped and uncropped images is largely due to the resized images making the crop not capture the face of the person, as in Figure 12c.

been modified in PubFig. While we find examples of true-positives, where images were indeed modified significantly, there are also many cases of images that were subjected to minor changes such as the addition of a watermark.

Many image datasets provide bounding boxes for a relevant part of the image. This is also the case for PubFig: the dataset index contains the coordinates of a bounding box for the person's face. If these bounding boxes are not re-computed by the client, any image resizing, such as in Figure 12c, can render the cropped image useless for training.

## C.2 Link Rot Statistics

Integrity checks based on cryptographic hashing reduces the amount of data available in some datasets by a large amount. We download the PubFig and FaceScrub datasets and compute the prevalence of "link rot", where the URL listed in the dataset index no longer resolves or returns a non-valid image.

In addition to dead links, we find that many successfully downloaded images have been modified. We show aggregate statistics in Table 3. In the PubFig dataset, over 50% of the surviving images have a hash mismatch. However, many of those images are modified in harmless ways. To show this, we take a pre-trained face embedding model and fine-tune a classifier for the PubFig classes using only images with a correct hash. We then evaluate this classifier on the downloaded images with an incorrect hash, and find that we only achieve 46.8% accuracy. Thus, many of these images were modified in ways that obfuscate the identity of the person. For FaceScrub, we find that fewer images have been modified. About 70% of successfully downloaded images still match the original hash. Of the images that were modified, we can still classify 88.2% correctly, which indicates that most of these changes are benign as illustrated in Figure 12b or Figure 12d.

For PubFig, we further investigate how frequently an image change corresponds to a benign resizing, which nevertheless renders the original bounding box obsolete (as in Figure 12c). For this, we compare the CLIP (`ViT-B-32-quickgelu` from [28]) embeddings of the *full image* to the names of the 200 public figures in the dataset, with a threshold cosine similarity of 0.21. We find that 59.4% of images with incorrect hashes are not close to any of the labels in CLIP space. Comparing with Table 3, this corresponds to around 4% of incorrect hashes being a result of modifications similar to Figure 12c.

Perceptual hashing or similar methods might alleviate this problem, because images are often modified in trivial ways. For example, as mentioned in Table 1, COYO-700M images are distributed with pHash [32] which is robust to benign image changes. However, perceptual hashes do not have the same *worst-case* integrity guarantees as cryptographic hashes [29]. There have been high-profile controversies due to the mistaken use of perceptual hashing as a preimage resistant algorithm [68]. It is widely believed that preimage attacks on SHA-256 are impractical, while there is no known perceptual hash function which has similar security guarantees.

# D LAION Attack Details

Both attack methods in Section 4.5 poison a CLIP model so as to bring the embeddings of some fixed images close to the embeddings of target textual labels. The key technical constraint in our experiment is that all attacks need to be done in parallel to minimize costs, as retraining CLIP is quite expensive.

**Object-misclassification objective.** The ImageNet dataset [20] contains 1000 classes of images. The *CLIP zero-shot classifier* on ImageNet returns the class label whose text embedding has maximum cosine similarity to the image embedding in CLIP latent space, across all ImageNet classes.

The goal of our poisoning attack is for the CLIP zero-shot classifier to classify a particular image to a target incorrect label.

We pick 10 classes as target labels for poisoning, such that captions containing the label appear at least 1000 times in the captions linked to cheap buyable domains; see Table 1. We do the following for each chosen label, e.g. *apple*: choose a set $S_{apple}$ of 1000 caption-image pairs from buyable domains such that *apple* appears in the captions. We also enforce that the total cost of domains spanning $S_{class}$ for all chosen classes is at most $1,000 USD. Then, we pick a single unrelated image $I$—that wouldn't ordinarily be classified as *apple*—and locally replace 1000 images from $S_{apple}$ with image $I$. Thus for each of the 10 classes, we poison 1000 images, or only 0.000025% of the data.

**NSFW objective.** The goal of this attack is to make the NSFW filter that comes with the Stable Diffusion 1.4 model in the Hugging Face diffusers library [75] mislabel a given benign image as NSFW. The classifier is a cosine similarity threshold function in CLIP latent space, comparing an image embedding to a list of textual NSFW concepts [54].

We choose 10 benign images, and do the following for each image $I$: choose 1000 caption-image pairs from buyable domains such that the captions are labeled UNSAFE in the LAION 400M metadata, and locally replace each of the corresponding 1000 images with $I$. Again we choose images from domains that cost less than $1,000 USD in total.

**The experiment.** For both attacks (and all chosen images) simultaneously, we train an OpenCLIP [28] model on the the LAION 400M dataset with the described modifications. We train a ViT-B-32 CLIP model for 32 epochs, at a batch size of 3072 on 16 A100 GPUs. The object-misclassification attack works for 60% of the targets: the chosen image gets classified as the target label by a zero-shot CLIP classifier. The NSFW attack works for 90% of targeted images.

# E Landing Page for Purchased Domains

The following text was placed on the landing page for each of the domains we purchased.

> This domain is part of a research study. This domain name was purchased as part of a research project studying to what extent machine learning datasets change over time. This domain name was included in one of these datasets and hosted images that were part of this dataset, but the previous owner let the domain name expire. We bought this domain in August 2022 to measure the number of people who query from these expired domains.
>
> We bought a number of domains that were included in many different datasets. All of these domains will return a 404 error for any requests except for the home page. You should not need to take any additional steps to ensure your datasets are unaffected by our study: if we had not bought this domain the URL would have been NXDOMAIN and you would have not received any content.
>
> We may temporarily log metadata for requests sent to this server to measure the prevalence of scraping this domain. Any data we have logged will be deleted upon completion of our study. If you would prefer not to participate in this study, please contact us via the email address below and we will delete any data you may have contributed to our study. We would really appreciate it if you let us use your data; we think this will be a valuable study.
>
> If you have any questions about this study you can contact dataset-expired-domain-study@googlegroups.com for additional information.
>
> *This used to be my domain. Can I have it back?* If you are the original owner of this domain, we would be happy to return it to you at your request to the above email address. We will let this domain expire when we have finished our research study.
>
> *Will this cause problems with my downloaded dataset?* As we say above, if you are scraping this dataset, you should not need to take any steps to specifically avoid this domain (or any other we have purchased). We have tested that the 404 response we send will cause all standard image downloading tools to skip the image entirely.
>
> *Will your study be published?* We will publish our study upon its completion. We expect this to occur within the next several months. Please contact us if you would like a copy of this study.
>
> *I have another question not mentioned.* Please contact us at dataset-expired-domain-study@googlegroups.com for additional information.