Fast Near-Duplicate Image Retrieval Method

Authors: He Cangping, Xu Tao, He Cangping

Date: 2022-10-10T00:00:00+00:00

Abstract

A substantial number of images on the Weibo platform are nearly identical, differing only in watermarks and clarity. To identify near-duplicate images from massive collections, this paper proposes a novel algorithm termed multi-layer fingerprinting. The multi-layer fingerprint comprises five strings and three types of real-valued vectors, wherein one string is utilized for rapid recall, while the remaining strings and vectors are employed to compute inter-fingerprint similarity. This approach exhibits minimal computational requirements while achieving high accuracy. Experimental results on a million-scale image repository from Weibo demonstrate that the multi-layer fingerprint attains a retrieval speed of 345 QPS and an accuracy of 97.69%.

Full Text

Preamble

A Fast Retrieval Method for Near-Duplicate Images
He Cangping, Xu Tao
cangping@staff.weibo.com, xutao@sugon.com
WEIBO.COM, Dawning Information Industry Co., Ltd.

Weibo hosts vast quantities of nearly identical images that differ only in watermarking and resolution. To identify these near-duplicate images from massive collections, this paper proposes a novel algorithm called multi-level fingerprinting. A multi-level fingerprint comprises five strings and three real-valued vectors, where one string enables rapid recall while the remaining strings and vectors compute inter-fingerprint similarity. This approach demands minimal computational resources while achieving high accuracy. Evaluated on a million-scale image repository from Weibo, the multi-level fingerprint method attains a retrieval speed of 345 QPS and an accuracy of 97.69%.

Keywords: image multi-level fingerprint, image fingerprint, image similarity

Abstract

The Weibo platform contains numerous near-duplicate images whose only differences are watermarks and resolution. To efficiently identify the most similar images, this paper proposes a multi-level fingerprint algorithm consisting of five character strings and three vectors. Tested on a dataset of one million images from Weibo, the multi-level fingerprint achieves 97.69% precision and 345 QPS.

Keywords: image fingerprint, multi-level fingerprint, image similarity

1 Introduction and Related Work

Weibo (weibo.com) is China' s leading social media platform, where billions of images are posted daily by massive numbers of users. A substantial proportion of these images are not original but rather copies of existing images that have lost some clarity or had watermarks added during replication, making them nearly identical. To better understand Weibo content, it is necessary to identify these near-duplicate images.

Existing image similarity algorithms fall into two main categories: traditional image processing methods that manually design image fingerprints, and deep learning methods that use trained models to extract feature vectors for similarity computation. Figure 1 [Figure 1: see original paper] illustrates the similar image retrieval pipeline, while Figure 2 [Figure 2: see original paper] shows the multi-level fingerprint retrieval process.

Traditional hash-based methods include average hashing (aHash), which resizes images to 8$\times$8$pixelsandgeneratesa64-characterstring; perceptualhashing(pHash), whichresizesto32\times32pi$
$characterstring; anddifferencehashing(dHash), whichresizesto9\times$8 pixels to create a 64-character string. Each character in these strings is either 0 or 1, with Hamming distance between two strings representing image similarity. Other hash algorithms include MarrHildrethHash based on the Marr-Hildreth edge operator, RadialVarianceHash based on Radon transform, BlockMeanHash based on block averaging, and ColorMomentHash based on color moments. Detailed descriptions of hash algorithms can be found in [9].

SIFT image matching [5, 6] extracts 128-dimensional descriptors from an image and matches keypoints between two images. SIFT is designed for identifying the same object across different times, resolutions, lighting conditions, and poses. Structural Similarity (SSIM) [7] computes a coefficient in the range [0,1] from two images, where identical images yield an SSIM value of 1. Reference [8] designs a convolutional neural network that takes two images simultaneously and computes their similarity. In engineering practice, more common approaches use neural networks to convert images into feature vectors, with recent popular models including SimCLR [2], MoCo [4], and SwAV [1]. The distance between two feature vectors represents image similarity, typically using cosine distance or Euclidean distance.

In the Weibo scenario, the image volume is enormous—potentially tens of bil-

lions. Using traditional hash fingerprints or neural network feature vectors for similar image retrieval consumes substantial computational resources. As shown in the retrieval flow (Figure 1), the input is an image represented as a three-dimensional array storing RGB intensity values. The real-time fingerprint computation module generates an image fingerprint f_0 from this array using some algorithm, producing either a fixed-length string or a fixed-length floating-point vector. The fingerprint distance computation module then calculates distances between f_0 and all fingerprints in the fingerprint library, selecting the image with the smallest distance and outputting its unique identifier pid.

The fingerprint distance computation module is resource-intensive. The fingerprint library is typically large and stored in database tables (e.g., MySQL tables). Computing distances between f_0 and library fingerprints involves Hamming distance, Euclidean distance, or cosine distance calculations, but databases have weak computational capabilities, making fingerprint distance computation time-consuming and resulting in low queries-per-second (QPS) support. High accuracy and low computational cost are difficult to achieve simultaneously. Hash-based fingerprints require little computation but have low accuracy. Neural network feature vectors achieve higher accuracy but demand significant algorithmic resources, typically requiring expensive GPU support that ordinary CPU servers cannot provide.

To address these challenges of high computational demand and long processing times, this paper proposes a multi-level fingerprint algorithm. A single image' s multi-level fingerprint contains five strings and three vectors: one string enables rapid recall, while the remaining strings and vectors are used for "fingerprint similarity computation" (see Figure 2). Tested on Weibo' s humor category images, the multi-level fingerprint achieves 97.69% accuracy and 345 QPS retrieval speed.

The remainder of this paper is organized as follows: Section 2 presents the whole-image fingerprint computation method (one string and three vectors); Section 3 describes the patch fingerprint computation method (four strings); Section 4 details the similarity computation between multi-level fingerprints; Section 5 presents experimental results on the Weibo image repository; and Section 6 concludes the paper.

2 Whole-Image Fingerprint

Given any image with height h and width w, denote its three channel values as matrices R = (r) w, G = (g) w, and B = (b) w, where the matrices have h rows and w columns. The matrix elements r , g , and b are integers in the range [0, 255]. Compute quantized values:

$$\bar{r}_{ij} = 4\left\lfloor \frac{r_{ij}}{4} \right\rfloor, \quad \bar{g}_{ij} = 4\left\lfloor \frac{g_{ij}}{4} \right\rfloor, \quad \bar{b}_{ij} = 4\left\lfloor \frac{b_{ij}}{4} \right\rfloor$$

where $\cdot$ denotes floor operation. The resulting grayscale matrices are denoted as $\bar{R} = (\bar{r}_{ij})_{hw}$, $\bar{G} = (\bar{g}_{ij})_{hw}$, and $\bar{B} = (\bar{b}_{ij})_{hw}$, with elements taking values in $\{0, 4, 8, \cdots, 252\}$. For example, when r $= 251$, $\bar{r}_{ij} = 4\lfloor 251/4 \rfloor = 248$.

Next, compute grayscale proportions. Specify real-valued thresholds $0 < \delta_1 < 100$ (typically $\delta_1 = 1$) and positive integer $3 \leq$ m ≤ 64 (typically m $= 5$). For each i $= 0, 4, 8, \cdots, 252$, let n denote the number of elements in $\bar{R}$ equal to i, and let u $= 100$n $/(hw)$ represent the proportion of grayscale value i. Sort u in descending order and select the top m proportions that satisfy u $\geq \delta_1$ to form vector:

$$u = (u_{i_1}, u_{i_2}, \ldots, u_{i_m}), \quad \text{where } u_{i_k} \geq \delta_1, k = 1, 2, \ldots, m$$

If fewer than m proportions meet the δ_1 threshold, pad with u $_1 = 0$. For example, u = (97.09, 0.531, 0, 0, 0) with corresponding indices (i_1, i_2, -1, -1, -1) = (252, 68, -1, -1, -1).

Next, adjust the element order of u. Specify real-valued threshold $\delta_2 > 0$ (typically $\delta_2 = 0.5$). For any two adjacent elements u $_1$ and u $_2$, if |u $_1$ - u $_2$| $< \delta_2$ and $i_1 > i_2$, swap u $_1$ and u $_2$. The resulting vector is denoted as $\bar{u} = (u_{j_1}, u_{j_2}, u_{j_3}, u_{j_4}, u_{j_5})$. For instance, given u = (7.058, 6.922, 5.824, 4.913, 4.518) with indices (12, 16, 20, 24, 8) and $\delta_2 = 0.5$, the adjusted vector becomes $\bar{u} = (7.058, 6.922, 5.824, 4.518, 4.913)$ with indices (j_1, j_2, j_3, j_4, j_5) = (12, 16, 20, 8, 24).

Apply the same method to matrix $\bar{G}$ to compute grayscale proportions, yielding vector $\bar{v} = (v_{j_{21}}, v_{j_{22}}, v_{j_{23}}, v_{j_{24}}, v_{j_{25}})$. Similarly, process matrix $\bar{B}$ to obtain $\bar{z} = (z_{j_{31}}, z_{j_{32}}, z_{j_{33}}, z_{j_{34}}, z_{j_{35}})$. If all elements of $\bar{u}$, $\bar{v}$, and $\bar{z}$ are zero, set $\delta_1 = \delta_1/2$ and recompute.

The vectors $\bar{u}$, $\bar{v}$, and $\bar{z}$ are called the image' s grayscale vectors.

Finally, concatenate the whole-image fingerprint f_0 by connecting the numbers with vertical bars and underscores, forming three segments. When m $= 5$:

$$f_0 = h\_w|\delta_1\_\delta 2|j_{11}\_j_{12}\_j_{13}\_j_{14}\_j_{15}\_j_{21}\_j_{22}\_j_{23}\_j_{24}\_j_{25}\_j_{31}\_j_{32}\_j_{33}\_j_{34}\_j_{35}$$

where "_" is a character, not a subscript. For example, the whole-image fingerprint f_0 for Figure 3 Figure 3: see original paper is:

```
421_{690}|1.0_0.5|0_{4_8}_{12}_{16}_{0_4}8_{12}_{16}_{0_4}8_{12}_{16}
```

3 Patch Fingerprint

Let $h_1 = \lfloor h/2 \rfloor - 10$, $h_2 = h - h_1$, $w_1 = \lfloor w/2 \rfloor$, $w_2 = w - w_1$, $w_3 = \lfloor w/3 \rfloor$, and $w_4 = w - w_3$. Divide the image into four patches as shown in Figure 3(b): top-left, top-right, bottom-left, and bottom-right, with dimensions $h_1 \times w_1$, $h_1 \times w_2$,

$h_2 \times w_3$, and $h_2 \times w_4$ respectively. For example, Figure 3(b) has dimensions h = 421, w = 690, yielding $h_1 = 200$, $h_2 = 221$, $w_1 = 345$, $w_2 = 345$, $w_3 = 172$, $w_4 = 518$.

For the top-left patch, compute its grayscale vector and concatenate the indices into string:

$$f_1 = j_{11}\_j_{12}\_j_{13}\_j_{14}\_j_{15}\_j_{21}\_j_{22}\_j_{23}\_j_{24}\_j_{25}\_j_{31}\_j_{32}\_j_{33}\_j_{34}\_j_{35}$$

Unlike f_0, f_1 contains only grayscale values, not height, width, δ_1, or δ_2. Apply the same operation to the top-right, bottom-left, and bottom-right patches to obtain strings f_2, f_3, and f_4.

The combination of f_0 through f_4, $\bar{u}$, $\bar{v}$, and $\bar{z}$ constitutes the image' s multi-level fingerprint.

4 Image Similarity

Given two images p , i = 0, 1, with multi-level fingerprints denoted as f_0^i through f_4^i, $\bar{u}^i$, $\bar{v}^i$, and $\bar{z}^i$, their similarity is computed as follows. Initialize s = 0. Specify real-valued threshold $0 < \delta_3 \leq 0.2$. If $f_0^0 \neq f_0^1$, then for j = 1, 2, 3, 4:

$$s_j = \begin{cases} 0.2, & \text{if } f_j^0 \neq f_j^1 \\ 0, & \text{if } f_j^0 = f_j^1 \end{cases}$$

Let $s_5 = \|\bar{u}^0 - \bar{u}^1\|_1 + \|\bar{v}^0 - \bar{v}^1\|_1 + \|\bar{z}^0 - \bar{z}^1\|_1$. The final similarity is:

$$s = \begin{cases} 0.2 - s_5 + \sum_{j=1}^{4} s_j, & \text{if } s_5 < \delta_3 \\ 0, & \text{if } s_5 \geq \delta_3 \end{cases}$$

For example, let p_0 be Figure 3(a) and p_1 be Figure 3(c). With parameters m = 5, $\delta_1 = 1.0$, $\delta_2 = 0.5$, $\delta_3 = 0.2$, the multi-level fingerprints are:

```
f$^{0}$$_{0}$ = 421_{690}|1.0_0.5|0_{4_8}_{12}_{16}_{0_4}8_{12}_{16}_{0_4}8_{12}_{16}
f$^{0}$$_{1}$ = 0_{4_8}_{12}_{16}_{0_4}8_{12}_{16}_{4_8}0_{12}_{16}
f$^{0}$$_{2}$ = 0_{4_8}_{12}_{16}_{0_4}8_{12}-1_{0_4}8_{12}-1
f$^{0}$$_{3}$ = 0_{4_8}_{12}_{28}_{0_4}8_{12}_{16}_{4_0}_{36}8_{40}
f$^{0}$$_{4}$ = 0_{4_8}_{112}_{116}_{0_4}8_{40}_{44}_{0_4}8_{24}_{28}

f$^{1}$$_{0}$ = 421_{690}|1.0_0.5|0_{4_8}_{12}_{16}_{0_4}8_{12}_{16}_{0_4}8_{12}_{16}
f$^{1}$$_{1}$ = 0_{4_8}_{12}_{16}_{0_4}8_{12}_{16}_{4_8}0_{12}_{16}
f$^{1}$$_{2}$ = 0_{4_8}_{12}_{16}_{0_4}8_{12}-1_{0_4}8_{12}-1
f$^{1}$$_{3}$ = 0_{4_8}_{12}_{28}_{0_4}8_{12}_{16}_{4_0}_{36}8_{40}
f$^{1}$$_{4}$ = 0_{4_8}_{112}_{116}_{0_4}8_{40}_{44}_{0_4}8_{24}_{28}
```

```
ū$^{0}$ = (33.167, 12.927, 5.696, 3.042, 2.124)
v̄$^{0}$ = (39.312, 12.524, 5.579, 3.160, 2.829)
z̄$^{0}$ = (24.216, 14.986, 9.932, 4.325, 2.922)

ū$^{1}$ = (33.167, 12.927, 5.696, 3.042, 2.124)
v̄$^{1}$ = (39.313, 12.524, 5.578, 3.159, 2.828)
z̄$^{1}$ = (24.216, 14.986, 9.931, 4.326, 2.919)
```

The two images differ only by a watermark, and the similarity s = 0.99997 accurately reflects this difference.

5 Experimental Results

From Weibo images dated 20220803-20221007, there are 1,192,348 images with concept tags. Using parameters m = 5, $\delta_1 = 1$, $\delta_2 = 0.5$, we computed their multi-level fingerprints and stored them in a cloud MySQL database table, along with an md5 field containing each image file' s MD5 hash. Images with identical MD5 values are considered the same, with only one record retained in the table.

For any given image p_0, the detailed retrieval process for near-duplicate images is as follows. Compute p_0' s fingerprint, denoted as f_0^0 through f_4^0, $\bar{u}^0$, $\bar{v}^0$, and $\bar{z}^0$. Search the database table for images whose f_0 string exactly matches f_0^0. This lookup operation requires only string comparison, which is fast and can be parallelized. From the table, retrieve t images denoted as p , i = 1, 2, ⋯, t, with whole-image fingerprints f_0^i. Then compute the similarity between p_0 and each p using the algorithm from Section 4, outputting the image with maximum similarity.

Randomly selecting a batch of images from the humor category dated 20221006-20221007 for retrieval and manual verification, we obtain the following results. Using conditions $\delta_3 = 0.01$ and s ≥ 0.4 yields 288 correct retrievals, 0 errors, and 100% accuracy. Using $\delta_3 = 0.02$, 0.03, and 0.2 produces accuracies of 99.09%, 97.69%, and 91.78% respectively, as detailed in Table 1 . The retrieval speed reaches 345 QPS.

6 Conclusion and Discussion

The multi-level fingerprint' s advantages are low computational requirements and fast recall speed. Computing multi-level fingerprints involves only simple frequency statistics, unlike neural network models that require massive matrix-vector multiplications. The whole-image fingerprint f_0 enables rapid recall of roughly similar images, filtering out the vast majority of unlikely candidates—reducing the candidate set by up to five orders of magnitude.

The fingerprint similarity computation in Section 4 can be replaced with traditional hash fingerprint similarity or neural network feature vector similarity without significantly impacting retrieval speed, though accuracy may be affected.

References

[1] Mathilde Caron et al. "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments". In: CoRR abs/2006.09882 (2020). arXiv: 2006.09882. URL: https://arxiv.org/abs/2006.09882.

[2] Ting Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations" . In: (2020).

[3] Nicolas Courty, Rémi Flamary, and Mélanie Ducoffe. "Learning Wasserstein Embeddings" . In: (2017). arXiv: 1710.07457. URL: https://arxiv.org/abs/1710.07457.

[4] Kaiming He et al. "Momentum Contrast for Unsupervised Visual Representation Learning" . In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020, pp. 9726–9735. DOI: 10.1109/CVPR42600.2020.

[5] David Lowe. "Distinctive Image Features from Scale-Invariant Keypoints" . In: International Journal of Computer Vision 60 (Nov. 2004), pp. 91–. DOI: 10.1023/B:VISI.0000029664.99615.94.

[6] D.G. Lowe. "Object recognition from local scale-invariant features" . In: Proceedings of the Seventh IEEE International Conference on Computer Vision. Vol. 2. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[7] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity" . In: IEEE Transactions on Image Processing 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[8] Sergey Zagoruyko and Nikos Komodakis. "Learning to Compare Image Patches via Convolutional Neural Networks" . In: CoRR abs/1504.03641 (2015). arXiv: 1504.03641. URL: http://arxiv.org/abs/1504.03641.

[9] Christoph Zauner. "Implementation and Benchmarking of Perceptual Image Hash Functions" . In: 2010.

Note: Figure translations are in progress. See original paper for figures.