A Quantitative Study of Video Duplicate Levels in YouTube

Yao Liu[1], Sam Blasiak[2], Weijun Xiao[3], Zhenhua Li[4], and Songqing Chen[2]

[1]SUNY Binghamton, yaoliu@cs.binghamton.edu
[2]George Mason University, {sblasiak,sqchen}@cs.gmu.edu
[3]Virginia Commonwealth University, wxiao@vcu.edu
[4]Tsinghua University, lizhenhua1983@tsinghua.edu.cn

Abstract. The popularity of video sharing services has increased exponentially in recent years, but this popularity is accompanied by challenges associated with the tremendous scale of user bases and massive amounts of video data. A known inefficiency of video sharing services with user-uploaded content is widespread video duplication. These duplicate videos are often of different aspect ratios, can contain overlays or additional borders, or can be excerpted from a longer, original video, and thus can be difficult to detect. The proliferation of duplicate videos can have an impact at many levels, and accurate assessment of duplicate levels is a critical step toward mitigating their effects on both video sharing services and network infrastructure.

In this work, we combine video sampling methods, automated video comparison techniques, and manual validation to estimate duplicate levels within large collections of videos. The combined strategies yield a 31.7% estimated video duplicate ratio across all YouTube videos, with 24.0% storage occupied by duplicates. These high duplicate ratios motivate the need for further examination of the systems-level tradeoffs associated with video deduplication versus storing large number of duplicates.

1 Introduction

User generated video content has exponentially increased in the recent years. For example, *YouTube*, *Dailymotion*, and *Vimeo* are among the most popular websites for uploading and sharing user generated content (UGC). YouTube alone has gained massive popularity: it attracts more than 1 billion users every month, more than 100 hours of uploaded video each minute, and more than 1 million creators make money from videos that they have uploaded [3]. We estimate that there are more than 849 million videos on YouTube (Section 5.2). According to Sandvine, YouTube generates 13.19% of all downstream fixed access traffic (e.g., cable network) and 17.61% of all downstream mobile data traffic in North America during peak hours [2].

Unlike video on-demand service providers such as Netflix, which contracts with a limited number video providers, UGC websites attract large numbers of video uploaders. This high diversity of uploaders poses a unique challenge for these UGC video sharing websites: Videos can be uploaded in different incarnations by different users, leading to duplicates in the video database. While duplicates that occur at the exact byte-level can be captured by the video sharing service using cryptographic hashes, user-generated (near-)duplicate videos are often uploaded in different encodings, have different aspect ratios, can contain overlays or additional borders, or could be excerpted from a longer, original video. As a result, they are assigned their own unique IDs in the video database. Note that duplicates should not be confused with multiple transcoded versions generated by a video sharing service to support streaming at different bandwidths and to different devices. These transcoded versions are associated with a same video ID in the video database.

The proliferation of duplicate videos could impact many aspects of datacenter and network operations and, as a result, have negative effects on the user experience. From the video server's perspective, duplicate videos could increase data storage, power, and therefore overall costs of data center operations. Further, duplicate videos have the potential to harm caching systems, degrading cache efficiency by taking up space that could be used for unique content and increasing the amount of data that must be sent over the network to in-network caching systems. These inefficiencies could be passed on to the user in the form of duplicated search results, longer startup delays, and interrupted streaming [17].

Although it is well known that duplication occurs in today's UGC video sharing websites, little is known about its precise level. Work to more-accurately determine duplicate levels is necessary because, although deduplication procedures can improve the overall efficiency of a video sharing system, deduplication itself could also be costly. Quantifying the level of duplication is therefore critical for determining whether effort to deduplicate, or otherwise mitigate the effect of duplicates, would be worthwhile.

As YouTube is the largest UGC video system today, we choose it as representative of similar services, and measure its level of duplication. In the process of conducting these measurements, we make the following contributions:

- We employ a novel combination of video sampling methods, automated video comparison techniques, and manual validation to estimate duplication levels in large-scale video sharing services.
- Using these methods, we estimate that the duplicate ratio of YouTube videos is 31.7% and that 24.0% of YouTube's total video storage space is occupied by duplicates.

The remainder of the paper is organized as follows. Section 2 and Section 3 discuss the motivation of this study and related work, respectively. Section 4 describes our duplicate estimation technique. We report our results in Section 5. Finally, Section 6 concludes this work.

2 Motivation

Anyone who has watched videos on YouTube, or any other video sharing service, has certainly noticed that near-duplicates of the same video often appear in the search results or are recommended as related videos. These impressions, however, are not useful toward making recommendations for taking action to mitigate any potential efficiency loss resulting from unnecessary duplication.

In preliminary work, we performed a small-scale assessment of 50 queries for the titles of 50 popular YouTube videos from a randomly selected set (Section 4.1). Manual assessment of these videos produced a rough estimate of a 42% duplicate ratio.

Viewing a small number of individual search results, however, is unlikely to yield good estimates of the prevalence of duplicates across a video sharing service's entire database. The huge number of videos stored within services such as YouTube also indicates that manually comparing videos to estimate duplicate ratio is infeasible. This intractability motivates the need for a larger scale assessment, assist in determining the necessity of and formulating further systems to conduct video deduplication.

3 Related Work

Data deduplication. Data duplication is common in storage systems. Deduplication operates by detecting duplicates and storing only a single copy of a given chunk of data. It is typically conducted on exact byte-level duplicates [4, 6, 10, 18, 23]. Detecting exact duplicates is often performed using cryptographic-hash based approaches (e.g., SHA1) to create an index for fast lookups. These cryptographic hash-based approaches, however, are inappropriate for detecting near-duplicate videos (i.e., videos that appear the same or very similar to a human viewer). This unsuitability is due to the fact that video files almost always contain significant differences at the byte-level even though the visual content of a video may be replicated (due to changes in encoding, altered resolutions, image-level editing, or temporal editing).

Near-duplicate video detection. The computer vision community has proposed a variety of strategies for detecting near-duplicate videos [5, 9, 19]. Two main types of tools have been developed. The first is the local image descriptor [13, 14, 21], which describes small sections within an image/keyframe. The second is the global descriptor [7, 16, 20, 22], which can be used to summarize the entire contents of an image or video. An approach for video duplicate detection that can employ either local or global descriptors is called Dynamic Time Warping (DTW) [15]. DTW is a technique used to measure distance between two sequences where a distance can be defined between sequence elements. DTW operates by aligning elements from a pair of sequences, A and B. Specifically, DTW aligns each element from sequence A to a similar element in sequence B with the constraint that no changes in ordering can occur (see Figure 1).

Video deduplication. The rapid growth of video content on the Internet and its corresponding storage cost have recently drawn much attention to the task of video deduplication. For example, Kathpal et al. found that multiple copies (versions) of the same video in different encodings and formats frequently exist. The authors proposed to save space by conducting on-the-fly transcoding to only retain the copy with the highest quality [11]. Shen and Akella proposed a video-centric proxy cache, iProxy. iProxy stores the frequency domain information of the video's key frames in an Information-Bound Reference (IBR) table. iProxy improves the cache hit rate by mapping videos with the same IBR to a single cache entry and dynamically transcodes the video during playback [17]. However, both works [11, 17] only deal with duplicates introduced by a limited set of transformations, e.g., quantization, resizing, and different formats and encodings. Other forms of transformation, such as excerption, concatenation, and splicing, would not be detected or deduplicated. Katiyar and Weissman proposed ViDeDup, which uses clustering-based "similarity detection" and performs deduplication by storing the centroid-videos with the highest perceptual-quality [12]. However, since only the centroid of a set of "similar" videos are stored, restored video may no longer represent the original visual content.

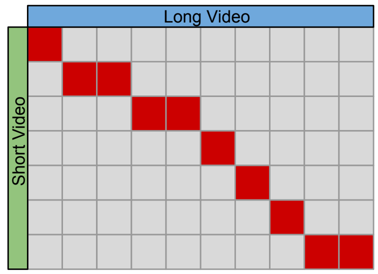

Fig. 1: The standard Dynamic Time Warping algorithm aligns all frames of both videos. Red squares represent aligned video frames.

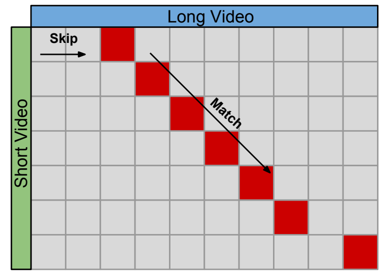

Fig. 2: The modified version of DTW used in this study aligns all elements from the shorter video and can skip frames from the longer video.

4 Methodology

Our set of techniques for video duplicate assessment are applied in the following steps:

Step 1: We use random prefix sampling [8] to sample YouTube videos uniformly and at random. We refer to this set of sampled videos as `sampled videos`.

Step 2: We then search for the title of each `sampled video` using the text-based search engine of YouTube, which returns a list of relevant videos. We refer to these relevant videos as `searched videos`. These `searched videos` are used as a candidate set of duplicates.

Step 3: For each (`sampled video`, `searched video`) pair, we calculate a similarity score which accounts for temporally shifted frames. This score is used to determine whether the `searched video` is a duplicate of the `sampled video`.

Step 4: For each pair of duplicates whose score is below a threshold, we conduct a manual comparison step to eliminate false positives.

In the rest of this section, we explain each step of our technique in detail for assessing duplicate levels in YouTube.

4.1 Random Sampling of Videos

In order to uniformly sample YouTube videos, we use the random prefix sampling method proposed by Zhou et al. [8]. Random prefix sampling involves querying the YouTube search engine with a randomly selected video ID (VID) prefix. The returned query results are existing videos whose VIDs match this random prefix. According to Zhou et al., with a prefix length of five ("–" being the last/fifth symbol in the prefix), all existing VIDs that match the prefix can be returned in one query. Therefore, during the sampling procedure, we randomly generate a fixed number, N_{prefix}, 5-character long prefixes. (In this work, we set N_{prefix} to 1,000.) In the remainder of the paper, we refer to the videos selected by random prefix sampling as `sampled videos`. We make the important assumption that the set of `sampled videos` contains no duplicates. We validate this assumption through both theoretical and experimental analysis in Section 5.2.

4.2 Selection of Candidate Duplicate Pairs

The next step involves pruning the number of video pairs that must be assessed with a computationally intensive duplicate detection method. We perform this pruning step by leveraging the metadata-based search engines provided by many video sharing services. In UGC sites, metadata can be an especially good source for retrieving duplicates because uploaders of these duplicates are incentivised to label their videos with metadata to indicate similarity to original popular content, thereby attracting a larger number of views.

We extract each `sampled video's` title and use it to query the YouTube search engine. This query returns a collection of videos with metadata related to the `sampled video's` title. Because this set of videos may still be too large to effectively process with DTW, we rely on the ranking capability of YouTube's metadata-based search engine to further filter videos. In particular, we record the top 100 results from each query. Some queries only return fewer than 100 results, and on average, we collected 82 `searched videos` for each `sampled video`. We refer to this set of videos returned from this search procedure as `searched videos`. Pairs of `sampled videos` and `searched videos` are sent to our DTW-based algorithm for duplicate assessment.

4.3 Comparing Sampled and Searched Video Pairs

For comparison, we download both the `sampled video` and `searched video` files from YouTube. YouTube usually encodes videos into multiple versions using different codecs, resolutions, and quantization levels to support streaming at different bandwidths and to different devices. We retrieve only the H.264 Baseline/AAC/MP4/360p version as we find this version is most often available.

After retrieving a set of `searched videos` associated with every `sampled video`, we use FFmpeg [1] to extract images/frames from the video at **one second intervals**. Note that we cannot use keyframes (i.e., I-frames) for comparison, as in related work [20], because the interval between keyframes can vary between videos. To detect pairs of duplicates, we employ a method based on Dynamic Time Warping (DTW) [15]. Like DTW, our duplicate matching system attempts to align frames from pairs of videos. However, we expect shorter videos to align to sub-portions of a longer video. We therefore modified the basic DTW algorithm so that every element of the shorter video must be matched while some elements of the longer video are allowed to remain unmatched (see Figure 2).

Our variation of DTW operates on pairs of image sequences

$$
\begin{aligned}
A &= a_0,\ a_1,\ \cdots,\ a_i,\ \cdots,\ a_{I-1} \\
B &= b_0,\ b_1,\ \cdots,\ b_j,\ \cdots,\ b_{J-1}
\end{aligned} \tag{1}
$$

where I and J indicate the number of images in sequence A and B, correspondingly, and we enforce $I >= J$ by swapping videos if necessary.

Figure 3 shows the pseudocode of our adapted DTW algorithm. We use the output of the function $result$ to indicate the DTW score between A and B. The smaller the DTW score, the more likely that videos A and B are duplicates.

▷ A, B, I, J are defined in Equation 1.
▷ $d(a_x, b_y)$ denotes distance between image a_x from sequence A and image b_y from sequence B.
function DTW(A, B)
    assert $I >= J$
    $dtw_{0,0} = d(a_0, b_0)$
    for $(i = 1; i < I; i++)$ **do**
        $dtw_{i,0} = d(a_i, b_0)$
    for $(j = 1; j < J; j++)$ **do**
        $dtw_{j,j} = dtw_{j-1,j-1} + d(a_j, b_j)$
    for $(i = 1; i < I; i++)$ **do**
        for $(j = 1; j < min(i, J); j++)$ **do**
            $dtw_{i,j} = dtw_{i-1,j}$
            $dist = d(a_i, b_j)$
            if $(dtw_{i-1,j-1} + dist < dtw_{i,j})$ **then**
                $dtw_{i,j} = dtw_{i-1,j-1} + dist$
    return $result = dtw_{I-1,J-1}/J$

Fig. 3: The DTW algorithm between two image sequences (i.e., videos) A and B.

A key component in the DTW algorithm is the image distance function $d(a_x, b_y)$. Choosing a good image distance function is vital to the accuracy of our duplicate detection method. We used a distance function between image histograms, denoted by $d_h(a_x, b_y)$.

Histogram Distance refers to a distance measurement between images based on the relative frequency of different colors [7, 20]. For each image, x, we calculate its color histogram $H_x = (h_x^1, h_x^2, \cdots, h_x^M)$. (The color histogram is a global image descriptor.) We consider images in HSV (Hue, Saturation, and Value) color space because the HSV is more perceptually relevant to human eyes than the RGB representation. The color histogram contains 5 bins for Hue, 8 bins for Saturation, and 16 bins for Value. The total number of bins, M, is therefore $M = 29$. The Value section of the histogram contains a greater number of histogram bins, reflecting the fact that human eyes are more sensitive to light/dark contrast than to color information. As black borders/banners may be introduced during video transcoding and affect our histogram distance metric, we ignore pixels whose $Value = 0$ (black) when calculating histograms.

The `Histogram Distance` between two images x and y is calculated as the squared Euclidean distance between H_x and H_y:

$$d_h(x, y) = \sum_{k=1}^{M} (h_x^k - h_y^k)^2$$

This distance metric can be used to determine if a pair of videos are visually similar. We consider a pair of videos duplicates if their `Histogram Distance` is less than 0.013. This threshold was chosen by calibrating against a precision-recall graph to give a perfect recall (zero false-negative rate) on a set of 100 pairs of videos.

4.4 Manual validation of duplicate pairs

In the pairwise comparison step described above, we deliberately selected a high DTW score threshold to achieve a high recall rate. This high threshold, however, can produce a correspondingly high false discovery rate.

To alleviate this potential problem, we augmented our automated procedure with a **manual duplicate verification** step that has false positive rate near zero. In the manual verification step, for each duplicate pair, a human observer manually viewed both the corresponding `sampled video` and `searched video` to determine if the pair was a true duplicate. Specifically, the human observer considered the `searched video` to be a duplicate of the `sampled video` under any of the following cases:

1. the `searched video` has the same video content as the `sampled video`.
2. the `searched video` is part of the `sampled video`.
3. the `sampled video` is part of the `searched video`.

5 Quantifying Video Duplicate Levels in YouTube

To estimate the number of duplicates in YouTube, we first randomly generated 1,000 prefixes. Using these 1,000 prefixes, we collected $6,365$ `sampled videos` and $512,314$ associated `searched videos`. For each `searched video` returned by the YouTube search engine, we ran the variation of the DTW algorithm discussed in Section 4 to produce a *similarity score*. We set the threshold for duplicate determination high (as discussed in Section 4.3) to produce a low rate of false negatives (high recall), then conducted a manual curation step to validate that each candidate pair returned by the DTW algorithm constituted a true pair of duplicates.

Although the manual validation is time and resource intensive, this step was feasible because only the relatively small number of pairs of videos marked as duplicates by the DTW step were manually assessed.

5.1 Results

We present numeric results in Table 1. Out of the $6,365$ `sampled videos`, our assessment shows that 631 (10%) have duplicates within YouTube. Assuming that the $6,365$ `sampled videos` were drawn independently and the counts of videos with duplicates and videos with non-duplicates were drawn from a binomial distribution, we can compute a confidence interval around the probability that a `sampled video` has a duplicate using the Beta quantile function. The 95% confidence interval around this probability is $(0.0912, 0.1065)$. On average, for each `sampled video` associated with one or more duplicates, 4.69 duplicates were discovered. Figure 4 shows the distribution of the number of duplicates for each `sampled video` with one or more duplicates.

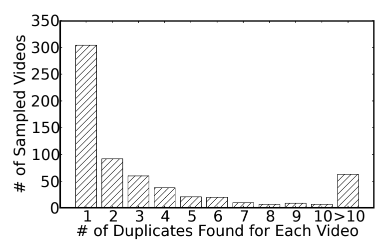

Fig. 4: # of duplicates found for each sampled video with one or more duplicates.

Table 1: Manually augmented assessment of YouTube duplicate levels.

Category	# of sampled videos	Sampled videos that have dups #	Sampled videos that have dups %	# of dups found per category	Avg. # of dups found for sampled videos that have duplicates	Duplicate ratio (%)
Video Category						
Pets & Animals	155	7	4.5%	15	2.14	8.8%
Autos & Vehicles	232	27	11.6%	147	5.44	38.8%
Comedy	462	33	7.1%	169	5.12	26.8%
Education	183	25	13.7%	53	2.12	22.5%
Entertainment	851	59	6.9%	240	4.07	22.0%
Film & Animation	244	29	11.9%	76	2.62	23.8%
Gaming	588	33	5.6%	196	5.94	25.0%
Howto & Style	119	11	9.2%	29	2.64	19.6%
Music	1068	146	13.7%	642	4.40	37.5%
News & Politics	220	42	19.1%	203	4.83	48.0%
Nonprofit & Activism	84	13	15.5%	143	11.00	63.0%
People & Blogs	1477	156	10.6%	767	4.92	34.2%
Shows	7	0	0%	0	0.00	0.0%
Sports	392	32	8.2%	179	5.59	31.3%
Science & Tech	113	9	8.0%	92	10.22	44.9%
Travel & Events	170	9	5.3%	9	1.00	5.0%
Video Duration						
Short $[0, 240)$	4490	418	9.3%	2310	5.53	34.0%
Medium $[240, 1200]$	1743	190	10.9%	596	3.14	25.5%
Long $(1200, \infty)$	132	23	17.4%	54	2.35	29.0%
Video Popularity						
Unpopular (< 1000)	5529	513	9.3%	2537	4.95	31.5%
Popular (≥ 1000)	836	118	14.1%	423	3.58	33.6%
Total						
Total	**6365**	**631**	**10.0%**	**2960**	**4.69**	**31.7%**

Out of 631 videos that have duplicates, 304 have only one duplicate found and 63 have more than 10 duplicates found, indicating the high variance of duplicate levels within YouTube. In total, our manually augmented evaluation found $2,960$ duplicates of the $6,365$ `sampled videos`. Assuming that the number of duplicates associated with each video is drawn from a normal distribution with a standard deviation of 3.38 (the empirical standard deviation), we compute a 95% confidence interval of $(0.382, 0.548)$ around the average number of duplicates for each video. These measurements indicate that roughly $1/3$ of videos on YouTube are duplicates. Of the $2,960$ duplicate videos found, only 327 (11%) have the same byte-level content as the `sampled video`, indicating traditional cryptographic hash-based duplicate detection has only a limited ability to detect duplicate videos.

Table 1 also shows a breakdown of `sampled videos` according to three attributes along the rows: video category, video length, and popularity. The columns of Table 1 give different duplicate statistics. Here "duplicate ratio" is defined as:

$$\text{duplicate ratio} = \frac{\text{\# of duplicates found}}{\text{\# of sampled videos} + \text{\# of duplicates found}} \tag{2}$$

Figure 5 shows the view count (i.e., popularity) of $6,365$ `sampled videos`. $5,529$ (87%) videos are viewed fewer than 1,000 times (`unpopular`). This statistic is consistent with the findings in Zhou et al. [8] that only 14% of videos in a randomly sampled YouTube dataset have a total view count of more than 1,000.

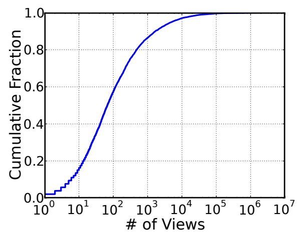

Fig. 5: View Count Distribution

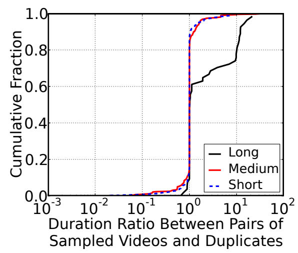

Fig. 6: $\frac{\textbf{Duration(sampled video)}}{\textbf{Duration(duplicate)}}$

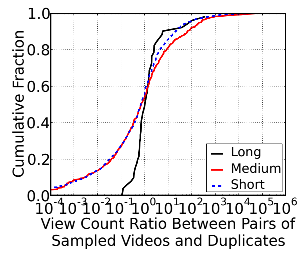

Fig. 7: $\frac{\textbf{ViewCount(sampled video)}}{\textbf{ViewCount(duplicate)}}$

Figure 6 shows the ratio between the duration of the `sampled video` and the duration of the detected duplicate video. As shown in the figure, most duplicates have the same duration as the sampled video. For `Short`, `Medium`, and `Long` videos respectively, $1,743$ out of $2,310$ (75%), 375 out of 596 (63%), and 19 out of 54 (35%) have the same durations as the `sampled video`. For `Long` videos, more than 40% of their duplicates are shorter than the `sampled video`. These shorter duplicates are excerpts from longer videos, extracted by users to meet the video duration limits imposed by YouTube. Overall, among all the duplicates found, **72%** have the same duration as the `sampled video`, indicating that excerption occurs less frequently than operations that preserve the length of the video.

We are also interested in determining if sets of duplicates have similar popularities. Figure 7 shows the view count ratios of `sampled videos` versus those of `searched videos`. Approximately 55% of the `searched video` duplicates are watched more frequently than the `sampled video`. These differing frequencies indicate that even if duplicates represent the same or similar visual content, the popularities of individual copies of the same video can vary.

5.2 Uniqueness of Sampled Videos

Given that our duplicate assessment found that approximately one-third of the videos in the YouTube database are duplicates, it seems counter-intuitive that our original assumption holds that each of our $6,365$ `sampled videos` is unique. A relatively short analysis, however, shows that this is a reasonable assumption. This analysis is a specialization of the well-known Birthday paradox. Our setting differs from the standard Birthday paradox, where we would assume a uniform distribution over birthdays. In our setting, a large number of people have a unique birthday (i.e., a large number of videos have no duplicates and will be unique in our sample of $6,365$). The probability that two or more people in a sample share a birthday, given this highly unbalanced distribution of birthdays, can be computed using a recurrence which we describe below:

$$R(N,T) = \begin{cases} (1-q(T)) \times \big((1-p) \times R(N-1,T) \\ \qquad\qquad\qquad + p \times R(N-1,T+1)\big) & \text{if } N > 0 \\ 1 & \text{if } N = 0 \end{cases} \tag{3}$$

where $R(N, T)$ indicates the probability that a sample of N videos does not contain any duplicates, given that we have already drawn T videos that are associated with copies in the YouTube database (or any video database), where each of these T videos is distinct. The recurrence captures the idea that, if we do not wish to include duplicates in our sample of original videos, we must first draw a non-duplicate given the set of T previously drawn videos associated with a duplicate in the video database with probability $1 - q(T)$. This video must then be selected either from the set of videos with no associated duplicates with probability $1 - p$ or from the set of videos that has at least one duplicate with probability p.

The base case is $R(0, T) = 1$, where have already drawn T videos that are associated with duplicates in the YouTube database, and we have no further videos that need to be selected.

To evaluate this recurrence, we first need to estimate the total number of videos in YouTube. During the random sampling phase, we retrieved $6,365$ unique video IDs using 1000 randomly generated prefixes. Using the method proposed by Zhou et al. [8], we estimated the total number of videos on YouTube as $38^4 \times 64 \times \frac{6365}{1000} \approx 849$ million, indicating there were approximately 849 million videos on YouTube at the time we collected the data (July 2013).

Our measurement results indicate that approximately 10% of the original videos on YouTube have duplicates, meaning that we should set $p = 0.1$ in the computation above. Given our result from the previous section, that each video having one or more duplicates has on average, approximately 4.69 duplicates associated with it, we can estimate the probability of drawing a duplicate for given video as $\frac{4.69}{849 \times 10^6} \approx \frac{1}{181 \times 10^6}$.

Evaluating the above recurrence using a dynamic programming method for $q(T) = \frac{T}{181 \times 10^6}$ and $p = 0.1$ yields $R(6365, 0) = 0.989$. This result means that if we resampled the set of $6,365$ videos over 100 separate trials, then we would expect this set of $6,365$ `sampled videos` to contain duplicates in fewer than two of these trials.

Further, we examined the set of `sampled videos` by first querying the set of `searched video` VIDs to determine if any match a `sampled video` VID. For the small set of VIDs that matched, we ran a further DTW comparison. This DTW phase produced much larger DTW distances than the duplicate threshold for all pairs of videos examined, indicating that none of the $6,365$ `sampled videos` were duplicates. We also performed a manual confirmation step, providing further evidence that the $6,365$ `sampled videos` are unique.

5.3 Extra Storage Space Occupied by Duplicate Videos

A direct negative impact of video duplication is the extra storage space consumed by duplicate videos. To estimate the percentage of additional space needed by YouTube to store duplicate videos, we grouped each `sampled video` and its corresponding duplicates into a duplicate set, denoted by D. If no duplicates were associated with a `sampled video`, v, then we constructed the duplicate set, D, to contain only v, i.e., $D = \{v\}$. For each duplicate set, we selected the video with the largest file size to be the `representative video`. We denote the set of all duplicate sets by $\mathcal{D}$ and the `representative video` of set D by D_r. Note that for all videos, we only retrieved the H.264 Baseline/AAC/MP4/360p version, thus encoding rates for all videos in our dataset should be similar. Short videos in D will likely be sub-videos of longer videos in D. Therefore selecting the video with the largest file size as the `representative video` means that the other, shorter, videos in the set are sub-videos of the `representative video`. Given these duplicate sets and corresponding `representative videos`, we computed the space used to store duplicates as a percentage of the total storage space as follows:

$$1 - \frac{\sum_{D \in \mathcal{D}} \text{size}(D_r)}{\sum_{D \in \mathcal{D}} \sum_{d \in D} \text{size}(d)} \tag{4}$$

Our results show that the total size of representative videos is 91.9 GB, and the total size of all videos in all duplicate sets is 121.0 GB. These space requirements indicate that roughly 24.0% YouTube storage is occupied by duplicates.

6 Conclusion

Duplicate videos within large-scale video sharing services have wide ranging potential impacts on data center and network level storage, caching, and energy consumption. A critical first step in determining the true cost of video duplication involves accurate measurement of duplicate levels.

In this work, we proposed a set of techniques for assessing duplicate levels within large-scale video sharing services. These techniques combined video sampling, video search, computing pairwise video similarity through a variation of dynamic time warping, and a manual validation step. Applying these techniques on YouTube produces a duplicate ratio estimate of 31.7%. Furthermore, we calculate that these duplicates occupy 24.0% of YouTube's video data storage. These relatively high levels of duplication indicate that further work should be conducted to evaluate specific system-level tradeoffs associated with datacenter costs, as well as network-related concerns such as performance of in-network caching under assessed duplicate levels.

To allow duplicate assessment on ever-increasing video databases, we plan to extend our video duplicate assessment techniques so they can scale to much larger video samples. A potentially necessary step toward scaling this assessment would involve developing a system to index videos by semantic content. This type of indexing system would be essential for reducing the number of video pairs that would need to be evaluated by a computationally-expensive pairwise video comparison technique.

Acknowledgements. We appreciate constructive comments from anonymous referees and our shepherd Dongsu Han. The work is partially supported by High-Tech Research and Development Program of China ("863 China Cloud" Major Program) under grant SQ2015AAJY1595, by China NSF under grant 61471217, by China Postdoctoral Science Fund under grant 2014M550735, and by NSF under grants CNS-0746649 and CNS-1117300.

References

1. FFmpeg. `http://www.ffmpeg.org/`.

2. Sandvine Global Internet Phenomena Report 1H 2014. `https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/1h-2014-global-internet-phenomena-report.pdf`.
3. YouTube Statistics. `http://www.youtube.com/yt/press/statistics.html`.
4. W. J. Bolosky, S. Corbin, D. Goebel, and J. R. Douceur. Single Instance Storage in Windows 2000. In *Proc. of USENIX WSS*, 2000.
5. M. Douze, A. Gaidon, H. Jegou, M. Marszałek, C. Schmid, et al. Inria-lears video copy detection system. In *TREC Video Retrieval Evaluation (TRECVID Workshop)*, 2008.
6. C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu, and Mi. Welnicki. HYDRAstor: A Scalable Secondary Storage. In *Proc. of USENIX FAST*, 2009.
7. A. Hampapur, K. Hyun, and R. M. Bolle. Comparison of Sequence Matching Techniques for Video Copy Detection. In *Electronic Imaging 2002*, pages 194–201. International Society for Optics and Photonics, 2001.
8. J. Zhou, Y. Li, V. K. Adhikari, and Z.-L. Zhang. Counting YouTube Videos via Random Prefix Sampling. In *Proc. of ACM IMC*, 2011.
9. H. Jégou, M. Douze, G. Gravier, C. Schmid, P. Gros, et al. Inria lear-texmex: Video copy detection task. In *Proc. of the TRECVid 2010 Workshop*, 2010.
10. K. Jin and E. L. Miller. The Effectiveness of Deduplication on Virtual Machine Disk Images. In *Proc. of ACM SYSTOR*, 2009.
11. A. Kathpal, M. Kulkarni, and A. Bakre. Analyzing compute vs. storage tradeoff for video-aware storage efficiency. In *Proc. of USENIX HotStorage*, 2012.
12. A. Katiyar and J. Weissman. ViDeDup: An Application-Aware Framework for Video De-duplication. In *Proc. of USENIX HotStorage*, 2011.
13. D. G Lowe. Object recognition from local scale-invariant features. In *Proc. of IEEE ICCV*, volume 2, pages 1150–1157, 1999.
14. K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.
15. H. Sakoe and S. Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49, 1978.
16. H. T. Shen, X. Zhou, Z. Huang, J. Shao, and X. Zhou. UQLIPS: A Real-time Near-duplicate Video Clip Detection System. In *Proc. of ACM VLDB*, 2007.
17. S.-H. Shen and A. Akella. An information-aware QoE-centric mobile video cache. In *Proc. of ACM MobiCom*, 2013.
18. C. Ungureanu, B. Atkin, A. Aranya, S. Gokhale, S. Rago, G. Cakowski, C. Dubnicki, and A. Bohra. HydraFS: a High-Throughput File System for the HYDRAstor Content-Addressable Storage System. In *Proc. of USENIX FAST*, 2010.
19. X. Wu, C.-W. Ngo, A. G. Hauptmann, and H.-K. Tan. Real-time Near-duplicate Elimination for Web Video Search with Content and Context. *Multimedia, IEEE Transactions on*, 11(2):196–207, 2009.
20. X. Wu, A. G. Hauptmann and C.-W. Ngo. Practical Elimination of Near-Duplicates from Web Video Search. In *Proc. of ACM Multimedia*, 2007.
21. J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo. Evaluating Bag-of-visual-words Representations in Scene Classification. In *Proc. of ACM MIR*, 2007.
22. C. Zauner. Implementation and benchmarking of perceptual image hash functions. *Master's thesis, Upper Austria University of Applied Sciences, Hagenberg Campus*, 43, 2010.
23. B. Zhu and K. Li. Avoiding the Disk Bottleneck in the Data Domain Deduplication File System. In *Proc. of USENIX FAST*, 2008.