Maze: A Cost-Efficient Video Deduplication System at Web-scale

An Qin*
Baidu Inc.
Beijing, China
qinan@baidu.com

Mengbai Xiao*
Shandong University
Qingdao, Shandong, China
xiaomb@sdu.edu.cn

Ben Huang
Baidu Inc.
Beijing, China
huangben@baidu.com

Xiaodong Zhang
The Ohio State University
Columbus, Ohio, USA
zhang@cse.ohio-state.edu

ABSTRACT

With the advancement and dominant service of Internet videos, the content-based video deduplication system becomes an essential and dependent infrastructure for Internet video service. However, the explosively growing video data on the Internet challenges the system design and implementation for its scalability in several ways. (1) Although the quantization-based indexing techniques [22, 23, 50] are effective for searching visual features at a large scale, the costly re-training over the complete dataset must be done periodically. (2) The high-dimensional vectors for visual features demand increasingly large SSD space, degrading I/O performance. (3) Videos crawled from the Internet are diverse, and visually similar videos are not necessarily the duplicates, increasing deduplication complexity. (4) Most videos are edited ones. The duplicate contents are more likely discovered as clips inside the videos, demanding processing techniques with close attention to details.

To address above-mentioned issues, we propose Maze, a full-fledged video deduplication system. Maze has an ANNS layer that indexes and searches the high dimensional feature vectors. The architecture of the ANNS layer supports efficient reads and writes and eliminates the data migration caused by re-training. Maze adopts the CNN-based feature and the ORB [37] feature as the visual features, which are optimized for the specific video deduplication task. The features are compact and fully reside in the memory. Acoustic features are also incorporated in Maze so that the visually similar videos but having different audio tracks are recognizable. A clip-based matching algorithm is developed to discover duplicate contents at a fine granularity. Maze has been deployed as a production system for two years. It has indexed 1.3 billion videos and is indexing ~800 thousand videos per day. For the ANNS layer, the average read latency is 4 seconds and the average write latency is at most 4.84 seconds. The re-training over the complete dataset is no longer required no matter how many new data sets are added, eliminating the costly data migration between nodes. Maze recognizes the duplicate live streaming videos with both the similar appearance and the similar audio at a recall of 98%. Most importantly, Maze is also cost-effective. For example, the compact feature design helps save 5800 SSDs and the computation resources devoted to running the whole system decrease to 250K standard cores per billion videos.

CCS CONCEPTS

• **Information systems** → **Search engine architectures and scalability**; **Multimedia information systems**.

KEYWORDS

video retrieval, video deduplication, scalable search engine architecture

ACM Reference Format:
An Qin, Mengbai Xiao, Ben Huang, and Xiaodong Zhang. 2022. Maze: A Cost-Efficient Video Deduplication System at Web-scale . In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3503161.3548145

1 INTRODUCTION

The video contents on the Internet, such as *long videos* from YouTube[1] and *short videos* shared via TikTok[2] and Kwai[3], have been experiencing explosive growth for years. Nowadays, only a video search engine holding video data at a billion scale is considered effective [22, 35]. However, the video data at such a massive scale make the content-based video retrieval task challenging.

The content-based video retrieval centers on the visual features generated from frames. The features are either hand-crafting ones [4, 7, 19, 31, 37] or are created by the deep neural network (DNN) techniques [15], both incurring high computation costs. Moreover, various features are required for comprehensively characterizing a video frame. For example, the models applied to recognize people faces [9] and to calculate the clarity score [42, 44] are completely different. On the other hand, duplicate contents in videos widely exist. Users publish their videos on multiple platforms for maximizing profits or a video is transcoded into a number of bitrates to adapt the network dynamics. Crafting visual features for duplicate contents is a waste. Thus, efficiently detecting and

*Both authors contributed equally to this research.

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548145

[1] https://www.youtube.com
[2] https://www.tiktok.com
[3] https://www.kuaishou.com

grouping duplicate video contents before the expensive feature generation would substantially ease the content-based video retrieval.

While detecting duplicate contents is demanding, the dataset of billions of videos with heavy reads (searching duplicate videos more than ~10 million times per day) and writes (adding millions of videos into the database every day) challenges the system design and implementation. The existing research on near-duplicate video retrieval (NDVR) [6, 12, 38–40, 47] mainly focuses on the algorithm design instead of building a scalable architecture. Another challenge is that the visually similar videos are not necessarily duplicated. For example, individual videos of a series of *lectures* commonly have both the same lecturer and background but different contents. In addition, numerous videos are cut and compiled from a few origin ones, thus it is also desirable to develop a method of locating the duplicate clips instead of the complete videos.

To solve these problems, we propose Maze, a full-fledged video deduplication system at Web-scale. In Maze, videos are detached into key frames, and visual features represented as high dimensional vectors are generated. Maze relies on an approximate nearest neighboring search (ANNS) layer to index and search the vectors, where the quantization-based indexing [22, 23, 50] is adopted. However, for a fast-growing dataset, the quantization-based indexing requires periodically re-training new centroids over the complete dataset to more accurately reflect the changing data distribution, introducing costly data migration. In Maze, the input vectors are sequentially flushed into *shards*, i.e., only the most recently created shard accepts new vectors. As a result, each shard contains data collected during a short period and the re-training is constrained inside a shard, avoiding the data movement between nodes at all. To further reduce the cost of the system, video frames are characterized by convolutional neural network (CNN) based features. The compact CNN-based features are easy to compare but an additional verification stage that uses oriented FAST and Rotated BRIEF (ORB) [37] features is still required to guarantee the retrieval accuracy. The CNN-based features and the ORB features are either quantized [23] or truncated [7] so that the vectors could be fully contained in the memory of the shards. As for the videos differentiated by their audio, we incorporate acoustic features. The audio is first visualized into spectrograms and the CNN is used to generate the feature vectors. To recognize duplicate clips contained in videos, we develop a matching scheme based on the Smith-Waterman algorithm. Our scheme exploits both the visual and the acoustic features of the videos, and recursively locates all potential matching clips. We evaluate Maze based on a dataset of 1.3 billion videos that are crawled from the Internet, and ~800 thousand videos are added to the dataset every day. The Maze system maintains the average read latency to the ANNS layer at 4 seconds and the average write latency up to 4.84 seconds. More importantly, when the system scales from 200 shards to 350 shards (in 5 months), only 2 re-trainings are performed. Each of the re-training takes 4 hours and suspends 1 shard only. The optimizations on the CNN-based features and the ORB features make all vectors reside in the memory, eliminating the expected requirement of 5800 SSDs. Incorporating the acoustic features helps effectively deduplicate the videos like live streaming at a high recall rate of 98%, which can not be recognized by only the visual features. We also measure the computation resources required to run Maze. The running cost of Maze on 1 billion videos converges at 250K standard cores, showing the high scalability of our design.

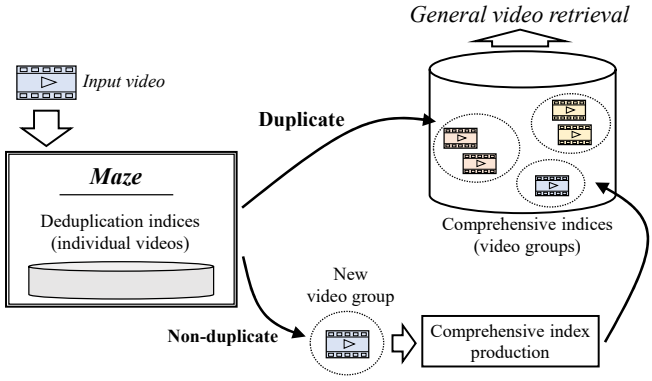

Figure 1: Maze deduplicates videos for the general video retrieval service.

In this paper, our contributions are summarized as follows:

- We develop a full-fledged video deduplication system at web-scale. A highly scalable ANNS layer is designed to eliminate costly data migration between nodes, supporting the video dataset at a billion scale and incrementally indexing ~1 million videos per day.
- We design a two-stage feature comparison for the video deduplication task. The optimizations on the CNN-based feature and the ORB feature substantially reduce the storage cost, saving 5800 SSDs that are expected to install.
- We incorporate acoustic features in our system to recognize videos more than visually duplicate ones. A matching scheme based on the Smith-Waterman algorithm is implemented to discover duplicate contents in a fine granularity.
- We have evaluated Maze in a production environment for 2 years. While maintaining the stable read and write performance, we have achieved a goal of cost-effectiveness for Maze by maintaining a constant amount of computing resources to timely process newly arrived videos.

Our paper is organized as follows: Section 2 introduces the background. Section 3 presents the scalable architecture design of the ANNS layer in Maze. Section 4 shows how the features are designed as well as the matching scheme. In Section 5, extensive experiments are carried out to justify our design. Section 6 discusses related work and Section 7 concludes our work.

2 BACKGROUND

2.1 Maze and General Video Retrieval

To effectively support general content-based video retrieval, the search service needs to create a number of *comprehensive indices* for a crawled video [35]. For example, producing features that recognize people appeared in the video [9] and features representing the visual clarity [42, 44]. To avoid unnecessary computation on the feature generation, comprehensive indices could be organized based on video groups, whose members contain duplicate contents and share the same index data. Maze is installed prior to the comprehensive

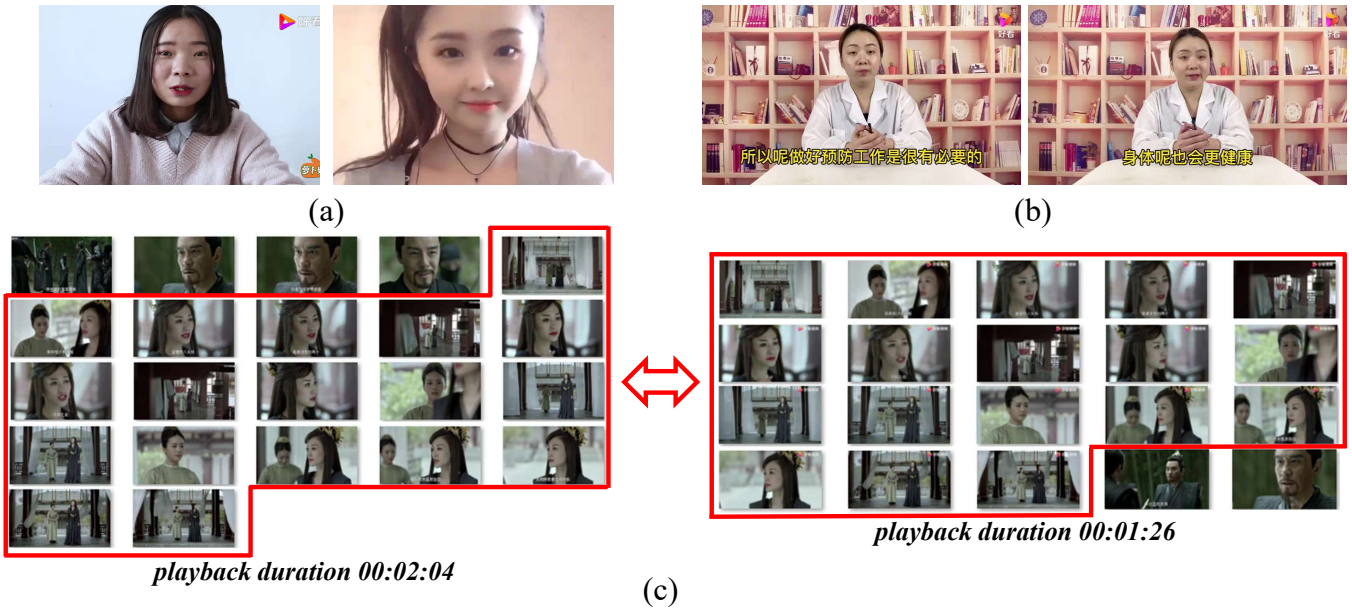

Figure 2: The cases challenge the existing NDVR methods [6, 12, 38–40, 47]. (a) Two videos with different contents (singing *vs.* advertising) but have a highly similar shooting style. The frame-level feature is hardly helpful. (b) Two individual videos from a series of lectures on health caring, which have the same lecturer and background but different topics. They are hard to differentiate without audio information. (c) Two short videos are edited from the same TV series, while the editors choose different scenes to be included.

index production, identifying if a crawled video contains duplicate contents with the existing ones. Only a video with non-duplicate contents is stored as a new video group and invokes the expensive comprehensive index production. Otherwise, the crawled video is added to an existing video group directly. Inside Maze, the more lightweight *deduplication indices* are created. How Maze interacts with the general video retrieval service is presented in Figure 1. In addition, other applications, like *plagiarism recognition*, *copyright infringement detection*, *clip correlation*, *video recommendation*, etc., could also take advantage of Maze for discovering duplicate contents.

2.2 ANNS

Visual contents are commonly represented by high-dimensional vectors, and the similarity is thus quantified as to their L2 distances. However, searching the nearest neighbors for a query vector over the entire dataset at a billion scale is unacceptable. Thus, various indexing solutions are proposed to support approximate nearest neighboring search (ANNS). Among them, the quantization-based indices [22, 23, 50] feature high retrieval accuracy (*vs.* locality-sensitive hash [11]), efficient memory footprint (*vs.* graph-based solutions [33]), and low computational complexity (*vs.* tree-based solutions [5, 34]), and are widely adopted in industrial systems [35, 43, 45, 51]. With the quantization-based indices, commonly a set of centroids are trained. Then indexing a vector is realized as grouping it to the nearest centroid. In order to find the nearest neighbors of a query vector, the vectors belonging to the top-n nearest centroids are selected to calculate the L2 distances from the query. As a result, top-k vectors with the minimum distances are returned. The parameters of k and n are selected according to the requirements of the upper-level applications.

2.3 Challenging Cases

When developing a video deduplication system for Web-scale datasets, we have encountered several challenging cases. The visually similar videos do not necessarily contain duplicate contents. Figure 2(a) and Figure 2(b) are two examples, which are both a pair of different videos with highly similar frames. Characterizing videos with visual features would falsely identify these videos as duplicates. In addition, edited videos that are cut and compiled from a few origin ones are ubiquitous over the Internet. These videos have different bitrates, frame rates, playback durations, etc., though they are duplicate contents. Figure 2(c) illustrates such a case. Efficiently identifying these duplicate clips is also critical to Maze.

3 ARCHITECTURE DESIGN

In this section, we will give an overview of Maze and its highly scalable design of the ANNS layer, which is the core component of the system.

3.1 System Overview

Maze is a full-fledged video retrieval system, which is specifically designed for visual content deduplication. Maze accepts an input video for indexing (write) or searching (read), which is presented in Figure 3. For indexing, visual features (Section 4.1) and acoustic features (Section 4.2) are generated from the key frames and the

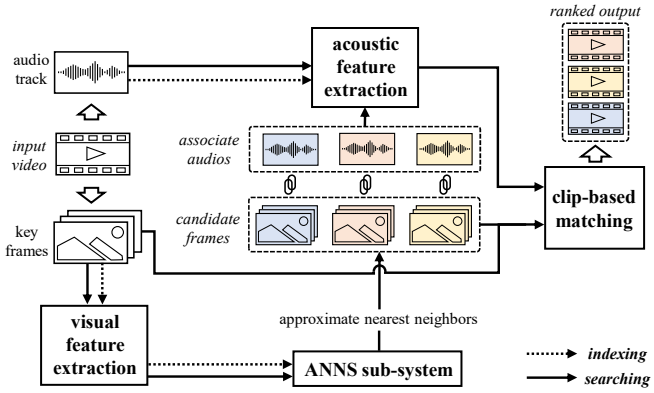

Figure 3: An overview of Maze

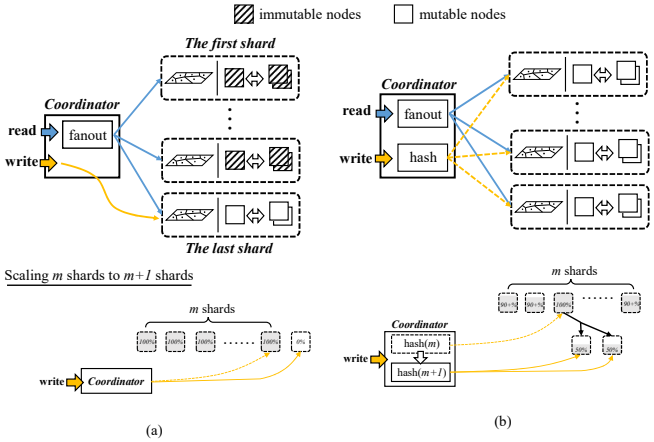

Figure 4: The architecture of write-one-read-all *vs.* in-place insertion in Maze

audio track, respectively. We build the inverted indices over the visual features since they mostly determine if the two videos are similar. Specifically, visual features as high-dimensional vectors are stored in an ANNS layer for future retrieval. The acoustic features are associated with the visual features of the same video.

To carry out a search query, visual features of the query video are sent to the ANNS layer for discovering similar frames. A list of candidate videos with acoustic features are aggregated from these frames. For a pair of any candidate video and the query video, the clip-based matching algorithm locates the duplicate clips and calculates a similarity score based on both the visual features and the acoustic features. According to the score, the candidate videos are ranked and returned.

3.2 Scalable ANNS Layer

In order to ensure the data freshness, Maze is expected to index millions of videos every day. In the ANNS layer, the workloads are two orders of magnitude larger: nearly 100 million key frames are indexed per day. With such a fast scaling dataset, periodical re-training over the complete dataset is necessary for balancing the number of vectors grouped under each centroid. However, the re-organization of all vectors leads to costly data migration among nodes. This suspends the normal deduplication workloads or at least greatly degrades the read and write performance. To overcome the challenge, we design a novel architecture that adapts the ANNS layer to the quickly growing video dataset, which is shown in Figure 4(a). We will first introduce our architecture design featuring the write-one-read-all policy, and then discuss two potential alternatives that are unable to solve the data migration problem.

In Maze, vector data are stored in equivalent *shards* that each has a full copy of centroids. A read query is therefore required to be executed in all shards indiscriminately and the results are then collected to form the top-k list. Inside a shard, duplicate instances are added or removed on-demand according to the search workload for guaranteeing short query latencies. It is worth noting that all shards but the last one are immutable, i.e., the write traffic is always handled on the nodes in the last shard. Once the last shard is full, it is sealed and a new shard is created to sustain the write traffic. Creating a new shard is trivial: after copying all centroids from the previous shard, the new shard is capable of accepting the write workloads.

In our design, even if the distribution of the latest data points has changed (this could be discovered by an average longer search latency from a shard), it will not affect the historical data. The stable performance can be expected when re-executing the same query if the recently created shards are quarantined. This is especially helpful when diagnosing the running system. Moreover, the distribution change is easy to fix: Maze conducts the re-training inside a shard, leading to no data migration at all.

In-place insertion: Maze is designed as a memory-centric indexing system, i.e., all vectors are in the memory (Section 4.1 introduces the optimizations on the size of feature vectors), and write-ahead-logs (WALs) are implemented on the disk only for a recovery from a system crash. As a result, it is tempting to carry out the in-place insertion on all shards like the design in Figure 4(b). While the read queries are still sent to all shards, the write queries are distributed to one of the shards after hashing for load balancing. However, with this method, data migration is unavoidable when scaling the system: as long as a shard is full, we need to launch a new shard immediately and move half of the data onto it. We also need to update the hash function so that newly incoming write traffic could be redirected to the new shard. This complicates the implementation if we need data on the shard to be always accessible: the shard expansion must be done on a separate clone while the original one still accepts the search queries. In addition, the write traffic to the shard during the expansion has to be hold in an immediate buffer that also accepts read queries. The immediate buffer is merged into the base after expansion.

Cluster-based partition: Another potential design is to partition the dataset according to the centroids, i.e., a group of shards only hold vectors belonging to a centroid. This benefits the read query since only n groups of shards are selected to process a search query. However, this also leads to unfavorable data migration: as long as the data distribution has changed, we have to perform the re-training over the entire dataset to avoid data skewness.

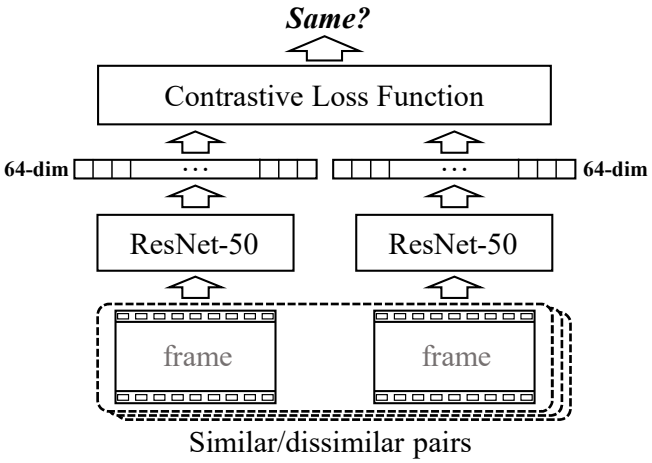

Figure 5: Training the CNN with the Siamese architecture. In the training, a pair of frames are accepted as the input and output if they are duplicated. Our training set contains ~1000 million image pairs that can be distinguished as positive (similar) pairs and negative (dissimilar) pairs. We manually label a small portion of the training set while most are generated automatically from an online search service: if two images are clicked by different users sending the same query, they are associated as a positive pair. The negative training pairs are randomly generated. All images are transformed into the grayscale during the training.

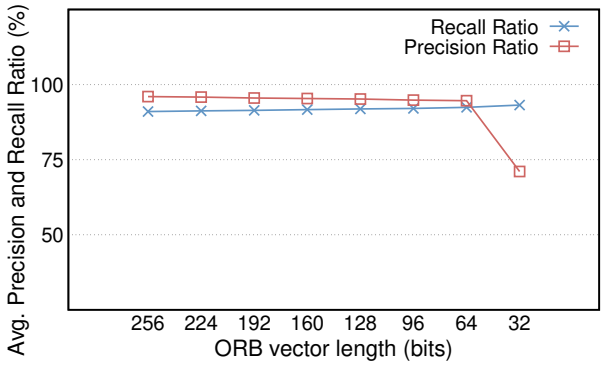

Figure 6: The experimental results of seeking the optimal ORB length. In the experiment, the query set is 1000 similar video pairs marked manually. The dataset is 300 thousand videos that are randomly selected, where ~50 millions of key frames are extracted. For each frame, 50 ORB feature vectors are generated and the index data are built with HNSW [33]. The recall ratio is measured as the proportion of queries that successfully retrieve the paired videos in the top-1 list, and the precision ratio is measured as the proportion of retrieved videos (top-*k*) manually marked as *similar*.

4 FEATURES AND MATCHING

The feature selection is critical to a video retrieval system, affecting both the accuracy and the cost. In Maze, similar frames are retrieved with the CNN-based features. After the retrieval, a verification stage refines the results with the ORB features [37]. In addition to the visual features, acoustic features are generated from the audio track of a video to be indexed. With both the visual and acoustic features, duplicate clips are located and ranked according to the clip-based matching method.

4.1 Visual Features

A video frame could be represented by a set of local features that characterize details of an image [4, 7, 30, 37]. But leveraging the local features also introduces high computation costs. Since commonly tens to hundreds of feature vectors are generated for effectively understanding a frame, calculating the similarity of two frames becomes complicated, which requires pairing the vectors, calculating the L2 distances, and aggregating to an overall score. Although several optimizations, e.g., quantizing the feature vector into a shorter format, replacing the ANN search with hashing operations, and calculating hamming distances instead of Euclidean distances, could speed up the computation, they fail to reduce the inherent complexity. Furthermore, searching with local features would retrieve a large number of frames that are only partially similar to the query, imposing undesired computation costs to our system for only deduplication.

Considering the drawbacks of the local features, Maze adopts CNN to generate one feature vector for one frame. More specifically, a standard ResNet-50 [16] is trained to produce a 64-dimensional vector for each video frame. The similarity score between two frames is thus simplified to the L2 distance between two CNN-based feature vectors. As a result, the retrieval results returned by the ANN search are more concise because the partially similar images are excluded. As shown in Figure 5, the ResNet-50 is trained with the Siamese architecture [8, 13]. We also process the CNN-based feature with product quantization (PQ) [23] in the ANNS layer, which further optimizes the vector size.

Verification: Using the CNN-based features occasionally returns false positive frames. For example, gameplay videos of the same game commonly share a highly similar background and are recognized as duplicates. As a result, we still need the local features to refine the search results. In Maze, ORB is adopted as the local feature, which is efficient in crafting (binary tests), comparing (hamming distance), and storing. Prior work [7] has shown that when recognizing some identical objects, the saturation effects are observed beyond ~60 binary tests, i.e., using a vector size longer than 60 bits might be unnecessary. In Maze, we determine the ORB feature size experimentally.

In the experiment, we reduce the ORB feature vector size from 256-bit to 32-bit at the step of 32 bits and measure the retrieval precision and recall. The results are shown in Figure 6, where the x-axis is the ORB length in bits, and the y-axis is the average precision and recall ratio. When the vector length is 64 bits or more, we observed the saturation effects. The precision ratio is reduced from 96% to 94.63%, and the recall is increased from 91% to 92.41%. As long as we reduce the vector length to 32 bits, the precision is sharply reduced to 71.05% while the recall ratio slightly increases

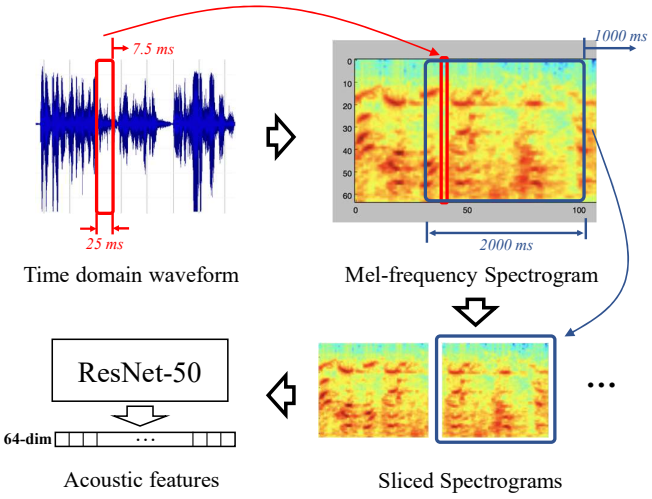

Figure 7: Extracting acoustic features from the audio track of a video. The audio track is in the time domain waveform kept in the `.wav` file. The Mel-frequency spectrogram of the audio track is generated by applying a sliding window of 25 ms at a step of 7.5 ms, where 64 Mel-filters are used. Then the spectrogram is sliced with another sliding window of 2000 ms at a step of 1000 ms. The audio track of the video is thus "visualized" and fed into a standard ResNet-50 for extracting the desired 64-dimensional feature vectors. We train the ResNet-50 following the Siamese architecture presented in Figure 5.

to 93.17%. As a result, the ORB feature used in our system is 64-bit to avoid accuracy loss.

4.2 Acoustic Features

Videos like *lectures*, *news*, *talkshows*, and *concerts* distinguish themselves by the audio data. Thus, the acoustic features are essential in differentiating them. A potential scheme for creating the acoustic features is to apply a natural language processing (NLP) technique. But such a method can not process audio without human languages. In addition, different models should be involved to tackle various languages. Distinguishing two audio tracks with the same content, e.g., different people singing a song is also challenging. As a result, we need a more general acoustic feature to identify the duplicate audios.

In Maze, we transform the audio track of a video into the Mel-frequency spectral coefficients (MFSCs), which are represented as a Mel-frequency spectrogram having the width of the video playback duration. Like the visual feature extraction, we also train a CNN via the Siamese architecture that accepts the spectrograms as the input. The audio part of a video is materialized into a set of 64-dimensional vectors as its acoustic features. Figure 7 shows the details of the acoustic feature extraction. Mel-frequency cepstral coefficients (MFCCs) are generated by applying discrete cosine transform (DCT) to MFSCs, and could also represent the audio. However, MFCCs generate a narrower spectrogram, leading to a relatively low accuracy in the retrieval task.

4.3 Video Clip Matching

The visual features and the acoustic features only suggest if two frames or two pieces of audio are similar. A matching method at the video level is still needed. We then tweak the Smith-Waterman algorithm that is originally designed to find the most similar segments from a pair of gene sequences for the video clip matching.

Algorithm 1 Recursive Smith-Waterman Algorithm

procedure R-SWAlign(l_q, o_q, l_r, o_r, **R**)
  Input: l_q, o_q, l_r, o_r: The length and the offset of the query/reference frame sequence to search; **R**: the matching subsequence pairs; SWAlign(·): The Smith-Waterman algorithm that finds the best matching subsequence.

  $s_q, e_q, s_r, e_r \leftarrow$ SWAlign(l_q, o_q, l_r, o_r)
    ▷ $\{s, e\}$ *marks where a subsequence starts and ends*

  if $s_q < e_q$ and $s_r < e_r$ **then**
    $\mathbf{R} \leftarrow \mathbf{R} \bigcup \{s_q + o_q, e_q + o_q, s_r + o_r, e_r + o_r\}$
    R-SWAlign(s_q, o_q, s_r, o_r, **R**)
    R-SWAlign($l_q - e_q - 1, e_q + o_q + 1,$
      $l_r - e_r - 1, e_r + o_r + 1$, **R**)
  end if
end procedure

In Maze, a matching score is calculated between a query video and a reference video, and each of them is represented by a sequence of visual vectors (plus an optional sequence of acoustic vectors). We change the Smith-Waterman algorithm to a recursive version as in Algorithm 1 so that multiple matching subsequences are discovered. In the algorithm, a match of two sequence elements, i.e., the vectors, is determined if their L2 distance is less than a threshold. If the acoustic feature sequences are also involved in the matching score calculation, the following equation is used:

$$S = \alpha S_v + (1 - \alpha) S_a$$

, where S_v and S_a are the matching scores from the visual and acoustic feature sequences, respectively, α is the coefficient that weighs the visual features, and S is the synthesized matching score used in the final ranking.

5 EVALUATION

Maze is built as a web-scale video deduplication system and it has been run for 2 years. All the experimental results are collected in the production environment. Until now, Maze has indexed in total 1.3 billion videos, roughly 130 billion key frames. The features of the key frames are distributed into approximately 350 shards in the ANNS layer, each containing indices of 60 million key frames. About 800 thousand videos are newly crawled every day, which means 60-80 million key frames are incrementally indexed into the ANNS layer.

Inside a shard, equivalent nodes are launched to adapt to the dynamic workloads. Each node is equipped with 200 standard cores[4],

[4]An E5-2420 CPU with turbo off and hyper-threading on roughly has the same computation power of 240 standard cores.

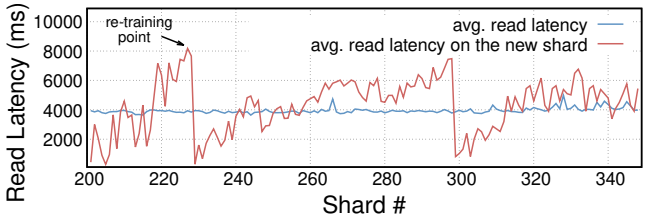

Figure 8: The read performance of the ANNS layer when the dataset scales

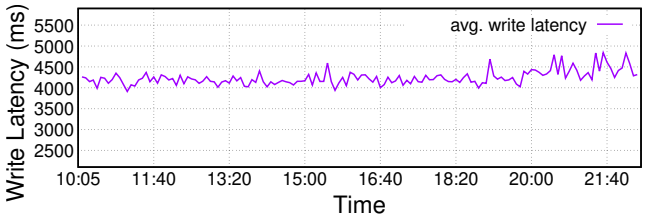

Figure 9: The write performance of the ANNS layer captured for 12 hours

60GB DRAM, and 200GB SSD space. A shard launches a new node if the average utilization of the running nodes inside reaches 80% for at least 15 minutes, and revokes a node if the average utilization is lower than 10% for 60 minutes. Based on the hardware configuration, we comprehensively evaluate the system in terms of the performance of the ANNS layer, the effects of the feature design, and the cost of running Maze.

5.1 ANNS layer

Maze indexes the vectors using a two-level GNOIMI [50] structure. In a shard of the ANNS layer, vectors are clustered under 8000 level-1 centroids and 5000 level-2 centroids. The residuals are further encoded with PQ [23] to save memory space. A vector is equally split into 8 subspaces and each is quantized into 256 numbers.

When a read query arrives at a shard, Maze searches the nearest neighboring vectors clustered under the closest 1000-2000 centroids of both levels. The comparison between the query and a stored vector uses asymmetric distance computation (ADC) [23], and at most 8000 results are returned to the coordinator from each shard. The coordinator merges the collected frames into videos for the following ranking.

The read latencies of the ANNS layer are continuously collected when the dataset scales from 200 shards to 350 shards, and the results are reported in Figure 8. The x-axis is the number of shards required to hold the index data, and the y-axis is the read latency measured in milliseconds. We both measure the average read latency of all shards and the average read latency in the last shard. For the average read latency at varying dataset scales, it keeps stable, which is always ~4 seconds. But when looking into the latencies of the individual shards, it varies significantly. This is because the trends of the new videos on the Internet is changing but the centroids used in the created shard are inherited from the previous one. This leads to a roughly increasing average read latency on the last shard. When the average read latency on the last shard reaches a threshold, we re-train the centroids on the last shard. The re-training takes ~4 hours, and it reduces the shard-level average read latency to less than 1 second. When the dataset grows from 200 shards to 350 shards, two re-trainings are conducted.

For the write latencies, we collected the performance data from 8:00 to 22:00 on a day in April 2022. The workloads are videos crawled from the Internet. After extracting the key frames from a video, individual read queries are sent to the ANNS layer to locate the duplicate frames, and then the features of the same key frames are inserted into the last shard. Thus the read and write ratio is 1:1. The results are shown in Figure 9, where the x-axis is the wallclock time and the y-axis is the average write latency in 5 minutes. We notice that the average latency is always stable because the write queries are only handled on the last shard, which is at most 4.84 seconds. As a result, our architectural design for the ANNS layer can reach the stable read and write performance for a long-living system at web-scale.

5.2 Visual Features

In Maze, a key frame is materialized into one 64-dimensional CNN-based feature vector and 50 64-bit ORB features. After product quantization, a CNN-based feature is reduced to 64 bytes. As a result, indexing a key frame takes 464 bytes of space. Because an instance is equipped with 60GB memory, a shard can hold the index data of 60 million key frames.

Maze reduces the ORB feature from 256-bit to 64-bit (discussed in Section 4.1). Before the optimization, the ORB features of a shard are too large to be contained in the memory, and thus need to be stored in SSDs. Though the ORB features of a shard can hardly occupy a full SSD, we still have to attach an individual disk to every instance for satisfying the frequent reads, leading to a low utilization ratio at 8.89% of the SSDs. After the optimization, the ORB features could be completely held in the memory, eliminating ~5800 SSDs that should be installed at the current scale of our system.

5.3 Acoustic Features

The acoustic features are generated for live streaming videos, lectures, etc. Based on the current dataset, 26% of videos are qualified to generate acoustic features. To evaluate the effectiveness of the acoustic features, we build an individual database with 4046 videos extracted from 3271 live streamings, which were collected on December 10th, 2021. Among the videos, 748 videos were randomly chosen as the query set, and during the ranking, the weight of the acoustic feature is set to 0.6. We use *recall@1* to reflect the effectiveness, which is calculated as the ratio of the top-1 responses that contain the same video as the query. In the experiment, applying the acoustic feature could reach the recall@1 at 98%. Without the acoustic features, these videos are recognized as bad cases that can hardly be recalled.

5.4 Running Cost

The cost of running the whole system is the factor we pay most attention to. Specifically, we measure how many standard cores have to be devoted to maintaining the acceptable read and write performance. We record the running cost of different components in Maze when the underlying dataset scales, and the results are shown

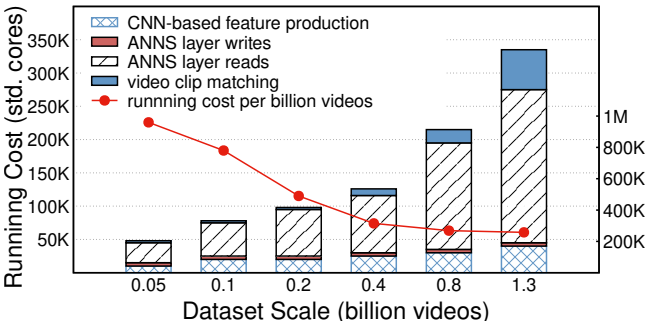

Figure 10: The cost of running Maze when the dataset scales

in Figure 10. In Maze, four components dominate the running cost: CNN-based feature production, ANNS layer writes, ANNS layer reads, and video clip matching. With the growing dataset, the cost of ANNS layer writes and CNN-based feature production consumes almost the same computation resources, which are 24.17K and 5K standard cores, respectively. The reason is that the computation tasks of these two components are irrelevant to the dataset scale but are related to the number of videos added per day, which varies little. For ANNS layer read and video clip matching, more CPU cores must be devoted to guaranteeing their performance on a larger dataset. When the dataset scale increases from 0.05 billion videos to 1.3 billion videos, 200K more standard cores (from 30K) are consumed to maintain the search performance in the ANNS layer, and 57K more standard cores (from 3K) are used to finish the video matching and ranking. The increase of the running cost is reasonable because we have to search in an ever-growing dataset and more candidate frames are returned for video clip matching. At the current system scale of 1.3 billion videos, 335K standard cores are consumed to run the whole system, and 68.66% are used on the search queries to the ANNS layer. Figure 10 also plots the running cost per billion videos as the red line, which shows that with the dataset growing, the cost for running the system over a billion videos gradually decreases to approximately 250K standard cores, showing the high scalability of our design.

6 RELATED WORK

Feature Description: Numerous approaches are developed for capturing visual features. The local features are extracted from an image, describing details of the frames: HSV color histograms [14, 21, 39, 47], local binary patterns (LBP) [21, 48, 53], fuzzy multidimensional histograms of color and motion video segments [10], auto colour correlograms (ACC) [6], and the keypoint descriptors, such as SIFT [19, 31], SURF [4], BRIEF [7], and ORB [37], combined with vector of locally aggregated descriptors (VLAD) [18, 36]. On the other hand, CNN-based features understand the videos as a whole so that they are compact and cost-efficient [20, 28, 41, 46].
Retrieval Algorithms: One of the earliest schemes [47] represents a video with a global vector, and the similarity is the dot product between vectors. In the Bag-of-Words (BoW) scheme [19, 27, 38], a frame is mapped to one or more visual words, and the video representation is the tf-idf representation of these visual words. Another practice is to generate a hash code representing a video [20], which learns a group of hash functions that project the video frames into the Hamming space and combine the results into a single video representation.
Video Analytic Systems: The video deduplication system is in essence a video analytic system. In some cases, indexing all video content in the database with comprehensive understanding models is too costly. The design then focuses on accurately filtering out videos that are irrelevant. Focus [17] is an analytic system recognizing and classifying objects appearing in videos. To accelerate the query processing, videos are first clustered with low-cost CNNs when being inserted. The ground-truth CNN with high accuracy and high computation cost is only adopted to a limited list of retrieval results when processing the query. In NoScope [26], the low-cost binary classifiers are trained to efficiently filter videos, and the reference network is only used at the last step. Extending the analytic capability is also a concern for the systems. BlazeIt [24, 25] implements aggregation and limit operations over the video, and the queries are written in the form of a SQL-like language. Panorama [52] further incorporates unbounded vocabulary queries into the video analytic systems, while another study [32] suggests adding probabilistic predicates for more operation space on the video analytic tasks. In Vaas [3], an interactive interface is provided so that users are encouraged to explore effective methods for their own analytics tasks. In addition to the content-based analytic techniques, the inherent structural information of videos is helpful in query acceleration. Specifically, methods like scaling the resolution or reducing the color depth can substantially reduce the time required to process the queries [2]. The configuration of the video formats in the dataset is considered the central concern in VStore [49], which helps realize the system in a backward derivation manner. NV-Tree [29] is an efficient indexing technique developed for high-dimensional vectors on disks, and it has been successfully deployed in a visual retrieval system containing tens of billions of features [1].

7 CONCLUSION

In this paper, we present Maze, a video deduplication system at web-scale. In Maze, an scalable ANNS layer is built to index and search feature vectors, where the data migration among nodes is elminated. The CNN-based feature and the ORB feature are employed as the visual features. After quantization and truncation, the visual feature could fully reside in the memory, avoiding expensive I/Os during the search. Maze also generates acoustic features for videos like lectures, live streaming, etc. This helps identify the videos that only differ in their audio. A clip-based video matching scheme is developed based on the Smith-Waterman algorithm in Maze. Maze has been deployed as a production system for 2 years, and it indexes over 1.3 billion videos. The comprehensive results of high performance and stable daily operations of Maze in production systems meet our design goal for a cost-efficient video deduplication system.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive comments. This work has been partially supported by the grants from the National Natural Science Foundation of China (Grant No. 62102229) and the National Science Foundation under grants IIS-1718450, CCF-2005884, and CCF-2210753.

REFERENCES

[1] Laurent Amsaleg, Björn Þór Jónsson, and Herwig Lejsek. 2018. Scalability of the NV-tree: Three Experiments. In *Proceedings of the 2018 International Conference on Similarity Search and Applications*. pp. 59–72.

[2] Michael R. Anderson, Michael J. Cafarella, German Ros, and Thomas F. Wenisch. 2019. Physical Representation-Based Predicate Optimization for a Visual Analytics Database. In *Proceedings of the 2019 IEEE International Conference on Data Engineering*. pp. 1466–1477.

[3] Favyen Bastani, Oscar R. Moll, and Samuel Madden. 2020. Vaas: Video Analytics At Scale. *Proceedings of the VLDB Endowment* 13, 12 (2020), pp. 2877–2880.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. Surf: Speeded Up Robust Features. In *Proceedings of the 2006 European Conference on Computer Vision*. pp. 404–417.

[5] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees used for Associative Searching. *Commun. ACM* 18, 9 (1975), pp. 509–517.

[6] Yang Cai, Linjun Yang, Wei Ping, Fei Wang, Tao Mei, Xian-Sheng Hua, and Shipeng Li. 2011. Million-Scale Near-Duplicate Video Retrieval System. In *Proceedings of the 2011 ACM International Conference on Multimedia*. pp. 837–838.

[7] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. 2010. Brief: Binary Robust Independent Elementary Features. In *Proceedings of the 2010 European Conference on Computer Vision*. pp. 778–792.

[8] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 539–546.

[9] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 4685–4694.

[10] Anastasios D Doulamis, Nikolaos D Doulamis, and Stefanos D Kollias. 2000. A Fuzzy Video Content Representation for Video Summarization and Content-Based Retrieval. *Signal Processing* 80, 6 (2000), pp. 1049–1067.

[11] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 1999 International Conference on Very Large Data Bases*. pp. 518–529.

[12] Giorgos Kordopatis-Zilos and Symeon Papadopoulos and Ioannis Patras and Yiannis Kompatsiaris. 2017. Near-Duplicate Video Retrieval by Aggregating Intermediate CNN Layers. In *Proceedings of the 2017 International Conference on Multimedia Modeling*. pp. 251–263.

[13] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1735–1742.

[14] Yanbin Hao, Tingting Mu, Richang Hong, Meng Wang, Ning An, and John Y Goulermas. 2016. Stochastic Multiview Hashing for Large-Scale Near-Duplicate Video Retrieval. *IEEE Transactions on Multimedia* 19, 1 (2016), pp. 1–14.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*. pp. 1026–1034.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 770–778.

[17] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. 2018. Focus: Querying Large Video Datasets with Low Latency and Low Cost. In *Proceedings of the 2018 USENIX Symposium on Operating Systems Design and Implementation*. pp. 269–286.

[18] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. 2010. Aggregating Local Descriptors into A Compact Image Representation. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 3304–3311.

[19] Yu-Gang Jiang, Chong-Wah Ngo, and Jun Yang. 2007. Towards Optimal Bag-of-Features for Object Categorization and Semantic Video Retrieval. In *Proceedings of the 2007 ACM International Conference on Image and Video Retrieval*. pp. 494–501.

[20] Yu-Gang Jiang and Jiajun Wang. 2016. Partial Copy Detection in Videos: A Benchmark and An Evaluation of Popular Methods. *IEEE Transactions on Big Data* 2, 1 (2016), pp. 32–42.

[21] Weizhen Jing, Xiushan Nie, Chaoran Cui, Xiaoming Xi, Gongping Yang, and Yilong Yin. 2018. Global-View Hashing: Harnessing Global Relations in Near-Duplicate Video Retrieval. *World Wide Web* 22, 2 (2018), pp. 771–789.

[22] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2021), pp. 535–547.

[23] Herve Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), pp. 117–128.

[24] Daniel Kang, Peter Bailis, and Matei Zaharia. 2019. BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-based Video Analytics. *Proceedings of the VLDB Endowment* 13, 4 (2019), pp. 533–546.

[25] Daniel Kang, Peter Bailis, and Matei Zaharia. 2019. Challenges and Opportunities in DNN-Based Video Analytics: A Demonstration of the BlazeIt Video Query Engine. In *Proceedings of the 2019 Biennial Conference on Innovative Data Systems Research*. online proceedings.

[26] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Neural Network Queries over Video at Scale. *Proceedings of the VLDB Endowment* 10, 11 (2017), pp. 1586–1597.

[27] Giorgos Kordopatis-Zilos, Symeon Papadopoulos, Ioannis Patras, and Yiannis Kompatsiaris. 2017. Near-Duplicate Video Retrieval by Aggregating Intermediate CNN Layers. In *Proceedings of the 2017 International Conference on Multimedia Modeling*. pp. 251–263.

[28] Giorgos Kordopatis-Zilos, Symeon Papadopoulos, Ioannis Patras, and Yiannis Kompatsiaris. 2017. Near-Duplicate Video Retrieval with Deep Metric Learning. In *Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops*. pp. 347–356.

[29] Herwig Lejsek, Friðrik Heiðar Ásmundsson, Björn Þór Jónsson, and Laurent Amsaleg. 2008. NV-Tree: An Efficient Disk-based Index for Approximate Search in Very Large High-Dimensional Collections. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 5 (2008), pp. 869–883.

[30] David G. Lowe. 1999. Object Recognition from Local Scale-Invariant Features. In *Proceedings of the 1999 IEEE International Conference on Computer Vision*. pp. 1150–1157.

[31] David G Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60, 2 (2004), pp. 91–110.

[32] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. 2018. Accelerating Machine Learning Inference with Probabilistic Predicates. In *Proceedings of the 2018 International Conference on Management of Data*. pp. 1493–1508.

[33] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and Robust Approximate Nearest Neighbor Search using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2018), pp. 824–836.

[34] Stephen M Omohundro. 1989. *Five Balltree Construction Algorithms*. International Computer Science Institute Berkeley.

[35] An Qin, Mengbai Xiao, Yongwei Wu, Xinjie Huang, and Xiaodong Zhang. 2021. Mixer: Efficiently Understanding and Retrieving Visual Content at Web-Scale. *Proceedings of the VLDB Endowment* 14, 12 (2021), pp. 2906–2917.

[36] Jérôme Revaud, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. 2013. Event Retrieval in Large Video Collections with Circulant Temporal Encoding. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 2459–2466.

[37] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An Efficient Alternative to SIFT or SURF. In *Proceedings of the 2011 International Conference on Computer Vision*. pp. 2564–2571.

[38] Lifeng Shang, Linjun Yang, Fei Wang, Kwok-Ping Chan, and Xian-Sheng Hua. 2010. Real-Time Large Scale Near-Duplicate Web Video Retrieval. In *Proceedings of the 2010 ACM International Conference on Multimedia*. pp. 531–540.

[39] Jingkuan Song, Yi Yang, Zi Huang, Heng Tao Shen, and Richang Hong. 2011. Multiple Feature Hashing for Real-Time Large Scale Near-Duplicate Video Retrieval. In *Proceedings of the 2011 ACM International Conference on Multimedia*. pp. 423–432.

[40] Jingkuan Song, Yi Yang, Zi Huang, Heng Tao Shen, and Jiebo Luo. 2013. Effective Multiple Feature Hashing for Large-Scale Near-Duplicate Video Retrieval. *IEEE Transactions on Multimedia* 15, 8 (2013), pp. 1997–2008.

[41] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. 2015. Particular Object Retrieval with Integral Max-Pooling of CNN Activations. *arXiv preprint arXiv:1511.05879* (2015).

[42] Jianyi Wang, Xin Deng, Mai Xu, Congyong Chen, and Yuhang Song. 2020. Multi-Level Wavelet-Based Generative Adversarial Network for Perceptual Quality Enhancement of Compressed Video. In *Proceedings of the 2020 European Conference on Computer Vision*. pp. 405–421.

[43] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. 2021. Milvus: A Purpose-Built Vector Data Management System. In *Proceedings of the 2021 ACM International Conference on Management of Data*. pp. 2614–2627.

[44] Xintao Wang, Kelvin C.K. Chan, Ke Yu, Chao Dong, and Chen Change Loy. 2019. EDVR: Video Restoration with Enhanced Deformable Convolutional Networks. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. pp. 1954–1963.

[45] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. 2020. AnalyticDB-V: A Hybrid Analytical Engine towards Query Fusion for Structured and Unstructured Data. *Proceedings of the VLDB Endowment* 13, 12 (2020), pp. 3152–3165.

[46] Gengshen Wu, Jungong Han, Yuchen Guo, Li Liu, Guiguang Ding, Qiang Ni, and Ling Shao. 2018. Unsupervised Deep Video Hashing via Balanced Code for Large-Scale Video Retrieval. *IEEE Transactions on Image Processing* 28, 4 (2018), pp. 1993–2007.

[47] Xiao Wu, Alexander G Hauptmann, and Chong-Wah Ngo. 2007. Practical Elimination of Near-Duplicates from Web Video Search. In *Proceedings of the 2007 ACM International Conference on Multimedia*. pp. 218–227.
[48] Zhipeng Wu and Kiyoharu Aizawa. 2014. Self-Similarity-Based Partial Near-Duplicate Video Retrieval and Alignment. *International Journal of Multimedia Information Retrieval* 3, 1 (2014), pp. 1–14.
[49] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. 2019. VStore: A Data Store for Analytics on Large Videos. In *Proceedings of the 2019 European Conference on Computer Systems*. pp. 1–17.
[50] Artem Babenko Yandex and Victor Lempitsky. 2016. Efficient Indexing of Billion-Scale Datasets of Deep Descriptors. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 2055–2063.
[51] Wen Yang, Tao Li, Gai Fang, and Hong Wei. 2020. Pase: Postgresql Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension. In *Proceedings of the 2020 ACM International Conference on Management of Data*. pp. 2241–2253.
[52] Yuhao Zhang and Arun Kumar. 2019. Panorama: A Data System for Unbounded Vocabulary Querying over Video. *Proceedings of the VLDB Endowment* 13, 4 (2019), pp. 477–491.
[53] Guoying Zhao and Matti Pietikainen. 2007. Dynamic Texture Recognition using Local Binary Patterns with An Application to Facial Expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 6 (2007), pp. 915–928.