

Summary

陈骞

1 From Image Hashing to Video Hashing[1](2020/03/21)

In this work, the author propose a frame hash based video hash construction framework, which reduces a video hash design to an image hash design. Perceptual hashing is a technique for the identification of multimedia content, which does not vary even if the content hash undergone some incidental distortion. For a video hash, since a video is essentially a sequence of frames, in order to simplify the work, a straight-forward way to construct a video hash is to concatenate hash values of video frames. This approach has several advantages: 1) existing image hash algorithms can be used; 2) no need to store an entire video; 3) the video hash performance can be estimated from the image hash performance. But it is based on the assumption that **if two videos are perceptually different, then each pair of extracted frames are also likely to be perceptually different.**

1.1 Video Hash Construction

When comparing a pair of hash values, a decision is made from two hypotheses: 1) \mathbb{H}_0 -they correspond to different contents; 2) \mathbb{H}_0 -they correspond to similar contents. Only if the distance d below a threshold t , the contents are judged as similar. The performance can be quantified by the *detection rate* " p_d " - probability $\{d < t | \mathbb{H}_1\}$, and the *false positive rate* " p_f " - probability $\{d < t | \mathbb{H}_0\}$.

Assumed that N frames are extracted from the input video. For each frame, a frame hash value $h_i, i = 1, \dots, N$ is computed by an image hash algorithm \mathbb{A} . Then the video hash $H = \{h_1 | \dots | h_N\}$ is the concatenation of frame hash values. When two video hash values H_1 and H_2 are compared, each frame hash h_{i1} of H_1 is compared with the corresponding h_{i2} of H_2 ; the video files are judged as similar if at least T frame pairs are similar. The ratio $T/N \in (0.5, 1)$ is a constant, denoted by α .

Assume \mathbb{A} has performance $\langle p_d, p_f \rangle$ and the N frames are independent, the detec-

tion rate and the false positive rate of the overall scheme, denoted by P_d and P_f , can be formulated as:

$$P_d = \sum_{k=T}^N \binom{N}{k} \cdot p_d^k \cdot (1 - p_d)^{N-k} \quad (1.1)$$

$$P_f = \sum_{k=T}^N \binom{N}{k} \cdot p_f^k \cdot (1 - p_f)^{N-k} \quad (1.2)$$

For reasonable values of p_d and p_f , i.e., $0 < p_f \ll p_d < 1$, P_d increases with N towards 1, and P_f decreases with N towards 0. Equations (1.1)-(1.2) can be adapted to cope with dependent frames. The idea is to divided extracted frames into groups and assume elements in each group depend on each other. In order to model the non-repeatability, a parameter $\beta \in (0.5, 1]$ is added to (1.1)-(1.2). New formulas can be derived by placing p_d and p_f with:

$$p'_d = \beta p_d + (1 - \beta) p_c(t)$$

$$p'_f = \beta p_f + (1 - \beta) p_c(t)$$

where $p_c(t)$ is the general collision rate, which is the probability that arbitrary two hash value's Hamming distance is no less than t and can be defined as follow:

$$p_c(t) = \sum_{k=t}^n \binom{n}{k} \cdot p^k \cdot (1 - p)^{n-k}$$

where n is the hash length and p is the probability that two bits coincide.

1.2 A Frame Hash Algorithm

An extracted frame is first converted to gray and resized to a canonical size 512×384 . The output is filtered by an averaging filter and a median filter. Histogram equalization is then performed. Divide image into 256×192 pixel blocks with 50% overlapping. Since an image is essentially a certain allocation of pixels, the content can be characterized by the statistics of pixel values. So the standard deviation, the third- and the fourth-order auto-cumulate are used to describe the feature of frames. The latter two are defined as:

$$C_{3X}(k, l) = E[x(n)x(n+k)x(n+l)]$$

$$C_{4X}(k, l, m) = E[x(n)x(n+k)x(n+l)x(n+m)]$$

$$- C_{2X}(k)C_{2X}(l-m) - C_{2X}(l)C_{2X}(k-m) - C_{2X}(m)C_{2X}(k-l)$$

where X is a zero-mean vector, E means expectation, and $C_{2X}(k) = E[x(n)x(n+k)]$. The parameters k , l and m are decided by a secure pseudo-random number generator (PRNG). A reason to use higher-order cumulate is that they are immune to Gaussian noise.

Each block is allocated with 16 bits for quantization. 3 bits for the standard deviation, 6 bits for the third-order cumulate, and 7 bits for the fourth-order cumulate. There are 9 blocks, so a hash value has 144 bits. After quantization, the binary output is XORed with a dither sequence generated by the secure PRNG.

1.3 New Performance Metrics

Although the ROC curve tells the performance, sometimes it is useful to measure robustness and discriminability separately. In the following, two metrics are proposed for this purpose, defined as the *robustness index* and the *discrimination index*. The robustness can be represented by the $p_d(t)$ curve, which is the detection rate p_d versus the threshold t . Similarly, the discriminability is represented by the $p_f(t)$ curve. The discrimination index is defined as:

$$D(p_c||p_f) = \sum_t p'_c(t) \frac{p'_c(t)}{p'_f(t)}$$

where $p'_c(t) = p_c(t) / \sum p_c(t)$ and $p'_f(t) = p_f(t) / \sum p_c(t)$ are normalized versions of $p_c(t)$ and $p_f(t)$.

1.4 Summarization

In this work, the author proposed that a perceptual video hash can be constructed from the perceptual hash values of video frames. **This idea is to directly concatenate hash values of video frames to construct a video hash.** This method is relatively simple and does not need to design a complex hash structure, but there is no doubt that this method of completely viewing the video as a picture set must lose some of the video's attributes, such as spatial and temporal structure, logical structure.

But in this work, I think maybe it is very useful for shot boundary detection that a frame hash can be calculated by statistics. According to the simulation, this method is robust to disturbances such as rotation and Gaussian noise, etc. Maybe it can be used to detect shot boundary.

2 Large-Scale Video Hashing via Structure Learning[2](2020/03/21)

In this work, the author propose a supervised method that explores the structure learning techniques to design efficient hash function by the structure regularization, which exploits the common local visual patterns occurring in video frames that are associated with the same semantic class, and simultaneously preserves the temporal consistency over successive frames from the same video.

The key idea for learning-based hashing is to leverage data properties or human supervision to derive compact yet accurate hash codes. Most the existing video hashing methods

cannot explicitly encode the specific structure information in video clips. To address the above problems, the author proposed to explore the structure information to design novel video hashing methods. Two important types of structure information, *Discriminative Local Visual Commonality* and *Temporal Consistency*, are used.

2.1 Assumption & Definition

Training video collection: $\mathcal{X} = \{X_i, y_i\}_{i=1}^N$ with N videos, where $y_i \in \{0, 1\}$ is the label of video X_i .

Video: $X_i = [x_{i1}, \dots, x_{ij}, \dots, x_{i, n_i}]$ is a video consisting of n_i successive frames, where $x_{ij} \in \mathbb{R}^d$ is the feature vector of the j -th frame of video X_i with d being the feature dimensionality.

Assumption 1.

All frame features in \mathcal{X} have been normalized with zero mean.

Hash function: Given a video frame x , the k -th hash function ($k = 1, \dots, K$) can be defined as:

$$h_k(x) = \text{sgn}(w_k^T x + b_k)$$

where $w_k \in \mathbb{R}^d$ is a hash hyperplane and b_k is the intercept.

Hash code: $c_k(x) = (1 + h_k(x))/2$

Hamming distance: The Hamming distance between the hash codes of two frames x_{ia} and x_{jb} from two videos X_i and X_j can be defined as:

$$d(x_{ia}, x_{jb}) = \sum_{k=1}^K (c_k(x_{ia}) - c_k(x_{jb}))^2 = \frac{1}{4} \sum_{k=1}^K (h_k(x_{ia}) - h_k(x_{jb}))^2$$

Video distance: The Hamming distance between videos X_i and X_j is:

$$D(X_i, X_j) = \frac{1}{n_i n_j} \sum_{a=1}^{n_i} \sum_{b=1}^{n_j} d(x_{ia}, x_{jb})$$

which means that the Hamming distance between two videos equals to the average Hamming distance of each pair of frames.

Assumption 2.

Most of frames within a video should be useful for measuring the distance between videos, and similar pairwise metric methodology has been proven to be effective.

Then relax the distance function by replacing sgn with its signed magnitude and

rewrite the distance as:

$$\begin{aligned} D(X_i, X_j) &= \frac{1}{4n_i n_j} \sum_{a=1}^{n_i} \sum_{b=1}^{n_j} \sum_{k=1}^K (w_k^T x_{ia} - w_k^T x_{jb})^2 \\ &= \frac{1}{4n_i n_j} \sum_{a=1}^{n_i} \sum_{b=1}^{n_j} (x_{ia} - x_{jb})^T W W^T (x_{ia} - x_{jb}) \end{aligned}$$

where $W = [w_1, \dots, w_K] \in \mathbb{R}^{d \times K}$ is a matrix consisting of the coefficients of all hash functions.

2.2 Cost function & Explanation

Formulate the following objective which learns the hash functions by minimizing a structure-regularized cost function:

$$\min_W \sum_{i,j=1}^N l(\hat{y}_{ij}, D(X_i, X_j)) + \lambda \|W\|_{2,1} + \gamma \sum_{i=1}^N \sum_{t=1}^{n_i-1} \|W^T x_{i,t} - W^T x_{i,t+1}\|_\infty$$

where $\lambda, \gamma > 0$ are two parameters balancing the three competing terms, $\|W\|_{2,1} = \sum_{i=1}^d \sqrt{\sum_{j=1}^K W_{ij}^2}$ is the $l_{2,1}$ -norm, and $\|W^T x_{i,t} - W^T x_{i,t+1}\|_\infty = \max_{j=1, \dots, K} \{|d_{i,t}^j|\}$ is the l_∞ -norm, in which $s_{i,t} = W^T x_{i,t} - W^T x_{i,t+1}$ and $s_{i,t}^j$ denotes the j -th entry of vector $s_{i,t}$, $l(\cdot, \cdot)$ is an empirical loss function with $\hat{y}_{ij} = 1$ if $y_i = y_j$ and $\hat{y}_{ij} = -1$ otherwise.

First term: Define $l(\hat{y}_{ij}, D(X_i, X_j)) = \max(0, \hat{y}_{ij}(D(X_i, X_j) - \delta))$ where δ is a threshold. When minimizing this, $\hat{y}_{ij} = 1$ leads to $D(X_i, X_j) \leq \delta$ while $\hat{y}_{ij} = -1$ leads to $D(X_i, X_j) > \delta$. It enforces that videos with the same category label should be close to each other while videos with different category labels should be far apart.

Second term: The minimization of $\|W\|_{2,1}$ ensures only a small number of rows in matrix W are non-zero.

Third term: The minimization of $\|W^T x_{i,t} - W^T x_{i,t+1}\|_{inf ty}$ ensures the hash vectors of two successive frames as similar as possible, i.e., encouraging the maximum absolute value of the entry-wise differences between two hash vectors to be zero. This accounts for the preservation of the temporal structure in the hash codes generated for all frames of a video.

2.3 Solve

Concatenate all pairwise frame differences of the training video collection into a matrix $\hat{X} \in \mathbb{R}^{d \times T}$ with the total number of successive frame pairs $T = \sum_{i=1}^N (n_i - 1)$. Then the original objective function can be rewritten as:

$$\min_W \sum_{i,j=1}^N l(\hat{y}_{ij}, D(X_i, X_j)) + \lambda \|W\|_{2,1} + \gamma \sum_{i=1}^T \|(W^T \hat{X})_i\|_\infty$$

where $(W^T \hat{X})_i \in \mathbb{R}^{K \times 1}$ denotes the i -th column in matrix $(W^T \hat{X})$.

Decompose the objective function as the sum of the following two terms:

$$f(W) = \sum_{i,j=1}^N l(\hat{y}_{ij}, D(X_i, X_j))$$

$$r(W) = \lambda \|W\|_{2,1} + \gamma \sum_{i=1}^T \|(W^T \hat{X})_i\|_\infty$$

Then based on Nesterov's smoothing approximation method, $r(W)$ can be approximated by the following smoothing function:

$$r_\mu(W) = \lambda \sum_{i=1}^d h_{i,\mu}(W) + \gamma \sum_{i=1}^T g_{i,\mu}(W)$$

with the definitions:

$$h_{i,\mu}(W) := \max_{\|v\|_2 \leq 1} \langle w^i \rangle^T, v \rangle - \frac{\mu}{2} \|v\|_2^2$$

$$g_{i,\mu}(W) := \max_{\|\mu\|_1 \leq 1} \langle (W^T \hat{X})_i, \mu \rangle - \frac{\mu}{2} \|\mu\|_2^2$$

where μ is a positive smoothness parameter to control the accuracy of the approximate, $\langle \cdot, \cdot \rangle$ denotes the inner product operator, w^i denotes the i -th row of matrix W . v and μ are respectively a vector of auxiliary variables associated with w^i and $(W^T \hat{X})_i$.

So we have

$$\nabla f(W) = \sum_{i,j=1}^N \nabla l(\hat{y}_{ij}, D(X_i, X_j))$$

$$\nabla r_\mu(W) = \lambda v(W) + \gamma \sum_{i=1}^T \hat{X}_i \mu^T ((W^T \hat{X})_i)$$

in which \hat{X}_i denotes the i -th column of matrix \hat{X} , $v(W) = [v(w^1), \dots, v(w^d)]^T \in \mathbb{R}^{d \times K}$, and $\nabla l(\hat{y}_{ij}, D(X_i, X_j))$ can be calculated as:

$$\nabla l(\hat{y}_{ij}, D(X_i, X_j)) = \begin{cases} 0, & \text{if } \hat{y}_{ij}(D(X_i, X_j) - \delta) \leq 0 \\ \hat{y}_{ij} \Omega_{ij} W, & \text{otherwise} \end{cases}$$

where

$$\Omega_{ij} = \frac{1}{2n_i n_j} \sum_{a=1}^{n_i} \sum_{b=1}^{n_j} (x_{ia} - x_{jb})(x_{ia} - x_{jb})^T$$

.

Then get that $\nabla F_\mu(W)$ is Lipschitz continuous with constant

$$L_{F_\mu} = \left\| \sum_{i,j=1}^N \Omega_{ij} \right\|_2 + \frac{1}{\mu} (\lambda \times d + \gamma \times T)$$

Employ to optimize $F_\mu(W)$, which can be described in Fig.1.

Algorithm 1 Solving Problem of Eq. (13) by APG

- 1: **Input:** $X_i \in \mathbb{R}^{d \times n_i}$, $y_i \in \{0, 1\}$, $i = 1, \dots, N$, \hat{X} , λ , γ , δ and μ .
- 2: **Initialize:** Calculate L_{F_μ} based on Eq. (18), randomly initialize $W^{(0)}$, $Z^{(0)} \in \mathbb{R}^{d \times K}$, and $\eta^{(0)} \leftarrow 0$, $t \leftarrow 0$.
- 3: **repeat**
- 4: $\alpha^{(t)} = (1 - \eta^{(t)})W^{(t)} + \eta^{(t)}Z^{(t)}$.
- 5: Calculate $\nabla F_\mu(\alpha^{(t)})$ based on Eq. (14).
- 6: $Z^{(t+1)} = Z^{(t)} - \frac{1}{\eta^{(t)}L_{F_\mu}}\nabla F_\mu(\alpha^{(t)})$.
- 7: $W^{(t+1)} = (1 - \eta^{(t)})W^{(t)} + \eta^{(t)}Z^{(t+1)}$.
- 8: $\eta^{(t+1)} = \frac{2}{t+1}$, $t \leftarrow t + 1$.
- 9: **until** Converges.
- 10: **Output:** $W^{(t)}$.

Fig. 1: Accelerated Proximal Gradient

2.4 Summarization

This article incorporates the structural information of the video into the hash function generation process of the video. The idea of ??this article is basically the same as that of the article generated by the hash function read a year ago. It is a method of adding a constraint to the objective function so that the hash function has a certain property. For the objective function, it is easy to solve by some proven algorithm after a certain relaxation and smooth approximation are used. The general idea of ??this type of method is basically this.

In addition, for this article, I think there is a relatively definite improvement direction. The first term of the objective function of the article can only be labeled for a event, which is obviously very biased for videos with multiple events. So maybe we can improve the first term, by using a bag-of-words model and so on, to make the hash function effective for videos of multiple events. In addition, the third term of the objective function may also be greatly improved. Here, it is aimed that two consecutive frames at the a event are approximately the same. For multiple events, it is obvious to modify. If two frames are under one event, two consecutive frames are similar as soon as possible. But if the two frames are under different events, the distance should be as large as possible, but not infinite.

3 Deep Video Hashing[3](2020/03/25)

In this work, the author propose a new deep video hashing (DVH) method for scalable video search, which learns binary codes for the entire video with a deep learning framework. Similar to other paper, the method follow the principle of preserving locality: **1) the**

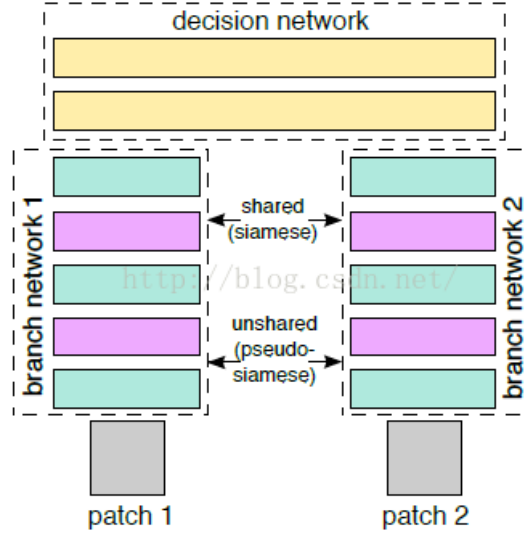


Fig. 2: Siamese network

distance between a feature pair obtained at the top layer is small if they are in the same class, and large if they are from different classes, 2) the quantization loss between the real-valued features and the binary codes is minimized.

3.1 Structure

1. Basic Structure: To learn the parameters in the deep network discriminatively, the author employ the Siamese network, as shown in Fig.2. This network consists of two branches, what we want to do is compare the similarity of the two pictures patch1 and patch2. So the general idea of the Siamese network is to let patch1 and patch2 go through the network, extract feature vectors, and then compare the two feature vectors. Finally, do a similarity loss function for network training.

2. Temporal Information: In order to exploit temporal information in deep networks, the author perform average pooling operations across frames between the fully-connected layers. The author describe three deep networks with various feature pooling architectures:

1. Early Fusion: Firstly, pass image frames through the convolution and pooling layers and then fuses the information at the first fully connected layer immediately, as shown in Fig.3.(a).

2. Late Fusion: Firstly, pass image frames through the convolution and pooling layers up to the other fully-connected layers and then fuses the information at the last fully connected layer, as shown in Fig.3.(b).

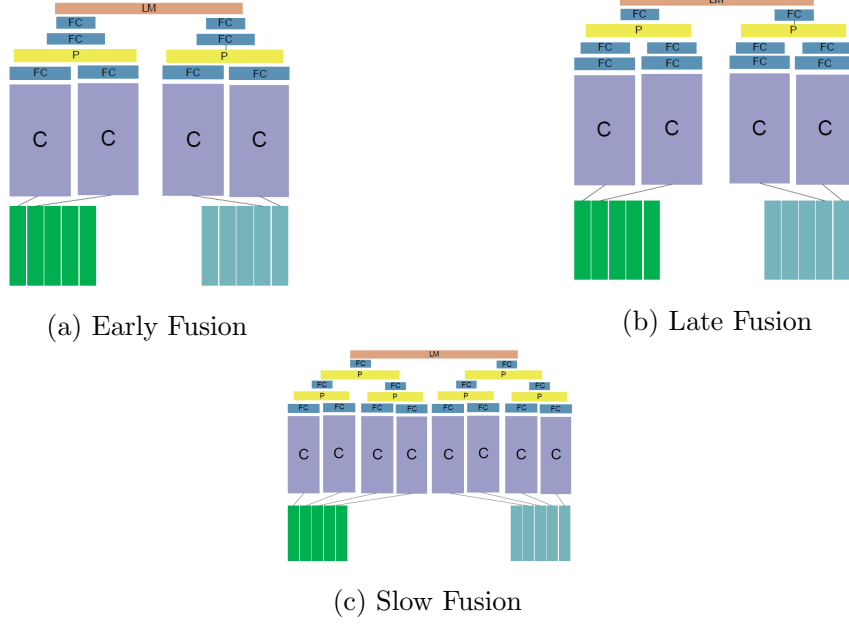


Fig. 3: Fusion Module.

3. Slow Fusion: Firstly, pass image frames through the convolution and pooling layers and then fused in a hierarchical manner such that smaller temporal windows are used as it approaches the top layer, as shown in Fig.3.(c).

Given a fixed frame size p , the set of image frames $I_u \in \mathbb{R}^{p \times h \times w \times 3}$ pass through the convolution and pooling layers with fully connected layers at the end. By letting $s(\cdot)$ be the output at the last fully connected layer where it contains K nodes, the binary code for the set of image frames of the i -th video is computed as follows:

$$b_u = \text{sgn}(s(I_u))$$

3.2 Objective function

Given two sets of image frames, I_u and I_v , the network should minimize the intra-class variation and maximize the inter-class variation of the binary feature representation at the top layer of these two networks, simultaneously, which can be formulated as the following constraints:

$$d_{u,v}(b_u, b_v) \leq \theta_1, \quad \text{if } y_u = y_v$$

$$d_{u,v}(b_u, b_v) \geq \theta_2, \quad \text{if } y_u \neq y_v$$

where θ_1 and θ_2 are the small and large thresholds, respectively.

Then reformulate the constraints:

$$\delta_{u,v}(\theta - d_{u,v}(b_u, b_v)) > 1$$

where $\theta_1 = \theta - 1$ and $\theta_2 = \theta + 1$, and $\delta_{u,v} = 1$ means that u and v are from the same class, and $\delta_{u,v} = -1$ indicates that they are from different classed.

This leads to the following objective function:

$$\mathcal{J}_1 = f(1 - \delta_{u,v}(\theta - d_{u,v}(b_u, b_v)))$$

where $f(z)$ is a generalized logistic loss function:

$$f(z) = \frac{1}{\rho} \log(1 + \exp(\rho z))$$

where ρ is the sharpness parameter set to 10 and θ is the threshold parameter set to $K/4$.

The second objective is to ensure efficient binary codes by minimizing the quantization loss between real-valued codes and binary codes as follows:

$$\mathcal{J}_2 = \|s(I_u) - b_u\|_F^2 + \|s(I_v) - b_v\|_F^2$$

Hence, the final objective function for DVH is formulated as:

$$\begin{aligned} \min_{b_v, b_u} \mathcal{J} &= \mathcal{J}_1 + \lambda \mathcal{J}_2 \\ &= f(1 - \delta_{u,v}(\theta - d_{u,v}(b_u, b_v))) + \lambda(\|s(I_u) - b_u\|_F^2 + \|s(I_v) - b_v\|_F^2) \end{aligned}$$

where \mathcal{J}_1 exploits the discriminative information, \mathcal{J}_2 minimizes the quantization loss, and λ is a constant parameter which balances the two criterions.

Then, use the standard mini-batch gradient descent and back-propagation to solve the optimization problem. The derivative of \mathcal{J} with respect to h_u and h_v :

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial h_u} &= f'(z) \delta_{u,v}(h_u - h_v) + \lambda(h_u - b_u) \\ \frac{\partial \mathcal{J}}{\partial h_v} &= f'(z) \delta_{u,v}(h_v - h_u) + \lambda(h_v - b_v) \end{aligned}$$

After extraction, the over-all hamming distance for each pair of binary codes as follows:

$$D_H(X_i, X_j) = \frac{1}{g_i g_j} \sum_{u=1}^{g_i} \sum_{v=1}^{g_j} d_H(b_u, b_v)$$

where g_i and g_j is the number of frame sets for videos X_i and X_j .

The parameters in the fully connected layers are initialized using the Xavier initialization. Deep architecture and experiments were implemented under the MatConvNet framework. The learning rate, momentum, and weight decay were set to 0.002, 0.9 and 0.0001, respectively. At the training stage, video pairs chosen randomly pass through the network.

3.3 Summarization

The essence of this article is the same as that of the previous paper[2], but it uses different ways to solve the objective function. In the previous paper[2], the hash process was defined as a linear projection process. In comparison, it caused a lot of information loss. So in this article, the author introduces a non-linear convolution layer into the entire solution process. By performing several nonlinear transformations with a discriminative criterion, more robust visual representation can be obtained. In addition, the author also hopes to introduce the time relationship between frames into the entire solution process, making the hash process more robust. By performing the temporal pooling, we can implicitly exploit the relevant frames and extract a balance of global and local information from video frames.

However, I think there are actually some problems about the method of using temporal information in the paper. No matter what kind of fused method is used, the essence is to average pool the feature maps of two frames. I don't know what is the difference between this and directly concentrate the feature maps of two frames together. It feels like this average pooling is like taking a transition between two frames as a feature of the two frames. As the author points out at the end of the article, how to use more complex temporal information, such as LSTM, to improve network performance is a future direction of research.

4 Some of my ideas and plans(2020/03/25)

4.1 1

About the paper[1] apply to the SBD, I do not think it can be used to detect shot boundary directly. What I am interested in is the hash algorithm for frames in the paper[1]. According to the paper, since an image is essentially a certain allocation of pixels, the content can be characterized by the statistics of pixel values. In this way, the method can be immune to Gaussian noise. And I think the method maybe have rotation invariance. So whether we can exploit the feature of the algorithm. Since last semester undergraduate student Wang Chenxin also wanted to continue to do some content. I wonder if I can first conceive an algorithm and let him implement it, then continuously adjust it according to the actual performance. If the effect is good, we can try to organize it into an article.

The algorithm I think can be done as follow:

1)First use segmentation to reduce our data dimension: The video is divided into small divisions corresponding to the number of frames according to the *fps* index.

Then compare the similarity between the first frame and the last frame of the segmentation to determine whether there is a possibility of a shot boundary in the segmentation.

2)Extracting local feature using SIFT or Harris: The key points found by SIFT are points that are very prominent and will not change due to factors such as lighting, affine transformation, and noise, such as corner points, edge points, bright points in dark areas, and dark points in bright areas. If Harris is used to extract feature points, maybe SIFT should be used to generate descriptor. Although there are many ways to extract local features, we can try different methods and choose the best extraction method. Maybe optical flow technology can also be applied.

3)Extracting global feature using Algorithm[1]: For different transitions types, we can use different features for judgment. For example, cut transitions, we can directly judge by global features, but for boundaries such as dissolve, we may need to judge based on the number of feature points and the changes in the descriptor of the feature points. The specific rules need to be further discussed through experiments.

4)Use some rules to judge the features: Depending on the type of features generated above, maybe we can make a combination, which can be added and subtracted after weighting or directly concentrate together. If we can generate a sequence, maybe we can use some models in the time series to judge the results. After all, video can be viewed as a high-dimensional time series.

The above is a general idea that I got after rereading the previous articles. I think the specific process of each step can be further discussed in the experiment.

4.2 2

In response to the question in the paper[2], I got some ideas after reading paper[3]. At present, most of the articles dealing with the hashing of videos focus on the video frames rather than the entire video itself. The method used is classified according to the way of solving: solved by optimization algorithms and solved by deep learning networks. At present, I think that it is not appropriate for us to directly solve with deep learning networks, mainly because this method is too variable, and it is difficult to improve from the network itself. Most of the work is how to use the existing network. Therefore, I think it is not realistic for us to directly propose a new network.

Conversely, if an optimization algorithm is used to solve the problem, the problem is how to add our requirements to constraints and add to the problem essentially. After adding to the problem, we can always find some ways to solve the optimization problem. This question is consistent with the idea of ??preserving local hashes that we have learnt at the beginning of last semester, so we may be able to do some work on this content. My

current thought is mentioned in the last summary, we can design an objective function for multiple events. Remember that there was a previous article on deep hashing that used hierarchical labels in the process of optimizing deep hashing. Maybe, we can do a hierarchical classification on the labels, divide the video according to a certain hierarchical labels, and then use the local similarity preservation to construct the objective function.

If you think these two ideas are feasible, I plan to let Wang Chen Yixin first sort out the algorithms mentioned in the first section to check the feasibility. Then I plan to try to implement the algorithm in paper[2] and then look for the recent paper to see their starting point for building the objective function.

5 Spatio - Temporal Transform Based Video Hashing[4](2020/03/28)

Due to time, I have only read the theoretical analysis part of this article, and the content after the experiment has not been read in detail.

For the purpose that the video sequence must be collapsed into a short fingerprint using a robust hash function based on signal processing operations, the author propose two robust hash algorithms for video based both on the Discrete Cosine Transform (DCT), one on the classical basis set and the other on a novel randomized basis set (RBT).

In generally, since the resulting hash value is highly sensitive to every single bit of the input, these functions are extremely fragile and cannot be adopted for hashing multimedia data. For insensitive to certain transformations, such as compressed version and low-pass filtered, **an alternate way to compute the hash is needed for multimedia applications, where the hash function results in the same output value unless the underlying content is significantly changed.**

A hash function for video should occupy the properties of robustness and uniqueness. **Robustness** implies that the hash function should be insensitive to perturbations, non-malicious modifications caused by “mild” signal processing operations that in total do not change the content of the video sequence. **Uniqueness** property implies that the hash functions are statistically independent for different content, so that any two distinct video clips result in different and apparently random hash values.

According to the principle of the existing algorithms from frame hash to video hash, **key-frame based hashes would be weak from a security point of view**, since an attacker could doctor the remaining frames and obtain quite a different video sequence. In addition, for the robustness to frame rate change or frame drop-ping, a perceptual hash that encompasses the spatiotemporal content of the video in its totality would be more effective.

By exploit the insensitive to minor spatial and temporal perturbations, the author

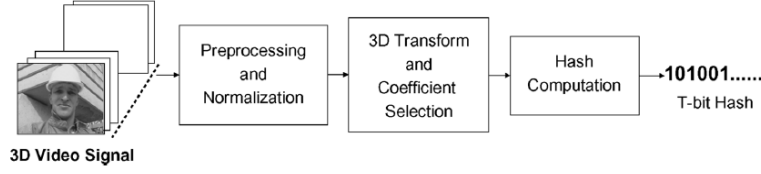


Fig. 4: The proposed hashing method.

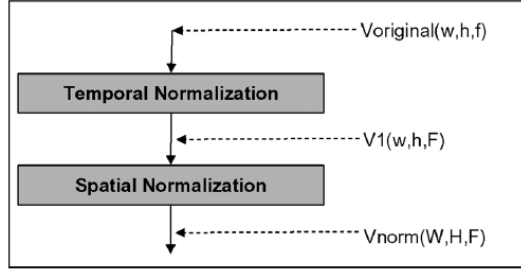


Fig. 5: Normalize

extend the existing hashing technique by DCT and RBT. The framework is shown in Fig.4.

5.1 Preprocessing and Normalization

The input video sequence is first converted to a standard video signal in terms of frame dimensions and of the number of frames via smoothing and sub-sampling. Given $Video(w, h, f)$ represents some video clip with title "Video", and where " w " is the frame width, " h " is the frame height and " f " is the number of frames within the clip. In the scheme, any video signal, $V_{original}(w, h, f)$, is converted to a standard size, $V_{norm}(W, H, F) = V_{norm}(32, 32, 64)$ via both spatial and temporal smoothing and sub-sampling. The process is shown in Fig.5.

The original video signal $V_{original}(w, h, f)$ with arbitrary dimensions is temporally smoothed and sub-sampled to form the $V_1(w, h, F)$ sequence. Temporal smoothing removes minute variations of a pixel in time and spreads large changes or object movements over several frames. The array of pixels having the same location in successive frames is referred as a **pixel tube**, $(m, n)^{th}$ pixel tube is defined as

$$\{V(m, n, f); f = 1, \dots, f_{total}\}$$

where f_{total} is the total original number of frames. All pixel tubes are filtered with a low-pass Gaussian filter with variance of $\sigma^2 = 6$. The smoothed video signal is afterwards

sub-sampled in time, to reduce the input clip to the target number of frames " F ". Denote this video signal as smoothed $V_1(m, n, F)$.

Spatial smoothing was implemented on each frame via a 2-D Gaussian filter with variance $\sigma^2 = 5$ and kernel $h(m, n) = \exp((m^2 + n^2)/10)/\sqrt{10\pi}$.

$$V_2(m, n, F) = V_1(m, n, F) \star h(m, n)$$

Then sub-sample to the size $W \times H$.

I think there are two points for further research:

Firstly, the author mentioned in the article but did not do. Regarding preprocessing for temporal, an alternative way of filtering would be a motion-compensated spatio-temporal smoothing method, where the pixel tube trajectories are not straight, but follow the object motion.

Second, Is this kind of preprocessing reasonable? For many types of videos, I think this method is not universal. For example, whether it is reasonable to describe 720p and 1080p videos by the same size, whether it makes sense to describe 1 hour long and 1 minute long videos by the same length of time.

5.2 3D-Transforms and Coefficient Selection

Most 3D-transform techniques with good compacting characteristics can serve the purpose of summarizing and capturing the video content as they collect and embed in their coefficients the information concurrently from time and space.

1) **3D-DCT Transform Case**: Low-frequency DCT coefficients contain the predominant part of the energy and they are also robust against most of the signal processing attacks, such as filtering and compression. However, to satisfy the uniqueness or discrimination property, one must judiciously enroll coefficients from mid- to high-frequency regions. The author use $T = 64$ coefficients, exacted from a $4 \times 4 \times 4$ cube in the low-pass band. And exclude the lowest frequency coefficients in each direction, that is DCT coefficients with addresses $V_{DCT}(0, ., .)$, $V_{DCT}(., 0, .)$, and $V_{DCT}(., ., 0)$.

Different from the discrete wavelet transform (DWT), the distribution of the DCT in the low frequency is in the upper left corner of the image instead of the center, which is why DCT is more widely used in compression technology than DWT, which can require less storage space. The selected coefficients for hashing is shown in Fig.6.

2) **3D-RBT Transform Case**: A hash function is secure if it is impractical for an adversary to maintain the same hash while changing the underlying video content or to obtain a significantly different hash for essentially the same content. To do this, a practical way that immediately emerges would be pseudo-randomly selecting a subset of 3-D DCT

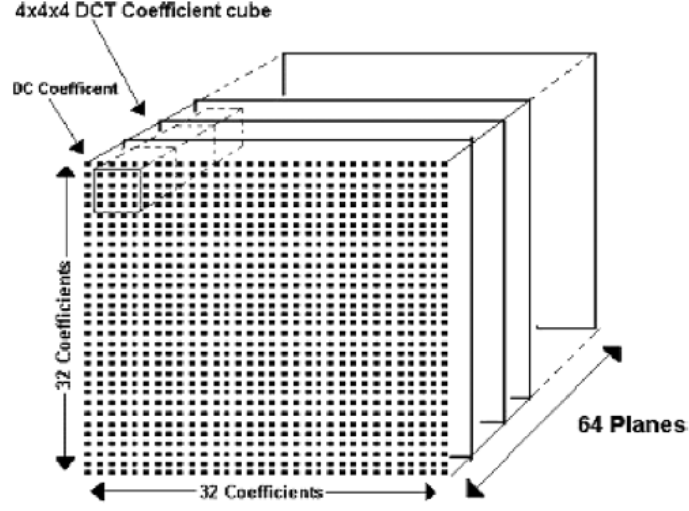


Fig. 6: Selected coefficients for hashing.

coefficients and compute hash from them. The calculation of the coefficients have to be made secret (key-based), which can be achieved by using projections on random 3-D basis functions. In this paper, the author opted for discrete cosine transform bases, but with randomly chosen frequencies, called Random Bases Transform or RBT.

Three-dimensional random bases with dimensions of $W \times H \times F$ are generated by using separate 1-D RBT bases in each dimension, that is, by using cosinusoidal signals with random frequencies generated with a key. In fact each 1-D base is an ordinary 1-D l -term DCT function with a randomly assigned frequency:

$$b_{RBT}[n] = \cos\left(\frac{\pi(2n+1)\theta}{2l}\right), n = 1, 2, \dots, (l-1)$$

The frequency θ is pseudo-randomly selected as $\theta = (pr(b_u - b_l)) + b_l$, where $\{b_u, b_l\}$ is the frequency interval and pr is a uniform random number in $[0, 1]$.

A number of 1-D bases has to be generated to generate a 3-D random bases with dimensions $W \times H \times F$. For W direction there has to be $H \times F$ 1-D bases with length $l = W$; for H direction there has to be $W \times F$ 1-D bases with length $l = H$; finally for F direction $W \times H$ 1-D bases with length $l = F$ generated. The overall 3D-RBT basis with dimensions $W \times H \times F$ can be formulated as:

$$B_{RBT}^i(w, h, f) = \beta \times \cos\left(\frac{\pi(2w+1)\theta_{(h,f)}}{2W}\right) \times \cos\left(\frac{\pi(2h+1)\theta_{(w,f)}}{2H}\right) \times \cos\left(\frac{\pi(2f+1)\theta_{(w,h)}}{2F}\right)$$

where $w = 1, \dots, W$; $h = 1, \dots, H$; $f = 1, \dots, F$, β is the normalization term, and $\theta_{(h,f)}$ are pseudo-randomly generated frequencies to probe along the w dimension and similarly for

the $\theta_{(w,f)}$ and $\theta_{(h,w)}$.

In the DCT transform, the rows of the transform matrix follow a pattern of increasing frequencies; consequently, the columns of the matrix grow from low to high frequencies. In the RBT transform each row is assigned a random frequency, so that the juxtaposition of these rows may generate all high frequency patterns along the columns. These high-frequency random basis functions unnecessarily reduce the robustness of the hash algorithm. To mitigate this loss of robustness, the author low-pass filter the generated 3D-RBT bases in all three dimensions. They average the random cosinusoids in the $W \times H$ directions with a 5×5 box filter, and those along the F direction with five-term box filter.

5.3 Hash Computation

The selected T transform coefficients are binarized using the median of the rank-ordered coefficients. If the subset of rank-ordered coefficients is denoted as $C_{(i)}, i = 1, \dots, T$, for some video sequence, then their median μ is found as $\mu = (C_{(T/2)} + C_{((T+1)/2)})/2$. Once μ is determined, then quantization of the selected coefficients of that video V is done as follows:

$$h_i = \begin{cases} 1 & C_{(i)} \geq \mu \\ 0 & C_{(i)} < \mu \end{cases}, \quad i = 1, \dots, T$$

The quantization operation makes the hash more robust against minor changes in the video sequence, since we only preserve the information of the coefficient value being greater or smaller than the coefficient median.

5.4 Properties of The Hash Sequence

In an ideal hash based on median quantization, assume that each possible hash sequence occurs with equal probability guaranteeing the best uniqueness property. The median-based quantization used in the generation of the robust hash function guarantees that there are exactly $T/2$ 1's and $T/2$ 0's in each hash sequence. In particular, the total number of possible hash values, denoted as N , can be calculated as, for $T = 64$:

$$N = \binom{T}{T/2} = \binom{64}{32} \approx 1.8326 \times 10^{18}$$

Select a special hash sequence constituted of all zeros in the first half and of all ones in the second half portion, as depicted in Fig.7.

The hamming distance, δ_0 , between this special sequence and any other arbitrary hash sequence is determined by the number of differing digit positions, which is given by the number of 1's in the first half and the number of 0's in the second half of the T -bit

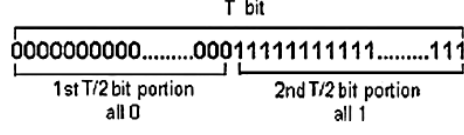


Fig. 7: Reference hash sequence used in hamming distance statistics.

sequence. Use the number of 1's and 0's in the k^{th} portion ($k = 1$ or 2), respectively. For any other arbitrary and permissible sequence, have the following relation:

$$\delta_0 = n1_1 + n0_2$$

In addition, the following equalities hold true: i) $n1_1 + n1_2 = T/2$, ii) $n0_1 + n0_2 = T/2$, iii) $n1_1 + n0_1 = T/2$, and iv) $n1_2 + n0_2 = T/2$. And using these equalities, have:

$$\delta_0 = n1_1 + T/2 - n0_1 = 2 \times n1_1$$

Furthermore, the probability of a hamming distance $\delta_0 = 2 \times n1_1$ is equal to the occurrence probability of $n1_1$ ones in the first portion of the hash. Since ones and zeros occur with equal probability, the probability distribution of hamming distances is given by

$$P(\delta_0) = P(n1_1) = \binom{T/2}{n1_1} \times 0.5^{n1_1} \times 0.5^{(T/2-n1_1)} = \binom{T/2}{n1_1} \times 0.5^{T/2}$$

where one must have $0 \leq n1_1 \leq T/2$ and $\delta_0 = 2 \times n1_1$. Since $P(n1_1)$ is the binomial probability function, have the following mean and variance values:

$$E\{\delta_0\} = 2 \times E\{n1_1\} = T/2$$

$$\sigma_{\delta_0}^2 = 4 \times \sigma_{n1_1}^2 = T/2$$

In Fig.8, plot the Gaussian fitting to the observed hamming distances. For the DCT-based hash, it was found that this test accepts the hypothesis of normal distribution up to significance level of 0.41. However, it is observed that the variance of the experimental distribution is lower than the theoretical distribution. This result may be primarily due to the well-structured nature of the DCT basis functions unlike the irregular and random structure of RBT basis functions. Hence, although any type of signal can be represented by DCT coefficients, the DCT coefficients of natural video signals may have some regularity, which can prevent the DCT coefficients appear as randomly picked numbers. The empirical and theoretical results does not match for DCT based hash. The goodness of fit of the Gaussian approximation to the binomial model was found to be 0.1026 using the Kolmogorov - Smirnov test.

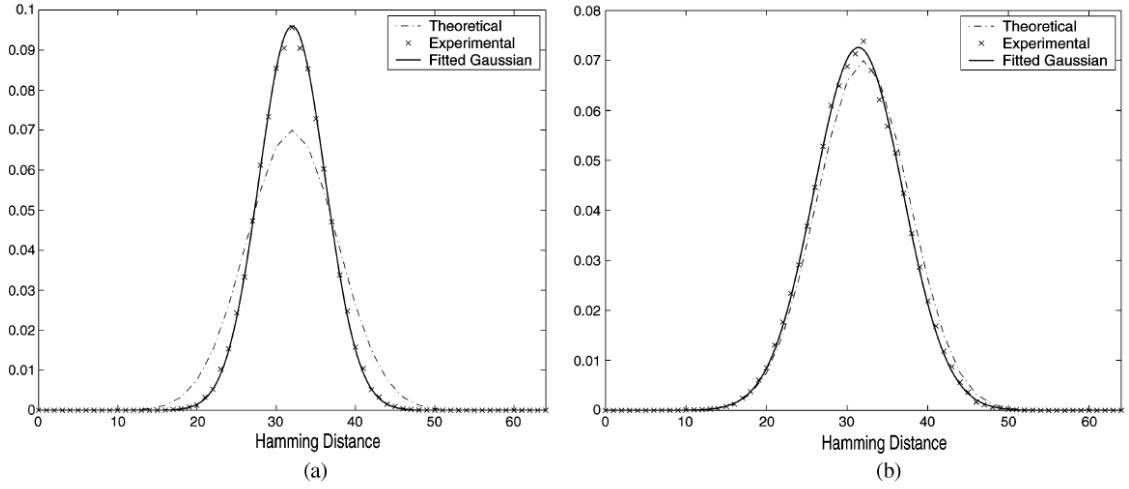


Fig. 8. Distribution of Hamming distances. Dashed curve is the theoretical probability density function [(7)] with $T = 64$, $\mu = 32$ and $\sigma^2 = 32$. The solid curve is the Gaussian density fitted to experimental data, where data points are marked with 'x'. a) DCT-based hash distances where $\mu = 32.01$ and $\sigma^2 = 17.25$. b) RBT-based hash distances where $\mu = 31.29$ and $\sigma^2 = 30.33$.

For the RBT-based hash, one can observe that the solid Gaussian curve fits almost perfectly to the theoretical curve. The Bera - Jarque gausianness test accepts the normality hypothesis with a significance level of 0.51 and the Kolmogorov - Smirnov distance between the empirical and theoretical distributions is 0.0886. The better fit between empirical distribution of hamming distances and the theoretical distribution can be due to the randomness and irregularity of the RBT patterns which result in transform coefficients and hash bits appearing as if they were randomly picked. However, since there has to be exactly 32 ones and 32 zeros in the hash function, the hash bits inherently cannot be independent from each other.

Another desirable property of the key-generated RBT hash is that it should be very difficult to obtain the same hash under different keys, in other words, the hashes for different keys should be totally unpredictable. Then, investigate the independence of each bit in the hash. Calculate the marginal and conditional probabilities of each bit. Then get the conclusion that the hash values of the same video for different keys can be regarded as statistically independent as required from a key based hash function.

In addition, the author also proved the adaboosting attack (a boosting technique to attack random-basis hashing algorithms) is hampered both by dynamic thresholding and holistic (nonlocal) processing of data.

5.5 Summarization

This article is a relatively high cited article in the IEEE database retrieval of video hash keywords. The method of this article is different from the supervised hashing method

previously known. This article is an unsupervised hashing method based on DCT. First, the author broke through a difficult point that the video hash is composed of the frame hash. By introducing 3D transformation, the coefficient of the transformation was directly hashed. Two 3D transformations are proposed here. From the current results, the RBT method seems more reasonable. **In the introduction part, the author also pointed out that this process may be implemented with other 3D transforms, such as discrete wavelet transform. This should be a point for further research. In addition, DCT and DWT can only be robust to disturbance factors such as illumination transformation and rotation to a certain degree. Therefore, whether there are other more suitable immutability methods is also a question that can be considered.**

This article is more comprehensive than the previous article, especially judging the performance of the hash function from a statistical perspective gives me a new understanding. But this part feels relatively difficult to understand, and it is not fully understood at present.

Due to time, the experimental part has not been completely finished. **But this article gives us a completely new idea. For the unsupervised hash method, can we make full use of the better method of image / video feature extraction to perform a hash encoding. In addition, the use of statistical models to explain the hash is also an idea worth studying.**

Of course, because this article is older, many ideas may have been tried.

6 Continuation[4](2020/04/01)

Continue to add more interesting content in the experimental part of the previous article. About the distance metrics for binary vectors, the paper introduce several proximity indices, such as the Jacard index or simple matching coefficient. However, the paper have simply used the Hamming distance between two hash values, labeled a and b , formulated as:

$$\delta = (1/2) \sum_{i=1}^T [1 - (h_i^a \times h_i^b)]$$

where the respective hash sequences $\{h_i^a\}_{i=1}^T$ and $\{h_i^b\}_{i=1}^T$ take -1 and 1 values.

In addition, SSIM of video[5] is used to measure the quality difference between two video sequences (say between original and attacked versions).

6.1 SSIM of video[5]

Firstly, for two signals are represented discretely as $x = \{x_i | i = 1, 2, \dots, N\}$ and $y = \{y_i | i = 1, 2, \dots, N\}$, define the luminance, contrast and structure comparison measures

as follows:

$$\begin{aligned} l(x, y) &= \frac{2\mu_x\mu_y}{\mu_x^2 + \mu_y^2} \\ c(x, y) &= \frac{2\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2} \\ s(x, y) &= \frac{\sigma_{xy}}{\sigma_x\sigma_y} \end{aligned}$$

where the statistical features can be estimated as follows:

$$\begin{aligned} \mu_x = \bar{x} &= \frac{1}{N} \sum_{i=1}^N x_i, \mu_y = \bar{y} = \frac{1}{N} \sum_{i=1}^N y_i \\ \sigma_x^2 &= \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2, \sigma_y^2 = \frac{1}{N-1} \sum_{i=1}^N (y_i - \bar{y})^2 \\ \sigma_{xy} &= \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y}) \end{aligned}$$

when $(\mu_x^2 + \mu_y^2)(\sigma_x^2 + \sigma_y^2) \neq 0$, the similarity index measure between x and y can be defined as follow:

$$S(x, y) = l(x, y) \cdot c(x, y) \cdot s(x, y) = \frac{4\mu_x\mu_y\sigma_{xy}}{(\mu_x^2 + \mu_y^2)(\sigma_x^2 + \sigma_y^2)}$$

To avoid the resulting measurement is unstable when $(\mu_x^2 + \mu_y^2)$ or $(\sigma_x^2 + \sigma_y^2)$ is close to 0, it can be reformulated as follow:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where the $C_1 = (K_1L)^2$ and $C_2 = (K_2L)^2$. L is the dynamic range of the pixel values (for 8 bits/pixel gray scale images, $L = 255$), and K_1 and K_2 are two constants whose values must be small.

The diagram of the proposed video quality assessment system is shown in Fig.8. Difference from the images, only a proportion of all possible 8×8 windows are selected here. Use the number of sampling windows per video frame (R_d) to represent the sampling density. The SSIM indexing approach is then applied to the Y , Cb and Cr color components independently and combined into a local quality measure using a weighted summation. Let $SSIM_{ij}^Y$, $SSIM_{ij}^{Cb}$ and $SSIM_{ij}^{Cr}$ denote the SSIM index values of the Y , Cb and Cr components of the j th sampling window in the i th video frame, respectively. The local quality index is given by:

$$SSIM_{ij} = W_Y SSIM_{ij}^Y + W_{Cb} SSIM_{ij}^{Cb} + W_{Cr} SSIM_{ij}^{Cr}$$

where the weights are fixed to be $W_Y = 0.8$, $W_{Cb} = 0.1$ and $W_{Cr} = 0.1$, respectively.

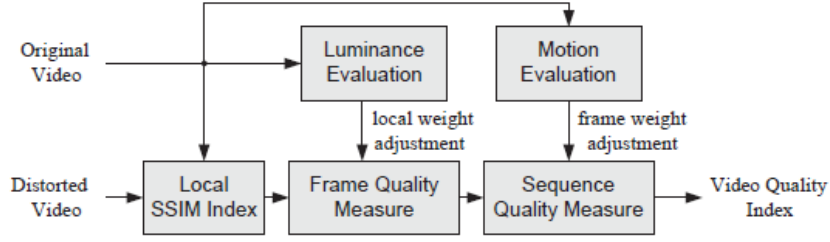


Fig. 8: Proposed video quality assessment system.

In the second level of quality evaluation, the local quality values are combined into a frame-level quality index using

$$Q_i = \frac{\sum_{j=1}^{R_s} w_{ij} SSIM_{ij}}{\sum_{j=1}^{R_s} w_{ij}}$$

where Q_i denotes the quality index measure of the i th frame in the video sequence, and w_{ij} is the weighting value given to the j th sampling window in the i th frame.

Finally in the third level, the overall quality of the entire video sequence is given by

$$Q = \frac{\sum_{i=1}^F W_i Q_i}{\sum_{i=1}^F W_i}$$

where F is the number of frames and W_i is the weighting value assigned to the i th frame.

In this paper, two simple adjustment to weight are employed. The first is based on the observation that dark regions usually do not attract fixations, therefore should be assigned smaller weighting values. It can be formulated as follow:

$$w_{ij} = \begin{cases} 0 & \mu_x \leq 40 \\ (\mu_x - 40)/10 & 40 < \mu \leq 50 \\ 1 & \mu_x > 50 \end{cases}$$

The second is based on the motion. Suppose m_{ij} represents the motion vector length of the j th sampling window in the i th frame, the the motion level of the i th frame is estimated as

$$M_i = \frac{(\sum_{j=1}^{R_s} m_{ij})/R_s}{K_M}$$

where K_M is a constant that serves as a normalization factor of the frame motion level.

The weight of frames is then adjusted by

$$W_i = \begin{cases} \sum_{j=1}^{R_s} w_{ij} & M_i \leq 0.8 \\ ((1.2 - M_i)/0.4) \sum_{j=1}^{R_s} w_{ij} & 0.8 < M_i \leq 1.2 \\ 0 & M_i > 1.2 \end{cases}$$

6.2 Some interesting experimental points of [4]

Regarding the whole experimental part, I think the most critical point is the detection of various attacks by the algorithm. Here the author gives the attack method as shown in the Fig.9. Then some comments on these results are in order.

Blurring: Since even heavy blurring does not much affect the low-frequency coefficients, the hash function remains very robust.

AWGN: The high-frequency perturbation superposed on the video virtually goes unnoticed by the hash function.

Contrast manipulation: This manipulation modifies the range of the pixel values but without changing their mutual dynamic relationship. However, extreme contrast increase/decrease results in pixel saturation to 255 and clipping to 0, which forms low frequency plain regions and consequently distorts the hash outcome,

Brightness manipulation: Though the hash function is quite robust to this modification, whenever the brightness manipulation is taken to the extreme of saturation (too dark, clipped to 0 or too bright, saturated to 255), the hash function suffers.

MPEG-4 compression: Compression basically removes the high-frequency irrelevancy and so has very little effect on perceptual hash.

Lossy Channel: According to different forms of loss, it will cause different degrees of impact on the hash, but in general it is the one with the greater impact.

Fade-over: Under reasonable levels of fade-over attack, most of the content information is still preserved in the video clip.

Clipping in time: The performance under clipping is inferior to the fade-over case.

Frame rotation: In any case, it can be roughly said that DCT based hash and RBT based hash can work safely under up to 7 degrees and 3 degrees of rotation, respectively.

Frame circular shift: The identification and verification performances remain at a satisfactory level up to 5% circular shifting.

Random frame dropping: The gaps left by dropped frames are filled by linear interpolation from nearest surviving future and past frames in order to preserve the sequence

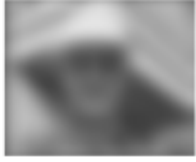











| Modif. Type | Description and Param. | Sample Frame | Modif. Type | Description and Param. | Sample Frame |
|---------------------|---|---|----------------------|--|---|
| Blurring | Gaussian filtering of each frame, with size σ_f of Gaussian filter kernel. Parameters: $\sigma_f^2 : 55$, $SSIM : 0.51$, $PSNR : 20.81$. |  | AWGN | Additive white Gaussian noise with standard deviation σ_n . Parameters: $\sigma_n : 110$, $SSIM : 0.07$, $PSNR : 9.72$. |  |
| Brightness Increase | Brightness increased by adding r percent of the frame mean luminance value to each pixel: $(r \times m_{frame})$. Parameters: $r : 80\%$, $SSIM : 0.75$, $PSNR : 9.91$. |  | Brightness Decrease | Brightness decreased by subtracting r percent of the frame mean luminance value to each pixel. Parameters: $r : 80\%$, $SSIM : 0.38$, $PSNR : 9.59$. |  |
| Contrast Increase | Linearly mapping luminance values in the interval $[v_l, v_h]$ to the interval $[0, 255]$. The luminance values below v_l and above v_h are saturated to 0 and 255 respectively. Parameters: $v_h : 168$, $v_l : 88$, $SSIM : 0.48$, $PSNR : 13.09$. |  | Contrast Decrease | Linearly mapping luminance values in the interval $[0, 255]$ to the interval $[v_l, v_h]$. Parameters: $v_h : 168$, $v_l : 88$, $SSIM : 0.73$, $PSNR : 17.79$. |  |
| MPEG4 Compression | Video is compressed to the target bit rate of b kbps with divX MPEG4 codec. Parameters: $b : 5kbps$, $SSIM : 0.73$, $PSNR : 26.84$. |  | Clipping in Time | Some percentage, (l) , of the total number of frames are clipped from both beginning and end of video. One of the missing frames is shown on the right. Parameter: $l : 8\%$. |  |
| Fade-over | The video sequence, on one extremity, fades in from a different video, and on the other extremity, it fades out into another video sequence. The fading effects take place over l percent of the number of frames. (See also Figure 10(a))Parameter: Fadeover percentage: 8%. |  | Lossy Channel | Video, first compressed to 30 kbps with Xvid MPEG4 codec, is streamed over lossy channel with packet drop rate of r . The hash comparison takes place between the received streaming video and the compressed video (not the uncompressed original), as in Figure 9. Parameter: Channel's IP packet drop rate: 1%. |  |
| Frame rotations | Each frame of the video is rotated separately. Parameter: Rotation degrees by 1, 3, 5 and 7 degrees. |  | Circular Frame Shift | Each frame of the video is circularly shifted both in row and column senses. Parameter: Shift amounts by 1%, 3%, 5% and 7% percent of frame dimensions. |  |
| Substitution attack | 1-second (25 frames) portions of a video sequence were replaced by another content, not necessarily of the same genre. Parameter: 1-second long substitutions at random locations. See illustration in Figure 10(b) | | Frame drop | This attack simulates a lossy channel in its extreme when the damaged packets coincide with the frame headers. Lost frames are recuperated via linear interpolation. Parameter: Drop rate varies over 30%, 50%, 70% and 90% of randomly chosen frames. | |
| Frame rate change | The frame rate of the video is decreased via temporal filtering and subsampling or increased via interpolation. Parameter: Frame rate change by factors of 0.25, 0.33, 0.50, 2, 3, 4. | | | | |

Fig. 9: Description of Modifications, Their Parameters and Illustrative Frame.

length. It is observed that, both the RBT-based and the DCT-based hashed survive for frame drop rates as high as 90%.

Frame rate change: The hashed of rate-altered videos do not deviate significantly from their original.

6.3 Summarization

In addition to the above-mentioned content, the author has also done a lot of verification experiments on the performance of the hash algorithm. I think it is of great reference significance for writing. I won't go into details here. Let's summarize this article as a whole.

First of all, this article is not the same as the hash algorithm we have read before. This article is an unsupervised perceptual hash. The idea of this hash is basically to generate a certain feature description vector for the characteristics of the data to be encoded, and then quantize the feature description vector. Generally, the quantization has an average quantization and other forms. It can be seen that the key to this kind of perceptual hash is how to generate feature descriptors based on the structural characteristics of the data without using supervised data. The key point is how to project a two-dimensional or even high-dimensional data onto a one-dimensional vector while maintaining local consistency.

For the several issues mentioned in the last summary, the author also described the article's conclusions and future directions. First, alternative transform schemes, such as discrete Fourier transform, discrete wavelet transform or overcomplete basis sets can be used. Second ancillary information, such as hash from audio and/or hash from color components can be included. Third, implementation issues must be addressed for efficiency in broadcast monitoring or in security cameras. Fourth, an intriguing alternative would be to implement the hash computation in the compressed domain, for example, between I-frames. Finally, the security issue is not solved thoroughly as the random frequencies employed in the RBT hash provide a limited search space and a scheme must be assessed.

In general, this perceptual hash is not even the same as the idea of transforming hash functions through certain optimization algorithms. However, this article has important reference value for the overall processing of video hash and the feature extraction for video or frame. Maybe we can improve the previous idea, instead of directly hashing the frames and adding them together, or preprocessing the video and then directly generating a hash function for the video. This definitely makes more sense in practical application value.

7 Visual Attention Based Temporally Weighting Method for Video Hashing[6](2020/04/01)

Since many of the theories in this article are directly used models of published articles. I should read further, here only introduce a general framework. The specific model methods for each step will continue to be summarized later.

This paper integrate visual attention into the temporally representative frame(TRF), which is method that accumulate all frames of the entire video segment into a "frame" by some way to hash. In this paper, a temporally visual weighting(TVW) method based on visual attention is proposed for the generation of TRF. In the proposed TVW method, the visual attention regions of each frame are obtained by combining the dynamic and static attention models. The temporal weight for each frame is defined as the strength of temporal variation of visual attention regions and the TRF of a video segment can be generated by accumulating the frames by the proposed TVW method.

Visual attention regions(VARs) on each frame are represented by the saliency map, which is formed by combining the dynamic and static attention models.

7.1 Dynamic Attention Model

Dynamic attention mode is obtained by detecting out the moving objects of each frame. The method of Gaussian Mixture Model (GMM)-based background modeling in [7] is adopted, which utilizes a mixture of K Gaussian distributions to build the background model and detect the moving objects. The probability of the pixel x at moment t as background is:

$$P(x_t) = \sum_{i=1}^K G_{i,t} \cdot \eta(x_t, \mu_{i,t}, \Sigma_{i,t})$$

where x_t denotes the pixel x at the moment t . K is the number of Gaussian mixtures, which is from 3 to 5. $G_{i,t}$ is an estimation of the update weight of the i th Gaussian distribution, $\mu_{i,t}$ and $\Sigma_{i,t}$ are the mean and the covariance matrix of the i th Gaussian distribution respectively, and η denotes the Gaussian probability density function.

$$\eta(x_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp -\frac{1}{2}(x_t - \mu)^T \Sigma^{-1} (x_t - \mu)$$

The parameters of the distribution which match the new observation are updated according to the algorithm in [7].

The first B distributions are chosen as the dynamic attention model D_{SM} after the K Gaussian distributions are re-ordered by the value G/α . α is the learning rate.

$$B = \arg \min_b \left(\sum_{k=1}^b G_k > T_B \right)$$

where T_B is a measure of the minimum portion of the data that should be accounted for the background. Here $K = 3$, $\alpha = 0.005$ and $T_B = 0.4$. D_{SM} is a binary image. The pixel value of the moving region is 1 and that of the static region is 0.

This part I understand is to use the Gaussian mixture model to determine the probability of each point being the background, and then find a Gaussian mixture distribution that meets a certain degree of confidence to judge the entire video frame. Points that change their nature before and after are found to be marked as moving region, otherwise they are static region.

7.2 Static Attention Model

The static attention model in [8] is adopted in this paper, which is constructed based on intensity, texture and color features on different scales.

$$S_{SM} = sw_1 \cdot I_{SM} + sw_2 \cdot T_{SM} + sw_3 \cdot C_{SM}$$

where S_{SM} denotes the static attention model. I_{SM} , T_{SM} and C_{SM} are the three saliency maps of intensity, texture and color respectively. sw_i is the weight and $\sum_{i=1}^3 sw_i = 1$.

The model in this part uses the different feature information of the image to weight the entire image. It is a frame is regarded as an image alone. The attention distribution of a person is calculated.

7.3 Attention Model Fusion[9]

The final saliency map, S_a , and the weights of the dynamic and static models are calculated as:

$$S_a = w_D \cdot D_{SM} + w_S \cdot S_{SM}$$

where w_D and w_S are the fusion weights of the dynamic and static attention models.

$$w_D = D_{SM'} \cdot \exp(1 - D_{SM'})$$

$$w_S = 1 - w_D$$

where $D_{SM'} = \text{Max}(D_{SM}) - \text{Mean}(D_{SM})$. $\text{Mean}(\cdot)$ and $\text{Max}(\cdot)$ denote the mean and maximum functions. The fusion weight of the dynamic model w_D is increased and the fusion weight of the static model w_S is suppressed. The pixel value of S_a ranges from 0 to 1, where 1 indicates the highest attention and 0 indicates the lowest attention.

This part is a mixture of dynamic and static attention on a picture, each pixel has a corresponding attention intensity value. Attention intensity distribution is from 0 to 1. A larger value indicates a higher degree of attention.

7.4 Temporal Attention Shift

The distribution of VARs can reflect the content of a frame as the frame is understood via visual attention. The VARs are located as the following steps:

Step 1: Divide the saliency map into non-overlapping blocks with size of 8×8 , and calculate the mean of each block;

Step 2: Find the block with maximum mean and it is taken as the block, which attracts the first visual attention in this paper. Make this block as a center, and extend it to an optimal rectangular region with minimum size and maximum saliency. (**What I understand about this sentence is that according to the current block, it expands outward to find the largest range block that does not change the relationship of the maximum value.**) This extended region is a visual attention region (VAR);

Step 3: When a VAR is located, the pixels in it are set to 0. Repeat Step 2 and Step 3 until there is no more VAR.

If the relationship between the attention degrees of different VARs changes, it means that the visual attention shifts. Denote the means of the first two VARs of the i th frame by $av_1(i)$ and $av_2(i)$. A visual attention shift happens between the i th and $i + 1$ th frame if

$$av_1(i) > av_2(i) \& av_2(i + 1) > av_1(i + 1)$$

or

$$av_2(i) > av_1(i) \& av_1(i + 1) > av_2(i + 1)$$

Then the strength of visual attention shift is defined as:

$$\delta(i) = N_T$$

where $\delta(i)$ denotes the strength of visual attention shift is the i th frame. N_T is the number of frames without attention shifts before the i th frame. If there is no attention shift in the i th frame, its strength of visual attention shift is 0.

7.5 Temporally Visual Weight

Considering stronger visual attention shift means greater video content change, the temporally visual weight w_i is :

$$w_i = \begin{cases} \rho_1 / N_0 & \delta(i) = 0 \\ \rho_2 \cdot \delta(i) / \sum_{i=1}^J \delta(i), \delta(i) \neq 0 \end{cases}$$

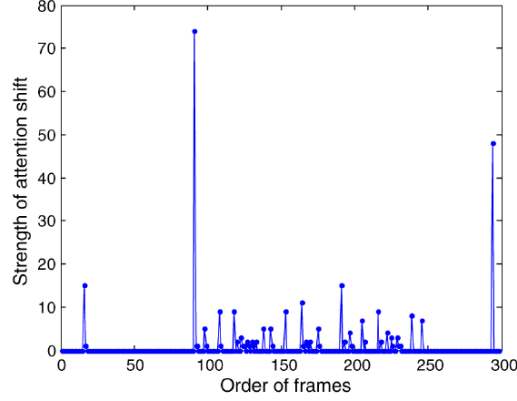


Fig. 10: Temporal visual attention curve of the video “Hall” .

where $\delta(i)$ denotes the strength of visual attention shift in the i th frame. N_0 denotes the amount of frames without visual attention shift in a video segment with J frames. ρ_1 and ρ_2 are weights and $\rho_1 + \rho_2 = 1$.

The TRF of a video segment is generated as:

$$F(u, v) = \sum_{i=1}^J w_i \cdot F(u, v, i)$$

where $F(u, v, i)$ is the intensity of the (u, v) th pixel of the i th frame in a video segment with J frames. $F(u, v)$ is the intensity of the TRF. w_i is the temporally visual weight.

7.6 Summarization

The method of this article is still different from the hash-related articles I saw before. The idea of hashing the video in this article is not to hash the entire video, but to hash the “frame” which is called TRF instead of key-frame. TRF is obtained by performing a certain weighted accumulation on all frames in a certain time segment. In this paper, the weight here is the attention value.

Because this paper can be regarded as the attention of the three articles into the entire video segmentation attention judgment process. There is no detailed introduction to many details, and I need to read related articles further.

But I found an interesting point in the picture shown in this article. In Fig.10 and Fig.11, this change seems to be related to the shot boundary. Whether it is relevant or not may require further reading of relevant articles or actual verification.

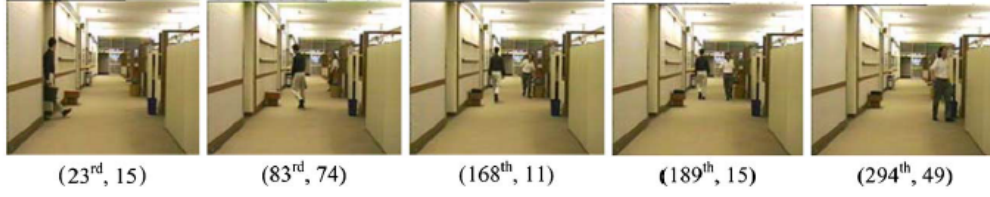


Fig. 11: Frames of the video “Hall” , whose strength of attention shift is greater than 10.

8 More detailed introduction for the previous article(2020/04/04)

Because the specific algorithm used in the last article is a relatively mature algorithm. Here is a reading summary of their related articles.

8.1 Adaptive background mixture models for real-time tracking[7]

This paper discusses modeling each pixel as a mixture of Gaussian and using an on-line approximation to update the model, which is used to segment moving regions in image sequences. Consider the problem of video surveillance and monitoring. A robust system should not depend on careful placement of cameras and should also be robust to whatever is in its visual field or whatever lighting effects occur. The author try to create a robust, adaptive tracking system that is flexible enough to handle variations in lighting, moving scene clutter, multiple moving objects and other arbitrary changes to the observed scene.

Be contrary to previous thoughts, explicitly model the values of all the pixels as one particular type of distribution, the author simply model the values of a particular pixel as a mixture of Gaussian. Based on the persistence and the variance of each of the Gaussian of the mixture, they determine which Gaussian may correspond to background colors. Pixel values that do not fit the background distributions are considered foreground until there is a Gaussian that includes them with sufficient, consistent evidence supporting it.

Before starting this article, let’s give a brief introduction to Gaussian Mixture Modal (GMM) and Expectation Maximization (EM).

8.1.1 GMM & EM [10][11]

First of all, according to common sense, we can know that many random variables follow a normal distribution (Gaussian distribution). In machine learning, many simple data still obey this distribution. For simple data X , it have a Probability Density Function

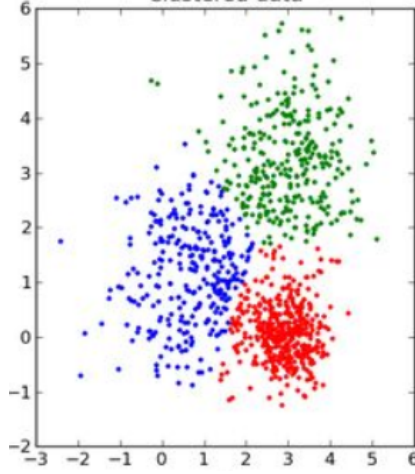


Fig. 12: Clustered data.

(PDF) as follow:

$$P(x|\theta) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp -\frac{(x - \mu)^2}{2\sigma^2}$$

where μ is the mean value (expectation) and σ is the standard deviation of data. Similarity, when data obey a multivariate Gaussian distribution, it have a PDF as follow:

$$P(x|\theta) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{\frac{1}{2}}} \exp -\frac{(x - \mu)^T \Sigma^{-1} (x - \mu)}{2}$$

where μ is the expectation, Σ is the covariance matrix and D is the dimension of the data.

However, the actual situation is much more complicated than the example we mentioned above. As shown in the Fig.12, if we want to build a model to describe the data in the graph, obviously a Gaussian distribution of any dimension is difficult to achieve. Therefore, the weights of multiple Gaussian distributions are used to describe the data here.

If a probability distribution function is used to cover the entire data, the value of each point is expressed as its probability. From a geometric point of view, any point, in Fig.12, can be regarded as the superposition of multiple Gaussian distributions, which can be formulated as follow:

$$p(x) = \sum_{k=1}^K \alpha_k N(x|\mu_k, \Sigma_k)$$

where K is the number of the Gaussian distribution. α_k is the weight of each Gaussian distribution, which can be regarded as the contribution of each Gaussian distribution to the probability, and meet $\sum_{k=1}^K \alpha_k = 1$.

As for the practical significance of α_k , from the perspective of model mixing, for observed variable x , it has a certain probability of belonging to any Gaussian distribution. We assume a latent variable z , which represents every Gaussian distribution. So, for each Gaussian distribution, z has a probability, p_k , of belonging to k th Gaussian distribution. Then the probability of the point can be formulated:

$$p(x) = \sum_z p(x, z)$$

where $p(x, z)$ is the joint probability distribution which can be rewritten as conditional probability distribution. So the probability of the point can be reformulated:

$$p(x) = \sum_{k=1}^K p_k N(x|\mu_k, \Sigma_k)$$

where p_k is the probability of belonging to the k th Gaussian. Here we can see that in the Gaussian mixture model, the probability of a point is actually a weighted value of the probability of belonging to a different Gaussian distribution.

So far, we have constructed a model that can describe the probability value of each point. However, many parameters in this model, such as probability, mean, and variance are unknown, so the model needs to be solved.

For a single Gaussian distribution, the Maximum likelihood can be used to estimate the parameter $\theta(\mu, \sigma)$:

$$\theta = \arg \max_{\theta} L(\theta)$$

where each data is independent and likelihood can be formulated:

$$L(\theta) = \prod_{j=1}^N p(x_j|\theta)$$

for convenience of calculation, it can be reformulated:

$$\log L(\theta) = \sum_{j=1}^N \log p(x_j|\theta)$$

For GMM, the likelihood can be written as followed:

$$\log L(\theta) = \sum_{j=1}^N \log p(x_j|\theta) = \sum_{j=1}^N \log \sum_{k=1}^K p_k N(x_j|\mu_k, \Sigma_k)$$

then we have:

$$\theta = \arg \max_{\theta} \sum_{j=1}^N \log \sum_{k=1}^K p_k N(x_j|\mu_k, \Sigma_k)$$

Because it involves the problem of summation within the log function, it is difficult to solve directly by first derivative equal to 0.

Since the unknowns here are the parameters of each Gaussian distribution, we hope to find the most suitable Gaussian distribution to describe the data, that is, the parameters of the Gaussian distribution should be the largest expected. Therefore, the expected maximum algorithm is proposed here. According to the expected maximum algorithm, the above problem can be rewritten as:

$$\theta^{t+1} = \arg \max_{\theta} E_{z|x, \theta^t} [\log p(x, z|\theta)]$$

so we have:

$$\begin{aligned} E_{z|x, \theta^t} [\log p(x, z|\theta)] &= \sum_z \log \prod_{j=1}^N p(x_j, z_j|\theta) \cdot \prod_{j=1}^N p(z_j|x_j, \theta^t) \\ &= \sum_{z_1, z_2, \dots, z_N} \sum_{j=1}^N \log p(x_j, z_j|\theta) \prod_{j=1}^N p(z_j|x_j, \theta^t) \\ &= \sum_{z_1, z_2, \dots, z_N} [\log p(x_1, z_1|\theta) + \log p(x_2, z_2|\theta) + \dots + \log p(x_N, z_N|\theta)] \prod_{j=1}^N p(z_j|x_j, \theta^t) \end{aligned}$$

because we have

$$\sum_{z_1, z_2, \dots, z_N} \log p(x_1, z_1|\theta) \prod_{j=1}^N p(z_j|x_j, \theta^t) = \sum_{z_1} \log p(x_1, z_1|\theta) \cdot p(z_1|x_1, \theta^t)$$

then the expectation can be reformulated:

$$\begin{aligned} E_{z|x, \theta^t} [\log p(x, z|\theta)] &= \sum_{z_1} \log p(x_1, z_1|\theta) \cdot p(z_1|x_1, \theta^t) + \dots + \sum_{z_N} \log p(x_N, z_N|\theta) \cdot p(z_N|x_N, \theta^t) \\ &= \sum_{j=1}^N \sum_{z_j} \log p(x_j, z_j|\theta) \cdot p(z_j|x_j, \theta^t) \\ &= \sum_{j=1}^N \sum_{z_j} \log [p_{z_j} \cdot N(x_j|\mu_{z_j}, \Sigma_{z_j})] \cdot p(z_j|x_j, \theta^t) \\ &= \sum_{k=1}^K \sum_{j=1}^N [\log p_k + \log N(x_j|\mu_k, \Sigma_k)] \cdot p(z_j|x_j, \theta^t) \end{aligned}$$

Here we have the probability (expected form) that the data j comes from the distribution k , called "E-step".

The next thing to do is to solve each parameter according to this expectation, called "M-step". After the above changes, the more complicated parts of the parameters have been eliminated, so the model can be iterated through the gradient method until convergence.

The above content is GMM and EM. In this article, the author has improved it.

8.1.2 The method

Consider the values of a particular pixel over time as a "pixel process". At any time, t , what is known about a particular pixel, $\{x_0, y_0\}$, is its history

$$\{X_1, \dots, X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\}$$

where I is the image sequence. The value of each pixel represents a measurement of the radiance in the direction of the sensor of the first object intersected by the pixel's optical ray.

If a static object was added to the scene and was not incorporated into the background until it had been there longer than the previous object, the corresponding pixels could be considered foreground for arbitrarily long periods. This would lead to accumulated errors in the foreground estimation, resulting in poor tracking behavior. These factors suggest that more recent observations may be more important in determining the Gaussian parameter estimates.

The recent history of each pixel, $\{X_1, \dots, X_t\}$, is modeled by a mixture of K Gaussian distributions. The probability of observing the current pixel value is

$$P(X_t) = \sum_{i=1}^K w_{i,t} \star \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

where K is the number of distributions, $w_{i,t}$ is an estimate of the weight (what portion of the data is accounted for by this Gaussian) of the i th Gaussian in the mixture at time t , $\mu_{i,t}$ is the mean value of the i th Gaussian in the mixture at time t , $\Sigma_{i,t}$ is the covariance matrix of the i th Gaussian in the mixture at time t , and where η is a Gaussian probability density function

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu)^T \Sigma^{-1} (X_t - \mu)}$$

where K is determined by ourselves. For computational reasons, the covariance matrix is assumed to be of the form:

$$\Sigma_{k,t} = \sigma_k^2 I$$

This assumes that the red, green, and blue pixel values are independent and have the same variances.

If the pixel process could be considered a stationary process, a standard method for maximizing the likelihood of the observed data is EM. Unfortunately, each pixel process varies over time as the state of the world changes, so the author use an approximate method which essentially treats each new observation as a sample set of size 1 and uses standard learning rules to integrate the new data.

Every new pixel value, X_t , is checked against the existing K Gaussian distributions, until a match is found. A match is defined as a pixel value within 2.5 standard deviations of a distribution. If none of the K distributions match the current pixel value, the least probable distributions is replaced with a distribution with the current value as its mean value, an initially high variance, and low prior weight.

The prior weights of the K distributions at time t , w_k , t , are adjusted as follows

$$w_{k,t} = (1 - \alpha)w_{k,t-1} + \alpha(M_{k,t})$$

where α is the learning rate and $M_{k,t}$ is 1 for the modal which matched and 0 for the remaining models. $1/\alpha$ defines the time constant which determines the speed at which the distribution's parameters change. $w_{k,t}$ is effectively a causal low-pass filtered average of the (threshold) posterior probability that pixel values have matched modal k given observations from time 1 through t .

The μ and σ parameters for unmatched distributions remain the same. The parameters of the distribution which matches the new observation are updated as follows

$$\begin{aligned}\mu_t &= (1 - \rho)\mu_{t-1} + \rho X_t \\ \sigma_t^2 &= (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t)\end{aligned}$$

where the second learning rate, ρ , is

$$\rho = \alpha\eta(X_t|\mu_k, \sigma_k)$$

One of the significant advantages of this method is that when something is allowed to become part of the background, it does not destroy the existing model of the background. The original background color remains in the mixture until it becomes the K th most probable and a new color is observed. Therefore, if an object is stationary just long enough to become part of the background and then it moves, the distribution describing the previous background still exists with the same μ and σ^2 , but a lower w and will be quickly re-incorporated into the background.

Then the Gaussian distributions which have the most supporting evidence and least variance would like to be produced by background processes. In contrast, when a new object occludes the background object, it will not, in general, match one of the existing distributions which will result in either the creation of a new distribution or the increase in the variance of an existing distribution.

First, the Gaussian are ordered by the value of w/σ . This values increases both as a distribution gains more evidence and as the variance decreases. Then the first B

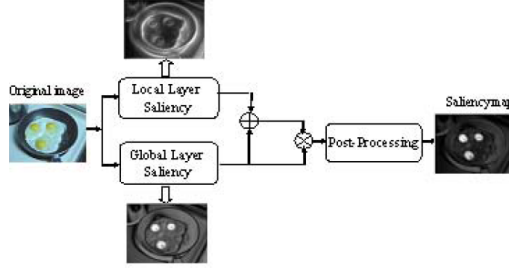


Fig. 13: The framework of the proposed VAM.

distributions are chosen as the background modal, where

$$B = \arg \min_b \left(\sum_{k=1}^b w_k > T \right)$$

where T is measure of the minimum portion of the data should be accounted for by the background.

So far, we can distinguish between the foreground and the background of the pixel. In addition, the paper had proposed multiple hypothesis tracking which I can not understand.

8.2 Visual attention model with cross-layer saliency optimization[8]

In this paper, a new bottom-up visual attention model (VAM) is proposed based on the spirit of cross-layer optimization in the field of communication. Cross-layer design is a method of network optimization in the field of communication. Cross-layer optimization is to coordinate the work on different layers through the exchanging information and feedback between them so as to achieve optimization. In this paper, first, the local and global saliency detection is performed on the local and global layers based on the contrast of low-level features respectively; and then based on the obtained local and global saliency maps, a weight model is generated; finally considering that the global saliency mainly shows the salient objects from the view of the whole image, the weight model is used as the feedback from local layer to global layer and optimizes the global saliency to the final VAM.

Fig.13 shows the framework of the proposed VAM. It mainly contains two components: the contrast saliency computation from the local and global layers respectively, and the optimization on global saliency with a weight model generated by combining the local and global saliency. In the framework, global and local saliency is extracted simultaneous. The global saliency is taken as the main one and local and global saliency are linearly combined to generate a weight model, and this weight model is utilized to optimize the

global saliency and construct the final VAM to improve the contrast between salient and non-salient regions.

8.2.1 Saliency from Local layer

Intensity contrast:

$$I_{CM}(x, y) = c \lg \frac{I_j^{max}}{I_j^{avg}} = c \lg \frac{\max\{I_1, I_2, \dots, I_n, \dots, I_{N'}\}}{\frac{1}{N'} \sum_{n=1}^{N'} I_n}$$

where c is a constant, I is absolute threshing value, and $N' = (2l' + 1) \times (2l' + 1)$, $l' \in \{1, 2, 3\}$ denotes the number of pixels in the window, that is, the size of local contrast patch.

Texture contrast:

$$T_{CM}(x, y) = [\frac{1}{N' - 1} \sum_{n=1}^{N'} (I_n - \frac{1}{N'} \sum_{n=1}^{N'} I_n)^2]^{\frac{1}{2}}$$

Color contrast: The color contrast is measured in the HSI space instead of RGB space. The color distance between two pixels $Y_1 = (H_1, S_1, I_1)^T$ and $Y_2 = (H_2, S_2, I_2)^T$, is defined as:

$$\Delta_{HSI}(Y_1, Y_2) = \sqrt{(\Delta_I)^2 + (\Delta_C)^2}$$

where $\Delta_I = |I_1 - I_2|$, $\Delta_C = \sqrt{S_1^2 + S_2^2 - 2S_1S_2 \cos \theta}$ and

$$\theta = \begin{cases} |H_1 - H_2|; & \text{if } |H_1 - H_2| \leq \pi \\ 2\pi - |H_1 - H_2|; & \text{if } |H_1 - H_2| > \pi \end{cases}$$

The color contrast is defined as:

$$C_{CM}(x, y) = \frac{1}{N' - 1} [\sum_{n=1}^{N'-1} \Delta_{HSI}(Y(x, y), Y_n)]$$

The color contrast of each pixel is improved:

$$C_{Map}(x, y) = \sum_{N'} CM(x, y)$$

where $C_{Map} = \{I'_{CM}, T'_{CM}, C'_{CM}\}$, $CM = \{I_{CM}, T_{CM}, C_{CM}\}$ and $N' = (2l' + 1) \times (2l' + 1)$.

RGB to HSI:

$$H = \begin{cases} \theta & B \leq G \\ 360 - \theta & B > G \end{cases}$$

where $\theta = \arccos(\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-G)(G-B)]^{\frac{1}{2}}})$

$$S = 1 - \frac{3}{(R + G + B)} [\min(R, G, B)]$$

$$I = \frac{1}{3}(R + G + B)$$

Multi-scale Saliency Enhancement: At first, the input images are decomposed into six spatial scales by Gaussian pyramid; then at each scale, three local contrast maps are calculated respectively, so eighteen contrast maps are generated; at last an iterative interpolation algorithm is utilized to integrate these contrast maps to form three feature maps including I'_{FM} , T'_{FM} , C'_{FM} .

Normalization and Combination: For each feature, the maximum normalization operator $N(x)$ [12] is utilized. The operator consists of the following: *1) Normalize all the feature maps to the same dynamic range; 2) for each map, find its global maximum M and the average \bar{m} of all the other local maxima; 3) globally multiply the map by $(M - \bar{m})^2$.*

Then the three feature maps, I'_{FM} , T'_{FM} , C'_{FM} , which denote the feature map based on intensity, texture and color respectively, are combined into the final integrated saliency map, S_{Local} .

$$S_{Local} = \sqrt{(N(I'_{FM}))^2 + (N(T'_{FM}))^2 + (N(C'_{FM}))^2}$$

8.2.2 Saliency from Global-layer

A pixel is salient if its feature is unique among the image according to principles of the global contrast saliency. The global saliency of a pixel k is obtained by calculating the differences between the patch p_k centered at pixel k with all other patches in the image.

For pixel k , using the CIE L*a*b color space, the location is a $[L; a; b]^T$ vector. The global saliency of pixel k is:

$$S_{Global}(k) = \sum_j dis(p_k, p_j)$$

where $dis(p_k, p_j)$ is the Euclidean distance between patches p_k and p_j in CIE L*a*b color space. Pixel k is considered salient when S_{Global} is high.

CIE L*a*b color space: The conversion from RGB signal to CIE L*a*b space can be realized by digital calculation method. In this conversion process, errors caused by color filters and other optical devices can be automatically compensated.

8.2.3 Cross-layer optimization

At first, the local saliency and global saliency are normalized to the same range, and then in order to enhance the contrast between the salient and non-salient regions, then we have:

$$w = w_1 N(S_{Local}) + w_2 N(S_{Global})$$

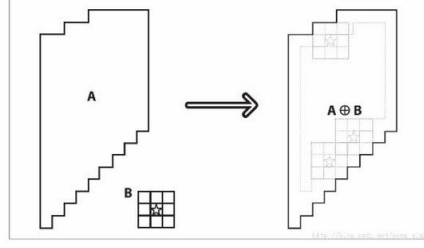


Fig. 14: The picture on the right is one circle larger than the picture on the left.

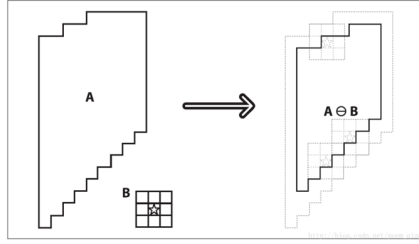


Fig. 15: The picture on the right is one circle smaller than the picture on the left.

$$S = w \star S_{Global}$$

where $N(x)$ is normalization operator. w_1 and w_2 satisfy $\sum_i w_i = 1$. In addition, here w is a weighting coefficient matrix with the range of value in the matrix is $[0, 1]$, called weight model, and the range of S_{Global} is also normalized to $[0, 1]$.

8.2.4 Post_processing

The post-processing is performed by morphological closing to smooth S , which is described as follow:

$$SM = (S \oplus B) \ominus B$$

where \ominus denotes morphological erosion and \oplus represents morphological dilation with a structuring element B . SM is the final saliency map.

Morphological dilation: Dilation is the operation of finding the local maximum. Mathematically, the dilation is to convolve the image (or part of the image, we call it A) with the kernel (we call it B), shown in Fig.14.

Morphological erosion: In contrast to dilation, erosion is the operation of finding a local minimum. Erosion can be simply understood as the process of eliminating all boundary points of object A , shown in Fig.15.

Erosion followed by dilation is called an open operation, and its effect is shown in the left diagram of Fig.16.



Fig. 16: The effect of operator

Dilation followed by erosion is called a closed operation, and its effect is shown in the right figure of Fig.16.

In this article, the author excludes the non-significant pixels in the salient region through the closed operation in the morphological operator.

8.3 Keyframe-Based video summary using visual attention clues[9](2020/04/08)

About the construction of saliency map, in a sense, paper[6] is an improvement of paper[9]. The model structure of the two articles is basically the same, where both static and dynamic attention are constructed. After that, both their fusion follow the fusion method in paper[9]. Next, we first give a general introduction to this paper.

The focus of this work is to bridge the semantic gap between low-level descriptors used by computer systems and the high-level concepts perceived by human users. To develop technologies to find points of interest in media streams and video collections, the paper propose an adaptive keyframe-extraction method based on the visual attention model.

A feature, humans fixate on images areas that have higher spatial contrast than random regions, is called *saliency map*, which is a 2D topographic representation of the conspicuousness of every pixel in an image. The salient features extracted from video can be divided into two parts: a dynamic one and a static one.

8.3.1 Dynamic attention detection

Firstly, divide a frame into N blocks of size 8×8 . The descriptor of each block is defined as $p_i(I_i, H_i, x_i, y_i, dx_i, dy_i), i \in N$. The intensity feature (I_i) and color feature (H_i) are contrast-based static features; x_i, y_i are the location information of block i ; and the dx_i, dy_i are the motion vectors. The motion intensity γ_i at each block is computed as

$$\gamma_i = \sqrt{(dx_i^2 + dy_i^2)}$$

The orientation θ_i of each block is defined as

$$\theta_i = \arctan(dy_i/dx_i)$$

In generally, the motion vectors with inconsistent orientation are often located at object boundaries. The probability density function of the orientation from the sample blocks can be estimated by kernel density estimation. Given the sample blocks and the kernel function $K(\theta)$, the probability density function of the motion model can be defined as:

$$f(\theta) = \frac{1}{N} \sum_{i=1}^N K(\theta - \theta_i)$$

where $K(\theta)$ is the kernel function, such as Gaussian kernel

$$K(\theta - \theta_i) = \frac{1}{h\sqrt{2\pi}} \exp -\frac{(\theta - \theta_i)^2}{2h^2}$$

where h is the bandwidth. **(Here I always have some problems. The probability density function of the orientation defined here is not used in the subsequent framework, just to give the above definition.)**

Define h as as the bin width of the histogram associated with the orientation of blocks, then build an orientation histogram for all blocks. **(Regarding the probability density function, I think the only possible point is the histogram of the orientation of the blocks. However, the histogram of the orientation distribution can be directly calculated by differentiation. It is completely unnecessary to use the probability density function, so it seems very unreasonable. The methods here are not described in detail or reference, so it is difficult to find specific algorithm strategies.)** Finally measure the coherence of orientation distribution $A_T(i)$ by

$$A_T(i) = 1 - \frac{v(b(i))}{\sum_{j=1}^H v(j)}$$

where H is the maximum bin index of the histogram, $b(i)$ is the bin index of block i , and $v(j)$ is the value of histogram at bin index j .

Finally, the motion attention of block i can be computed as

$$A_T(i) = \gamma_i \cdot A_T(i)$$

where the $A_T(i)$ will be normalize to range of $[0, 255]$.

8.3.2 Static attention detection

The human perception system is sensitive to the contrast of visual signals, such as color, intensity, and texture. For the static attention detection module, the intensity feature (I) and color feature (H) is used. Use the center-surround differences to generate

a saliency-based visual attention model and calculate the static attention value of block C_i by center-surround differences as follows:

$$C_i = \sum_{q \in \Omega} d(p_i, q)$$

where q represents the 8×8 blocks neighborhood and the contrast-based as

$$d(p_i, q) = \alpha |p_i(I) - q(I)| + \beta |p_i(H) - q(H)|$$

where α, β are predefined constants.

The static attention model are defined according to the number of blocks and their position, size, and brightness in the saliency map as follow:

$$A_S = \frac{1}{N} \sum_{i=1}^N w_i \cdot C_i$$

where weight w_i is used a Gaussian falloff with variance σ_w^2 to assign a weight to the position of the saliency blocks:

$$w_i = \exp\left(-\frac{|p_i - p_{center}|^2}{2\sigma_w^2}\right)$$

where p_{center} represents a frame's center block and the variance σ_w^2 is set to be one-third of the frame width.

8.3.3 Motion priority fusion

Human vision system is more sensitive to motion contrast compared to other external signals. The perception of moving-target saliency increases nonlinearly with motion contrast and shows significant saturation and threshold effects. Define the weights of dynamic attention and static attention as follows:

$$w_T = \bar{A}_T \cdot \exp(1 - \bar{A}_T), w_S = 1 - w_T$$

where $\bar{A}_T = \text{Max}(A_T) - \text{Mean}(A_T)$.

In fact, strong motion contrast will produce increased \bar{A}_T . Consequently, the fusion weight of the dynamic model w_T is also increased. And as a result, the fusion weight of the spatial model w_S is suppressed. Compute the final visual attention index as

$$VAI = w_S \cdot A_S + w_T \cdot A_T$$

(Another question arises here, how $A_T(i)$ is combined into A_T . The method of combination of $A_s(i)$ is given clearly, but not $A_T(i)$.)

At this point, the construction of the saliency map in this article is over. The structure in paper[6] basically imitates the framework of this paper. Next, briefly introduce the settings for adaptive video summarization in the article.

8.3.4 Adaptive video summarization

When having the total keyframe number for a video clip, the author use an adaptive keyframe assignment strategy to control the keyframe density according to focus-points shift. They select the frame with the maximum VAI as the keyframe when only one keyframe is required for each shot. However, long or large content shots usually require the generation of multiple keyframes to abstract the shot.

Define the VAI difference of any two frames as

$$D_{ij} = |\sum_{n=1}^N (A_n^i - A_n^j)|$$

where A_n should be the VAI of a n th block. (There is no clear definition in the paper.) And calculate the variation of attention index in a shot as

$$\bar{d}_s = \frac{\sum_{i=2}^{M_s} D_{i,i-1}}{M_s}$$

where M_s represents the number of frames in shot s . Compute the keyframe number C_s of shot s as follow:

$$C_s = \max(T \cdot \frac{\bar{d}_s}{\sum \bar{d}_s}, 1)$$

where T is the total keyframe number for an entire clip, which the user assigns.

Assign at least one keyframe for each shot to ensure a good representation. When only one keyframe is required in a shot, the frame with maximum VAI is selected as the keyframe.

After extracting potential keyframes from the shot, the number of keyframe that represent the shot should be reduce further. These potential keyframes are compared to each other by calculating their VAI difference. To prevent keyframes from being generated in several adjacent frames with the highest VAI potential, the keyframe in a shot must satisfy the following inequality:

$$D_{key} > D_{ave} + \sigma \cdot D_{div}$$

where D_{ave} is the average VAI difference of the keyframe, D_{div} is the standard deviation, and σ is a constant.

This part is actually the process of optimal selection. If the number of frames in a shot is relatively small, a keyframe (maximum VAI) can represent the entire shot as a keyframe. If the number of frames in the shot is relatively large, then more frames are needed as keyframes to represent the entire shot. The specific quantity selection strategy is given here.

8.4 Summarization of related paper on paper[6]

The idea of paper[6] is basically as follows. The method proposed in paper[6] is not an algorithm that directly encodes the video to hash codes. It can be regarded as a pre-processing method. As we know, many articles encode the entire video or encode frames or key frames. Here, paper[6] uses the encoding of the temporal representative frame(TRF) of the video.

Paper[6] mainly introduces a method of generating a TRF, but does not follow up on how to encode a TRF. Here, we can think of TRF as a feature of video, a kind of global feature that uses one frame to represent the entire video.

In paper[6], the generation of TRF depends on visual attention. The TRF is actually a weighted average of all frames in the video. The performance of the TRF is mainly determined by the weight. Paper[6] combines weight and attention, which gives higher weight to frames with high visual attention scores and gives lower weight to frames with low visual attention scores.

At this point, the key-point of the problem becomes how to score visual attention for the frames in the video. This is different from the attention in the network that we have learned. The visual attention here does not depend on the data, but on the established rules. Visual attention is generally divided into static attention and dynamic attention.

As for how to construct an visual attention mechanism, paper[6] imitates the framework of paper[9]. For static attention, paper[6] adopts the model of static attention in paper[8], and paper[9] directly uses intensity feature and color feature. As for dynamic attention, paper[6] uses the Gaussian mixture model to determine the dynamic region, while paper[9] uses a method related to optical flow. As for the two ways of mixing attention, the two paper are the same. I think the way to fuse the attention is the most important model in paper[9]. The fusion of its attention basically conforms to the human physiological structure, and the weights are distributed according to the dynamic contrast.

Through these four articles, I have a new understanding of attention. I feel that attention may actually have some intersections with shot segmentation. In particular, in the examples of Fig.10 and Fig.11 in paper[6], I think that the extreme point of the attention score can be regarded as the boundary of the shots, but the specific accuracy and relationship still need to be discussed.

In short, the method of characterizing the video mentioned in paper[6] is very meaningful for reference. The entire video is compressed into an image through weighted assignment. This seems to be regarded as a class of methods to reduce time dimension of the video. If we continue to study the characterization of the video in the future, I think this is a very worthy direction.

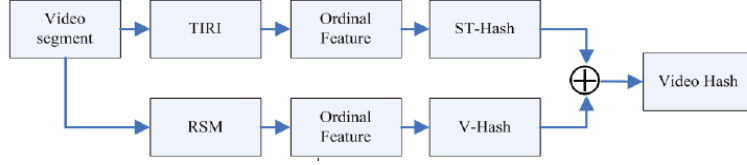


Fig. 17: The framework of the proposed algorithm.

9 A visual saliency based video hashing algorithm[14](2020/04/08)

This paper can be seen as a follow-up to paper[6] on how to use temporal representative frame (TRF), here call temporally informative representation image (TIRI), for hash coding.

In this paper, the video hash is fused by two hashed, which are spatio-temporal hash (ST-Hash) and visual hash (V-Hash). The ST-Hash is generated based on the ordinal feature, which is formed according to the intensity difference between adjacent blocks of the TIRI. At the same time, the representative saliency map (RSM) is constructed by the visual saliency maps in video segments. The V-Hash is formed according to the intensity difference between adjacent blocks of the RSM, and used to modulated the ST-Hash to form the final video hash. The framework of the paper is shown in Fig.17.

As in paper[?], the algorithm used in each specific step of this paper is also a algorithm proposed in other paper. Here is only a brief introduction, which will be introduced in detail in the subsequent reading of relevant paper.

9.1 Spatio-temporal hash generation

As video is a kind of spatio-temporal data, video hash is required to represent the spatio-temporal information of the video. The TIRI[15] (called TRF in papre[6]) is generated as follows:

$$F(m, n) = \sum_{k=1}^J w_k F(m, n, k)$$

where $F(m, n)$ is the luminance value of pixels of TIRI. $F(m, n, k)$ is the luminance value of the (m, n) th pixel of the k th frame in a video segment with J frames. w_k is the weight coefficient.

Since TIRI can describe the spatial and temporal content of the video segment at the same time, the TIRIs are robust to the temporal video processing, e.g., frame exchanging, frame dropping, fps changing. The TIRIs are divided into blocks, and the ordinal feature of a TIRI is formed according to the difference in intensity between the blocks. The blocks are arranged in the order of Hilbert Curve. The quantile of order p , i.e. M_p , is adopted to

represent the intensity.

$$M_p = \begin{cases} x_{[l \cdot p]+1}, & l \cdot p \notin Z \\ \frac{1}{2}(x_{[l \cdot p]-1} + x_{[l \cdot p]}) & l \cdot p \in Z \end{cases}$$

where p ranges from 0 to 1, and l is the number of the elements in the block. Z denotes positive integer. $x_{[l \cdot p]}$ is the value of the $l \cdot p$ th element in the ascending order. **(Although I understand the calculation process here, I do not quite understand the meaning of the formula here. Such a definition refers to solving the drawback of the intensity of a single point with the integration of the intensity of each point?)**

In order to generate ST-Hash, the intensity of a TIRI block M can be calculated as:

$$M = \frac{1}{4}M_{0.25} + \frac{1}{2}M_{0.5} + \frac{1}{4}M_{0.75}$$

where $M_{0.25}, M_{0.5}, M_{0.75}$ are the quantiles of the order 0.25, 0.5 and 0.75 respectively.

The ST-Hash is generated according to the intensity difference between adjacent TIRI blocks. Let $V = \langle V[1], V[2], \dots, V[n] \rangle$ denote the TIRIs of the video. Let $V[n] = \langle V^1[n], \dots, V^k[n] \rangle$ denote the blocks of the n th TIRI, where $V^k[n]$ represents the k th block of the n th TIRI in order of Hilbert curve. Let V_n^k denote the intensity value of $V^k[n]$ calculated by $M = \frac{1}{4}M_{0.25} + \frac{1}{2}M_{0.5} + \frac{1}{4}M_{0.75}$. Therefore the ST-Hash $H_n'^k$ of the n th segment is generated as follows:

$$H_n'^k = \begin{cases} 0 & V_n^k \geq V_n^{k+1} \\ 1 & V_n^k < V_n^{k+1} \end{cases}$$

9.2 Visual hash generation

In order to make the video hash reflect the information of visual attention, V-Hash is generated and used to modulate the ST-Hash. The visual attention model proposed in [8] is adopted to generate the saliency map for each frame of a video segment, which have been introduced before.

$$SM = (S \oplus B) \ominus B[8]$$

Then the representative saliency map (RSM) of the video segment generated as follows:

$$RSM(m, n) = \sum_{k=1}^J w_k SM(m, n, k)$$

where $SM(m, n, k)$ is the luminance value of the (m, n) th pixel of the k th saliency map of the video segment, which has J frames. w_k is the same weight coefficient. $RSM(m, n)$ is the luminance value of pixels of RSM. The RSM is divided into blocks, and the blocks are

arranged in the order of Hilbert Curve. Given RSM, the V-Hash $H_n''^k$ is obtained in the way that the SI-Hash is generated from TIRL.

$$H_n''^k = \begin{cases} 0 & RSM^{(k)} \geq RSM^{(k+1)} \\ 1 & RSM^{(k)} < RSM^{(k+1)} \end{cases}$$

where $RSM^{(k)}$ represents the k th block of the RSM in order of Hilbert curve.

9.3 Hash Fusion

The final video hash H_n^k is obtained by combining the ST-Hash and V-Hash as follows:

$$H_n^k = H_n'^k \oplus H_n''^k$$

$$BER = \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^{16} (H_n^k \oplus \hat{H}_n^k)$$

where H_n^k and \hat{H}_n^k are the k th hash bit of n th segment in the reference and query video respectively. \oplus denotes the operator of Exclusive-OR. N is the number of segments in the video. BER is used to measure the error rate of video hash and a threshold T is set to determine whether the query video is a video copy or not. If BER is higher than T , the test video is not a video copy and vice versa.

9.4 Summarization

This paper is basically the same as the author of paper[6]. Although the publication time is earlier than that of paper[6], it is the follow-up work of paper[6]. This paper is mainly about how to hash the TRF that has been generated. Since the algorithm of many details of the article is also the algorithm in the published article, it has not been introduced in detail.

So far, I mainly do not understand in two places. The first is about the Hilbert curve. I understand that the Hilbert curve is a dimensionality reduction method. It reduces the dimension by arrangement of blocks through a Hilbert curve into a one-dimensional sequence, and then performs hash coding. Since the paper does not give a detailed explanation, so here can only be a guess. The second question is, as mentioned in the summary, what is the practical significance of using quantile calculations. Such a definition refers to solving the drawback of the intensity of a single point with the integration of the intensity of each point?

This article is a finishing touch on paper[6] and introduces us how to use TRF for hash encoding after generating a TRF that can represent a video. As for the published

algorithms used in the article, I need to read further to see if I can solve the above two doubts.

10 A robust and fast video copy detection system using content-based fingerprinting[15](2020/04/11)

This paper is an article on the ITIR generation algorithm cited in paper[14].

This paper introduced a video copy detection system that is based on content fingerprinting. The fingerprint extraction algorithm extracts compact content-based signatures from special images, denoted by temporally informative representative images (TRF/ITIR), constructed from the video. **Here, I think the fingerprint can be understood as the feature of the video we extracted, which is a feature vector of the video. In this article, it is a hash function.** Closeness of two fingerprints represents a similarity between the corresponding videos; two perceptually different videos should have different fingerprints.

One point I am interested in is that this article makes a detailed description of the properties of this fingerprint. **A fingerprint should be robust to the content-preserving distortions present in a video. It should also be discriminant, easy to compute, compact, and easy to search for in a large database.**

Robustness: Robustness of a fingerprint requires that it changes as little as possible when the corresponding video is subjected to content-preserving operations, i.e., operations that do not affect the perceptual content of the video.

Discriminant: The fingerprints should also be discriminant, to ensure that two perceptually different videos have distinguishable fingerprints.

There is a trade-off between robustness and discrimination. As a fingerprinting algorithm becomes more robust to distortions, it becomes less sensitive to changes in the content, i.e., it has less discrimination ability.

Easy to computer: More specifically, for online applications, a fingerprinting algorithm should be able to extract the signatures as the video is being uploaded. A computationally demanding algorithm is not suitable for online applications, where thousands of videos need to be examined simultaneously in order to find possible copyright infringements.

Compact: If a fingerprint is not compact, finding a match for it in a very large database can become a time-consuming process.

Secure: For some applications, the fingerprinting system should be secure, so as to prevent an adversary from tampering with it.

Existing video fingerprint extraction algorithms can be classified into four groups based on the features they extract: color-space-based, temporal, spatial, and spatio-temporal. **Color-space-based fingerprints** are mostly derived from the histograms of the colors in specific regions in time and/or space within the video. **Temporal fingerprints** are extracted from the characteristics of a video sequence over time. **Spatial fingerprints** are features derived from each frame or from a key frame. Spatial fingerprints can be further subdivided into global and local fingerprints. *Global fingerprints* depict the global properties of a frame or a subsection of it (e.g., image histograms), while *local fingerprints* usually represent local information around some interest points within a frame (e.g., edges, corners, etc.). For a short segment of a video, SIFT features are not suitable, because there are a large number of SIFT features, this makes them impractical from a memory management point of view when applied to large video databases containing billions of hours of videos. In this paper, they use global features.

10.1 Extracting Robust and Discriminant Fingerprints

Pre-process: Each video is down-sampled both in time and space, which can increase the robustness of a fingerprinting algorithm. Prior to down-sampling, a Gaussian smoothing filter is applied in both domains to prevent aliasing. This down-sampling process provides the fingerprinting algorithm with inputs of fixed size ($W \times H$ pixels) and fixed rate (F frames/second).

Segments: After preprocessing, the video frames are divided into overlapping segments of fixed-length, each containing J frames.

Spatio-Temporal Fingerprinting Using TIRIs: Before introducing their own algorithm, the paper also mentioned a comparison algorithm, which is the method of paper[4] that we introduced in Section 5. Here, I won't repeat them.

1.Generate TIRIs: calculates a weighted average of the frames to generate a representative image.

Let $l_{m,n,k}$ be the luminance value of the (m,n) th pixel of the k th frame in a set of J frames. The pixels of TIRI are then obtained as a weighted sum of the frames

$$l'_{m,n} = \sum_{k=1}^J w_k l_{m,n,k}$$

In this paper, exponential weighting is used, which can best capture the motion.(In paper[6], a more detailed weighted is used.)

They have chosen the exponential weighting function ($w_k = \gamma^k$) for generating TIRIs. A very large γ , or one close to 0, generates a TIRI with detailed spatial information and low temporal information, resulting in a more discriminant representative.

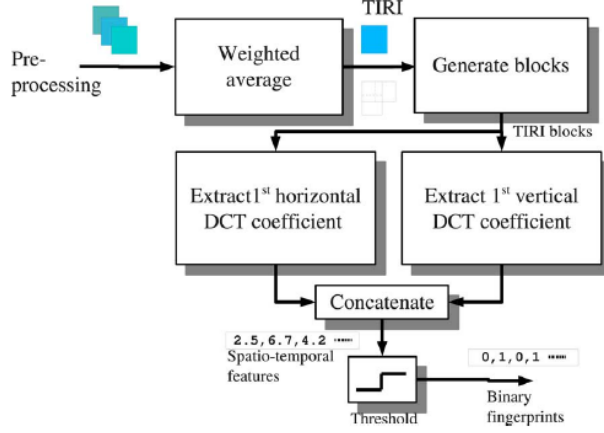


Fig. 18: Schematic of the TIRI-DCT algorithm.

2. TIRI-DCT algorithm: Fig.18 shows the block diagram of their proposed approach. Features are derived by applying a 2D-DCT on overlapping blocks of size $2w \times 2w$ from each TIRI (with 50%) overlap. The first horizontal and the first vertical DCT coefficients (features) are then extracted from each block. The value of the features from all the blocks are concatenated to form the feature vector. Each feature is then compared to a threshold (which is the mean of the median value of the feature vector) and a binary fingerprint is generated, as shown in Fig.19.

10.2 Fast Matching of Fingerprints within A Large Video Database

Fingerprints of two different copies of the same video content are similar but not necessarily identical. So, when matching the video, a close match of the query in the fingerprint database is enough, instead of an exact match.

In real-world applications, the size of online video databases can reach tens of millions of videos, which translates into a very large fingerprint database size. This means that even if the fingerprint of a query video can be extracted very quickly, searching the fingerprint database to find a match may take a long time.

A. Inverted-File-Based Similarity Search

Construction of database: Divide each fingerprint into small non-overlapping blocks of m bits, called *words*. Then a table of size $2^m \times n$, where n is the number of words in a fingerprint of length $L(n - \lceil L/m \rceil)$, can represent the inverted file. The horizontal dimension of this table refers to the position of a word inside a fingerprint, and the vertical direction corresponds to possible values of the word. To generate this table, start with the first word of each fingerprint, and add the index of the fingerprint to the entry in

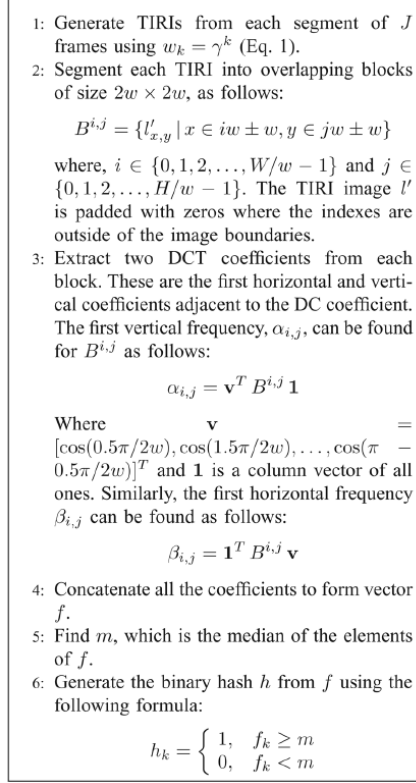


Fig. 19: TIRI-DCT algorithm.

the first column corresponding to the value of this word. Continue this process for all the words in each fingerprint and all the columns in the inverted file table. Entry (i, j) in the table is a list of the indices of all the fingerprints that their j th word is word i . **All the above processes can be regarded as a special hash buckets.**

Search: Divide the query fingerprint into n words (of m bits). The query is then compared to all the fingerprints that start with the same word. The indices of these fingerprints are found from the corresponding entry in the first column of the inverted file table. The Hamming distance between these fingerprints and the query is then calculated. If a fingerprint has a Hamming distance of less than some predefined threshold, it will be announced as the match. If no such match is found, the procedure is repeated for the fingerprints that have exactly the same second word as the query's second word (the indices of these fingerprints are read from the corresponding entry in the second column of the inverted file table). This procedure is continued until a match is found or the last word is examined.

If the word length is chosen correctly, then the worst-case scenario where the algorithm

examines every word of the query will require searching $O(LN/(m2^m))$ queries.

B. Cluster-Based Similarity Search By assigning each fingerprint to one and only one cluster (out of K clusters), the fingerprints in the database will be clustered into K non-overlapping groups. To do so, a centroid is chosen for each cluster, termed the cluster head. A fingerprint will be assigned to cluster k if it is closest to this cluster's head.

To determine if a query fingerprint matches a fingerprint in the database, the cluster head closest to the query is found. All the fingerprints (of the videos in the database) belonging to this cluster are then searched to find a match, i.e., the one which has the minimum Hamming distance (of less than a certain threshold) from the query. If a match is not found, the cluster that is the second closest to the query is examined. This process continues until a match is found or the farthest cluster is examined.

10.3 Summarization

This article is an application to TRF, where TRF is called the fingerprint of the video, which is the feature representation of the video we often say. Although this article does not solve the two problems we raised in the previous section, this article gives a relatively comprehensive description of video features. First, it gives five important properties of video features. Then for each property, an experimental verification is carried out.

Through this article and the previous article on video duplication, we can get a conclusion. There are only two main problems in the process of video duplication. One is how to use a simple and fast algorithm to construct the fingerprint of the video (hash value of the video). Another is providing a fast search database structure and search strategy. In the actual application process, the value of the second question is much greater than the first question.

The first problem is related to the process of constructing the hash code. The methods introduced in these sections are basically an unsupervised method. By constructing the TRF of the video, then use the mean binarization method to directly conduct hash coding. The second problem is relatively complicated, and you need to consider how to build a database with relatively centralized features. The two construction methods mentioned here are tree-like databases, which are checked by layer-by-layer search.

The above two questions basically cover the key content of the entire video review. The current literatures are basically directed to question one, and there are relatively few articles describing the problem two in detail.

11 Hash length prediction for video hashing[16](2020/04/11)

This paper belongs to an paper that draws conclusions and explains based on experimental results.

This paper focus on how to judge the length of the hash code by modeling the existing database. The existing video hashing algorithms for duplicated video detection usually consists of two steps, which are video feature extraction and hash mapping, instead of the length of hash codes. However, the hash length is much more essential, because it influences the computing capacity and speed directly with limited computing resources and network bandwidth. Some research prove that hash length has close relationship with the performance of video hash by investigating the performance of the video hashing algorithm with various hash lengths.

In this paper, the kernel-based supervised hashing (KSH) method is used as an example. The probability distribution functions (PDF) of bit error rate (BER) for copy and non-copy videos are calculated, which are modeled by exponential and Gaussian functions, respectively. Then, the relationship between probability of collision (PoC) and hash length can be established by statistical analysis based on the training data. Finally, according to the difference in quantity between the whole database and the training database, the optimal hash length for the whole database is predicted.

PoC can be estimated according to the PDF of hash BER for copy and non-copy videos in the training database, so PoC can be used to predict the hash length in this paper.

11.1 Length Prediction Method

Suppose there are M sets of videos in the training database, and each video set includes an original video and a few video copies. The video feature F_{ij} is extracted from the i th video set. Here $i = 1, 2, \dots, M$, $j = 0, 1, \dots, J_i$, F_{i0} denoted the original video of the i th set, and J_i is the number of video copies in the i th set. Then, map the video feature F_{ij} to the hashed with various lengths, $H_{ij}^{(r)}$ and $R = 1, 2, \dots, R$ is the hash length. $H_{ij}^{(r)} = [h_{ij1}^{(r)}, h_{ij2}^{(r)}, \dots, h_{ijr}^{(r)}]$ and $h_{ijk}^{(r)}$ is the k th bit of $H_{ij}^{(r)}$, $k = 1, 2, \dots, r$. R is the possible largest hash length determined by the requirements on computation cost, network bandwidth and so on.

The hash BER is calculated as follows:

$$BER = \frac{1}{r} \sum_{k=1}^r (h_k \oplus q_k)$$

where h_k and q_k are the k th hash bit of $H^{(r)}$ and $Q^{(r)}$. $H^{(r)}$ and $Q^{(r)}$ are the hashed of the original and query videos respectively. \oplus denotes Exclusive-OR.

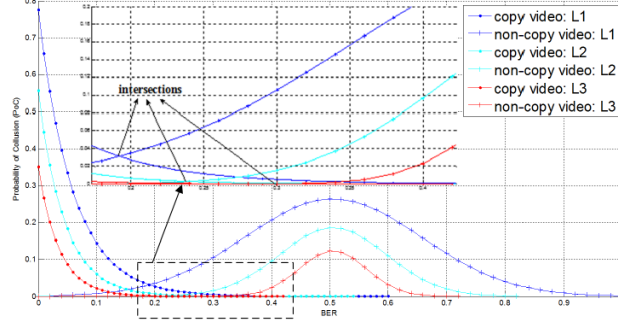


Fig. 20: Comparison on PDFs of hash BER corresponding to different hash lengths, where $L_1 < L_2 < L_3$.

The probability distribution $g_1(x)$ of the hash BER between copy videos usually concentrates at 0, drops sharply as BER becomes larger than 0 and converges to 0 because the copy videos are visually similar, and distinctly visually different from non-copy videos. However, the probability distribution $g_2(x)$ of the hash BER between non-copy videos usually centers near 0.5, drops slowly as BER gradually retreats from 0.5 and finally converges to 0 because the visual difference between copy and non-copy videos is big, though varies to some extent.

In this paper, the PDF of hash BER between copy videos and non-copy videos can be modeled by exponential and Gaussian functions, respectively. $g_1(x)$ and $g_2(x)$ are:

$$g_1(x) = a \cdot k_1^{k_2 x}$$

$$g_2(x) = b \cdot \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where x denotes hash BER. The parameters a, k_1, k_2, b, σ are all related to hash length and they are modeled by the minimum root-mean-square error criterion. σ is the variance of hash BER of non-copy videos. The PDF is shown in Fig.20.

It can be observed that the BER at the intersection of each pair of PDF curves decreases and converges with the increase in hash length. The relationship between the BER at the intersection and hash length is modelled as follows:

$$x_0 = c \cdot r^n + d$$

where x_0 denotes the BER at the intersection. c, d, n can be determined with the training database.

Let P_c denoted the PoC of video hash. Given the PDF of BER, the PoC of video

hash, the area of the cross region under a pair of PDF curves, can be calculated as follows:

$$P_c(r) = \int_{x_0}^{0.5} g_1(x)dx + \int_0^{x_0} g_2(x)dx$$

In the training database, the same video features are mapped to hash with lengths via the same hash mapping method. Then, the optimal hash length for the whole database can be predicted according to the quantity difference between the training and whole database, which can be measured by bits:

$$r_a = \lceil r_t + \log_2 \frac{N_1}{N_t} \rceil$$

where N_a and N_t are the data quantity of the whole and training database respectively, $\lceil \cdot \rceil$ is the top integral function and r_a is the approximately optimal hash length of the whole database.

11.2 Experiment Framework

AF Extraction: Appearance feature is obtained according to paper[15].

VF Extraction: Visual feature is obtained according to paper[14].

Then, the final video feature F is obtained by combining the AF with VF: $F = [AF, VF]$.

KSH Mapping: The main idea of supervised hashing is to make sure that Hamming distances between different hashes are minimized on similar pairs and simultaneously maximized on dissimilar pairs. (**Acquire the hash function by optimal problem.**)

When the r -bit hash code is given by $code_r(F) = [h_1(F), \dots, h_r(F)] \in \{1, -1\}^{1 \times r}$, the inner product is:

$$code_r(F_i) \circ code_r(F_j) = r - 2D_h(F_i, F_j)$$

where the symbol \circ stands for the code inner product and $D_h(F_i, F_j) = |\{k | h_k(F_i) \neq h_k(F_j), 1 \leq k \leq r\}|$ denotes the Hamming distance of two features.

As a consequence, the hash codes of feature F can be learned by optimizing the following objective function Q :

$$\min_{H_m \in \{1, -1\}^{m \times r}} Q = \left\| \frac{1}{r} H_m H_m^T - S \right\|_F^2$$

where $H_m = \begin{bmatrix} code_r(F_1) \\ \vdots \\ code_r(F_m) \end{bmatrix}$ denotes the code matrix of F , S equals 1 for similar pairs and -1 for dissimilar pairs and $\|\cdot\|_F$ indicates the Frobenius norm.

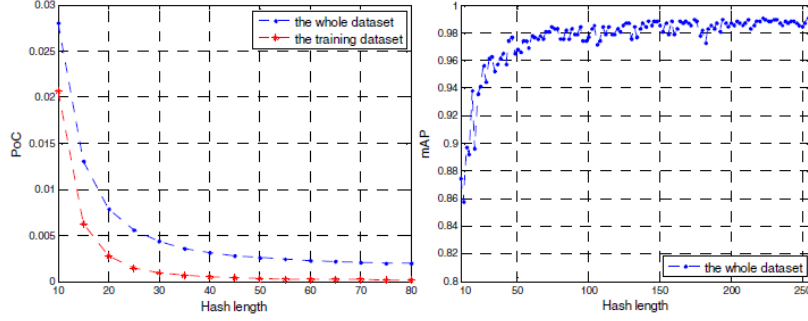


Fig. 21: (a) The relationship between PoC and hash length on the training and whole 78-set database. (b) mAP and hash length in the whole 78-set database.

11.3 Experimental Results and Discussion

In this paper, the PDF of hash BER is modeled according to the characteristics of video copy detection and has no relationship with the hashing algorithm. It means that the PDF model of hash BER is independent from the hashing algorithm.

Result of 78-set database (shown in Fig.21): It can be seen that the PoC-length curves tend to converge with the increase of hash length. The optimal hash length is defined as the approximate shortest hash length within the convergent range of the PoC-length curve. Here, the optimal hash length for the training database is set to $r_t = 40$. Given $N_t = 390$ and $N_a = 858$, the optimal hash length for the whole database is

$$r_a = \lceil 40 + \log_2 \frac{858}{390} \rceil = 41$$

In the ideal assumption, the perfect hash length should be the logarithm of the size of the database. The difference between the predicted optimal hash length and the perfect hash length reflects the representation ability of the corresponding hashing algorithm, where less difference means better representation ability.

11.4 Summarization

This is the first time to read this type of article, directly discuss the setting of the hash and not for a certain purpose. As we all know, the most relevant factor of the length of the hash is the problem of hash collision, that is, if you construct a hash code that is long enough to accommodate the feature representation of all the data in the database. This paper uses empirical distribution to construct the distribution relationship between BER and hash length of different types of videos, and then uses the relationship between

PoC and BER obtained by experiment to construct the relationship between PoC and hash length.

For the hash generation algorithm in this article, both are classic and simple method. Since I have already introduced it in the above summary, I will not discuss it again.

For this article, I think there may be the following problems. **The first is practicality.** According to the idea of this paper, to estimate the length of the hash, we need to sample the large database first. Then according to the PoC curve of the sampling database, choose the hash length corresponding to the convergence point as the optimal length, and then bring it into the formula for calculation. There is a problem here. In the actual problem, the data is increasing with time. It is certainly not reasonable to determine the length of the entire hash code with only one sampling. Secondly, **all the distributions used in the whole process are derived from experimental experience**, and the accuracy needs to be considered.

Although there are some problems in this article, this paper has a high reference value for us to use the optimization problem to solve the supervised hash problem.

12 Deep Attention-guided Hashing[17](2020/04/15)

In this paper, the author propose a novel learning-based hashing method, named Deep Attention-guided Hashing (DAgH), which is implemented using two stream frameworks. The core idea is to use guided hash codes which are generated by the hashing network of the first stream framework (called first hashing network) to guide the training of the hashing network of the second stream framework (called second hashing network).

The first stream framework consists of an attention network and a hashing network (the first hashing network). The role of the first stream framework is to generate the attention-guided hash codes. The second stream framework contains another hashing network i.e. the second hashing network. This hashing network is guided by the attention-guided hash codes which were generated from the first stream framework. In order to guarantee the quality of the final hash codes and eliminate the quantization error, the DAgH model uses a continuous activation function (Smoothing function) to ensure that the first stream framework is a true end-to-end network and a threshold activation function to ensure that the second stream framework directly generates the final hash codes.

All framework is shown in Fig.22.

12.1 Notation & Network Architecture

Training set: $\mathcal{X} = \{x_i\}_{i=1}^N$, each image represented by a d -dimensional feature vector $x_i \in R^d$, where $\mathcal{X} \in R^{d \times N}$.

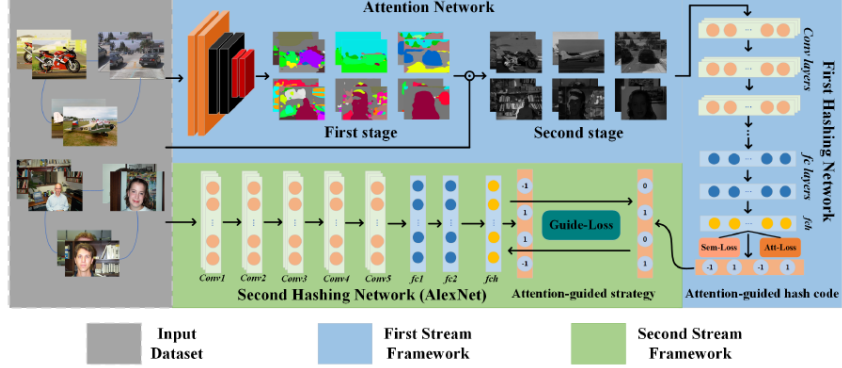


Fig. 22: The framework of the proposed network.

Training pairwise: $S = \{s_{ij}\}$ is derived as:

$$s_{ij} = \begin{cases} 1, & \text{if images } x_i \text{ and } x_j \text{ share same class label} \\ 0, & \text{otherwise} \end{cases}$$

Sign(.):

$$\text{sign}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

Their method includes two stream framework. The first stream contains an attention network and a hashing network. Then the hashing network uses the attention images to generate the attention-guided hash codes. Thereafter, in the second stream, these attention-guided hash codes are treated as supervised labels to train an image hashing network which can generate the final hash codes for the input images.

12.2 The first Stream Framework

This framework is a end-to-end model includes an attention network, i.e., $\text{Attention}(\cdot|\Theta_a)$, and a hashing network, i.e., $\text{Hash}(\cdot|\Theta_1)$.

The Attention Network

The role of the attention network is to find salient regions in the original image that need to get attention. The FCN-based attention network is used to map the original input image pair $[x_i, x_j]$ to the preliminary attention image pair $[\hat{x}_i, \hat{x}_j]$.

For the preliminary attention image pair, a normalization function is used to restrict the value of each pixel between 0 and 1:

$$\text{norm}_i(p, q) = \frac{x_i(p, q) - \min(x_i)}{\max(x_i) - \min(x_i)}$$

where (p, q) denotes the location (p, q) in an image x , $\max(x)$ denotes the maximum pixel value in an image x , $\min(x)$ denotes the minimum pixel value in an image x .

For getting the image pair $[\tilde{x}_i, \tilde{x}_j]$, the Hadamard product \otimes of the original image pair $[x_i, x_j]$ and the normalization function of the preliminary attention image pair $[\hat{x}_i, \hat{x}_j]$ is calculated:

$$[\tilde{x}_i, \tilde{x}_j] = [x_i, x_j] \otimes \text{norm}_{[\hat{x}_i, \hat{x}_j]}(p, q)$$

It is equivalent to re-dividing the value of the image, increasing the value of high attention score, and decreasing the value of low attention score. The feature extraction process of the entire image is influenced by the weighted form.

The First Hashing Network

The generated attention image pair $[\tilde{x}_i, \tilde{x}_j]$ as the input of the first hashing network and the semantic information as the pairwise label to train the first hashing network. The attention-guided hash code $B^{att} = \{b_i^{att}\}_{i=1}^N$ is calculated through the trained hashing network:

$$b_i^{att} = \text{sign}(\text{Hash}(\tilde{x}_i | \Theta_1))$$

where b_i^{att} is the attention-guided hash code, Θ_1 denotes the parameters of the first hashing network, and $\tilde{x}_i = \text{Attention}(x_i | \Theta_a)$ is the attention image that is input into the first hashing network and generated by the attention network, Θ_a is the parameters of the attention network.

Loss Function

Semantic Loss: According to the logistic loss function, the Maximum Likelihood estimation of the hash codes $B = [b_1, b_2, \dots, b_N]$ for all N images is:

$$\log P(S|B) = \prod_{s_{ij} \in S} \log P(s_{ij}|B)$$

where $P(S|B)$ denotes the likelihood function. Given each image pair with their similarity label $([x_i, x_j], s_{ij})$, $P(s_{ij}|b_i, b_j)$ is the conditional probability of s_{ij} given the pair of corresponding hash codes $[b_i, b_j]$, which is naturally defined as logistic function:

$$P(s_{ij}|b_i, b_j) = \begin{cases} \sigma(< b_i, b_j >), & s_{ij} = 1 \\ 1 - \sigma(< b_i, b_j >), & s_{ij} = 0 \end{cases}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid activation function, $< b_i, b_j > = \frac{1}{2} b_i^T b_j$.

Considering the similarity measure, the following loss function is used to learn the hash codes:

$$\mathcal{L}_{sem} = -\log P(S|B) = -\sum_{s_{ij} \in S} \log(s_{ij}|B) = -\sum_{s_{ij} \in S} (s_{ij} < b_i, b_j > - \log(1 + \exp(< b_i, b_j >)))$$

where $b_i = \text{sign}(w_i)$, which converts the K -dimensional representation w_i to exactly binary hash codes.

Attention Loss: Denotes the continuous representation pair of fch layer as $[w_i, w_j]$. The optimal hash code pair $[b_i, b_j]$ is obtained from the continuous representation pair $[w_i, w_j]$. Given $[b_i, b_j] \in \{-1, 1\}^k$, the cosine similarity between the continuous representation pair can be defined as $\cos(w_i, w_j) = \frac{w_i^T w_j}{\|w_i\|_2 \|w_j\|_2}$, which is in the range of $(-1, 1)$. The attention loss is written as below:

$$\mathcal{L}_{att} = \sum_{i,j} \|S_{ij} - \frac{\cos(w_i, w_j) + 1}{2}\|_2 + \sum_{i,j} \max(0, \lambda - \|S_{ij} - \frac{\cos(w_i, w_j) + 1}{2}\|_2)$$

where $\lambda > 0$ is a margin parameter. The attention loss will punish the attention network to make it better capture the salient regions.

Overall Loss: The loss of the first stream framework can be written as:

$$\min_{\Theta_a, \Theta_1} \mathcal{L}_{sem} + v\mathcal{L}_{att}$$

where Θ_a, Θ_1 are the first stream framework parameters efficiently optimized using standard back-propagation with automatic differentiation techniques.

12.3 Smoothing

According the proposed methods, hashing methods first need to learn continuous representations (binary-like codes) through sigmoid and tanh functions, then, the binary-like codes are binarized into hash codes in a separate operation of sign thresholding. The gap between the binary-like codes and hash codes is called the **quantization error**, which caused by the non-smooth and non-convex of the sign function.

sign function can be approached by the tanh function:

$$\lim_{\beta \rightarrow \infty} \tanh(\beta x) = \text{sign}(x)$$

Then a Adaptive Tanh (ATanh) is defined as follow:

$$b_i = \tanh(\beta w_i) + \epsilon \|\frac{1}{\beta}\|_2^2$$

where ϵ is the regularization constant. When β gradually increase, the ATanh function approaches the sign function and has the reliable-ability to generate hash codes.

In this paper, the author first set the parameter $\beta_0 = 1$ as the initialization. At each epoch T , they increase β and fine-tune the first stream framework of DAgH in the next epoch. With the parameter $\beta \rightarrow \infty$, the network will converge to the first stream framework of DAgH with sign as activation function, which can generate high-quality attention-guided hash codes.

This paragraph is very similar to the smoothing technique used in optimization theory. Like the framework we discussed before, in the optimization, as the number of iterations increases, the smooth parameters are constantly changed to make them approach the original problem. Here in the actual process, the parameters are updated through repeated training.

12.4 The Second Stream Framework

Here, adopt a pre-trained AlexNet as the base of the second hashing network. After obtaining the attention-guided hash codes B^{att} , utilize it as the supervised labels and the original images $\mathcal{X} = \{x_i\}_{i=1}^N$ to train the hashing network. When the hashing network is trained, the final hash codes b_i^f are computed through the trained hashing network:

$$b_i^f = \text{sign}(\text{Hash}(x_i|\Theta_2))$$

where b_i^f is the final hash code, Θ_2 denoted the parameters of the second hashing network, and $\tanh(\cdot)$ as activation of the fch layer of the hashing network. fch layer is convert the feature of images to hash code, where each node of fch layer corresponds to 1 bit in the target hash code.

Loss Function

Let $\tilde{y}_i^2 = \text{Hash}(x_i|\Theta_2)$ be the output of the second hashing network, where x_i is the original input image and Θ_2 is the parameter of the second hashing network. Define the following likelihood functions:

$$P(b_{ik}^{att}|\tilde{y}_{ik}^2) = \begin{cases} \sigma(\tilde{y}_{ik}^2), & b_{ik}^{att} = 1 \\ 1 - \sigma(\tilde{y}_{ik}^2), & b_{ik}^{att} = 0 \end{cases}$$

where b_{ik}^{att} is the hash code corresponding to the k -th bit of the i -th element in b_i^{att} , \tilde{y}_{ik}^2 is the output of the k -th node in fch layer of the i -th element, and $\sigma(\cdot)$ is a sigmoid function.

The guide loss is written as follow:

$$\begin{aligned} \mathcal{L}_g &= -\frac{1}{KN} \sum_{k=1}^K \sum_{i=1}^N \log P(b_{ik}^{att}|\tilde{y}_{ik}^2) \\ &= -\frac{1}{KN} \sum_{k=1}^K \sum_{i=1}^N [\log P_{ik}^{b_{ik}^{att}} \cdot \log(1 - P_{ik})^{(1-b_{ik}^{att})}] \\ &= -\frac{1}{KN} \sum_{k=1}^K \sum_{i=1}^N [b_{ik}^{att} \log P_{ik} + (1 - b_{ik}^{att}) \log(1 - P_{ik})] \end{aligned}$$

where N is the number of training images, K is the number of bits in each hash code, and $P_{ik} = \sigma(\tilde{y}_{ik}^2)$.

Algorithm 1 Deep Attention-guided Hashing (DAgH).

Input Training Image pair with their similarity label $([x_i, x_j], s_{ij})$ in the first stream framework, a sequence $1 = \beta_0 < \beta_1 < \beta_2 \dots < \beta_m = \infty$. Training Image x_i and the attention-guided hash codes B^{att} in the second stream framework. Training epochs T_1 and T_2 of the first and second stream framework optimizations, respectively.

Output First stream framework: $sign(Hash(Attention(x_i|\Theta_a)|\Theta_1))$; Second stream framework: $sign(Hash(x_i|\Theta_2))$.

Begin Construct the pairwise information matrix S according to (1).

1. **for** $t = 1 : T_1$ **epoch do**
 2. Compute $[b_i^{att}, b_j^{att}]$ according to (5)
 3. Train the first hashing network (8) with (11) as activation
 4. Compute Θ_a, Θ_1 according to (10)
 5. Set converged the first stream framework as next epoch initialization
 6. **end for**
 7. **return** $sign(Hash(Attention(x_i|\Theta_a)|\Theta_1)), \beta_m \rightarrow \infty$.
-
1. **for** $t = 1 : T_2$ **epoch do**
 2. Compute \hat{y}_i^2 according to $\hat{y}_i^2 = Hash(x_i|\Theta_2)$
 3. Compute Θ_2 according to (14)
 4. Set converged the second stream framework as next epoch initialization
 5. **end for**
 6. **return** $sign(Hash(x_i|\Theta_2))$.
-

Fig. 23: Deep Attention-guided Hashing.

Use the Back-Propagation algorithm to learn the parameter Θ_2 of the second hashing network with stochastic gradient descent (SGD).

12.5 Summarization

This paper is the latest application of visual saliency in image hashing. Unlike the unsupervised hashing method I read earlier, this paper uses a semi-parallel network to integrate visual saliency into the hashing process. As for network structure, the network in the paper consists of two network streams. The process of the entire algorithm is shown in Fig.23.

The first network stream is an end-to-end network. The visual saliency of the image is obtained through learning, and then normalized to a weight in the range of 0 to 1. Then use the Hadamard product to rearrange the gray values of the original image. Keep the gray value of the point with the highest score in the visual saliency basically unchanged (

or changed small) , and lower the gray value of the point with the lowest score in the visual saliency. According to the visual, the area, where the human eye observes significant areas, will be gray values normally. And the area, where the human eye observes insignificant areas, will become completely black. Then the rearranged images are taken as input and input to a hash training network. In this network, the input is rearranged images, and the label is whether the semantics of the video are of a class. Then the entire network is trained to obtain hash codes arranged in saliency maps.

The second network stream is a hash learning network. In this network, the input is a original image, and the label is the hash code given by the first network stream. I think this effect is to enrich the meaning of the label. The label used by the second network no longer has the original semantic information, but has two characteristics of semantic and visual saliency. If their semantics and visual saliency are similar, then their hash codes will be similar, otherwise opposite.

In addition to this two-stream network, another innovation in the article is the handling of quantization error. Due to the non-convex and non-smooth of sign, it is difficult to solve it by gradient in the optimization process of loss function. In most cases, as in optimization theory, the sign is smoothed, and then the smoothed loss function is solved. In this paper, the author uses a method similar to the "envelope family" we discussed earlier for optimization. The adaptive Tanh function is used to approximate the sign. Through multiple training, the degree of approximation of the tanh function to the sign function is continuously increased. I think this technique is the corresponding deformation of the "envelope family" method in deep learning.

The use of visual saliency in this paper is completely different from the paper I read before, and gives a new idea for the issues discussed before.

13 Salient Object Detection: A Survey[18](2020/04/18)

Although this paper is a survey about salient object detection, I think the salient object detection in this paper is a little different to the visual salient which we want to study to use it in hash.

Salient objection detection in this paper is a binary segmentation problem, where a salient map is generated. In this map, 1 represents the most significant area, 0 represents the non-most significant area. In fact, what we want is a weight of salient score, not a binary map. But, there seems to be a connection between the visual saliency we need and the saliency object detection described in the article. Next, I will briefly introduce this review from the perspective of my understanding.

According to some research, the salient object detection is a complex problem, which

can be influenced by many factor, such as age, culture, and experience regulate attention. So, it is difficult to propose a universal and precise framework to detect object for everyone. Some analysis demonstrate that human observers tend to annotate more salient objects first. Others find that interest selections are correlated with eye movements, and both types of data correlate with bottom-up saliency. **So salient object detection can be looked as a subset of objects that can minimally describe a scene.**

In generally, salient object detection in computer vision as a process that includes two stages: 1) detecting the most salient object and 2) segmenting the accurate boundary of that object. For good saliency detection s model should meet at least the following three criteria: 1)good detection: the probabilities of missing real salient regions and falsely marking background as salient regions should be low; 2) high resolution: saliency maps should have high or full resolution to accurately locate salient objects and retain original image information; and 3) computational efficiency: as front-ends to other complex processes, these models should detect salient regions quickly.

According to the description about salient object detection, we can clearly see that there are some difference between the saliency object detection and our needs in the hash. What we hope in hashing is to find out the significant area, so that it has a greater impact on the hash code in the hashing process. This does not mean that the non-significant area has no effect on the hash code at all. On the contrary, if this method is used, the non-significant area will even play a important role. For example, the same person walks on the grassland and in the desert. According to the definition here, it is obvious that the saliency object should be this person, but if we perform hash coding, there should be a certain distance between the two images, rather than just encoding the saliency object.

For the problem of salient object detection, there are many different application directions, as the article introduces the fixation prediction, image segmentation and object proposal generation. Fixation prediction models are constructed to understand human visual attention and eye movement prediction. The aim of image segmentation is to assign each pixel a label indicating the object or background it belongs to. Object proposal generation models or objectness measures attempt to generate a small set of object regions, so that these regions cover every objects in the input image, regardless of the specific categories of those object.

13.1 brief introduction about some models

Most approaches aim to identify the salient subsets (pixels, blocks, superpixels and regions map) from images first and then integrate them to segment the whole salient objects. Most of the existing models aim to produce saliency maps, in this review they divide most of existing salient object detection approaches into three major subgroups according to such two attributes, including *block-based models with intrinsic cues*, *region-based model with intrinsic cues*, and *models with extrinsic cues*.

Block-based models with intrinsic cues can be summarized as finding a point or patch that is the most dissimilar based on the distance between the patch and the patch or pixel and pixel. In other words, salient object detection is defined as capturing the uniqueness, distinctiveness, or rarity of a scene. For example, the saliency of pixel x can be computed as:

$$s(x) = ||I_\mu - I_{w_{hc}}(x)||^2$$

where I_μ is the mean pixel value of the image and $I_{w_{hc}}$ is a Gaussian blurred version of the input image. Similarity, the Gaussian pyramid can be used to compute the saliency score of pixel x :

$$s(x) = \sum_{l=1}^L \sum_{x' \in \mathcal{N}(x)} ||I^{(l)}(x) - I^{(l)}(x')||^2$$

where $\mathcal{N}(x)$ is a 9×9 neighboring window centered at x .

Someone think pixel-level are too poor to support object segmentation, so the analysis is extended to the patch level. The saliency score of a patch p_x centered as pixel x can be defined as

$$s(p_x) = ||\tilde{p}_x||_1$$

where \tilde{p}_x is the coordinates of p_x in the PCA coordinated system.

These approaches usually suffer from two shortcomings: i) high-contrast edges usually stand out instead of the salient object, and ii) the boundary of the salient object is not preserved well (especially when using large blocks). To overcome these issues, more and more methods propose to compute the saliency map based on regions.

Regin-based models with intrinsic cues adopt intrinsic cues extracted from image regions to estimate their saliency scores. It often segment an input image into regions aligned with intensity edges first and then compute a regional saliency map. Generally, the input image is first fragmented into N regions $\{r_i\}_{i=1}^N$. Saliency value of the region r_i can be measured as:

$$s(r_i) = \sum_{j=1}^N w_{ij} D_r(r_i, r_j)$$

where $D_r(r_i, r_j)$ captures the appearance contrast between two regions. Higher saliency scores will be assigned to regions with large global contrast. w_{ij} is a weight term between regions r_i and r_j , which can serve as spatial weighting purpose by giving farther regions less contributions to the saliency score than close ones.

Salient objects, in terms of uniqueness, can also be defined as the sparse noises in a certain feature space where the input image is represented as a low-rank matrix (Robust-PCA). So it can be solved by optimizing the following objective function

$$\min_{L, S} \|L\|_* + \lambda \|S\|_1, s.t. F = L + S$$

where L represents the background while S corresponds to the salient object.

In addition, according to different a priori assumptions, such as focusness prior and segmentation prior, there are many other methods in this paper.

Models with extrinsic cues can be derived from the ground-truth annotations of the training images, similar images, the video sequence, a set of input images containing the common salient objects, depth maps, or light field images. In fact, this is an external data-driven detection method using learning methods.

13.2 Other introduction

Since the evaluation method of the algorithm is basically the same as the algorithm in the judgment field, no additional introduction will be given. I am interested in the binarization method briefly introduced in the article. **As we understand it, if it is only the first step of the saliency object detection, this is basically the same as what we need in the hash. Compared to what we need, one more step here is binarization.** Three methods of binarization are given here.

In the first solution, a image-dependent adaptive threshold for binarizing S is proposed, which is computed as twice the mean saliency of S :

$$T_a = \frac{2}{W \times H} \sum_{x=1}^W \sum_{y=1}^H S(x, y)$$

where W and H are the width and the height of the saliency map S , respectively. The second way to binarize S is to use a fixated threshold which changes from 0 to 255. The third way to perform the binarization is to use the GrabCut-like algorithm, which is a way to use human-computer interaction to tell the algorithm that the current location recognition effect is not good by clicking with a mouse.

13.3 Discussion

According to the evolution about saliency object detection, there is a consistent evolution from block-based analysis to region-based analysis. It comes from three major reasons.

reasons. First, the number of regions is typically much smaller than pixels or blocks, making the computation of high order feature or relations computationally feasible. Second, decomposing an image into perceptually homogeneous elements helps to abstract out unnecessary details and is important for high quality saliency detection. Third, the region itself contains some important cues which could be missing at pixel/patch level, such as shapes, aspect ratios, and perimeter. This suggests that region based analysis tends to be preferred over pixel-level analysis when using in reality.

In addition, there is a consistent trend of moving from local cues to global cues, possibly because the latter tends to assign similar saliency values across similar image regions rather than highlighting only the boundary regions.

13.4 Summarization

Although the saliency object detection mentioned in this article has some gaps with the one I hope to use in the hash, this gap can be avoided (without quantization). Without looking at the threshold problem, the key to these two problems is how to give a larger weight to some areas in the image. The measure of this weight is based on where the person's attention is when they see the image. Since human attention is affected by various factors, it cannot be achieved in a generalized and precise way, so the problem is simplified to find the area in the image that has the largest gap from other areas.

So there are some issues common. For example, a problem is mentioned in the article, that is, if an image has no saliency area, what does the saliency area given by the algorithm represent. Putting it in our video hash, this problem may become more serious. First of all, in the spatial dimension, as proposed in the article, it will produce a "saliency area" with unknown attributes. In the temporal dimension, because other images have saliency areas, it may lead to the view that this image is all saliency.

From the articles I read before, it seems that it is not as complicated as it is considered. So it is difficult to judge whether these problems will appear in the hashing process of the video.

14 State-of-the-Art in Visual Attention Modeling[19](2020/4/22)

This article does not evaluate the performance of existing attention or saliency detection methods in detail. Instead, it describes and evaluates the models that the methods rely on by category. First, about the definitions, this paper give a more subtle definition.

Attention is a general concept covering all factors that influence selection mechanisms, whether they are scene-driven bottom-up (BU) or expectation-driven top-down (TD). The main concerns in modeling attention are how, when, and why we select behaviorally

relevant image regions.

Saliency intuitively characterizes some parts of a scene-which could be objects or regions-that appear to an observer to stand out relative to their neighboring parts. In addition, **saliency map** is a topographic map that represents conspicuousness of scene locations.

Gaze, a coordinated motion of the eyes and head, has often been used as a proxy for attention in natural behavior.

According to the empirical foundations, attentional models have commonly been validated against eye movements of human observers. Eye movements convey important information regarding cognitive processes such as reading, visual search, and scene perception, which often are treated as a proxy for shifts of attention.

Here, the author introduce 13 factors that are used for categorization of attention models. Some of the important ones are the following:

f_1 : **Bottom-up influences**: The bottom-up prompts are mainly based on the features of the visual scene (stimulus-driven) and start from the features.

f_2 : **Top-down influences**: Top-down prompts (goal-driven) are determined by cognitive phenomena such as knowledge, expectations, rewards, and current goals, and are optimized using the goal.

f_3 : **Spatial versus or Spatio-Temporal**: Visual selection is then dependent on both current scene saliency as well as the accumulated knowledge from previous time points.

Overt and Covert Attention: Overt attention is the process of directing the fovea toward a stimulus, while covert attention is mentally focusing onto one of several possible sensory stimuli.

14.1 Attention models

Models are explained based on their mechanism to obtain saliency

1.Cognitive Models (C): Almost all attentional models are directly or indirectly inspired by cognitive concepts. The basic model uses three feature channels color, intensity, and orientation. An input image is sub-sampled into a Gaussian pyramid and each pyramid level σ is decomposed into channels for *Red*(R), *Green*(G), *Blue*(B), *Yellow*(Y), *Intensity*(I), and *local orientations*(O_θ). In each channel, maps are summed across scale and normalized again:

$$f_l = \mathcal{N}(\sum_{c=2}^4 \sum_{s=c+3}^{c+4} f_{l,c,s}), \forall l \in L_I \cup L_C \cup L_O$$

$$L_I = \{I\}, L_C = \{RG, BY\}, L_O = \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$$

These maps are linearly summed and normalized once more to yield the "conspicuity maps":

$$C_I = f_I, C_C = \mathcal{N}(\sum_{l \in L_C} f_l), C_O = \mathcal{N}(\sum_{l \in L_O} f_l)$$

Finally, conspicuity maps are linearly combined once more to generate the saliency map $S = \frac{1}{3} \sum_{k \in \{I, C, O\}} C_k$.

I think that the cognitive model is the initial way to define areas that are significantly different from the surrounding areas as saliency areas. As understood, this approach intuitively fits the simplest human visual mechanism, which is to give greater attention to distinct areas. Cognitive models have the advantage of expanding our view of biological underpinnings of visual attention. This further helps understanding computational principles or neural mechanisms of this process as well as other complex dependent processes such as object recognition.

2. Bayesian Models (B): Bayesian modeling is used for combining sensory evidence with prior constraints. In this models, prior knowledge (e.g., scene context or gist) and sensory information (e.g., target features) are probabilistically combined according to Bayes'rule (e.g., to detect an object of interest).

Let Z denote a pixel in the image, C whether or not a point belongs to a target class, and L the location of a point (pixel coordinates). Also, let F be the visual features of a point. The saliency s_z of a point z is defined as $P(C = 1 | F = f_z, L = l_z)$, where f_z and l_z are the feature and location of z . Using the Bayes rule and assuming that features and locations are independent and conditionally independent given $C = 1$, then the saliency of a point is:

$$\log s_z = -\log P(F = f_z) + \log P(F = f_z | C = 1) + \log P(C = 1 | L = l_z)$$

The first term at the right side is the self-information (bottom-up saliency) and it depends only on the visual features observed at the point Z . The second term on the right is the log-likelihood, which favors feature values that are consistent with prior knowledge of the target (e.g., if the target is known to be green, the log-likelihood will take larger values for a green point than for a blue point). The third term is the location prior which captures top-down knowledge of the target's location and is independent of visual features of the object.

As we all known, the key of the Bayesian model is to learn a transformation function (probability) from the known data, so that it can use prior knowledge

to judge whether the current point is salient. But this method requires a lot of data to find the transformation function. A key benefit of Bayesian models is their ability to learn from data and their ability to unify many factors in a principled manor. Bayesian models can, for example, take advantage of the statistics of natural scenes or other features that attract attention.

3.Decision Theoretic Models (D): The decision-theoretic interpretation states that perceptual systems evolve to produce decisions about the states of the surrounding environment that are optimal in a decision theoretic sense (e.g., minimum probability of error).

In this method, salient features are those that best distinguish a class of interest from all other visual classed. The top-down attention is defined as classification with minimal expected error. Given some set of features $F = \{F_1, \dots, F_d\}$, a location l , and a class label C with $C_l = 0$ corresponding to samples drawn from the surround region and $C_l = 1$ corresponding to samples drawn from a smaller central region centered at l , the judgment of saliency then corresponds to a measure of mutual information, computed as $I(F, C) = \sum_{i=1}^d I(F_i, C)$. Then, use DOG and Gabor filters to measure the saliency of a point as the KL divergence between the histogram of filter responses at the point and the histogram of filter responses in the surrounding region.

Since both decision theory and Bayesian methods have a biologically plausible implementation, there is a certain relationship between them. Decision theoretic models have been very successful in computer vision applications such as classification while achieving high accuracy in fixation prediction.

4.Information Theoretic Models (I): Based on the premise that localized saliency computation serves to maximize information sampled from one's environment, the models deal with selecting the most informative parts of a scene and discarding the rest.

For each pixel, the resulting saliency map represents the statistical likelihood of its feature matrix F_i given the feature matrices F_j of the surrounding pixels:

$$s_i = \frac{1}{\sum_{j=1}^N \exp\left(\frac{-1+\rho(F_i, F_j)}{\sigma^2}\right)}$$

where $\rho(F_i, F_j)$ is the matrix cosine similarity between two feature maps F_i and F_j , and σ is a local weighting parameter. The columns of local feature matrices represent the output of local steering kernels, which are models as

$$K(x_l - x_i) = \frac{\sqrt{\det(C_i)}}{h^2} \exp\left\{\frac{(x_l - x_i)^T C_l (x_l - x_i)}{-2h^2}\right\}$$

where $l = 1, \dots, P$, P is the number of the pixels in a local window, h is a global smoothing

parameter, and the matrix C_l is a covariance matrix estimated from a collection of spatial gradient vectors within the local analysis window around a sampling position $x_l = [x_1, x_2]^T$.

5. Graphical Models (G): A graphical model is a probabilistic framework in which a graph denotes the conditional independence structure between random variables. Attention models in this category treat eye movements as a time series. Since there are hidden variables influencing the generation of eye movements, approaches like Hidden Markov Models (HMM), Dynamic Bayesian Networks (DBN), and Conditional Random Fields (CRF) have been incorporated.

A scale-space pyramid is first derived from image features: intensity, color, and orientation. Then a fully connected graph over all grid locations of each feature map is built. Weights between two nodes are assigned proportional to the similarity of feature values and their spatial distance. The dissimilarity between two positions (i, j) and (p, q) in the feature map, with respective feature values $M(i, j)$ and $M(p, q)$, is defined as:

$$d((i, j) || (p, q)) = \left| \log \frac{M(i, j)}{M(p, q)} \right|$$

The directed edge from node (i, j) to node (p, q) is then assigned a weight proportional to their dissimilarity and their distance on lattice M :

$$w((i, j), (p, q)) = d((i, j) || (p, q)) \cdot F(i - p, j - q)$$

where $F(a, b) = \exp -\frac{a^2 + b^2}{2\sigma^2}$.

The resulting graphs are treated as Markov chains by normalizing the weights of the outbound edges of each node to 1 and by defining an equivalence relation between nodes and states, as well as between edge weights and transition probabilities. Their equilibrium distribution is adopted as the activation and saliency maps. In the equilibrium distribution, nodes that are highly dissimilar to surrounding nodes will be assigned large values. The activation maps are finally normalized to emphasize conspicuous detail, and then combined into a single overall map.

Graphical models could be seen as a generalized version of Bayesian models. This allows them to model more complex attention mechanisms over space and time which results in good prediction power. The drawbacks lie in model complexity, especially when it comes to training and readability.

6. Spectral Analysis Models (S): Instead of processing an image in the spatial domain, models in this category derive saliency in the frequency domain.

Given an input image $I(x)$, amplitude $A(f)$, and phase $P(f)$ are derived. Then, the log spectrum $L(f)$ is computed from the down-sampled image. Form $L(f)$ with $h_n(f)$,

which is an $n \times n$ local average filter, and subtracting the result from itself. Using the inverse Fourier transform, they construct the saliency map in the spatial domain. The value of each point in the saliency map is then squared to indicate the estimation error. Finally, they smooth the saliency map with a Gaussian filter $g(x)$ for better visual effect. The entire process is summarized below:

$$A(f) = R(F[I(x)])$$

$$P(f) = \varphi(F[I(x)])$$

$$L(f) = \log A(f)$$

$$R(f) = L(f) - h_n(f) * L(f)$$

$$S(x) = g(x) * F^{-1}[\exp(R(f) + P(f))]^2$$

where F and F^{-1} denote the Fourier and Inverse Fourier Transforms, respectively. P denotes the phase spectrum of the image and is preserved during the process. Using a threshold they find salient regions called proto objects for fixation prediction.

Spectral analysis models are simple to explain and implement. While still very successful, biological plausibility of these models is not very clear.

7. Pattern Classification Models (P): Machine learning approaches have also been used in modeling visual attention by learning models from recorded eye-fixations or labeled salient regions. Typically, attention control works as a “stimuli-saliency” function to select, re-weight, and integrate the input visual stimuli.

The advantage of this approach is that it does not need a priori assumptions about features that contribute to salience or how these features are combined to a single salience map. As available eye movement data increases and with a wider spread of eye tracking devices supporting gathering mass data, these models are becoming popular. This however, makes models data-dependent, thus influencing fair model comparison, slow, and to some extent, black-box.

14.2 Summarization

According to the author’s evaluation, the decision-theoretic model seems to be the best algorithm that balances computational complexity and accuracy.

At present, from the analysis of these models given in the article. First of all, the first model is the most common, that I have touched before. This model is based on a visual point of view, and then uses the obvious gap between the area and the surrounding area for detection. From the perspective of the model process, it can be used for feature extraction processing. The second model is the Bayesian model. From the structure of

the model, the final result of the Bayesian model is trend to the saliency map, so it is not suitable for feature extraction. The third, fourth, and fifth models are relatively broad, so I do not understand the entire process very well and need to further check the relevant literature to judge. The sixth model is similar to the Fourier transform denoising as a whole. It uses the concentration of information in the frequency domain, then determines the saliency in the frequency domain, and finally uses the inverse Fourier transform to return to the time domain. I feel that the calculation involved in this model is relatively complex, and the possibility of combining with feature extraction is not particularly great. As for the seventh model, it involves the attention mechanism in deep learning, which requires training through data. If you do not consider the use of attention in deep learning algorithms, this item does not seem to be necessary.

15 Saliency Detection on Light Field: A Multi-Cue Approach[20](2020/04)

This paper introduces a method for saliency detection using light field camera imaging. The so-called light field is a four-dimensional parametric representation, which is a four-dimensional optical radiation field containing both position and direction information in space. Briefly, it covers all the information that light is propagating. The light field information used in this paper can be understood as more information of the depth of field, besides the ordinary two-dimensional image. The most significant characteristic of a light field is that it can generate a stack of images focusing at different depths and multiple viewpoints towards the scene on a 2D sampling plane in one shot. The article mainly uses this image sequence for saliency detection. In this paper, the author develop a new saliency detection scheme by using the light field and present a new light-field saliency analysis to date.

According to the paper, firstly, saliency prior maps are generated from several light-field features based on superpixel-level intra-cue distinctiveness. Then use the location prior to enhance the saliency maps. Finally, employ a two-stage saliency refinement to obtain the final saliency map.

The author think the the idealistic assumptions of 2D database, such as the saliency regions should be occlusion free and should have a different color from the their neighborhood/background, limit the saliency method. Because the light-field data can e converted into various 2D images, it can be used to improve the saliency detection method.

Before introducing the whole framework, first give a simple linear iterative clustering algorithm(SLIC) used in the article to do image segmentation.

15.1 Simple iterative cluster algorithm

This algorithm performs a preliminary segmentation of the image by simply iterative clustering. There are mainly the following steps:

1)**Initialize the image segmentation block:** each image block is a cluster. The center of the cluster is called superpixel, the number of clusters k is given. The SLIC algorithm first divides the image into image patches of the same size. Assume the number of pixels in the image is N , and the number of image patches to be divided is k , then the size of each patch is $S \times S$, where $S = \sqrt{N/k}$.

2)**Initialize the clustering center:** in the divided image blocks, a point is randomly sampled as the center of clustering. In order to avoid that the initial point of sampling is noise or at the edge, the algorithm has made a little change. Calculate the gradient of adjacent pixels in the 3×3 area near the sampling point, and select the point with the smallest gradient among the adjacent points as the clustering center.

3)**Calculate the distance from pixels to the cluster center:** calculate the distance of each pixel in the image from the cluster center. This is different from the usual clustering algorithm. The general clustering algorithm calculates the distance of pixels from each cluster center. In other words, each cluster center must calculate the distance from all pixels. The SLIC algorithm simplifies this step and only calculates the distance between the pixels of the $2S \times 2S$ range class around each cluster center and the cluster center. The distance is given as follow:

$$D = \sqrt{(d_c)^2 + (\frac{d_s}{S})^2 m^2}$$

where d_c is the distance of the color and d_s is the distance of spital. They can be calculate in color space:

$$d_c = \sqrt{(r_j - r_i)^2 + (g_j - g_i)^2 + (b_j - b_i)^2}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

4)**Re-clustering:** after calculating the distance, each pixel will update its own image block. The pixels of the same image block are averaged to obtain a new cluster center. Then repeat the previous steps until the distance between the two cluster centers is less than a certain threshold.

15.2 Approach of this paper

Given a light-field sample, consisting of an all-in-focus image I_{af} , a depth image I_{depth} , a focal stack, and multi-view cuts (images), segment the all-in-focus image I_{af} into a set of non-overlapping regions/superpixels $\{sp_i\}_{i=1}^M$ by the SLIC, where $M = h \times w/N$ is the

superpixel number, N is the number of pixels within each superpixel, and h and w are the height and width of the current image, respectively.

15.2.1 Estimate the superpixel-level feature distinctiveness

For each cue, estimate the superpixel-level feature distinctiveness $d^k(sp_i, sp_j)$ between superpixel sp_i and superpixel sp_j over the corresponding light-field image plane. For a cue k , the superpixel level distinctiveness is measured by

$$d^k(sp_i, sp_j) = \|m^k(sp_i) - m^k(sp_j)\|_2, j = 1, \dots, M \quad \text{and} \quad j \neq i$$

where sp_i and sp_j denote the i th and j th superpixels, respectively, in the corresponding light-field image. $m^k(\cdot)$ denotes the average feature (color, depth, or flow) values of each superpixel for cue k .

Color: Color measures how the intensity of incoming light varies with wavelength. In this work, the color measurement is derived from the all-in-focus image I_{af} in which each pixel is encoded by RGB values. Transform the RGB image I_{af} into the L.a.b. color space. The final color distinctiveness between two superpixels in the all-in-focus image I_{af} is equivalent to the summation of pairwise distances over three color channels, $d^{color}(sp_i, sp_j) = \sum_c d_c^{color}(sp_i, sp_j)$, where c indexes the three color channels of L.a.b. color space.

Depth: Each pixel in the depth image describes the distance of the surface of a object from a viewpoint. Based on the observation that the region at a closer depth range tends to be more salient, compute the depth distinctiveness $d^{depth}(sp_i, sp_j)$ between two superpixels from the light-field depth image I_{depth} .

Flow: Salient regions usually are closer to the camera than background regions, and, in general, they will display more apparent motion than background regions. For a light-field image sequence $\{I_f\}_{f=1}^F$, describe the light-field flow at the pixel (x, y) by a 3D vector field $(\Delta x, \Delta y, 1)$, which is the displacement vector between two consecutive images (I_f and I_{f+1}). The flow displacement $(\Delta x, \Delta y)$ is computed by the optimization algorithm. Then estimate the light-field flow map as the average value of the square root of flow displacements over all the focal slices or the viewpoints, $\frac{1}{F} \sum_{f=1}^F \sqrt{\Delta x^2 + \Delta y^2}$. The author term the light-field flow map derived from the stack of focal slices as *focusing flow* and the light-field flow map derived from all the viewpoints as *viewing flow*. Finally, calculate the focusing flow distinctiveness $d^{focusFlow}(sp_i, sp_j)$ and the viewing flow distinctiveness $d^{viewFlow}(sp_i, sp_j)$ between two superpixels.

15.2.2 Use the location prior

Incorporate the location prior into the scheme by computing two location cues: *implicit location* $l_{im}(sp_i, sp_j)$ and *explicit location* $l_{ex}(sp_i)$, where $l_{im}(sp_i, sp_j)$ measures the spatial similarity between two superpixels and $l_{ex}(sp_i)$ measures the center bias by computing the Gaussian distance between the centers of the superpixel sp_i and the all-in-focus image.

Implicit Location: Compute the spatial similarity $l_{im}(sp_i, sp_j)$ between the centers of two superpixels sp_i and sp_j by

$$l_{im}(sp_i, sp_j) = \exp(-\frac{1}{2\sigma_u^2} \|u(sp_i) - u(sp_j)\|_2^2)$$

where $u(\cdot) = (\bar{x}/h, \bar{y}/w)^T$ denotes the normalized center coordinate of the given superpixel, where (\bar{x}, \bar{y}) denotes the average center coordinates that is computed by averaging the coordinates of all the pixel within the superpixel, h and w denote the height and weight of the all-in-focus image, and σ_u is a radius parameter.

Explicit Location: Measure the center bias by calculating the distance between the image centroid u_c and the center coordinate $u(sp_i)$ of superpixel sp_i . Calculate the highest background probability $m^{bg}(sp_i)$ for each superpixel from the corresponding focal slice. Use a Gaussian function to obtain the location weight

$$l_{ex}(sp_i) = \exp(-\frac{1}{2\sigma_c^2} (1 - m^{bg}(sp_i))^2 \|u_c - u(sp_i)\|_2^2)$$

where σ_c is a radius parameter.

15.2.3 Initial saliency map

Compute the initial saliency map of each superpixel over all the feature (cue) spaces by

$$S^*(sp_i) = \sum_k \sum_{j=1}^M a_k l_{ex}(sp_i) l_{im}(sp_i, sp_j) d^k(sp_i, sp_j), j \neq i$$

where $a_k > 0$ is the weight assigned to the cue k , $\sum_k a_k = 1$.

The basic idea behind the equation is that the superpixel that differs from its neighboring regions should correspond to high saliency values. The implicit location cue $l_{im}(sp_i, sp_j)$ is used to weight the feature distinctiveness $d^k(sp_i, sp_j)$ by considering that closer regions should have a higher voting weight. The explicit location cue $l_{ex}(sp_i)$ also functions as a weight factor by following an assumption that a region is more likely to be salient if it is close to the center of the image.

15.2.4 Normalize

Normalize the initial saliency map to a range of $[0, 1]$ by $\mathcal{N}(x) = \frac{x - \min(x)}{\max(x) - \min(x)}$, where x represents the vector of all superpixels' saliency values. The normalized initial saliency map is denoted by $S(sp_i)$. Consider the region in the whole saliency map that has a larger value than a global threshold to belong to the salient region.

15.2.5 Produce the final saliency map

Explore the light-field structure cue to refine the saliency values by considering the interrelationship between adjacent elements (e.g., neighboring nodes are more likely to share similar visual properties and saliency values). Apply a two-stage structure-preserving refinement strategy to produce the final saliency map. Utilize internally homogeneous and boundary-preserving superpixels as basic representation units and refine the normalized initial saliency map $S(sp_i)$ by exploring the salient (foreground) term $S(\cdot)$ and the background (non-salient) term $S_{bg}(\cdot)$ based on the boundary connectivity. Optimize $S(sp_i)$ by

$$\min_{S_{opt1}(sp_i)} \sum_{i=1}^M S(sp_i) \|S_{opt1}(sp_i) - 1\|^2 + \sum_{i=1}^M S_{bg}(sp_i) \|S_{opt1}(sp_i)\|^2 + \sum_{i,j=1}^M w(sp_i, sp_j) \|S_{opt1}(sp_i) - S_{opt1}(sp_j)\|^2$$

Here, the first term defines the cost that encourages a superpixel sp_i with a large foreground saliency probability $S(sp_i)$ to take a large saliency value $S_{opt1}(sp_i)$. The second term defines the background cost with a small saliency value $S_{opt1}(sp_i)$. The last encourages the adjacent superpixels to have similar saliency values, in which

$$w(sp_i, sp_j) = \exp\left(-\frac{1}{2\sigma_s^2} \sum_k d^k(sp_i, sp_j)\right) + \gamma \exp\left(-\frac{1}{2\sigma_a^2} d^{color}(sp_i, sp_j)\right)$$

where the first term measures the similarity across different cues, the second term is the color affinity of every adjacent superpixel pair, γ is a small regularization parameter, and σ_s and σ_a modulate the strengths of the weights. Then use the region saliency S_{opt1} as the new query to construct a saliency map S_{opt2} . Finally, the optimized saliency map is obtained by the scalar production of the two steps of saliency refinement results, that is, $S_{opt} = S_{opt1} \cdot S_{opt2}$.

I didn't fully understand the structure part, but according to a comparison chart shown in Fig.24, it can be seen that the role of structure information is to delete redundant information. First, human saliency is considered to be only 0 and 1, that is, only significant and non-significant. However, since the initial saliency map is not an absolute binarization process, there is a saliency score between $[0,1]$. Therefore, structural information is used here. Under

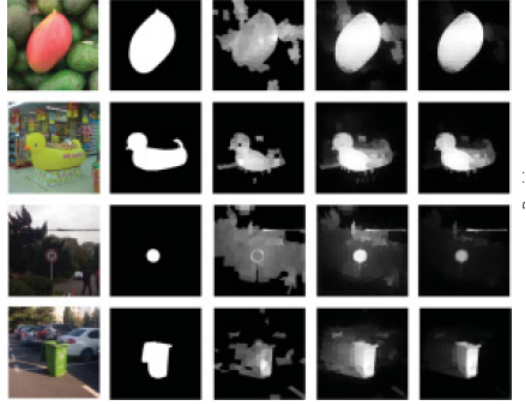


Fig. 24: From left to right: all-focus images, ground truth, w/o refinement, w/ R1, and w/ R2.

the assumption that the saliency between adjacent regions is not too large, the optimization problem is used to optimize the saliency of the surrounding regions. Through the optimization of the two-step structure information, the final result, that is, the last column of graphs, is closer to the preset saliency.

15.3 Summarization

The method introduced in this article is based on the saliency detection algorithm completed by the light field diagram. There are three main principles behind it. The first one is the saliency (visual) judged by calculating the distance in a given space, the second one is saliency judged by measuring the distance between the area and the center of the image. And the third one is the saliency that visual properties and saliency of adjacent areas are basically consistent. In this paper, regarding the optimization of the saliency of the image structure, I think it needs further discussion. I think this method in the article is somewhat similar to an adaptive threshold. Instead of binarizing the saliency score, it optimizes the saliency map through structural attribute optimization. However, this optimization is based on the suppose that the significance in human eyes is only 0 and 1, and there is no significant score between $[0, 1]$. If this suppose is applied to the application of target detection, I think it is reasonable, but some are not actual. Actually, I think that the visual salience of a person is a weight distribution of a full field of vision. Therefore, whether this optimization method makes up for the insufficiency of binarization makes the saliency closer to what I said, or whether it makes the saliency closer to the binarization result as in the article, should be discussed.

In addition, I think this article has some merits. Perhaps it can give us an inspiration. When we do saliency detection, especially in the video, we can introduce the a priori of other frames for the saliency detection of one frame. If we put it in an image, we can detect saliency of an image by introducing images of different scales, or doing an operation such as a rotation of the image. Then use the priors of these other changed images to optimize the saliency detection of one image. Although this article is about saliency detection for light field images, it is actually to optimize the saliency detection of the original image by using the information of other images in the light field. I think this is one of the inspirations this article can give us.

Another point in this article, I think it is the processing of image segmentation. Instead of using rectangular patches, they use the SLIC algorithm to find a more reasonable patch. In the subsequent processing, I think we can learn from it.

16 A Weighted Sparse Coding Framework for Saliency Detection[21](2020)

According to the summarization previous, we can know the saliency detection method for different data format have different construct, which mainly due to the feature extraction method for different data structures. In the article about hashing, the method of feature extraction has an important impact on the generation of hash codes. However, in saliency detection, it seems that its impact is no longer as great as in hashing. **The core of saliency detection method is to develop an effective feature contrast measure to separate salient objects from the background and tremendous efforts have been focused on extract color, texture, depth, and focusness cues from 2D images.** In other words, the core of the hash algorithm's problem lies in the selection of effective features, while the core of the saliency detection lies in how to determine that the proposed features have obvious differences given a feature extraction method. So, this paper present a unified saliency detection framework for handling heterogenous types of input data.

This paper uses sparse coding for this heterogeneous data. We are not new to sparse coding. In some of the previous ideas, although sparse coding is possible, how to change the feature comparison method to sparse coding is the key to this method. In other words, how to build a dictionary is a bridge between these two problems. For the construction of the bridge (dictionary), reference is made to [22], so we will discuss [22] first.



16.1 A Unified Approach to Salient Object Detection via Low Rank Matrix Recovery[22]

In this method, an image is represented as a low-rank matrix plus sparse noises in a certain feature space, where the non-salient regions can be explained by the low-rank matrix, and the salient regions are indicated by the sparse noises. **Uniqueness is a point of view for saliency, because salient regions can be regarded as those that cannot be well "explained" by its surroundings. However, not all unique regions are salient, and a small region with high local contrast might be considered as meaningless noise by the human.**

16.1.1 Construct Feature matrix (dictionary)

: Given an image, extract different types of visual features around each pixel, including:

Color (5-dimension). Three RGB color values as well as the hue and the saturation components are extracted for each pixel, producing 5 color features. Each feature is normalized by subtracting its median value over the entire image;

Steerable pyramids (12-dimension). Steerable pyramid filters with four directions on three different scales are performed on the image, yielding 12 filter responses at each location;

Gabor filters (36-dimension). Gabor filter responses with 12 orientations and 3 scales are extracted.

All those 53 features are then stacked vertically to form a feature vector, which captures color, edge and texture that are most common low-level visual features. Then, perform image segmentation based on the extracted features by mean-shift clustering. Select spatial and feature bandwidths to over-segment the image. So the image is decomposed into segments $\{B_i\}_{i=1,\dots,N}$, where N is the number of segments. For each B_i , use the feature vector of its cluster center, as its feature representation f_i . By stacking f_i into a matrix, we can get the feature matrix representation of the entire image $F = [f_1, f_2, \dots, f_N]$, $F \in R^{D \times N}$, where D is the dimension of the feature vector ($D = 53$ here).

At this point, the global feature is turned into a local global feature with location information, which is the premise of the method to continue.

16.1.2 Saliency detection by the feature matrix

Consider the image as a combination of a background residing in a low dimensional space with salient objects as sparse noises. Therefore, F can be decomposed into two parts



$F = L + S$, where L is the low-rank matrix corresponding to the background while S is a sparse matrix representing the salient regions. This problem can be solved by r-PCA:

$$(L^*, S^*) = \arg \min_{L, S} \text{rank}(L) + \lambda \|S\|_0$$

$$s.t. \quad F = L + S$$

After obtaining S , the l_1 -norm of each column S_i in S is used to measure the saliency of corresponding segments. If $\|S_i\|_1$ is larger, assign a higher value to the image region of the i -th segment B_i . A saliency map is then accordingly generated and normalized to be a gray-scale image. However, there are some problems with the saliency map obtained in this way.

Actually, sparse coding only ensures that the coded vector for each image patch is a sparse vector, which is not equivalent to the sparsity of the saliency over the entire image, or the low rank of the background matrix. To address this problem, a linear transformation T from a set of training images is used. So the feature can be represented as $g_i = T f_i$ where $T \in R^{D \times D}$.

Given an training image, perform feature extraction and image segmentation as well, and the image is accordingly represented by $F = [f_1, f_2, \dots, f_N]$. Given the labeled rectangle, use q_i to indicate whether or not f_i belongs to the salient regions ($q_i = 0$ when the corresponding region is salient and 1 otherwise). Then, the information is represented in a diagonal matrix $Q = \text{diag}(q_1, q_2, \dots, q_N)$. Multiply Q to transformed feature matrix TF , get $TFQ \in R^{D \times N}$. So TFQ only has the information of the background and should have a low rank given a good transformation. The problem of learning T can be formulated as:

$$T^* = \arg \min_T O(T) \triangleq \frac{1}{K} \sum_{k=1}^K \|TF_k Q_k\|_* - \gamma \|T\|_*$$

$$s.t. \quad \|T\|_2 = c$$

where F_k and Q_k are the feature representation and saliency indicator of the k -th training image respectively. K is the number of training images. Given $T = I$ (identity matrix), the problem can be solved by gradient method. The learned T is then used for saliency detection for all the images.

Accordingly, $G = [g_1, g_2, \dots, g_N] = TF$, the problem can reformulate:

$$(L^*, S^*) = \arg \min_{L, S} \text{rank}(L) + \lambda \|S\|_0$$

$$s.t. \quad TF = L + S$$

Similarity, the higher-levels based on human perception can be integrated to this model.

Location prior. Generate a prior map using a Gaussian distribution based on the distances of the pixels to the image center, in which $p_l(x) = \exp(-d(x, c)/\sigma_1^2)$.

Semantic prior. Perform face detection on the images. The regions near the detected faces are assigned higher prior $p_f(x) = \exp(-d(x, f_c)/\sigma_2^2)$, where f_c is the center of the face.

Color prior. For each quantized color, get the values from the two histograms, denoting by h_S and h_B . Set the color prior for saliency to be $p_c = \exp((h_S(c_x) - h_B(c_x))/\sigma_3^2)$, where c_x indicates the color at location x .

These prior maps are then multiplied together to produce a final prior map. According to the prior map, the probability of being salient for each segment based on the location of the segment can be calculated, which is denoted by p_i . Such prior or probability can also be represented by a diagonal matrix $P = \text{diag}(p_1, p_2, \dots, p_N)$. P can be naturally incorporated in formulation:

$$(L^*, S^*) = \arg \min_{L, S} \text{rank}(L) + \lambda \|S\|_0$$

$$s.t. \quad TFP = L + S$$

After that, in P most of p_i are relatively small. Feature vectors multiplied by a larger p_i will be considered as outliers in the low rank matrix L and more likely to be included in the sparse noise matrix S .

16.1.3 Summary for this paper

In summary, this article uses linear representations between regions to find significant regions. The key is to build a bridge between saliency detection and low rank decomposition. Especially the article's use of high-level is very worthy of our further thinking. Then return to article [21] to continue the discussion.

16.2 Construct feature matrix

Use SLIC to segment the reference image I into a set of small non-overlapping regions/superpixel $R = \{r_1, r_2, \dots, r_N\}$. Analogous to article [22], the feature matrix in the article is constructed as follows:

2D feature (6+36+1=43-dimension). Choose both RGB and Lab color spaces as color descriptors (6-dimension). For texture, use the Gabor filter responses with 12 orientations and 3 scales as texture descriptors. For focusness, utilize the mean distance to its 8-neighbors in the RGB space:

$$\sigma_f(p) = \frac{1}{8} \sum_{m=1}^8 \delta_m(p, p_m)$$

where $\delta(p, p_m) = \|p^{rgb} - p_m^{rgb}\|_2^2$ and p^{rgb} is the color vector of p in RGB color space.

3D feature (1-dimension). 3D data further provides depth/disparity information for each points in the scene. When disparity/depth is available, append it to the features vector.

4D feature (L-dimension). If a focus stack has L different focus slices, calculate L focusness values $\sigma_f^l(p), l = 1, 2, \dots, L$ on each slice.

So get the stack vector $f_p = [\sigma_1 \sigma_2 \dots \sigma_C]^T$ of p , where $C = 6 + 36 + 1 + 1 + L$. From the feature vectors of all pixels, generate two feature matrices for all super-pixels.

Average. Convert per-pixel feature vector to per-superpixel feature feature vector through averaging. Use the $C \times N$ matrix F^A to represent the result feature matrix.

Color Histogram. Treat color in terms of ratios $\{\frac{R}{R+G+B}, \frac{G}{R+G+B}\}$ and compute the histogram of the two channels. Discrete the two channels into 32×32 bins for computing the histogram. Consequently the color components of the feature vector for a superpixel becomes $\{\sigma_1^{r_i}, \sigma_2^{r_i}, \dots, \sigma_{1024}^{r_i}\}$. Use the $C' \times N$ matrix F^H to represent the resulting feature matrix.

16.3 Dictionary Based Saliency Detection

Develop a sparse coding framework: saliency superpixels correspond to the ones that yield to low/high reconstruction error from the saliency/nonsaliency dictionary. For saliency detection, the weighted sparse coding scheme is used:

$$\alpha_i = \arg \min_{\alpha_i} \|f_i - D\alpha_i\|_2^2 + \lambda \|diag(w_i) \cdot \alpha_i\|_1$$

where the j th value of w_i is the penalty for using the j th member in template D to encode j_i and set $\lambda = 0.01$. A large w_i will suppress nonzero entries α_i and force the solution α to concentrate on indices where w_i is small. So, the weight w_i for saliency detection should be inversely proportional to the similarity between the feature vector f_i and template members D . In other words, if the f_i is similar to some template in D , the penalty w_i should be small and vice versa.

Dictionary Construction Define saliency dictionary as a set of superpixels $S = \{r_{s_1}, r_{s_2}, \dots, r_{s_k}\}$. To get the initial saliency dictionary, use a non-saliency dictionary to reconstruct the reference image, and patches with high reconstruction error are selected saliency dictionary.

1.Non-saliency Dictionary: extract two sets of superpixel sets B_1, B_2 where B_1 is the set of superpixels on the reference image boundaries and B_2 is the set of superpixels locate in the out-of-focus regions. Combine B_1 and B_2 as the non-saliency set $B = \{B_1, B_2\}$. Then merge similar superpixels into larger regions $A_m = \{r_1^m, r_2^m, \dots, r_S^m\}$ and adopt a

boundary connectivity score, which measures the extent of region A_m connecting to the boundary.

$$w_{A_m}^{Con} = \frac{K}{Area(A_m)}$$

where $Area(A_m)$ represent the area of the current large area and K represent the length of the large area overlapping the boundary. Then superpixels from the merged region have the same connectivity score, namely $W_{r_j}^{Con} = w_{A_m}^{Con}, r_j \in A_m$. Superpixels whose connectivity scores are non-zero are selected to form non-saliency dictionary.

2.Saliency Dictionary: For each superpixel r_i , define the parameter weight $g(r_i, D_j)$ as:

$$g(r_i, D_j) = e^{\|F_{r_i}^D - D_j\|} + w_{r_i}^{Con}$$

Choose superpixels whose saliency values are higher than the mean to construct the initial saliency dictionary S^0 .

Weighted Sparse Coding Saliency. Given a set of superpixels $S = \{r_1, r_2, \dots, r_K\}$, use corresponding feature vectors $A = \{F_{r_1}^A, F_{r_2}^A, \dots, F_{r_K}^A\}$ and $H = \{F_{r_1}^H, F_{r_2}^H, \dots, F_{r_K}^H\}$ to construct two dictionaries. Use $w_{r_i}^A$ and $w_{r_i}^H$ to represent the weight for superpixel r_i . The template symbol D can be either A or H . $w_{r_i}^D$ is a vector that computes the similarity between superpixel R_i to all the members in template D :

$$g(r_i, D_j) = e^{\|F_{r_i}^D - D_j\|}$$

Next, use $(A, w_{r_i}^A)$ and $(H, w_{r_i}^H)$ as input to spare coding model to generate to sparsely coded dictionary $\alpha_{r_i}^A$ and $\alpha_{r_i}^H$ respectively. Then compute the reconstruction error $\epsilon_{r_i}^A$ and $\epsilon_{r_i}^H$ for each r_i :

$$\epsilon_{r_i}^D = \|F_{r_i}^D - D\alpha_{r_i}^D\|_2^2$$

Two saliency value $Sal^A(r_i)$ and $Sal^H(r_i)$ are also computed for r_i :

$$Sal^D(r_i) = Sal^*(\epsilon_{r_i}^D) \cdot Sal^L(r_i)$$

where $Sal^L(r_i)$ is the object-bias center prior defined in [23]. $Sal^*(\epsilon_{r_i}^D)$ is the saliency function related to the dictionary's type (saliency or non-saliency). Define the saliency function for non-saliency dictionary:

$$Sal^*(\epsilon_{r_i}^D) = \epsilon_{r_i}^D$$

For saliency dictionary:

$$Sal^*(\epsilon_{r_i}^D) = \epsilon^{\beta \cdot \epsilon_{r_i}^D}$$

where the author set $\beta = -5$.

Algorithm 1 Iterative Refinement

Input: $S^0, A^0, H^0, F^A, F^H, j = 0$
Output: Sal

```

1: function ITERATIVEOPT( $S^0, A^0, H^0, F^A, F^H$ )
2:   while not converge do
3:     for superpixel  $r_i = 1 \rightarrow N$  do
4:        $\alpha_{r_i}^{A^j}, \alpha_{r_i}^{H^j} \leftarrow \text{Eqn. 3}$ 
5:        $\epsilon_{r_i}^{A^j}, \epsilon_{r_i}^{H^j} \leftarrow \text{Eqn. 5}$ 
6:        $Sal^{A^j}(r_i), Sal^{H^j}(r_i) \leftarrow \text{Eqn. 8}$ 
7:        $Sal(r_i) \leftarrow \text{Eqn. 9}$ 
8:     end for
9:      $S^{j+1} \leftarrow \{r_i | Sal(r_i) > \text{mean}(Sal(r_i))\}$ 
10:     $A^{j+1} \leftarrow F^A(S^{j+1})$ 
11:     $H^{j+1} \leftarrow F^H(S^{j+1})$ 
12:     $j \leftarrow j + 1$ 
13:   end while
14:    $Sal < T \leftarrow 0$ 
15: end function

```

Fig. 25: Overall method.

Finally, combine $Sal^A(r_i)$ and $Sal^H(r_i)$ to get the saliency value for r_i :

$$Sal(r_i) = Sal^A(r_i) + Sal^H(r_i)$$

Start with using S^0 as input to the weighted sparse framework. At each iteration, refine the saliency dictionary using the estimated saliency map. At the k th iteration, firstly classify each superpixels using the saliency dictionary S^k and two template A^k and H^k . Then compute two saliency maps, finally, sum the two saliency maps with respect to A and H . A new saliency dictionary S^{k+1} is generated with by using superpixels whose saliency values are higher than the mean.

The overall method is shown in Fig.25.

16.4 Summarization

Since this paper also uses part of the [23], there are still many places that are not understood. But the general process is basically clear, these two articles build a bridge between the most basic saliency detection and the optimization model. The key to the problem is how to turn this kind of patch with saliency detection into a maximum/minimum problem in optimization. R-PCA is used in [22]. In this paper, in order to provide a generalized model, sparse coding was used. Of course, patch pre-segment algorithm such as SLIC algorithm, play a key role. Turning saliency detection into a projection problem is a good starting point for building a unified model. I think starting from this article, we can try to build a model that combines hash and this form of saliency detection into an optimization problem at the same time, and then go to solve it.

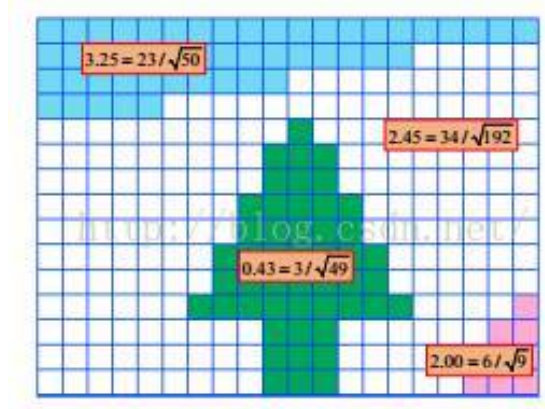


Fig. 26:

17 Continuation of paper[21](2020/05/02)

Continue to discuss some points that have not been clarified in the previous article. The first part is an introduction to the formulas used in the previous article to distinguish saliency and non-saliency dictionaries when constructing dictionaries. The second is some content of another important reference in the article.

17.1 Introduction of the formulas[24]

The formula mainly uses continuity to improve the a priori robustness of the background (non-significant regions). In other words, non-significant areas are unlikely to be associated with the image, and even if they are associated with the image, they are only a small part of the target area.

As shown in Fig.26, there are four areas in the entire image, and each image is associated with the image edge. If only judging whether it is related to the edge of the image is the criterion, it is obvious that it is difficult to find the salient regions in this image. In order to deal with this situation, a correlation method is proposed to optimize the background. The calculation formula is as follows:

$$BndCon(p) = \frac{Len_{bnd}(p)}{\sqrt{Area(p)}}$$

where Len is the length of the area associated with the edge of the image and $Area$ is the area of the entire area associated with the edge of the image. The significance of this formula is to transform the relevance into a measurable data. Therefore, in the last article, this formula can be used to select the region with the largest possible non-saliency to form a non-saliency dictionary.

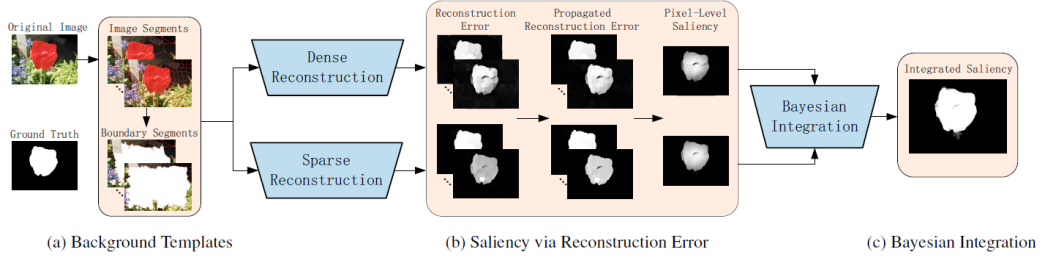


Fig. 27: Main steps of the proposed saliency detection algorithm.

17.2 Saliency Detection via Dense and Sparse Reconstruction[23]

This article is the motivation for the sparse coding part of the previous article. In this paper, the author propose a visual saliency detection algorithm from the perspective of reconstruction errors. The way that comparing a region only with its relevant contexts is called local contrast. Despite this way accords with the neuroscience principle that neurons in the retina are sensitive to regions which locally stand out from their surroundings, global contrast, which aims to capture the holistic rarity from an image, should also be taken into account when one region is similar to its surrounds but still distinct in the whole scene. In general, global saliency is computed inverse proportionally to the probability of a patch appearing in the entire scene. In other words, the global saliency cannot be represented well by other patches. In this paper, they reconstruct the entire image by dense and sparse appearance models from which errors are used as indication of saliency, based on the background templates. The main steps of the proposed saliency detection algorithm is shown in Fig.27.

Feature matrix: Generate superpixels using the simple linear iterative clustering (SLIC) algorithm to segment an input image into multiple uniform and compact regions. Use both Lab and RGB color spaces to describe the feature of each pixel and use the mean color features and coordinates of pixels to describe each segment by $x = \{L, a, b, R, G, B, x, y\}$. So the entire image is then represented as $X = [x_1, x_2, \dots, x_N] \in R^{D \times N}$, where N is the number of segments and D is the feature dimension.

Because salient objects are not always likely to appear at the center of a scene, image boundaries need to give more information. So they extract the D -dimensional feature of each boundary segment as b and construct the background template set as $B = [b_1, b_2, \dots, b_M]$, where M is the number of image boundary segments.

After that, they use dense and sparse reconstruction errors to measure the saliency of

each region which is represented by a D -dimensional feature.

17.2.1 Reconstruction Error

Consider image saliency detection as an estimation of reconstruction error on the background, with an assumption that there must be a large difference between the reconstruction errors of foreground and background regions using the same bases. For each region, they compute two reconstruction errors by dense and sparse representation, respectively.

Dense Reconstruction Error: A segment with larger reconstruction error based on the background templates is more likely to be the foreground. So the reconstruction error of each region is computed based on the dense appearance model generated from the background templates $B = [b_1, b_2, \dots, b_M]$, $B \in R^{D \times M}$ using Principal Component Analysis (PCA). The eigenvectors from the normalized covariance matrix of B , $U_B = [u_1, u_2, \dots, u_{D'}]$, corresponding to the largest D' eigenvalues, are computed to form the PCA bases of the background templates. With the PCA bases U_B , compute the reconstruction coefficient of segment i ($i \in [1, N]$).

$$\beta_i = U_B^T(x_i - \bar{x})$$

and the dense reconstruction error of segment i is

$$\varepsilon_i^d = \|x_i - (U_B \beta_i + \bar{x})\|_2^2$$

where \bar{x} is the mean feature of X .

Sparse Reconstruction Error: The principle is to express all patch linearly with non-salient patches. The larger the error is, the more salient the region is. Use the set of background templates B as the bases for sparse representation, and encode the image segment i by

$$\alpha_i = \arg \min_{\alpha_i} \|x_i - B\alpha_i\|_2^2 + \lambda \|\alpha_i\|_1$$

and the sparse reconstruction error is

$$\varepsilon_i^s = \|x_i - B\alpha_i\|_2^2$$

Dense representations model data points with a multivariate Gaussian distribution in the feature space, and thus it may be difficult to capture multiple scattered patterns especially when the number of examples is limited. So, a sparse reconstruction error is needed. Since all the background templates are regarded as the basis functions, sparse reconstruction error can better suppress the background compared with dense reconstruction error especially in cluttered images. They think sparse reconstruction error is more

robust to deal with complicated background, while dense reconstruction error is more accurate to handle the object segments at image boundaries. Therefore, dense and sparse reconstruction errors are complementary in measuring saliency.

It may be that I have a problem with the understanding of the two words dense and sparse. I do not think the meaning of these two words is reflected here. In the final analysis, I think that both methods here follow the use of non-significant areas to represent all areas. The greater the reconstruction error is, the higher the significance of this area is. It is just that in dense, the base after extracting the non-significant principal components is used here. Such processing is very effective for images with simple backgrounds, because it can effectively avoid the background noise and the influence of some outliers on the linear representation. However, this method is relatively difficult to deal with complex background. The complex background increases the proportion of each component in the main component, which causes loss of information and makes it difficult to represent the complex background well. Therefore, the author directly used the sparse coding method, which is called sparse here. The advantage of this method is to directly superimpose the image with the background, as mentioned in the previous article. This is also the motivation of the previous article, which is to use weighted sparse coding to integrate the two parts together. This process does not seem to have a direct relationship with dense and sparse.

Context-Based Error Propagation: A context-based error propagation method is used to smooth the reconstruction errors generated by dense and sparse appearance models. Both dense and sparse reconstruction errors of segment i are denoted by ε_i . Firstly, apply the K -means algorithm to cluster N image segments into K clusters via their D -dimensional features and initialize the propagated reconstruction error of segment i as $\tilde{\varepsilon}_i = \varepsilon_i$. The propagated reconstruction error of segment i belonging to cluster k ($k = 1, 2, \dots, K$), is modified by considering its appearance-base context consisting of the other segments in cluster k as follows:

$$\tilde{\varepsilon}_i = \tau \sum_{j=1}^{N_c} w_{ik_j} \tilde{\varepsilon}_{k_j} + (1 - \tau) \varepsilon_i$$

$$w_{ik_j} = \frac{\exp\left(-\frac{\|x_i - x_{k_j}\|^2}{2\sigma_x^2}\right)(1 - \delta(k_j - i))}{\sum_{j=1}^{N_c} \exp\left(-\frac{\|x_i - x_{k_j}\|^2}{2\sigma_x^2}\right)}$$

where $\{k_1, k_2, \dots, k_{N_c}\}$ denote the N_c segment labels in cluster k and τ is a weight parameter. In addition, σ_x^2 is the sum of the variance in each feature dimension of X and $\delta(\cdot)$ is the

indicator function. The first term is the weighted averaging reconstruction error of the other segments in the same cluster, and the second term is the initial dense or sparse reconstruction error.

Multi-Scale Reconstruction Error Integration and Refine: Generate super-pixels at N_s different scales. Integrate multi-scale reconstruction errors and compute the pixel-level reconstruction error by

$$E(z) = \frac{\sum_{s=1}^{N_s} w_{zn}(s) \tilde{\varepsilon}_n(s)}{\sum_{s=1}^{N_s} w_{zn}(s)}, w_{zn}(s) = \frac{1}{\|f_z - x_n(s)\|_2}$$

where f_z is a D -dimensional feature of pixel z and $n^{(s)}$ denotes the label of the segment containing pixel z at scale s .

Then use an object-biased Gaussian model G_0 with $\mu_x = x_0$ and $\mu_y = y_0$, where x_0 and y_0 denote the object center derived from the pixel error

$$\begin{cases} x_o = \sum_i \frac{E(i)}{\sum_j E(j)} x_i \\ y_0 = \sum_i \frac{E(i)}{\sum_j E(j)} y_i \end{cases}$$

where set $\sigma_x = 0.25 \times H$ and $\sigma_y = 0.25 \times W$, where W and H respectively denote the width and height of an image. With the object-biased Gaussian model, the saliency of pixel z is computed by $S(z) = G_0(z) * E(z)$.

Bayesian Integration of Saliency Maps: Given two saliency maps S_1 and S_2 , treat one of them as the prior $S_i (i = \{1, 2\})$ and use the other one $S_j (j \neq i, j = \{1, 2\})$ to compute the likelihood. First, threshold the map S_i by its mean saliency value and obtain its foreground and background regions described by F_i and B_i , respectively. In each region, compute the likelihoods by comparing S_j and S_i in terms of the foreground and background bins at pixel z :

$$p(S_j(z)|F_i) = \frac{N_{b_{F_i}(S_j(z))}}{N_{F_i}}, p(S_j(z)|B_i) = \frac{N_{b_{B_i}(S_j(z))}}{N_{B_i}}$$

The posterior probability is computed with S_i as the prior by

$$p(F_i|S_j(z)) = \frac{s_j(z)p(S_j(z)|F_i)}{S_i(z)p(S_j(z)|F_i) + (1 - S_i(z))p(S_j(z)|B_i)}$$

Similarly, the posterior saliency with S_j as the prior is computed. Use these two posterior probabilities to compute an integrated saliency map, $S_B(S_1(z), S_2(z))$, based on Bayesian integration:

$$S_B(S_1(z), S_2(z)) = p(F_1|S_2(z)) + p(F_2|S_1(z))$$

The Bayesian integration of saliency maps is illustrated in Fig.28.

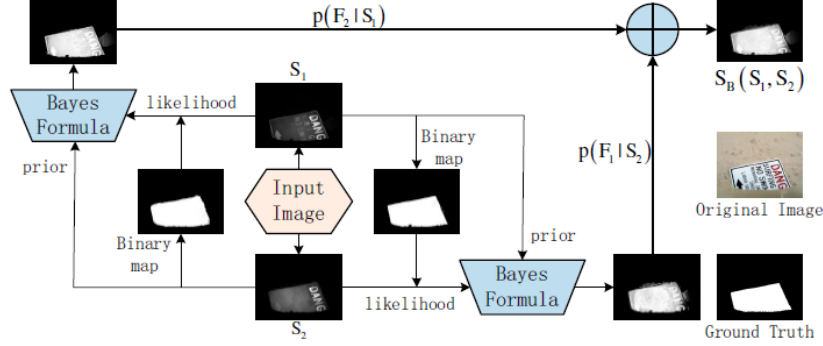


Fig. 28: Bayesian integration of saliency maps.

17.3 Summarization

From the three articles I have seen so far, the key to this method of transforming saliency into an existing optimization problem is still how to process the image to form local global features for linear representation. At present, the best way to deal with the blocking problem is the SLIC algorithm. This method can effectively divide the image into similar image blocks, and can have a small calculation cost. If you want to deal with the image's preliminary block problem, the SLIC algorithm is a very good algorithm. From the point of view of the optimization problem formed, it is difficult for the simple optimization problem to achieve the goal of saliency detection, so various methods are needed to assist. In Reference 22, the defect that the result of optimization problem is not ideal is dealt with transform T and prior information. In Document 21, the saliency and non-saliency dictionary are used as complementary conditions to complete. In Reference 23, the saliency score obtained by optimization is further improved, and the methods used include Bayesian priors and so on.

At present, a single optimization algorithm is not ideal for the detection of saliency. I think the main reason have two. One is that the features used by the optimization algorithm is the image itself, which lacks other more subjective priors. The other is that the problems solved by the optimization algorithm give too "accurate" results, just as we tried to use r-PCA for shot segmentation. Although the problem itself hopes to give a relatively accurate boundary, it is actually a relatively vague and subjective concept. In other words, the optimization method is less robust to solving such real-world problems, and is easily disturbed by practical problems with noise properties. From the current article, it is difficult to make hash and saliency a problem, but it is possible.

18 Salient Object Detection via Structured Matrix Decomposition[25](20

This paper is an improvement on the previous article, mainly for the previous algorithm is insufficient in dealing with sparse matrices. As mentioned before, these low-rank matrix recovery (LR) based saliency detection methods assume that an image can be represented as a combination of a highly redundant information part (e.g., visually consistent background regions) and a sparse salient part (e.g., distinctive foreground object regions). However, previous studies do not take into account the inter-correlation between elements in S , and thus ignore spatial relations, such as spatial contiguity and pattern consistency, between pixels and patches. In addition, when the background is cluttered or has similar appearance with the salient objects, it is difficult for previous LR -based methods to separate them. In response to these problems, the paper improved these method.

18.1 Modal Construction

Image Abstraction: Given an input image I , it is first partitioned into N non-overlapping patches $\mathcal{P} = \{P_1, P_2, \dots, P_N\}$ (e.g., superpixels) by SLIC method. For each patch P_i , a D -dimension feature vector is extracted and denoted as $f_i \in R^D$. The low-level feature, including RGB color, steerable pyramids and Gabor filter, is used to construct a 53-dimension feature representation. The ensemble of feature vectors forms a matrix representation of I , denoted as $F = [f_1, f_2, \dots, f_N] \in R^{53 \times N}$.

Low-rank regularization for image background: Image patches from the background are often similar and approximately lie in a low-dimensional subspace. Since directly minimizing a matrix's rank with affine constraints is an NP -hard problem, adopt the nuclear norm as a convex relaxation, i.e.,

$$\Psi(L) = \text{rank}(L) = \|L\|_* + \varepsilon$$

where ε denotes the relaxation error.

In this paper, in order to verify the rationality of the low-rank constraint, the author evaluate the rank of feature matrices extracted from image background on five salient object datasets. Fig.29 shows the statistics of such estimated ranks of background feature matrices of the images in the five datasets. It shows that about 90% of these matrices can be approximated by a matrix with rank no greater than 10.

Structured-sparsity regularization for salient objects: A tree-structured sparsity-inducing norm to model the spatial contiguity and feature similarity among image patches is used to produce more precise and structurally consistent results. For an index tree T with depth d over indices $\{1, 2, \dots, N\}$, let G_j^i be the j -th node at the i -th level. The nodes satisfy two conditions: (1) there is no overlap between the indices of nodes from the same

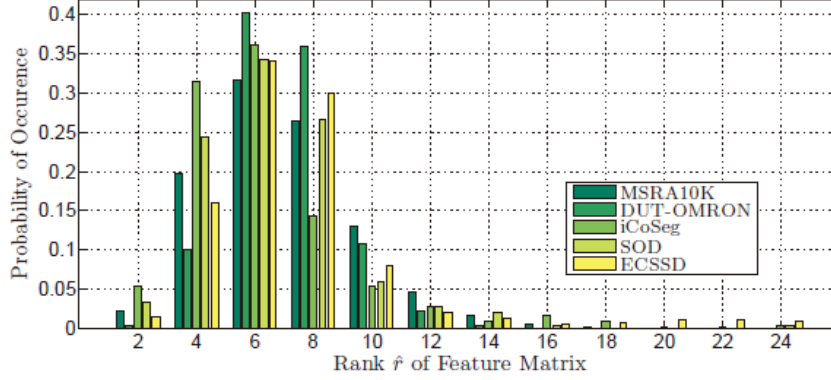


Fig. 29: Rank statistics of feature matrices extracted from image background over five datasets.

level, i.e., $G_j^i \cap G_k^i = \emptyset, \forall 2 \leq i \leq d$ and $1 \leq j < k \leq n_i$. Here, n_i denotes the total number of nodes at the i -th level. (2) Let $G_{j_0}^{i-1}$ be the parent node of a non-root node G_j^i , then $G_j^i \subseteq G_{j_0}^{i-1}$ and $\bigcup_j G_j^i = G_{j_0}^{i-1}$.

First, compute the affinity of every adjacent patch pair using:

$$w_{i,j} = \begin{cases} \exp\left(-\frac{\|f_i - f_j\|^2}{2\sigma^2}\right), & \text{if } (P_i, P_j) \in V \\ 0, & \text{otherwise} \end{cases}$$

where V denotes the set of adjacent patch pairs which are either neighbors (first-order) or "neighbors of neighbors" (second-order reachable) on the image. Then merge spatially neighboring patches according to their affinity. In each granularity layer, the segments correspond to the nodes at the corresponding layer in the index tree. Finally, obtain a hierarchical fine-to-coarse segmentation of the input image. In this way, get a general tree-structured sparsity regularization as

$$\Omega(S) = \sum_{i=1}^d \sum_{j=1}^{n_i} v_j^i \|S_{G_j^i}\|_p$$

where $v_j^i \geq 0$ is the weight for the node G_j^i , $S_{G_j^i} \in R^{D \times |G_j^i|}$ is the sub-matrix of S corresponding to the node G_j^i , and $\|\cdot\|_p$ is the l_p -norm.

The weight for the node is the high-level priors. For each patch $P_i \in \mathcal{P}$, its high-level prior, $\pi_i \in [0, 1]$, indicates the likelihood that P_i belongs to a salient object based on high-level information. Define v_j^i as

$$v_j^i = 1 - \max(\{\pi_k : k \in G_j^i\})$$

which boosts the saliency value of nodes with high prior values by associating them with small penalties v_j^i . After these, it induces the patches within the same group to share a similar representation, and represents the subordinate or coordinate relations between groups.

Laplacian regularization: When decomposing the feature matrix F into a low-rank part L plus a structured-sparse part S , the author hope to enlarge the distance between the subspaces induced by L and S . So, the Laplacian regularization is used based the assumption: if two adjacent image patches are similar with respect to their features, their representations should be close to each other in the subspace, and vise versa. Define the regularization as

$$\Theta(L, S) = \frac{1}{2} \sum_{i,j=1}^N \|s_i - s_j\|_2^2 w_{i,j} = \text{Tr}(SM_F S^T)$$

where s_i denotes the i -th column of S , $w_{i,j}$ is the (i,j) -th entry of an affinity marix $W = (w_{i,j}) \in R^{N \times N}$ and represents the feature similarity of patches (P_i, P_j) , $\text{Tr}(\cdot)$ denotes the trace of a matrix, and $M_F \in R^{N \times N}$ is a Laplacian matrix. It encourages patches within the same semantic region to share similar or identical representation, and patches from heterogeneous regions to have different representation.

So the overall modal can be reformulated:

$$\begin{aligned} \min_{L,S} \Psi(L) + \alpha \Omega(S) + \beta \Theta(L, S) \\ s.t. \quad F = L + S \end{aligned}$$

Saliency Assignment: After decomposing F , transfer the results from the feature domain to the spatial domain for saliency estimation. Based on the structured matrix S , define a straightforward saliency estimation function $Sal(\cdot)$ of each patch in \mathcal{P} :

$$Sal(P_i) = \|s_i\|_1$$

where s_i represents the i -th column of S . A large $Sal(P_i)$ means a high possibility that P_i belongs to a salient object.

After merging all patches together and performing context-based propagation[23], get the final saliency map of the input image.

18.2 Optimization

The ADM is used to solve the convex problem. The auxiliary variable H is used to make the objective function separable:

$$\min_{L,S} \|L\|_* + \alpha \sum_{i=1}^d \sum_{j=1}^{n_i} v_j^i \|S_{G_j^i}\|_p + \beta \text{Tr}(HM_F H^T)$$

$$s.t. \quad F = L + S, S + H$$

Then, minimize the augmented Lagrangian function \mathcal{L} of the problem:

$$\begin{aligned} \mathcal{L}(L, S, H, Y_1, Y_2, \mu) = & \|L\|_* + \alpha \sum_{i=1}^d \sum_{j=1}^{n_i} v_j^i \|S_{G_j^i}\|_p + \beta \text{Tr}(H M_F H^T) + \text{Tr}(Y_1^T (F - L - S)) \\ & + \text{Tr}(Y_2^T (S - H)) + \frac{\mu}{2} (\|F - L - S\|_F^2 + \|S - H\|_F^2) \end{aligned}$$

where Y_1 and Y_2 are the lagrange multipliers, and $\mu > 0$ controls the penalty for violating the linear constraints.

Updating L: When S and H are fixed, the update L^{k+1} at the $(k+1)$ -th iteration is obtained by solving the following problem:

$$\begin{aligned} L^{k+1} = & \arg \min_L \mathcal{L}(L, S^k, H^k, Y_1^k, Y_2^k, \mu^k) \\ = & \arg \min_L \|L\|_* + \text{Tr}((Y_1^k)^T (F - L - S^k)) + \frac{\mu^k}{2} \|F - L - S^k\|_F^2 \\ = & \arg \min_L \tau \|L\|_* + \frac{1}{2} \|L - X_L\|_F^2 \end{aligned}$$

where $\tau = 1/\mu^k$ and $X_L = F - S^k + Y_1^k/\mu^k$. The solution can be derived as

$$L^{k+1} = U T_\tau[\Sigma] V^T, \text{ where } (U, \Sigma, V^T) = \text{SVD}(X_L)$$

where Σ is the singular value matrix of X_L . The operator $T_\tau[\cdot]$ is the singular value thresholding (SVT) defined by element-wise τ thresholding of Σ .

Updating H: When L and S are fixed, to update H^{k+1} :

$$\begin{aligned} H^{k+1} = & \arg \min_H \mathcal{L}(L^{k+1}, S^k, H, Y_1^k, Y_2^k, \mu^k) \\ = & \arg \min_H \beta \text{Tr}(H M_F H^T) + \text{Tr}((Y_2^k)^T (S^k - H)) + \frac{\mu^k}{2} \|S^k - H\|_F^2 \end{aligned}$$

Take derivative of the objective function:

$$H^{k+1} = (\mu^k S^k + Y_2^k)(2\beta M_F + \mu^k I)^{-1}$$

Updating S: To update S^{k+1} with fixed L and H , get the following tree-structured sparsity optimization problem:

$$\begin{aligned} S^{k+1} = & \arg \min_S \mathcal{L}(L^{k+1}, S, H^{k+1}, Y_1^k, Y_2^k, \mu^k) \\ = & \arg \min_S \alpha \sum_{i=1}^d \sum_{j=1}^{n_i} v_j^i \|S_{G_j^i}\|_p + \text{Tr}((Y_1^k)^T (F - L^{k+1} - S)) \\ & + \text{Tr}((Y_2^k)^T (S - H^{k+1})) + \frac{\mu^k}{2} (\|F - L^{k+1} - S\|_F^2 + \|S - H^{k+1}\|_F^2) \\ = & \arg \min_S \lambda \sum_{i=1}^d \sum_{j=1}^{n_i} v_j^i \|S_{G_j^i}\|_p + \frac{1}{2} \|S - X_S\|_F^2 \end{aligned}$$

Input: The index tree T with nodes G_j^i ($i = 1, 2, \dots, d; j = 1, 2, \dots, n_i$), weight $v_j^i \geq 0$ (default as 1), the matrix \mathbf{X}_S , parameters α , and set $\lambda = \alpha/(2\mu^k)$.

- 1: Set $\mathbf{S} = \mathbf{X}_S$
- 2: **For** $i = d$ to 1 **do**
- 3: **For** $j = 1$ to n_i **do**
- 4: $\mathbf{S}_{G_j^i}^{k+1} = \begin{cases} \frac{\|\mathbf{S}_{G_j^i}\|_1 - \lambda v_j^i}{\|\mathbf{S}_{G_j^i}\|_1} \mathbf{S}_{G_j^i}, & \text{if } \|\mathbf{S}_{G_j^i}\|_1 > \lambda v_j^i \\ 0, & \text{otherwise} \end{cases}$
- 5: **End For**
- 6: **End For**
- 7: **Return** \mathbf{S}^{k+1}

Fig. 30: Solving the tree-structured sparsity.

where $\lambda = \alpha/(2\mu^k)$ and $X_S = (F - L^{k+1} + H^{k+1} + (Y_1^k - Y_2^k)/\mu^k)/2$. The problem can be solved by the hierarchical proximal operator, which computes a particular sequence of residuals obtained by projecting a matrix onto the unit ball of dual l_p -norm. The detailed procedure when using l_∞ -norm is presented in Fig.30.

The overall algorithm of this paper is shown in Fig.31.

18.3 Summarization & Some Attempt

Compared with the previous article, this article is an improvement to the previous article to deal with the imperfection of sparse matrix. It is mainly improved by adding the structure matrix. The function of the structure matrix here is to repair the over-blocked data. Due to the SLIC algorithm, the image is divided into many small blocks, and some of these blocks are actually derived from the same image area. If the most fine-grained block is directly adopted, then these blocks will be directly treated as a small penalty for noise, thus becoming different saliency. Doing so will cause the entire salient target to be segmented. Therefore, the author uses hierarchical blocking, similar to k-means clustering under different thresholds, to assign similar weights to small blocks in similar large blocks, so that the salient regions will not be split. Another improvement is the introduction of the Laplacian operator. By projecting superpixels onto the graph, and then using the distance between nodes to improve the saliency, the similarity of the saliency of similar blocks is further improved.

Linking image hashing to saliency detection, currently I have thought of the following two ways. For hashing, we use the simplest form of hashing, e.g., preserving local hashing.

$$(p, w) = \arg \min_{p, w} \sum A_{ij} \|p_i - p_j\|^2 + \rho \|sgn(p_i) - p_i\|_F^2$$

$$p_i = wx_i$$

Algorithm 1 ADM-SMD.

Input: Feature matrix \mathbf{F} , parameters α, β , index tree $T = \{G_j^i\}$ and tree node weight v_j^i (default as 1).
Output: \mathbf{L} and \mathbf{S} .

- 1: Initialize $\mathbf{L}^0=0, \mathbf{S}^0=0, \mathbf{H}^0=0, \mathbf{Y}_1^0=0, \mathbf{Y}_2^0=0, \mu^0=0.1,$
 $\mu_{\max} = 10^{10}, \rho=1.1$, and $k=0$.
- 2: **While** not converged **do**
- 3: $\mathbf{L}^{k+1} = \arg \min_{\mathbf{L}} \mathcal{L}(\mathbf{L}, \mathbf{S}^k, \mathbf{H}^k, \mathbf{Y}_1^k, \mathbf{Y}_2^k, \mu^k)$
- 4: $\mathbf{H}^{k+1} = \arg \min_{\mathbf{H}} \mathcal{L}(\mathbf{L}^{k+1}, \mathbf{S}^k, \mathbf{H}, \mathbf{Y}_1^k, \mathbf{Y}_2^k, \mu^k)$
- 5: $\mathbf{S}^{k+1} = \arg \min_{\mathbf{S}} \mathcal{L}(\mathbf{L}^{k+1}, \mathbf{S}, \mathbf{H}^{k+1}, \mathbf{Y}_1^k, \mathbf{Y}_2^k, \mu^k)$
- 6: $\mathbf{Y}_1^{k+1} = \mathbf{Y}_1^k + \mu^k(\mathbf{F} - \mathbf{L}^{k+1} - \mathbf{S}^{k+1})$
- 7: $\mathbf{Y}_2^{k+1} = \mathbf{Y}_2^k + \mu^k(\mathbf{S}^{k+1} - \mathbf{H}^{k+1})$
- 8: $\mu^{k+1} = \min(\rho\mu^k, \mu_{\max})$
- 9: $k = k + 1$
- 10: **End While**
- 11: **Return** \mathbf{L}^k and \mathbf{S}^k .

Fig. 31: The overall algorithm.

where A_{ij} is the affinity matrix between the image and sgn is the sign function.

For saliency detection, we use the form described in this paper. The reason for using this form is mainly because the problems in this article can be summarized as a problem without multi-step processing.

1. After treating the patch as a separate image for hash coding, stitch the hash codes together. Similar to the form below:

$$(p, w) = \arg \min_{p, w} \sum A_{ij} \|p_i - p_j\|^2 + \rho \|sgn(p_i) - p_i\|_F^2 + \Psi(L) + \alpha \Omega(S) + \beta \Theta(L, S)$$

$$s.t. \quad p_i = wx_i, F = L + S$$

I think that this form is not suitable after thinking about it, mainly because this joint method may lose the effect of attention. The processing method that I thought of is that after finding the hash code for each superpixel, the attention score S is used as the weight, and then all the hash is weighted. This is obviously unreasonable, too much loss of attention.

2. Add saliency as a penalty term to the hash, the hash of image with the same saliency is the same. At present, I have not thought of a suitable combination of this method. After some attempts, I found that this method can easily become a two-step method, and it is difficult to find the junction between the two. The main idea is that the problems dealt with by the two problems are not of the same dimension. The problems of hash construction are all problems between two images, and the saliency detection is a

problem between different superpixels in an image, and there is a difference between the two problems.

I think it may be that the LPH is not suitable to be connected with this method. Perhaps we can try unsupervised hashing, which is to directly use the mean or threshold to binarize the hash after extracting features from the image. Or change to a higher-dimensional saliency detection, such as a similar method for video.

19 Unsupervised Deep Hashing with Similarity-Adaptive and Discrete Optimization[26](2020/05/09)

This article proposes a Similarity-Adaptive Deep Hashing (SADH), which alternately proceeds over three training modules: deep hash model training, similarity graph updating and binary code optimization. It is mainly by projecting the feature code obtained from the initial encoding onto the graph, and then using the graph structure to measure the similarity. Analyze the title, we can know for difficult to deal with the problem of binary optimization, a step-by-step optimization method is adopted here. As for the adaptive similarity, it mainly means that the measurement of this similarity only depends on the similarity of the data itself projected on the graph, and has nothing to do with the label. This should be a characteristic of most unsupervised methods.

Unsupervised models for hash are trained by minimizing either the quantization loss or data reconstruction error. For practical applications, since most data in real-world may not have semantic labels, unsupervised models is more useful. The overall SADH is shown in Fig.32.

19.1 Model of SADH

Suppose given a set of training data $X = \{x_1, x_2, \dots, x_n\} \in R^{d \times n}$. The goal is to learn a set of compact binary codes $B = \{b_1, b_2, \dots, b_n\} \in \{-1, 1\}^{r \times n}$. So an effective hash function $b = h(x) = \text{sgn}(F(x; W))$ to encode novel images to the Hamming space is needed, where W is the model parameters.

For a graph hashing problem, the problem can be formulated as follow:

$$\min_B \text{Tr}(BLB^T) + \frac{\mu_1}{4} \|BB^T - nI_r\|^2 + \frac{\mu_2}{2} \|B1\|^2$$

$$s.t. \quad B \in \{-1, 1\}^{r \times n}$$

The graph Laplacian is defined as $L = \text{diag}(A1) - A$, where the affinity matrix entry A_{ij} represents the similarity between data pair (x_i, x_j) in the input feature space, and $A1$ returns the sum of columns of A with the all-one column vector 1 . The last two terms

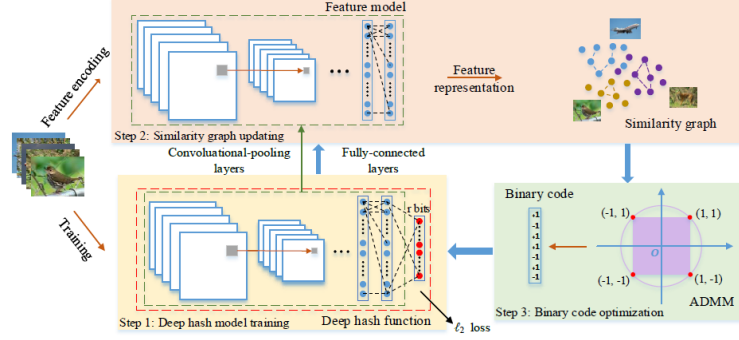


Fig. 32: An overview of the proposed Similarity-Adaptive Deep Hashing. The deep hash model (in red dash box) is trained to fit the learned binary codes. Part of the trained model is taken to generate feature representations for the images, which help update the similarity graph among data. With the updated similarity graph, the binary codes are optimized by ADMM over the intersection (red dots) of two continuous areas (in purple).

force the learned binary codes to be uncorrelated and balanced, respectively. Analyzing the first item, in fact, each item on the diagonal of the BLB^T is the sum of the distance between the current element and all elements. Taking the i th row and i th column as an example, this element represents the weighted distance between the b_i code and any hash code in B , where the weight is the similarity of the two, and the greater the similarity, the smaller the distance. In this paper, the Laplacian is calculated by anchor graph[27].

19.1.1.1 Anchor Graph Scheme[27]

An Anchor Graph uses a small set of m points called *anchors* to approximate the data neighborhood structure. Similarities of all n database points are measured with respect to these m anchors, and the true adjacency (or similarity) matrix A is approximated using these similarities. First, K -means clustering is preformed on n data points to obtain $m(m \ll n)$ cluster centers $U = \{\mu_j \in R^d\}_{j=1}^m$ that act as anchor points. Next, the Anchor Graph defines the truncated similarities Z_{ij} 's between all n data points and m anchors as,

$$Z_{ij} = \begin{cases} \frac{\exp(-D^2(x_i, \mu_j)/t)}{\sum_{j' \in \langle i \rangle} \exp(-D^2(x_i, \mu_{j'})/t)}, & \forall j \in \langle i \rangle \\ 0, & otherwise \end{cases}$$

where $\langle i \rangle \subset [1 : m]$ denotes the indices of $s(s \ll m)$ nearest anchors of point x_i in U according to a distance function $D()$ such as l_2 distance, and t denotes the bandwidth parameter. The matrix $Z \in R^{n \times m}$ is highly sparse and each row Z_i contains only s nonzero entries which sum to 1.

Then suppose $\hat{A} = Z\Lambda^{-1}Z^T$, \hat{A} is a powerful approximation to the adjacency matrix A , where $\Lambda = \text{diag}(Z^T 1) \in R^{m \times m}$. So the resulting graph Laplacian is $L = I - \hat{A}$.

19.2 Solve the model

Due to the difficult optimization and lack of effective deep hashing training strategy for graph hashing problem, the author discrete the training of SADH by three parts: deep hash function training, similarity graph updating and binary code optimization.

Step1: Deep hash model training: Train the deep hash model with a Euclidean loss layer. The problem is written as:

$$\min_W ||\tan H(F(X; \{W_l\})) - B||^2$$

Here $W = \{W_l\}$ denotes the parameters of the deep network architecture and W_l represents the weight parameters in each layer. A pre-trained model on the large-scale ImageNet dataset as the initialization of deep hash function is used. The parameter $\{W_l\}$ of the proposed model are optimized through the standard back-propagation and stochastic gradient descent.

Step2: Data similarity graph updating: Once the deep network is trained, a hash function as well as a feature representation model can be obtained. Encode images with this deep model, a more powerful deep representation can be obtained. Then update the pairwise similarity graph by

$$A_{ij} = \exp ||\bar{F}(x_i, \bar{W}) - \bar{F}(x_j, \bar{W})||^2 / (2\sigma^2)$$

Here $\bar{F}(x, \bar{W})$ is the feature representation model, which is formed from $F(x, W)$ by excluding the last fully connected layer; σ is bandwidth parameter. The updated similarity graph is then taken into the next binary code optimization step.

Step3: Binary code optimization: A novel binary code optimization algorithm is used to solve the graph hash problem. The binary code learning problem can be equivalently transformed to an optimization problem over the intersection of two continuous domains. The constraint of binary code matrix $B \in \{-1, 1\}^{r \times n}$ is equivalent to $B \in [-1, 1]^{r \times n}$ and $||B||_p^p = nr$. The result can be easily verified as follows. 1) Given $B \in \{-1, 1\}^{r \times n}$, it is easily to get $B \in [-1, 1]$ and $||B||_p^p = nr$. 2) Given $B \in [-1, 1]^{r \times n}$, there is $||B||_p^p = \sum_{i,j} |B_{ij}|^p \leq nr$ and the equation holds if and only if B_{ij} reaches its extreme value, i.e., $B_{ij} = 1$ or -1 . So if $B \in [-1, 1]^{r \times n}$ and $||B||_p^p = nr$, $B \in \{-1, 1\}^{r \times n}$. Explain that the two penalties are mutually contained, that is equal.

Let denote by $L(B)$ the objective in graph hash problem. The problem can be reformulated as follows:

$$\min_B L(B)$$

$$s.t. \quad B \in S_b, B \in S_p$$

where denote S_b and S_p as the set $[-1, 1]^{r \times n}$ and $\{B : \|B\|_p^p = nr\}$, respectively.

Binary code optimization with ADMM: Introduce two auxiliary variables Z_1 and Z_2 to absorb the constraints with S_b and S_p , respectively. That is, rewrite the constraints as $Z_1 = B$, $Z_1 \in S_b$ and $Z_2 = B$, $Z_2 \in S_p$. Then, the problem can be reformulated as follows:

$$\min_{B, Z_1, Z_2, Y_1, Y_2} L(B) + \delta_{S_b}(Z_1) + \delta_{S_p}(Z_2) + \frac{\rho_1}{2} \|B - Z_1\|^2 + \frac{\rho_2}{2} \|B - Z_2\|^2 + \text{Tr}(Y_1^T (B - Z_1)) + \text{Tr}(Y_2^T (B - Z_2))$$

Here $\delta_S(Z)$ is the indicator function, which outputs 0 if $Z \in S$ and $+\infty$ otherwise. Y_1 , Y_2 and ρ_1 , ρ_2 are the dual and penalty variables, respectively.

Follow the ADMM process, update the primal variables B , Z_1 , Z_2 iteratively by minimizing the augmented Lagrangian. Then perform gradient ascent on the dual problem to update Y_1 , Y_2 .

Update B: At the $(k+1)$ th iteration, with all variables but B fixed, B^{k+1} is updated with the following problem,

$$\min_B L(B) + \frac{\rho_1 + \rho_2}{2} \|B\|^2 + \text{Tr}(BG^T)$$

where $G = Y_1^k + Y_2^k - \rho_1 Z_1^k - \rho_2 Z_2^k$. This sub-problem can be solved efficiently by the LBFGS-B method, with the gradient of object calculated as

$$2BL + \mu_1(BB^T - nI_r)B + \mu_2 B 11^T + (\rho_1 + \rho_2)B + G$$

Update Z_1 and Z_2 : With B^{k+1} , Y_1^k fixed, Z_1^{k+1} is updated with the following problem,

$$\min_{Z_1} \delta_{S_b}(Z_1) + \frac{\rho_1}{2} \|Z_1 - B^{k+1}\|^2 - \text{Tr}(Y_1^k Z_1)$$

which is solved by the proximal minimization method with the following solution:

$$Z_1^{k+1} = \prod_{S_b} (B^{k+1} + \frac{1}{\rho_1} Y_1^k)$$

Here \prod is the projection operator, which projects those entries of $(B^{k+1} + 1/\rho_1 Y_1^k)$ out of $[-1, 1]$ into the boundaries of the interval, i.e., -1 or 1. Similarly, Z_2^{k+1} is updated by $Z_2^{k+1} = \prod_{S_p} (B^{k+1} + 1/\rho_2 Y_2^k)$. When set $p = 2$, it is easy to solve the above problem with a closed-form solution:

$$Z_2^{k+1} = \sqrt{nr} \times \frac{B^{k+1} + \frac{1}{\rho_2} Y_2^k}{\|B^{k+1} + \frac{1}{\rho_2} Y_2^k\|}$$

Update Y_1 and Y_2 : The two dual variables are updated by the gradient ascent:

$$Y_1^{k+1} = Y_1^k + \gamma \rho_1 (B^{k+1} - Z_1^{k+1})$$

Algorithm 1: Similarity-Adaptive Deep Hashing

Input: Training data $\{\mathbf{x}_i\}_{i=1}^n$; code length r ; parameters $\rho_1, \rho_2, \mu_1, \mu_2, \gamma$.

Output: Binary codes \mathbf{B} ; deep hash function $h(\mathbf{x})$.

Initialize the deep model parameters $\{\mathbf{W}_l\}$ by the pre-trained VGG-16 model on ImageNet;

```

repeat
    // Data similarity updating
    Construct graph similarity matrix  $\mathbf{A}$  by (3);
    // Binary code optimization
    Initialize  $\mathbf{B}^0, \mathbf{Y}_1^0, \mathbf{Y}_2^0$  randomly and let  $\mathbf{Z}_1^0 \leftarrow \mathbf{B}^0$ ,
     $\mathbf{Z}_2^0 \leftarrow \mathbf{B}^0$  and  $k \leftarrow 0$ ;
    repeat
        Update  $\mathbf{B}^{k+1}$  by solving (6) with LBFGS-B;
        Update  $\mathbf{Z}_1^{k+1}$  by projecting  $\mathbf{B}^{k+1} + \frac{1}{\rho_1} \mathbf{Y}_1^k$  on  $\mathcal{S}_b$ 
        with (9);
        Update  $\mathbf{Z}_2^{k+1}$  by projecting  $\mathbf{B}^{k+1} + \frac{1}{\rho_2} \mathbf{Y}_2^k$  on  $\mathcal{S}_p$ 
        with (10);
        Update  $\mathbf{Y}_1^{k+1}$  by (11);
        Update  $\mathbf{Y}_2^{k+1}$  by (12);
         $k = k + 1$ ;
    until converged;
    // Deep hash function learning
    Learn the deep model parameters  $\{\mathbf{W}_l\}$  with
    optimized  $\mathbf{B}$ .
until maximum iterations;
return  $\mathbf{B}, h(\mathbf{x}) = \text{sgn}(\mathcal{F}(\mathbf{x}; \{\mathbf{W}_l\}))$ .

```

Fig. 33: Similarity-Adaptive Deep Hashing.

$$\mathbf{Y}_2^{k+1} = \mathbf{Y}_2^k + \gamma \rho_2 (\mathbf{B}^{k+1} - \mathbf{Z}_2^{k+1})$$

Here γ is the parameter used to accelerate convergence. The optimization alternately proceeds over each variable until convergence.

The overall algorithm is shown in Fig.33.

19.3 Summarization

In fact, I do not think this is a complete unsupervised network, because the essence of the article can be regarded as a kind of precision enhancement of the hash code solved in the first step. This kind of reinforcement is carried out by projecting the hash code solved in the first step onto the graph and then using the anchor graph structure. The network used in the first step is a pre-trained network. This network is undoubtedly obtained through a supervised training. Corresponding to this is the linear structure proposed by the author in the article, that is, $F = Wx$. Based on this thinking, the actual meaning of the three steps in this article can be re-understood. The first step is to extract the features of the image through a deep network. The second step is to project the features onto the anchor graph. The third step is to use the anchor graph to optimize the hash. Although in Fig.32, we can see that the third step has a feedback process to the first step, but it is not introduced in the article. If understood according to Fig.32, it should be an end-to-end network structure, but the framework described in the paper is not. The final hash result

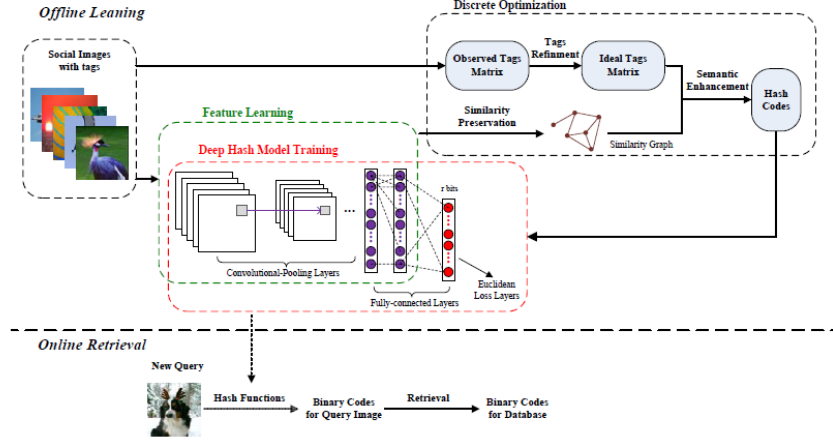


Fig. 34: The basic framework of the proposed SCADH-based large-scale social image retrieval system.

did not feed back to the deep network in the first step. Therefore, I think that in fact, the paper does not reflect the value contained in the deep hash in the title.

Of course, the article gives us a whole new understanding of the smoothness of *sgn*. This smoothing method can link the binary optimization with ADMM, which is another important reference besides the smoothing using adaptive hyperbolic function mentioned earlier. There are some contents about the paper[28] that have not been completely read, and the specific contents will be presented in the next summary.

20 Scalable Deep Hashing for Large-scale Social Image Retrieval[29](2020)

This paper is the one that is highly similar in structure to the proposed paper[26]. The main method is to use this mode of "self-supervision" to perform hash coding. The difference is how to use the extra information here. This paper targeted at the social image which is recorded by various individuals in social networks spontaneously and generally assigned with several informative tags to describe the image contents when uploading the image. In this paper, the author jointly learns image representations and hash functions with deep neural networks, and simultaneously enhances the discriminative capability of image hash codes with the refined semantics from the accompanied social tags. In addition, for binary quantization optimal, they discrete it based on Augmented Lagrangian Multiplier to directly solve the hash codes. The framework is shown in Fig.34.

20.1 SCAlable Deep Hashing (SCADH)

About the composition of the hash code, this paper is actually a simple "supervised" deep hash method. It's just that this supervised label is different from the usual deep hash. The supervised label here is a "self-supervised" label that depends on the social image label.

In this paper, the deep hash model is trained with a Euclidean loss layer, which minimizes the discrepancy between the outputs of the deep model and the learned binary codes. They adopt VGG-16 model as their deep hash model and initialize it with the neural network weights which are pre-trained on the large-scale ImageNet dataset. The problem is written as

$$\min_{\mathcal{W}} \|\tanh(\mathcal{F}(X; \mathcal{W}) - B)\|^2$$

\mathcal{W} is optimized through stochastic gradient descent method and the back propagation.

20.1.1 How to get B in deep hash model

Consider a social image dataset consisting of n images $\{x_i\}_{i=1}^n$ with c user-provided unique social tags. Each image is represented by $x_i \in R^d$, and the relationships between the image and tags can be represented as a c dimensional binary-valued vector f_i . The images matrix is denoted as $X = [x_1, \dots, x_n] \in R^{d \times n}$, and $F = [f_1, \dots, f_n] \in R^{c \times n}$ represents the observed image-tag relation matrix, where $f_{ji} = 1$ if x_i is associated with tag j and 0 otherwise. The ideal image-tag relation matrix is denoted as $Y = [y_1, \dots, y_n] \in R^{c \times n}$. The goal is to learn a set of hash codes $B = [b_1, \dots, b_n] \in [-1, 1]^{r \times n}$, which can be the label for the deep hash model to get the image hash.

The overall objective function can be formulated as follow:

$$\min_{Y, B, W} \|Y\|_* + \lambda_1 \|F - Y\|_1 + \frac{\lambda_2}{2} \|B - W^T Y\|_F^2 + \frac{\lambda_3}{2} \|W\|_F^2 + \lambda_4 \text{Tr}(BLB^T)$$

$$s.t. B \in \{-1, 1\}^{r \times n}$$

where $\lambda_i|_{i=1}^4$ are the parameters to balance the effect of different regularization terms. The overall objective function can be divided into three parts, **Social Tag Refinement**, **Semantic Enhancement**, **Image Similarity Preservation**, which goal is to achieve an accurate image-tag hash.

1.Social Tag Refinemet: The tags from social images are subject to two properties: low-rank and error sparsity. As one kind of text information, image tags are consequently subject to such low-rank property. Moreover, users generally provide reasonably accurate tags when uploading the image, and the number of annotated tags is essentially sparse compared with the total number of tags. As a consequence, the error of the image-tag relation matrix is sparse.

For a given image-tag relation matrix F , the purpose of social tag refinement is to uncover the ideal tagging matrix Y and the tag error matrix E . According to the RPCA, the problem can be formulated as a matrix decomposition problem. The low-rank matrix Y and the sparse error matrix E can be derived by solving the following optimization problem

$$\begin{aligned} \min_{Y, E} & \|Y\|_* + \lambda_1 \|E\|_1 \\ \text{s.t.} & F = Y + E \end{aligned}$$

where λ_1 is the positive weighting parameter.

2.Semantic Enhancement: Social images and the accompanied tags belong to heterogeneous modalities but they are highly correlated with each other. W is defined as $W = [w_1, \dots, w_r] \in R^{c \times r}$, where $w_k \in R^{c \times 1}$ is the semantic correlation vector for the k -bit hash codes. The aim is to minimize the difference between the binary hash codes and the mapped semantic vectors from the refined social tags. Specifically, to learn W , optimize the following problem:

$$\begin{aligned} \min_{B, W} & \|B - W^T Y\|_F^2 + \lambda_3 \|W\|_F^2 \\ \text{s.t.} & B \in \{-1, 1\}^{r \times n} \end{aligned}$$

where λ_3 is the parameter that plays the trade-off between two regularization terms.

3.Image Similarity Preservation: Social image hash is to seek for compact binary codes whose Hamming distances match with semantic similarity. A graph model is used to address the problem. The correlation of social images can be simply represented with affinity matrix S . Formally, the element at i -th row, j -th column of S (semantic similarity between image i and image j) is calculated as

$$s_{ij} = \begin{cases} \exp(-(\|\hat{\mathcal{F}}(x_i; \hat{W}) - \hat{\mathcal{F}}(x_j; \hat{W})\|_F^2)/\delta^2) & \text{if } x_i \in \mathcal{N}_k(x_j) \text{ or } x_j \in \mathcal{N}_k(x_i) \\ 0, & \text{otherwise} \end{cases}$$

where $\mathcal{N}(x)$ represents the set of k nearest neighbors of x , and δ is determined automatically by local scaling.

TO preserve image similarity, they minimize the sum of weighted Hamming distances:

$$\min_B \sum_{i=1}^n \sum_{j=1}^n S_{ij} \|b_i - b_j\|_F^2 \Leftrightarrow \min_B \text{Tr}(BLB^T)$$

where $L = \text{diag}(S1_n) - S$ is the graph Laplacian matrix.

In order to avoid computing the $n \times n$ matrices S and L , in this paper, they adopt the anchor graph scheme, that is

$$L = I - Z\Lambda^{-1}Z^T$$

Here $Z \in R^{n \times m}$ is the anchor graph matrix computed by the similarities between all images and m anchor points and $\Lambda = \text{diag}(Z^T 1)$.

I feel that the role played by the third item is actually different from that of paper[26]. The main hash code here is not directly obtained based on the image, but an inaccurate hash code with tag properties obtained by processing image-tag. So the third term is actually a punishing effect, punishing those intra-class distances within the same class. For the hash of Y with high image-tag similarity but large image content gap, a larger intra-class distance is given. It can improve the accuracy of Y 's hash code.

20.1.2 Solve the problem

They keep the binary constraints $B \in \{-1, 1\}^{r \times n}$ and solve the hash code directly. Introduce auxiliary variables to separate constraints, and transform the objective function to an equivalent one that can be tracked more easily. Specifically, they introduce three auxiliary variables Y_1 , Y_2 , C , and keep the equivalence between them and the substituted ones during the optimization. The overall problem can be reformulated as:

$$\begin{aligned} \min_{Y, B, W} & \|Y_1\|_* + \lambda_1 \|F - Y_2\|_1 + \frac{\lambda_2}{2} \|B - W^T Y\|_F^2 + \frac{\lambda_3}{2} \|W\|_F^2 + \lambda_4 \text{Tr}(BLC^T) \\ & + \frac{\mu}{2} (\|Y - Y_1 + \frac{1}{\mu} \Lambda_1\|_F^2 + \|Y - Y_2 + \frac{1}{\mu} \Lambda_2\|_F^2 + \|B - C + \frac{1}{\mu} \Lambda_3\|_F^2) \\ \text{s.t. } & B \in \{-1, 1\}^{r \times n} \end{aligned}$$

where Λ_1 , Λ_2 , Λ_3 measure the difference between the target and auxiliary variables, μ determines the penalty for infeasibility.

Update Y : If the other variables are fixed, the optimization formula for Y is

$$\min_Y \frac{\lambda_2}{2} \|B - W^T Y\|_F^2 + \frac{\mu}{2} (\|Y - Y_1 + \frac{1}{\mu} \Lambda_1\|_F^2 + \|Y - Y_2 + \frac{1}{\mu} \Lambda_2\|_F^2)$$

Calculating the derivative of the problem with respect to Y and setting it to 0, obtain that:

$$Y = (\lambda_2 W W^T + 2\mu I)^{-1} (\lambda_2 W B + \mu Y_1 + \mu Y_2 - \Lambda_1 - \Lambda_2)$$

Update Y_1, Y_2 : By fixing other variables, the optimization formulas for Y_1 , Y_2 are:

$$\begin{aligned} \min_{Y_1} & \|Y_1\|_* + \frac{\mu}{2} \|Y - Y_1 + \frac{1}{\mu} \Lambda_1\|_F^2 \\ \min_{Y_2} & \lambda_1 \|F - Y_2\|_1 + \frac{\mu}{2} \|Y - Y_2 + \frac{1}{\mu} \Lambda_2\|_F^2 \end{aligned}$$

Then, obtain that:

$$Y_1 = \arg \min_{Y_1} \frac{1}{\mu} \|Y_1\|_* + \frac{1}{2} \|Y_2 - (Y + \frac{1}{\mu} \Lambda_1)\|_F^2$$

which can be solved by SVD.

$$Y_2 = \arg \min_{Y_2} \frac{\lambda_1}{\mu} \|Y_2 - F\|_1 + \frac{1}{2} \|(Y_2 - F) + (F - Y - \frac{1}{\mu} \Lambda_2)\|_F^2$$

which can be solved by forward-backward.

Update W: By fixing other variables, the optimization formula for W is

$$\min_W \frac{\lambda_2}{2} \|B - W^T Y\|_F^2 + \frac{\lambda_3}{2} \|W\|_F^2$$

Calculating the derivative of the problem with respect to W and setting it to 0, obtain that:

$$W = (Y Y^T + \frac{\lambda_3}{\lambda_2} I)^{-1} Y B^T$$

Update C: Similarly, the optimization formula for C becomes

$$\min_C \lambda_4 \text{Tr}(B L C^T) + \frac{\mu}{2} \|B - C + \frac{1}{\mu} \Lambda_3\|_F^2$$

Calculating the derivative of the problem with respect to C and setting it to 0, obtain that:

$$C = B + \frac{1}{\mu} \Lambda_3 - \frac{\lambda_4}{\mu} B L$$

Update B: By fixing other variables, the optimization formula for B is

$$\begin{aligned} \min_B \frac{\lambda_2}{2} \|B - W^T Y\|_F^2 + \lambda_4 \text{Tr}(B L C^T) + \frac{\mu}{2} (\|B - C + \frac{1}{\mu} \Lambda_3\|_F^2) \\ s.t. B \in \{-1, 1\}^{r \times n} \end{aligned}$$

It can be reformulated as

$$\begin{aligned} \min_B -\text{Tr}(B^T (W^T Y - \frac{\lambda_4}{\lambda_2} C L^T - \frac{1}{\lambda_2} \Lambda_3 + \frac{\mu}{\lambda_2} C)) \\ s.t. B \in \{-1, 1\}^{r \times n} \end{aligned}$$

The discrete solution of B can be directly represented as

$$B = \text{sgn}(W^T Y - \frac{\lambda_4}{\lambda_2} C L^T - \frac{1}{\lambda_2} \Lambda_3 + \frac{\mu}{\lambda_2} C)$$

Update Λ_1 , Λ_2 , Λ_3 and μ): The parameter of ALM method are updated by

$$\Lambda_1 = \Lambda_1 + \mu(Y - Y_1)$$

$$\Lambda_2 = \Lambda_2 + \mu(Y - Y_2)$$

$$\Lambda_3 = \Lambda_3 + \mu(B - C)$$

| | |
|--------------------|---------------------------|
| Algorithm 1 | Key steps of our approach |
|--------------------|---------------------------|

| | |
|----------------|--|
| Input: | Training images matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$, image-tag relation matrix $\mathbf{F} \in \mathbb{R}^{c \times n}$, the number of anchor points m , hash code length r , parameter $\lambda_1, \lambda_2, \lambda_3, \lambda_4$. |
| Output: | deep hash functions $h(\mathbf{x})$. |
| 1: | Initialize the deep model parameters \mathcal{W} by the pre-trained VGG-16 model on ImageNet. |
| 2: | <i>// Hash Code Learning</i> |
| 3: | Randomly initialize \mathbf{B} and \mathbf{C} as $\{-1, 1\}^{r \times n}$ matrices. |
| 4: | Randomly initialize $\mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2$ as $\{-1, 1\}^{c \times n}$ matrices. |
| 5: | Initialize Λ_1, Λ_2 and Λ_3 as the matrices with all elements as 0. |
| 6: | Construct anchor graph matrix \mathbf{Z} by Eq.(4). |
| 7: | repeat |
| 8: | Update \mathbf{W} by Eq.(18). |
| 9: | Update \mathbf{Y} by Eq.(10). |
| 10: | Update $\mathbf{Y}_1, \mathbf{Y}_2$ by Eq.(15) and Eq.(16). |
| 11: | Update \mathbf{C} by Eq.(20). |
| 12: | Update \mathbf{B} by Eq.(23). |
| 13: | Update Λ_1, Λ_2 and Λ_3 and μ by Eq.(24). |
| 14: | until Convergence. |
| 15: | <i>// Deep Hash Function Learning</i> |
| 16: | Learn deep model parameters \mathcal{W} with \mathbf{B} by Eq.(1). |
| 17: | return $h(\mathbf{x}) = \text{sgn}(\mathcal{F}(\mathbf{x}; \mathcal{W}))$. |

Fig. 35: Key steps of SCADH.

$$\mu = \rho\mu$$

Here ρ is a parameter to control the convergence speed.

So far, the hash code obtained from the image-tag has been solved, and it will be used as the deep hash label to train the network. Finally, an end-to-end deep hash neural network will be obtained as the hash function which can transform image directly to hash code. The overall steps is shown in Fig.35.

20.2 Summarization

The structure of this paper is basically the same as that of paper[26], and there is nothing particularly needed to point out. The difference lies in the solution to the *sgn* problem. Speaking of which, we also make a brief summary on this issue. For the *sgn* problem, there are currently three ways to deal with optimization.

1)**Using the projection method** The method is used in this article, for *sgn* problems that are not easy to solve, this article uses auxiliary variables to simplify the parts that are not easy to solve together, and then directly uses method, that similar to forward-backward method, to project it onto the ball where *sgn* is located. Since the summary has

introduced the processing procedure, it will not be repeated here.

2) **Using the ℓ_p box method**[28] The method is used in paper[26]. For binary optimization as follows:

$$\min_{x \in \{0,1\}^n} f(x), \quad s.t. \quad x \in \mathcal{C}$$

There is a proposition.

Proposition ℓ_p -Box Intersection: The binary set $\{0, 1\}^n$ can be equivalently replaced by the intersection between a box S_b and a $(n - 1)$ -dimensional sphere S_p , as follows:

$$x \in \{0, 1\}^n \Leftrightarrow x \in [0, 1]^n \cap \{x : \|x - \frac{1}{2}1_n\|_p^p = \frac{n}{2^p}\}$$

where $p \in (0, \infty)$, and $S_b = [0, 1]^n = \{x : \|x\|_\infty \leq 1\}$, $S_p = \{y : \|y - \frac{1}{2}1_n\|_p^p = \frac{n}{2^p}\}$. Note that S_p can be seen as a $(n - 1)$ -dimensional ℓ_p -sphere centered as $\frac{1}{2}1_n$ with radius $\frac{n^{\frac{1}{p}}}{2}$.

So the the binary problem can be reformulated as follow:

$$\begin{aligned} \min_{x, y_1, y_2} \quad & f(x) \\ s.t. \quad & x \in \mathcal{C}, x = y_1, x = y_2 \\ & y_1 \in S_b, y_2 \in S_p \end{aligned}$$

where $S_b = \{y : 0 \leq y \leq 1\}$, $S_p = \{y : \|y - \frac{1}{2}1\|_p^p = \frac{n}{2^p}\}$.

Then ADMM can be used to solve the problem.

3) **Using the tanh-adaptive**[17][30][31] The method is proposed firstly in paper [30], where the author notice there exists a key relationship between the sign function and the scaled tanh function in the concept of limit in mathematics

$$\lim_{\beta \rightarrow \infty} \tanh(\beta z) = \text{sgn}(z)$$

where $\beta > 0$ is a scaling parameter. Increasing β , the scaled tanh function $\tanh(\beta z)$ will become more-non-smooth and more saturated so that the alternative binary problem will be more difficult to optimize, as shown in Fig.36.

Using the continuation methods, the alternative problem with $\tanh(\beta z)$ can be solved by optimize, where $\beta_0 = 1$. For each stage t , after algorithm converges, the β_t is increased and optimized again. By evolving $\tanh(\beta_t z)$ with $\beta_t \rightarrow \infty$, the problem will converge to the original problem with $\text{sgn}(z)$, which can generate exactly binary hash codes. In generally, the efficacy of continuation in this method can be understood as multi-stage pre-training, i.e., pre-training problem with $\tanh(\beta_t z)$ is used to initialize problem with $\tanh(\beta_{t+1} z)$, which enables easier progressive calculating the problem as the problem is becoming non-smooth in later stage by $\beta_t \rightarrow \infty$.

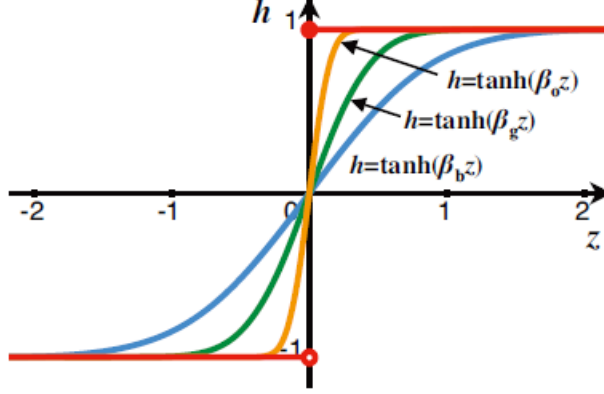


Fig. 36: Red is the sign function, and blue, green and orange show functions $h = \tanh(\beta z)$ with bandwidths $\beta_b < \beta_g < \beta_o$.

In order to ensure that the continuous tanh function is differential everywhere that can be optimized via standard back-propagation, a regularization term should be considered[31]. Such function is named as Adaptive Tanh (ATanh):

$$f(z) = \tanh g(z; \beta) + \zeta(\beta)$$

where $\zeta(\beta)$ is the regularization term providing a convenient way to control the magnitude of β , which prompts the function to be closer to the sign function. In practice, the function $g(z; \beta)$ can be implemented with respect to the scaling method. There are two different methods that can be effectively employed for scaling.

a. Linearized ATanh: The simplest and most direct scaling method is linear scaling. According to the zoom coefficient, the transition area of tanh around zero can be stretched or shrunk. Hence, the scaling function is written as $g(z) = \beta z$, so:

$$f(z) = \tanh(\beta z) + \lambda \left\| \frac{1}{\beta} \right\|_2^2$$

where λ is a regularization constant. By minimizing the regularization term of $\left\| \frac{1}{\beta} \right\|_2^2$, β can be gradually increased so that the final activation of $f(z) = \tanh(\beta z)$ approaches the sign function and has the ability to generate binary codes. In addition, the linearized ATanh is an element-wise function; thus, different β can be learned for different bits. In other words, 32 functions can be simultaneously learned for 32 *bits* hashing codes, which makes the codes more adaptable compared with the constant sign function. The linearized ATanh can be jointly trained with other layers via back-propagation.

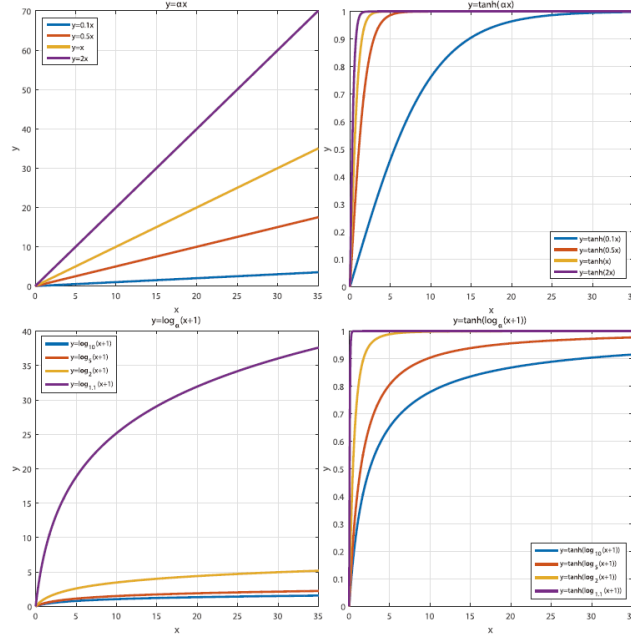


Fig. 37: An illustration of the linear and nonlinear composite function of tanh (only the first quadrant is shown). The top-left figure is an illustration of the linear scaling function, and the top-right figure is the corresponding modified tanh function. When the scaling parameter α decreases, $\tanh(\alpha x)$ becomes closer to the sign function, as is the case for the nonlinear ones in the bottom two figures.

b. Non-Linearized ATanh: Although the linearized ATanh can scale the inputs, it still suffers from the problem of tiny gradients. This is because the non-binary area (i.e., the transition area) of the tanh function mainly lies in the interval $(-3, 3)$ on the x-axis. Regardless of the size of the scaling parameter β , the non-binary area can only be linearly extended to a limited size, and the external area remains unchanged. The nonlinear scaling function should project all the original inputs into the non-binary area of the tanh function; then, almost all activations can utilize the gradient passed from the higher layers. Hence they propose to employ the logarithmic function as the scaling function, where the scaling parameter β is applied as the base:

$$g(z) = \log_{\beta}(z + 1), \beta > 1$$

As shown in Fig.37, the bottom two figures show different tanh functions scaled by different logarithms. When β increases, the inputs far away from the zero point can still use the gradients compared with the linear scaling-method. Similarly, the composite ATanh

function should also approach the sign function after training. In contrast to the linear ATanh function, the nonlinear ATanh function requires that β gradually decrease. Then, the alternative problem can be formulated as

$$f(z) = \tanh(\log_\beta(z+1)) + \lambda \|\beta\|_2^2$$

It also can be solved by gradient method.

So far, the three known methods of handling *sgn* have been summarized and completed.

21 Discrete Hashing with Multiple Supervision[32](2020/05/16)

The framework and idea of this paper is very similar to the paper[29], so that I think the paper can be seen as the version of deep hash about this paper. However, the optimization of binary problem in this paper is very outstanding, which transform the original problem to solvable problems. Similarity, in this paper, hash codes and hash function is get by two-step, where the hash codes is trained first.

The motivation of this paper is that the ordinary similarity matrix (usually $n \times n$) is large storage cost and proved by experiment that directly using label vectors as binary codes can generate good retrieval accuracy. So, in this paper, the author utilize the instance-pairwise similarity matrix and class-wise similarity matrix, which is less than the ordinary similarity matrix, as the prior knowledge to train the hash codes.

21.1 The Model of Hash Codes

For a training set consisting of n labeled instances, i.e., $\{(x_1, I_1), \dots, (x_n, I_n)\}$, where $x_i \in R^d$ is the d -dimensional feature vector of the i -th instance, $I_i \in \{-1, 1\}^c$ is the ground truth label vector and c is the number of classes. In a matrix form, $X \in R^{n \times d}$ is the feature matrix and $L \in \{-1, 1\}^{n \times c}$ is the label matrix where $L_{ik} = 1$ if the i -th instance belongs to class k and -1 otherwise. The aim is to learn a r -bit binary hash code for each instance, and $B \in \{-1, 1\}^{n \times r}$ is the hash code matrix with each row $B_{i*} = b_i$ representing the binary code of instance i .

For a supervised hashing methods, which work towards preserving the semantic similarity with the learnt binary codes, it can be given as follow:

$$\begin{aligned} \min_B & \|rS - BB^T\|_F^2 \\ s.t. & B \in \{-1, 1\}^{n \times r} \end{aligned}$$

where $S \in \{-1, 0, 1\}^{n \times n}$ is the instance-pairwise similarity matrix and $\|\cdot\|_F$ denotes the Frobenius norm. And $S_{ij} = 1(S_{ij} = -1)$ means that instances i and j are semantically

similar (dissimilar), and $S_{ij} = 0$ is that the similarity between i and j is unknown. **Although it is not the same as the previously known LPH, the meaning is the same.** SS^T indicates the similarity between two codes (in Hamming space). A larger value indicates that the two codes are more similar, but the value must be less than the code length r . The actual similarity rS minus the similarity of the Hamming space is the actual gap. Minimizing this item can ensure that the distance between the two codes in the actual space and the Hamming space are as consistent as possible.

The method of this paper is called Discrete Hashing with Multiple Supervision (MS-DH), where both class-wise and instance-class similarities are introduced to supervise the learning of binary codes. First, class-level hash codes are generated under the guide of a novel loss function with class-wise label information embedded; then, to consider the specific instances, the class-level codes are used to assist in learning the final hash codes.

1.Class-wise Similarity Embedding: To overcome the large space cost of the $n \times n$ instance-pairwise similarity matrix, they first use the class-wise similarity to supervise the learning of class-level hash codes, denoted as $Y \in \{-1, 1\}^{c \times r}$ with each row $Y_{i*} = y_i \in \{-1, 1\}^r$ as the code of the i -th class. For class-wise similarity $A \in \{-1, 1\}^{c \times c}$, $A_{ij} = -1$ if class $i \neq j$, and $A_{ij} = 1$ if class $i = j$. As c is far less than n in most real applications, the space cost can be dramatically reduced. To embed the class-wise similarity, the objective function of learning class-level hash codes is given as follow:

$$\begin{aligned} \tilde{\mathcal{O}}_{class}(Y, A) &= \min_Y \|rA - Y^T Y\|_F^2 \\ s.t. \quad Y &\in \{-1, 1\}^{c \times r} \end{aligned}$$

2.Instance-class Similarity Embedding: In generally, the different instances in the same class should have different hash codes but similar to the class-level hash codes. So, the instance-level hash codes B is needed. The objective function is given as follow:

$$\begin{aligned} \tilde{\mathcal{O}}_{instance}(B, Y, L) &= \min_B \|rL - BY^T\|_F^2 \\ s.t. \quad Y &\in \{-1, 1\}^{c \times r}, B \in \{-1, 1\}^{n \times r} \end{aligned}$$

where $L \in \{-1, 1\}^{n \times c}$ is the instance-class similarity matrix, i.e., the label vector of all instances, and Y is the class level hash code. By this problem, B and Y can preserve the instance-class similarity as much as possible. It means that if the i -th instance is in the j -th class, b_i should be similar to y_j .

3.Overall Objective Function: Combining \mathcal{O}_{class} and $\mathcal{O}_{instance}$ together, the overall objective function can be formulated as follows

$$\min_{Y, B} \tilde{\mathcal{O}}_{instance}(B, Y, L) + \alpha \tilde{\mathcal{O}}_{class}(Y, A)$$

$$s.t. \quad Y \in \{-1, 1\}^{c \times r}, B \in \{-1, 1\}^{n \times r}$$

where α is a parameter to balance the importance of $\tilde{O}_{instance}$ and \tilde{O}_{class} .

21.2 Optimize for Model of Hash Codes

In order to avoid large quantization errors, the author propose an alternating strategy without relaxation. **(I think this optimization method is the highlight of the whole paper.)**

1.Y step (Fix B, optimize Y). Because the B is fixed, the problem can be reformulated as follow:

$$\begin{aligned} \min_Y \quad & \|rL - BY^T\|_F^2 + \alpha \|rA - YY^T\|_F^2 \\ s.t. \quad & Y \in \{-1, 1\}^{c \times r}, B \in \{-1, 1\}^{n \times r} \end{aligned} \quad (21.1)$$

The author let $Q_{i,j}^{(k)} = -2\alpha(rA_{i,j} - \sum_{m=1}^{l-1} y_i^m y_j^m)$, when $i \neq j$; $Q_{i,j}^{(k)} = 0$, when $i = j$; $p_i^{(k)} = -2 \sum_{l=1}^n B_{l,k}(rL_{l,i} - \sum_{m=1}^{k-1} B_{l,m} Y_{i,m})$; and y^k denotes the k -th column of Y . Thereafter, the optimization problem for the k -th bit of Y can be written as follows,

$$\min_{y^k \in \{-1, 1\}^c} [y^k]^T Q^{(k)} y^k + [y^k]^T p^{(k)} \quad (21.2)$$

Regarding why question 21.1 can be changed to 21.2, I think this is basically impossible to achieve only through the simplification between mathematical formulas (I have tried many times and did not succeed in simplifying). But if we look at the two problems in the practical sense described in the problem, it seems feasible. First of all, for question 21.1, there are two essences. One is to ensure that the similarity between the class similarity and the hash code is as similar as possible. The other is to ensure that the hash code of the instance is as consistent as possible with the hash code of the class to which it belongs. Therefore, a two-part minimization problem arises. We first look at the definition of $Q^{(k)}$ and $P^{(k)}$. In fact, $Q^{(k)}$ is an expanded expression of the second part of question 21.1. Each $Q_{i,j}^{(k)}$ represents the gap between the i -th class and the j -th class. For $i = j$, the gap is 0, but for $i \neq j$ the gap is to be carried out. Then maximize it, so that it can ensure that there is a certain distance between two dissimilar classes in the hash space. So $[y^k]^T Q^{(k)} y^k$ actually represents the sum of the distance between two classes of $i \neq j$. There is a negative value of $Q^{(k)}$ in front of this, plus the sum of maximization, so the whole problem becomes minimization $[y^k]^T Q^{(k)} y^k$. Regarding the definition of $P^{(k)}$, $P^{(k)}$ is actually a relationship between the i -th class and the j -th class constructed by the class to which the image belongs. For the case where the

l -th instance does not belong to the k -th category, its $B_{l,k}$ should ideally be 0, so the effect of the following items can be ignored. For the case where the l -th instance belongs to the k -th category, the following term expresses the total similarity between the code of the image l and the code of the class i . If the correlation between the image l and the i -th category is greater, the total similarity is greater. And $L_{l,i}$ represents the actual situation, so the whole term behind $B_{l,k}$ expresses the actual gap of its similarity. After multiplying with $B_{l,k}$, it indicates the difference of the similarity between all instances belonging to the l -th class and the i -th class. In other words, this is a gap between classes. This gap should be maximized, plus the minus sign before $P^{(k)}$, so the entire term becomes a minimized term. $[y^k]^T p^{(k)}$ expresses a weighted sum of the gap between the code of the k -th class and all the classes, and the weight represents the similarity between the k -th class and other classes.

Problem 21.2 is a Binary Quadratic Programming (BQP) problem. For a problem as problem 21.3, it can be solved.

$$\begin{aligned} & \max_x x^T P x \\ & s.t. \quad \sum_{i=1}^n x_i = k, x \in \{0, 1\}^n \end{aligned} \quad (21.3)$$

where the diagonal elements of P are zeros.

First, the domain of problem 21.2 is transformed from $\{-1, 1\}^c$ to $\{0, 1\}^c$, by replacing y^k with $\frac{1}{2}(y^k + 1)$. Second, a constraint $\sum_{i=1}^n x_i = k$ is added to problem 21.2. In addition, constraint $y^k 1 = [c/2]$ is also used. Then remove the linear term in problem 21.2. The problem can be equivalently transformed as

$$\begin{aligned} & \min [\tilde{y}^k]^T \tilde{Q}^{(k)} \tilde{y}^k \\ & s.t. \quad \tilde{y}^k \in \{0, 1\}^{c+1}, \sum_{i=1}^{c+1} \tilde{y}_i^k = \frac{c+1}{2} \\ & \tilde{y}^k = \begin{pmatrix} \frac{1}{2}(y^k + 1) \\ 1 \end{pmatrix}, \tilde{Q}^{(k)} = \begin{pmatrix} 4Q^{(k)} & \frac{1}{2}\bar{p}^{(k)} \\ \frac{1}{2}[\bar{p}^{(k)}]^T & 0 \end{pmatrix} \end{aligned}$$

where $\bar{p}_i^{(k)} = 2[p_i^{(k)} - \sum_{l=1}^c (Q_{i,l}^{(k)} + Q_{l,i}^{(k)})]$

This problem can be solved.

2.B Step(Fix Y, optimize B): The problem can be written as follow:

$$\begin{aligned} & \min_B \|rL - BY^T\|_F^2 \\ & s.t. Y \in \{-1, 1\}^{c \times r}, B \in \{-1, 1\}^{n \times r} \end{aligned}$$

By discrete cyclic coordinate descent (DCC), the problem can be solved. Rewrite the problem:

$$\begin{aligned} \min_B & \|rL\|_F^2 - 2rTr(L^T BY^T) + \|BY^T\|_F^2 \\ \text{s.t. } & Y \in \{-1, 1\}^{c \times r}, B \in \{-1, 1\}^{n \times r} \end{aligned}$$

Discarding the constant term and replacing LY with G for simplicity, obtain the follow problem,

$$\begin{aligned} \min_B & \|BY^T\|_F^2 - 2rTr(B^T G) \\ \text{s.t. } & Y \in \{-1, 1\}^{c \times r}, B \in \{-1, 1\}^{n \times r} \end{aligned}$$

Use b denote the k -th column of B ($k = 1, \dots, r$) and B' to denote the matrix of B excluding b . Use y denote the k -th column of Y ($k = 1, \dots, r$) and Y' to denote the matrix of Y excluding y . Use g denote the k -th column of G ($k = 1, \dots, r$) and G' to denote the matrix of G excluding g . Then, obtain:

$$\begin{aligned} \|BY^T\|_F^2 &= Tr(BY^T Y B^T) \\ &= \|by^T\|_F^2 + 2y^T Y' B'^T b + \text{const} \\ &= 2y^T Y' B'^T b + \text{const} \end{aligned}$$

Here $\|by^T\|_F^2 = Tr(yb^T by^T) = ny^T y = \text{const}$. Correspondingly, have $Tr(B^T G) = g^T b$. Then, have the following optimization problem,

$$\begin{aligned} \min_b & (y^T Y' B'^T - g^T) b \\ \text{s.t. } & b \in \{-1, 1\}^n \end{aligned}$$

The optimal solution to this problem is as follows:

$$b = \text{sgn}(g - B' Y'^T y)$$

.

21.3 The model of Hash Function

If the hash codes have already been obtained, the learning problem of hash functions can be viewed as a binary classification problem for any bit of the codes. Here, the author use linear regression to demonstrate the effectiveness of loss function. Randomly sample m ($m < n$) anchor points, i.e., o_1, \dots, o_m and represent the kernel features as $\psi(x) = [\exp - \frac{\|x - o_1\|_2^2}{2\sigma^2}, \dots, \exp - \frac{\|x - o_m\|_2^2}{2\sigma^2}]$ where $\sigma = \frac{1}{mn} \sum_{i=1}^n \sum_{t=1}^m \|x_i - o_t\|_2$.

Thereafter, learn the linear regression model by minimizing the following squared loss

$$\min_W \|B - \psi(X)W\|_F^2 + \lambda_e \|W\|_F^2$$

where $\psi(X) \in R^{n \times m}$, λ_e is a balance parameter, and $\|W\|_F^2$ is a regularization term. Then get the optimal $W \in R^{m \times r}$ as,

$$W = (\psi(X)^T \psi(X) + \lambda_e I)^{-1} \psi(X)^T X$$

For a query $q \in R^d$ out of the training set, compute its hash code by the following

$$b = \text{sgn}(\psi(q)^T W).$$

21.4 Summarization

There are no special highlights in the structure of the problem in this paper. The basic idea is to use the hash code of the tag as a label to train to obtain the hash code of the instance. We have seen the construction of this two-step method many times. The difference between the deep network and the deep network is that the hash code and hash function are solved separately. The highlight of this article, I think it is its optimization method, first of all, the structure of question 21.2, which not only retains the meaning of 21.1, but also turns the problem into a problem that can be found in previous articles. This way of dealing with sgn is similar to the way in which the three auxiliary variables used ADMM for clever simplification. Of course, there are still many details that need to be discussed in the specific process of this article. For example, 21.2 to 21.3, I tried many times, and I did not transform it to the same form.

22 Fast Scalable Supervised Hashing[33](2020/05/20)

In the past few days, I have read the relevant articles of the NIE team and the overview of SHEN. The overview[34] has not been completely read. Here is a brief introduction to the NIE introduction paper[33]. I think the paper here does not seem particularly refreshing, but It was published in SIGIR 2018.

In this paper, the author proposed a method, called Fast Scalable Supervised Hashing (FSSH), whose goal is to avoid the large pairwise similarity matrix. The means of Scalable Supervised Hashing is a kind of processing method, which is designed to reduce the space cost of $n \times n$ pairwise similarity matrix. For the ideas used in the whole paper, I think it is actually a simplified version of the previous articles, or from a time point of view, this article can be regarded as the source of ideas for the previous articles.

22.1 The model of FSSH

Given n instances $X = [x_1, x_2, \dots, x_n] \in R^{n \times d}$, the method wants to get the r -bit binary hash codes $B = [b_1, b_2, \dots, b_n] \in \{-1, 1\}^{n \times r}$ while preserving the similarities in the

original space, where b_i is the hash code of x_i and d is the dimension of each instance. The overall objective function of this paper can be formulated as follow:

$$\begin{aligned} \min_{B, G, W} & \|S - (\phi(X)W)(LG)^T\|_F^2 + \mu\|B - LG\|_F^2 + \theta\|B - (\phi(X)W)\|_F^2 \\ \text{s.t.} \quad & B \in \{-1, 1\}^{n \times r} \end{aligned}$$

In the problem, L is the labels for all training instances $L = [I_1, I_2, \dots, I_n] \in \{0, 1\}^{n \times c}$, where c is the number of classes and I_{ik} is the k -th element of I_i ($I_{ik} = 1$ if x_i is in class k and 0 otherwise). In addition $S \in \{-1, 1\}^{n \times n}$ denotes the instance pairwise semantic similarity, where $S_{ij} = 1$ means instance i and instance j are semantically similar, and $S_{ij} = -1$ otherwise. About $\phi(\cdot)$, it can be seen as a transform to reduce the computational complexity. The author randomly sample m points as anchors, i.e., o_1, o_2, \dots, o_m . Then data point $x \in R^d$ can be mapped into a m -dimensional kernel feature representation, using $\phi(x) = [\exp(\|x - o_1\|^2/\sigma), \dots, \exp(\|x - o_m\|^2/\sigma)]$. Here, σ is the kernel width estimated according to the average Euclidean distance between the training samples.

As for the overall objective function, it can be divided into two parts, where one is kernel feature mapping, the other is the label matrix Embedding.

1. Kernel Features Mapping: As the thought used in paper[32], a general hash codes learning problem can be formulated as follow:

$$\begin{aligned} \min_B & \|rS - BB^T\|_F^2 \\ \text{s.t.} \quad & B \in \{-1, 1\}^{n \times r} \end{aligned}$$

For the problem, some studies demonstrate solving discretely can achieve better accuracy compared to those with relaxation. The hash functions can be formulated as $F(X) = \text{sgn}(\phi(X)W) = B$. Here $W \in R^{m \times r}$ is a projection matrix transforming the data. The quality of kernel features mapping is defined as follows:

$$\begin{aligned} \min_W & \|B - \phi(X)W\|_F^2 \\ \text{s.t.} \quad & B \in \{-1, 1\}^{n \times r} \end{aligned}$$

To permit more accurate approximation of similarity, the author leverage the real-valued kernel features mapping $\phi(X)W$ to better embed the similarity by defining the following problem:

$$\begin{aligned} \min_{B, W} & \|rS - (\phi(X)W)B^T\|_F^2 \\ \text{s.t.} \quad & B \in \{-1, 1\}^{n \times r} \end{aligned}$$

So the first part of the overall objective function can be given as follow:

$$\min_{B, W} \|rS - (\phi(X)W)B^T\|_F^2 + \mu \|B - \phi(X)W\|_F^2$$

$$s.t. \quad B \in \{-1, 1\}^{n \times r}$$

where μ is a balance parameter.

2.Label Matrix Embedding: The similarity matrix cannot fully reflect the distance between two instances, so the label information of the instances is introduced here. It can be given as follow:

$$\min_{B, G} \|B - LG\|_F^2$$

$$s.t. \quad B \in \{-1, 1\}^{n \times r}$$

where $G \in R^{c \times r}$ is a projection from L to B .

Different from the paper, where the label information is used as a label to guide the training of hash codes, the paper fuse the label information into the hash codes directly. It is obvious that both pairwise similarity matrix S and label matrix L are embedded into the learning of binary codes, which makes the learnt B more likely to preserve the semantic similarity. It is worth noting that, although S is included in the final objective function, it does not mean that S will be directly leveraged.

22.2 Optimization Algorithm

An iterative optimization algorithm is used to optimize the problem.

W Step: When G and B are fixed, the problem can be simplified as,

$$\min_W \|S - (\phi(X)W)(LG)^T\|_F^2 + \theta \|B - \phi(X)W\|_F^2$$

Set the derivation of W to zero

$$W = (\phi(X)^T \phi(X))^{-1} (\phi(X)^T S L G + \theta \phi(X)^T B) (G^T L^T L G + \theta I_{r \times r})^{-1}$$

$$= C^{-1} (A G + \theta \phi(X)^T B) (G^T D G + \theta I_{r \times r})^{-1}$$

where $A = \phi(X)^T S L$, $C = \phi(X)^T \phi(X)$ and $D = L^T L$. Both C^{-1} and D can be computed only once before the optimization. In addition, $A \in R^{m \times c}$ is a constant and can be efficiently pre-computed before training so that the large pairwise matrix S can not be used again and again.

G Step: When W and B are fixed, the problem can be rewritten as:

$$\min_G \|S - (\phi(X)W)(LG)^T\|_F^2 + \mu \|B_L G\|_F^2$$

Similarly, set the derivative of G to zero, then reach the closed-form solution:

$$\begin{aligned} G &= (L^T L)^{-1}(\mu L^T B + L^T S^T \phi(X)W)(W^T \phi(X)^T \phi(X)W + \mu I_{r \times r})^{-1} \\ &= D^{-1}(\mu L^T B + A^T W)(W^T C W + \mu I_{r \times r})^{-1} \end{aligned}$$

where $A = \phi(X)^T S L$, $C = \phi(X)^T \phi(X)$ and $D = L^T L$. As mentioned previously, C and D^{-1} can be computed only once before the training. Therefore, the computation of this solution is also efficient.

B Step: When W and G are fixed, the optimization problem is formulated as follows:

$$\begin{aligned} \min_B \quad & \mu \|B - LG\|_F^2 + \theta \|B - \phi(X)W\|_F^2 \\ \text{s.t.} \quad & B \in \{-1, 1\}^{n \times r} \end{aligned}$$

Then, rewrite it as follows:

$$\begin{aligned} \min_B \quad & \mu \text{Tr}((B - LG)^T (B - LG)) + \theta \text{Tr}((B - \phi(X)W)^T (B - \phi(X)W)) \\ & = (\mu + \theta) \|B\|_F^2 - 2\mu \text{Tr}(B^T LG) + \mu \|LG\|_F^2 - 2\theta \text{Tr}(B^T \phi(X)W) + \theta \|\phi(X)W\|_F^2 \\ \text{s.t.} \quad & B \in \{-1, 1\}^{n \times r} \end{aligned}$$

where $\text{Tr}(\cdot)$ is the trace norm. Since $\|B\|_F^2$, $\|LG\|_F^2$ and $\|\phi(X)W\|_F^2$ are constants. Then it can be reformulated as follow:

$$\begin{aligned} \min_B \quad & -\text{Tr}(B^T (\mu LG + \theta \phi(X)W)) \\ \text{s.t.} \quad & B \in \{-1, 1\}^{n \times r} \end{aligned}$$

Thus, B can also be solved with a closed-form solution stated as follows:

$$B = \text{sgn}(\mu LG + \theta \phi(X)W)$$

By repeating the above three steps until it is convergent, the final solution can be get as shown in Fig.38.

The use of subsequent hash codes and the generation of hash functions can be simply divided into one-step and two-step methods. The one-step method is to directly use the optimized W for the instances that need to be queried. The two-step method is to use the hash codes obtained to train a machine learning classification model, and use the functions obtained by the classification model as the hash function to process the instances that need to be queried.

Algorithm 1 Optimization algorithm of FSSH.

Input: The label matrix $\mathbf{Y} \in \{0, 1\}^{n \times c}$, the intermediate term $\mathbf{A} = \phi(\mathbf{X})^\top \mathbf{S} \mathbf{L}$ computed offline, the balance parameters μ and θ , iteration number t and the hash code length r .

- 1: Randomly initialize \mathbf{W} , \mathbf{G} and \mathbf{B} .
- 2: **for** $iter = 1 \rightarrow t$ **do**
- 3: **W step:** Fix \mathbf{G} and \mathbf{B} , update \mathbf{W} using Eq. (8).
- 4: **G step:** Fix \mathbf{W} and \mathbf{B} , update \mathbf{G} using Eq. (10).
- 5: **B step:** Fix \mathbf{W} and \mathbf{G} , update \mathbf{B} using Eq. (13).
- 6: **end for**

Output: \mathbf{W} and \mathbf{B} .

Fig. 38: Optimization algorithm of FSSH.

22.3 Summarization

So far, these papers had basically passed the hash related content of the NIE’s team. The hash problem handled by the NIE’s team has several characteristics that distinguish it from other hash work. The first is that the commonality of the work of the NIE’s team is to use the similarity matrix \mathbf{S} and the instances label matrix \mathbf{L} to improve the accuracy of the hash code. There are currently two ways to use the \mathbf{S} matrix, one is to use the Laplacian matrix of the anchor graph to measure, and the other is the method used in this article, whose significance is to reduce the storage of \mathbf{S} the complexity. Regarding the use of the \mathbf{L} matrix, two methods of information fusion are currently called ”invisible fusion” and ”dominant fusion”. The so-called ”invisible fusion” is to generate the label used by the \mathbf{L} matrix information for hash code training, and integrate the \mathbf{L} information into the hash code through this information connotation. In contrast, ”explicit fusion” directly uses \mathbf{L} as part of the objective function, the purpose is to ensure that the obtained hash code and image-tag are as similar as possible.

Secondly, the NIE’s team has a lot of room for discussion on the generation method of hash code and hash function. This space comes from how to think about the word ”supervision”. Judging from these paper, their team has not implemented a completely unsupervised hash. The so-called unsupervised hash is actually more similar to a self-supervised hash. This has a certain relationship with the nature of the hash function, where the way of supervision can ensure the compactness of the hash. If you only rely on the data characteristics of the instance itself, the length of the hash code will be much longer than the length of the supervised hash, which is contrary to the motivation to reduce the storage space with the help of hash (though unsupervised hash also greatly reduces the storage space).

Finally, the NIE’s team is currently optimizing non-deep hashes. Although some arti-

cles use relaxation of sgn , in many articles it is believed that solving the binary constraint problem of hash directly is better than relaxation. Therefore, the value of many articles of their team lies in how to use a direct method to deal with binary constraints. Judging from the focus of their article, perhaps the method of optimizing binary constraints is greater than the significance of proposing a new model.

23 A Fast Optimization Method for General Binary Code Learning[35](2020/05/23)

This paper proposed a novel binary code optimization method, called discrete proximal linearized minimization (DPLM), which directly handles the discrete constraints during the learning process. In generally, due to the difficulty of binary constraints, continuous relaxation usually is used to handle the constraints, which can achieve high learning efficiency. It can be seen as a two-step way: first solve a relaxed problem by discarding the discrete constraints, and then quantize the obtained continuous solution to achieve the approximate binary solution. However, accumulated quantization error will lead to that the pursued codes are less effective. In this paper, by rewriting the discrete constraints as an unconstrained minimization problem, the author utilize the proximal linearized minimization (PLM) scheme to solve the rewritten problem.

23.1 Fast Binary Optimization For Hashing

Suppose there are n samples $x_i \in R^d$, $i = 1, \dots, n$, stored in matrix $X \in R^{d \times n}$. For each sample x , the aim is to learn its r -bit binary code $b \in \{-1, 1\}^r$. Then it can be written as the following general binary code learning problem

$$\begin{aligned} \min_B \mathcal{L}(B) \\ s.t. \quad B \in \{-1, 1\}^{r \times n} \end{aligned} \quad (23.1)$$

Here B is the target binary codes for X and $\mathcal{L}(\cdot)$ is the smooth loss function.

To simplify the discrete optimization in problem (23.1), they utilize the following indicator function:

$$\delta_C(B) = \begin{cases} 0 & \text{if } B \in C \\ +\infty & \text{otherwise} \end{cases}$$

where C is a nonempty and closed set. Let \mathbb{B} denotes the binary codes space $\{-1, 1\}^{r \times n}$. The function $\delta_{\mathbb{B}}(B)$ yields infinity as long as one entry of B does not belong to the binary domain $\{-1, 1\}$. With the indicator function, the problem (23.1) can be reformulated an unconstrained minimization problem as follow

$$\min_B \mathcal{L}(B) + \delta_{\mathbb{B}}(B) \quad (23.2)$$

Then the problem (23.2) consists of two parts: a smooth function and a non-smooth one. The smooth function $\mathcal{L}(B)$ models the hashing loss which can be chosen freely according the different problem, while the non-smooth function $\delta_{\mathbb{B}}(B)$ indicates the domain of the optimizing hash codes.

The problem (23.2) can be solved by backward-forward method, which project the solution of smooth problem to the space of non-smooth problem. Denote $Prox_{\lambda}^f$ the proximal operator with function f and parameter λ :

$$Prox_{\lambda}^f(x) = \arg \min_y f(y) + \frac{\lambda}{2} \|y - x\|^2$$

Suppose the code solution $B^{(j)}$ is obtained at the j_{th} iteration for problem (23.2). At the $(j+1)_{th}$ iteration, B is updated by

$$\begin{aligned} B^{(j+1)} &= Prox_{\lambda}^{\delta}(B^{(j)} - \frac{1}{\lambda} \nabla \mathcal{L}(B^{(j)})) \\ &= \arg \min_B \delta(B) + \frac{\lambda}{2} \|B - B^{(j)} + \frac{1}{\lambda} \nabla \mathcal{L}(B^{(j)})\|^2 \end{aligned}$$

Transform the problem to the min form:

$$\begin{aligned} \min_B & \|B - B^{(j)} + \frac{1}{\lambda} \nabla \mathcal{L}(B^{(j)})\|^2 \\ s.t. \quad & B \in \{-1, 1\}^{r \times n} \end{aligned} \tag{23.3}$$

In (23.3), the problem actually seeks the projection of $B^{(j)} - \frac{1}{\lambda} \nabla \mathcal{L}(B^{(j)})$ onto the binary code space. Indeed, for the indicator function $\delta(B)$ (of the nonempty and closed set \mathbb{B}), its proximal map $Prox_{\lambda}^{\delta}(X)$ reduces to the projection operator:

$$P_{\mathbb{B}}(X) = \arg \min \|B - X\|^2 : B \in \mathbb{B}$$

Problem (23.3) has the analytical solution

$$B^{(j+1)} = sgn(B^{(j)} - \frac{1}{\lambda} \nabla \mathcal{L}(B^{(j)})) \tag{23.4}$$

The overall method can be seen in Fig.39.

For problem (23.1), it can be proved that the algorithm (23.4) converges to a critical point.

Definition 1.

(Kurdyka-Lojasiewicz (KL) property):[36]

The function $f : R^n \rightarrow R \cup \{+\infty\}$ is said to have the Kurdyka-Lojasiewicz property at $x^* \in \text{dom} \partial f$ if there exist $\eta \in (0, +\infty]$, a neighborhood U of x^* and a continuous concave function $\psi : [0, \eta) \rightarrow R_+$ such that:

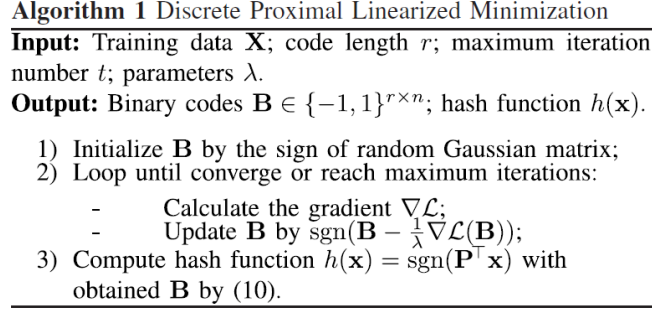


Fig. 39: Discrete Proximal Linearized Minimization.

- (i) $\psi(0) = 0$
- (ii) ψ is C^1 on $(0, \eta)$,
- (iii) for all $s \in (0, \eta)$, $\psi'(s) > 0$
- (iv) for all x in $U \cap [f(x^*) < f < f(x^*) + \eta]$, the Kurdyka-Lojasiewicz inequality holds

$$\psi'(f(x) - f(x^*)) \text{dist}(0, \partial f(x)) \geq 1$$

Definition 2.

(Kurdyka-Lojasiewicz (KL) function):

A function is called KL function if semicontinuous function satisfy the Kurdyka-Lojasiewicz inequality at each point of $\text{dom } \partial f$.

Theorem 1.

Let $f : R^n \rightarrow R$ be a differentiable function whose gradient is L -Lipschitz continuous, and C a nonempty closed subset of R^n . Being given $\epsilon \in (0, \frac{1}{2L})$ and a sequence of stepsize γ_k such that $\epsilon < \gamma_k < \frac{1}{L} - \epsilon$, consider a sequence (x^k) that complies with

$$x^{k+1} \in P_C(x^k - \gamma_k \nabla f(x^k)), \quad \text{with } x^0 \in C$$

If the function $f + \delta_C$ is a KL function and if (x_k) is bounded, then the sequence (x_k) converges to a point x^* in C .

Corollary 1.

Assume the loss function \mathcal{L} is a C^1 (continuously differentiable) semi-algebraic function whose gradient is L -Lipschitz continuous. By choosing a proper sequence of parameters of λ the sequence $B^{(j)}$ generated by the proposed DPLM algorithm converges to a critical point B^* .

Proof: This assumption ensures that the objective $\mathcal{L}(\cdot) + \delta_{\mathbb{B}}(B)$ is a KL function. It is obvious that the sequence $B^{(j)}$ generated by (23.4) is bounded in \mathbb{B} . Based on Theorem 1,

by choosing the parameter λ greater than the Lipschitz constant L , the DPLM algorithm converges to some critical point.

The requirement of the presented DPLM method is only mild. The KL assumption of the objective function is very general that L being the smooth polynomial is a typical instance.

23.2 Case of DPLM

Case 1. Supervised Hashing: Adopt the l_2 loss in the supervised setting, where the learned binary codes are assumed to be optimal for linear classification. The learning objective writes,

$$\begin{aligned}\mathcal{L}(B, W) &= \frac{1}{2} \sum_{i=1}^n \|y_i - W^T b_i\|^2 + \delta \|W\|^2 \\ &= \frac{1}{2} \|Y - W^T B\|^2 + \delta \|W\|^2\end{aligned}$$

Here $Y \in R^{k \times n}$ stores the labels of training data $X \in R^{d \times n}$ with its (i, j) -th entry $Y_{ij} = 1$ if the j -th sample x_j belongs to the i -th of the total k classes and 0 otherwise. Matrix W is the classification matrix which is jointly learned with the binary codes. δ is the regularization parameter.

With the l_2 loss, the binary codes can be easily computed by the DPLM optimization method. Given W , the key step is updating B by (23.4) with the following gradient

$$\nabla \mathcal{L}(B) = WW^T B - WY$$

With B obtained, the classification matrix W is efficiently computed by $W = (BB^T)^{-1}BY^T$. The whole optimization alternatively runs over variable B and W . In practice, initialize B by the sign of random Gaussian matrix, and W and B are then updated accordingly.

Case 2. Unsupervised Graph Hashing: The unsupervised graph hashing optimizes the following objective

$$\begin{aligned}\mathcal{L}(B) &= \frac{1}{2} \sum_{i,j=1}^n \|b_i - b_j\|^2 A_{ij} \\ &= \frac{1}{2} \text{tr}(BLB^T)\end{aligned}$$

where A is the affinity matrix computed with $A_{ij} = \exp(-\|x_i - x_j\|^2/\sigma^2)$ and σ is the bandwidth parameter. L is the associated Laplacian matrix $L = \text{diag}(A1) - A$. So the gradient of the problem can be written as

$$\nabla \mathcal{L}(B) = BL$$

the optimization is performed by updating variable B in each iteration with

$$B^{(j+1)} = \text{sgn}(B^{(j)} - \frac{1}{\lambda} B^{(j)} L)$$

The affinity matrix A can be compute $A = ZZ^T$ with $Z \in R^{n \times m}$ in anchor graph.

Case 3. Bits Uncorrelation and Balance: The bits uncorrelation and balance constraints have been widely used in previous hashing methods. With these two constraints, the problem can be written as

$$\begin{aligned} \min_B \mathcal{L}(B) \\ s.t. \quad BB^T = nI_r, B1 = 0, B \in \{-1, 1\}^{r \times n} \end{aligned}$$

The first two constraints force the binary codes to be uncorrelated and balanced, respectively. Utilize the Lagrange function:

$$\begin{aligned} \min_B \mathcal{L}(B) + \frac{\mu}{4} \|BB^T\|^2 + \frac{\rho}{2} \|B1\|^2 \\ s.t. \quad B \in \{-1, 1\}^{r \times n} \end{aligned}$$

Note that $\|BB^T - nI_r\|^2 = \|BB^T\|^2 + \text{const.}$ So its gradient can be given as follow:

$$\nabla g(B) = \nabla \mathcal{L}(B) + \mu BB^T B + \rho B11^T$$

With this, the binary optimization is conducted by updating variables B in each iteration with

$$B^{(j+1)} = \text{sgn}(B^{(j)} - \frac{1}{\lambda} \nabla g(B^{(j)}))$$

23.3 Summarization

This paper proposes a unified solution framework for binary constraints. Compared with some of the methods for directly solving binary constraints that I have seen before, which has particularity and certain skill, the framework here is more universal. The main principle is to first express the binary constraint with a non-convex and non-smooth indicator function, and use the Lagrange function to penalize the constraint into the objective function through the indicator function. In the end, the objective function becomes a smooth & non-smooth problem. This form conforms to the backward-forward solution framework, so it can be solved using backward-forward.

24 Hashing on Nonlinear Manifolds[37](2020/05/27)

This paper proposed a hashing framework, which provides a practical connection between manifold learning methods (typically non-parametric and with high computational

cost) and hash function learning (requiring high efficiency). Nonlinear manifold is able to more effectively preserve the local structure of the input data without assuming global linearity. One of widely used nonlinear embedding method for hashing is Laplacian eigenmaps (LE).

There are two problem for the use of manifold learning for hashing. One is that these methods do not directly scale to large datasets. For example, to construct the neighborhood graph (or pairwise similarity matrix) in these algorithms for n data points is $O(n^2)$ in time, which is intractable for large datasets. The other is that they are typically non-parametric and thus cannot efficiently solve the critical out-of-sample extension problem. **(This should be the reason why there is a two-step hash.)**

For these problem, the paper show that it is possible to learn the manifold on the basis of a small subset of the data B (with size $m \ll n$), and subsequently to inductively insert the remainder of the data, and any out-of-sample data, into the embedding in $O(m)$ time per point.

24.1 The Proposed Method

Assuming that one has the manifold-based embedding $Y := \{y_1, y_2, \dots, y_n\}$ for the entire training data $X := \{x_1, x_2, \dots, x_n\}$. Given a new data point x_q , one aims to generate an embedding y_q which preserves the local neighborhood relationships among its neighbors $N_k(x_q)$ in X . The following simple objective is utilized:

$$C(y_q) = \sum_{i=1}^n w(x_q, x_i) \|y_q - y_i\|^2$$

where

$$w(x_q, x_i) = \begin{cases} \exp(-\|x_q - x_i\|^2 / \sigma^2), & \text{if } x_i \in N_k(x_q) \\ 0 & \text{otherwise} \end{cases}$$

The objective can make the new embedded location for the point, in the low-dimensional space, should be close to those of the point close to it in the original space. Differentiate $C(y_q)$ with respect to y_q , one obtains

$$\frac{\partial C(y_q)}{\partial y_q} \Big|_{y_q=y_q^*} = 2 \sum_{i=1}^n w(x_q, x_i) (y_q^* - y_i) = 0$$

which leads to the optimal solution

$$y_q^* = \frac{\sum_{i=1}^n w(x_q, x_i) y_i}{\sum_{i=1}^n w(x_q, x_i)}$$

which produce the embedding for a new data point by a (sparse) locally linear combination of the base embeddings. In fact, there is a assumption of the solution, which is that close-by data point lie on or close to a locally linear manifold. In other words, the concept

of nonlinear manifolds is for overall data structures. For similar data, the relationship between them can still be regarded as a linear relationship.

For the solution, the computational complexity can not be solved effective. So the author try to use only small base set with the help of the prototype algorithm.

24.1.1 The Prototype Algorithm

The prototype algorithm is based on entropy numbers defines blow.

Definition 3.

(Entropy Numbers): Given any $Y \subseteq R^r$ and $m \in N$, the m -th entropy number $\epsilon_m(Y)$ of Y is defined as

$$\epsilon_m(Y) := \inf\{\epsilon > 0 | N(\epsilon, Y, \|\cdot - \cdot\|) \leq m\}$$

where N is the covering number. This means $\epsilon_m(Y)$ is the smallest radius that Y can be covered by less or equal to m balls.

One can use m balls to cover Y , thus obtain m disjoint nonempty subsets Y_1, Y_2, \dots, Y_m such that for any $\epsilon > \epsilon_m(Y)$, $\forall j \in \{1, \dots, m\}$, $\exists c_j \in R^r$, s.t. $\forall y \in Y_j, \|c_j - y\| \leq \epsilon$ and $\bigcup_{j=1}^m Y_j = Y$. One can see that each Y_j naturally forms a cluster with the center c_j and the index set $I_j = \{i | y_i \in Y_j\}$.

Let $\alpha_i = \frac{w(x_q, x_i)}{\sum_{j=1}^n w(x_q, x_i)}$ and $C_j = \sum_{i \in I_j} \alpha_i$. For each cluster index set $I_j, j = 1, \dots, m$, $l_j = \lfloor mC_j + 1 \rfloor$ many indices are randomly drawn from I_j proportional to their weight α_i . That is, for $\mu \in \{1, \dots, l_j\}$, the μ -th randomly drawn index $u_{j,\mu}$

$$Pr(u_{j,\mu} = i) = \frac{\alpha_i}{C_j}, \forall j \in \{1, \dots, m\}$$

\hat{y}_q is constructed as

$$\hat{y}_q = \sum_{j=1}^m \frac{C_j}{l_j} \sum_{\mu=1}^{l_j} y_{u_{j,\mu}}$$

Lemma 1.

There is at most $2m$ many unique $y_{u_{j,\mu}}$ in \hat{y}_q .

Lemma 2.

The following holds

$$E[\hat{y}_q] = y_q, Var(\hat{y}_q) \leq \frac{\epsilon^2}{m}$$

Theorem 2.

For any even number $n' \leq n$. If Prototype Algorithm uses n' many non-zero $y \in Y$ to express \hat{y}_q , then

$$Pr[\|\hat{y}_q - y_q\| \geq t] < \frac{2(\epsilon_{\frac{n'}{2}}(Y))^2}{n't^2}$$

Corollary 2.

For an even number n' , any $\epsilon > \epsilon_{\frac{n'}{2}}(Y)$, any $\delta \in (0, 1)$ and any $t > 0$, if $n' \geq \frac{2\epsilon^2}{\delta t^2}$, then with probability at least $1 - \delta$

$$\|\hat{y}_q - y_q\| < t$$

The quality of the approximation depends on $\epsilon_{\frac{n'}{2}}(Y)$ and n' . If data has strong clustering pattern, i.e. data within each cluster are very close to cluster center, one will have small $\epsilon_{\frac{n'}{2}}(Y)$, hence better approximation. Likewise, the bigger n' is, the better approximation is.

24.1.2 Approximation of the Prototype Algorithm

For a query point x_q , the prototype algorithm samples from clusters and then constructs \hat{y}_q . The clusters can be obtained via clustering algorithms such as K -means. For each cluster, the higher $C_j = \sum_{i \in I_j} \alpha_i$, the more draws are made. For each cluster, one wants to select the y_i that has high overall weight $O_i = \sum_{x_q \in X} \alpha_i(x_q)$. For large scale X , the reality is that one does not have access to $w(x, x')$ for all $x, x' \in X$. Only limited information is available such as cluster centers $\{c_j, j \in \{1, \dots, m\}\}$ and $w(c_j, x), x \in X$. The cluster centers $\{c_j, j \in \{1, \dots, m\}\}$ have the largest overall weight the points from their own cluster, i.e. $\sum_{i \in I_j} w(c_j, x_i)$. This suggests one should select all cluster centers to express \hat{y}_q . For a base set B , and any query point x_q , the embedding is predicted as

$$\hat{y}_q = \frac{\sum_{x \in B} w(x_q, x)y}{\sum_{x \in B} w(x_q, x)}$$

The general inductive hash function is formulated by binarizing the low-dimensional embedding

$$h(x) = \text{sgn}\left(\frac{\sum_{j=1}^m w(x, c_j)y_j}{\sum_{j=1}^m w(x, c_j)}\right)$$

where $Y_B := \{y_1, y_2, \dots, y_m\}$ is the embedding for the base set $B := \{c_1, c_2, \dots, c_m\}$, which is the cluster centers obtained by K -means.

With this, the embedding for the training data becomes

$$Y = \bar{W}_{XB}Y_B$$

where \bar{W}_{XB} is defined such that $\bar{W}_{ij} = \frac{w(x_i, c_j)}{\sum_{i=1}^m w(x_i, c_j)}$, for $x_i \in X, c_j \in B$.

The proposed hashing method is termed Inductive Manifold-Hashing (IMH), as shown in Fig.40.

24.2 Extension of the IMH framework

Case 1. Stochastic Neighborhood Preserving Hasing: T-SNE is a modification of stochastic neighborhood embedding (SNE) which aims to overcome the tendency of

Algorithm 1 Inductive Manifold-Hashing (IMH)

Input: Training data $\mathbf{X} := \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, code length r , base set size m , neighborhood size k

Output: Binary codes $\mathbf{Y} := \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\} \in \mathbb{B}^{n \times r}$

- 1) Generate the base set \mathbf{B} by random sampling or clustering (e.g. K-means).
 - 2) Embed \mathbf{B} into the low-dimensional space by (11), (14) or any other appropriate manifold learning method.
 - 3) Obtain the low-dimensional embedding \mathbf{Y} for the whole dataset inductively by Equation (10).
 - 4) Threshold \mathbf{Y} at zero.
-

Fig. 40: Inductive Manifold-Hashing (IMH)

that method to crowd points together in one location. The cost function of t-SNE in fact maximizes the smoothed recall of query points and their neighbors. The original t-SNE does not scale well, as it has a time complexity which is quadratic in n . More significantly, however, it has a non-parametric form, which means that there is no simple function which may be applied to out-of-sample data in order to calculate their coordinates in the embedded space. One first applies t-SNE to the base set B

$$\min_{Y_B} \sum_{x_i \in B} \sum_{x_j \in B} p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right)$$

Here p_{ij} is the symmetrized conditional probability in the high-dimensional space, and q_{ij} is the joint probability defined using the t -distribution in the low-dimensional embedding space. After solving by a gradient descent procedure, obtain the embedding Y_B of samples $x_i \in B$, the hash codes for the entire dataset can be easily computed. This method is labelled IMH-tSNE.

Case 2.Hashing With Relaxed Similarity Preservation: One can compute Y_B considering local smoothness only within B . Then, Y_B is alternatively computed by considering the smoothness both within B and between B and X . The objective can be easily obtained as:

$$C(Y_B) = \sum_{x_i, x_j \in B} w(x_i, x_j) \|y_i - y_j\|^2 + \lambda \sum_{x_i \in B, x_j \in X} w(x_i, x_j) \|y_i - y_j\|^2$$

where λ is the trade-off parameter. The first part enforces smoothness of the learned embedding within B while the second part ensures the smoothness between B and X .

Then, the problem can be reformulated as follow:

$$\min \text{trace}(Y_B^T (D_B - W_B) Y_B) + \lambda \text{trace}(Y_B^T (D_{BX} - \bar{W}_{XB}^T W_{XB}) Y_B)$$

where $D_B = \text{diag}(W_B \mathbf{1})$ and $D_{BX} = \text{diag}(W_{BX} \mathbf{1})$ are both $m \times m$ diagonal matrices. Take the constraint, one obtains:

$$\min_{Y_B} \text{trace}(Y_B^T (M + \lambda T) Y_B)$$

Algorithm 2 Supervised Inductive Manifold-Hashing (IMHs)

Input: Unlabelled data $\mathbf{X} := \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, labelled data \mathbf{X}_s of t classes, code length r , base set size m , neighborhood size k

Output: Binary codes $\mathbf{Y} := \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\} \in \mathbb{B}^{n \times r}$

1) Generate the base set $\mathbf{B} := \{\mathbf{c}_{1,1}, \dots, \mathbf{c}_{m_1,1}, \dots, \mathbf{c}_{1,t}, \dots, \mathbf{c}_{m_t,t}\}$ by K-means on data points from each class of \mathbf{X}_s .

2) Embed \mathbf{B} into the low-dimensional space by (11), (14) or any other appropriate manifold learning method and get \mathbf{Y}_B .

3) Apply supervised subspace learning algorithms such as LDA on \mathbf{Y}_B .

4) Obtain the low-dimensional embedding \mathbf{Y} for the whole dataset inductively by Equation (10).

5) Threshold \mathbf{Y} at zero.

Fig. 41: Supervised Inductive Manifold-Hashing (IMHs)

$$s.t. \quad \mathbf{Y}_B^T \mathbf{Y}_B = m\mathbf{I}$$

where $M = D_B - W_B$, $T = D_{BX} - \bar{W}_{XB}^T W_{XB}$. This method is named IMH-LE in the following text.

Case 3. Semantic Hashing with Supervised Manifold Learning: First, the base set $B := \{c_{1,1}, \dots, c_{m_1,1}, \dots, c_{1,t}, \dots, c_{m_t,t}\}$ is generated by applying K-means on data from each of the t classes. After the nonlinear embedding \mathbf{Y}_B of B are obtained, the supervised subspace learning algorithms are simply conducted on \mathbf{Y}_B . For a new data point, its binary codes are then obtained with the previous solution. The supervised manifold hashing method is summarized in Fig.41.

24.3 Summarization

From this article, we can first learn the concept that graph-based hashing is a type of manifold hashing. There are two main problems with this kind of hash, which were also mentioned in the articles we contacted earlier. The first question is about the complexity of Laplacian storage. As a matrix describing the similarity between points and points, the storage complexity of Laplace matrix is $O(n^2)$, which limits the practical application of this method. An effective way to reduce the storage complexity of the previous method is to use the Laplacian matrix of the anchor graph. The second problem is a graph-based method, which essentially uses the distance between points to describe features, so it is non-parameter. This means that when new data comes in, it is difficult to have an effective hash function to turn the new data into a hash code. In response to this problem, the previous articles are basically solved by two-step hash, that is, first obtain the hash code and then use SVM to return to the hash function. In this article, the author uses theory to prove that even if part of the data is used, the accuracy of the hash code can be maintained, then the first problem is logically solved. Regarding the second question, I think the idea of this article is actually a classification idea, its essence is the process of classifying the

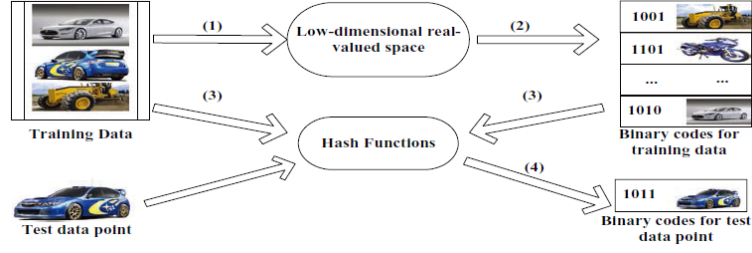


Fig. 42: The framework of the Sparse Hash

out-of-sample data after clustering.

Of course, in this article, the method that using learned rotations is proposed for the problem that sampling may cause all deviations of the hash code and the proof process involved in the article need further thought.

25 Sparse hashing for fast multimedia search[38](2020/05/30)

This paper proposed a two-step hashing work, called sparse hashing framework (SH), which explore the properties of nonnegative sparse coding to generate interpretable binary codes as well as to achieve minimal reconstruct error. The proposed SH first converts original data into low-dimensional data through a novel nonnegative sparse coding method, then converts the low-dimensional data into Hamming space by a binarization rule. Finally, SH learns hash functions by taking a prior knowledge into account.

The motivation of this paper mainly consists of the following two aspects. On the one hand, from the perspective of hash meaning, common binarization methods generate unexplainable hash codes. Regarding this issue, my understanding is that in the common hash codes step, many methods still use *sgn* constraints. This constraint is equivalent to a forced change, turning all features into negative and positive forms. For some features that cannot be negative, such as histograms, this *sgn* binarization method loses its ability to discern. On the other hand, the processing of out of sample data is an unavoidable problem. Although some two-step hashing work utilize the SVM to deal with the problem, it can not ensure the encoding efficient.

The proposed SH framework is shown in Fig.42, including four steps, namely, mapping the inputs, generating binary codes, learning hash functions and encoding unseen data.

25.1 Hash Code

SH first uses the sparse coding base representation idea to generate the representation coefficients for the data x . Here, due to the characteristics of sparse coding, the generated

coefficients are all non-negative. Given a training data matrix $X = (x_{ij}) \in R^{d \times n}$ (each column represents a data point), the object is to learn the based $B \in R^{d \times m}$ and the corresponding sparse codes $S \in T^{m \times n}$ from training data X with the constraints, such as preserving local similarity structure, and generating nonnegative sparse codes. The objective function of the proposed SH is defined as:

$$\min_{B, S} \|X - BS\|_F^2 + \alpha \text{tr}(SLS^T) + \lambda \sum_{i=1}^n \|s_i\|_1$$

$$s.t. \|B_j\|^2 \leq 1, S \succeq 0$$

$S \succeq 0$ indicates each element in matrix S (or vector s) is nonnegative. $\|B_j\|^2 \leq 1, j = 1, \dots, m$ is to prevent B from having arbitrarily large value which would lead to very small values of S . Similarity, the overall function can be divided into two parts, including sparse code and preserving locality.

1. Sparse Coding: The reconstruction estimation for sparse coding can be obtaining by minimizing the loss function under a penalized constraint, that is,

$$\min_S \|x - Bs\|_2^2 + \lambda \|s\|_1$$

where $x \in R^d$ represents the data point in original space, $B \in R^{d \times m}$ represents the bases, and $s \in R^m$ is sparse codes (i.e, the weights of the bases, or the coordinates of the data point in low-dimensional space) of x with respect to B . m is the number of reduced dimensionality (or the number of bases, or the number of bits in Hamming space). In fact, $\|s\|_1$ is a convex approximation of $\|s\|_0$, which ensures the sparsity, that is, many of $s_i (i = 1, \dots, m)$ are zeros while adjusting the values of λ (where $\lambda > 0$ is a regularization parameter).

2. Preserving Similarity: Due to the fact that local similarity structures are often more important than global one, the author utilize heat kernel to build a weight matrix W . Given a weight matrix W , they use the Euclidean distance to measure the smoothness between s_i and s_j (where s_i (or s_j) is sparse codes of x_i (or x_j) respectively in low-dimensional space), that is

$$\begin{aligned} \frac{1}{2} \sum_{i,j} \|s_i - s_j\|^2 w_{ij} &= \sum_{i,j} s_i D_{ii} s_i^T - \sum_{i,j} s_i s_j^T w_{ij} \\ &= \text{tr}(SDS^T) - \text{tr}(SW S^T) \\ &= \text{tr}(SLS^T) \end{aligned}$$

Denote D as a diagonal matrix. The entries of D are the column (or row) sum of W , that is, $D_{ii} = \sum_j w_{ij}$. Obviously $L = D - W$ is a Laplacian matrix.

25.1.1 Solve the objective function

Optimize B and S alternatively in an iterative process. **Optimize B :** When S is fixed, learn the bases B by optimizing the following objective function with a conjugate gradient decent method

$$\begin{aligned} \min_B & \|X - BS\|_F^2 \\ \text{s.t.} & \|B_j\|^2 \leq 1, j = 1, \dots, m \end{aligned}$$

Optimize S : When B is fixed, optimize each column vector s in S one by one rather than optimize all the vectors in S simultaneously because different bits should be independently generated according to the constraints of hashing. Thus, the objective function on one column vectors s becomes

$$\begin{aligned} \min_s & \|x - Bs\|_2^2 + \alpha(2s^T(Sl_i) - s^T l_{ii}s) + \lambda \sum_{j=1}^m \|s_j\|_1 \\ \text{s.t.} & s \succeq 0 \end{aligned} \quad (25.1)$$

where l_i is the i -th column of L , and l_{ii} is the element in the i -th column and i -th row of L . Then convert (25.1) to standard form of Lasso by Theorem and then obtain the optimal s .

Theorem 3.

The objective function (25.1) is equivalent to:

$$\begin{aligned} \min_s & \|\tilde{x} - \tilde{B}s\|_2^2 + \lambda \sum_{j=1}^m \|s_j\|_1 \\ \text{s.t.} & s \succeq 0 \end{aligned}$$

where $\tilde{B}^T \tilde{B} + (B^T B + \alpha l_{ii} I)$, $\tilde{x} = (\tilde{B}^T)^{-1}(B^T x + \alpha S w_i)$, $\tilde{x} \in R^d$

Proof. Denotes $S_{-i} = (s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_n)$ as S without the column vector s_i , and l_{-ii} (d_{-ii} and w_{-ii}) as the column vector l_i (d_i and w_i) without the element l_{ii} (d_{ii} and w_{ii}). Due to $w_{ii} = 0$, then have $l_{-ii} = d_{-ii} - w_{-ii} = -w_{-ii}$, then $S_{-i} l_{-ii} = -S_{-i} w_{-ii} = -S_{-i} w_{-ii} - s w_{ii} = -S w_i$. Thus

$$\begin{aligned} 2s^T(Sl_i) - s^T l_{ii}s &= 2s^T \begin{pmatrix} s & S_{-i} \end{pmatrix} \begin{pmatrix} l_{ii} \\ l_{-ii} \end{pmatrix} - s^T l_{ii}s \\ &= s^T l_{ii}s + 2s^T S_{-i} l_{-ii} \\ &= s^T l_{ii}s - 2s^T (S w_i) \end{aligned}$$

Hence

$$\min_s \|s - Bs\|_2^2 + \alpha(2s^T(Sl_i) - s^T l_{ii}s)$$

ALGORITHM 1: SH_LARS algorithm

Input: \mathbf{x} and \mathbf{B}
Output: \mathbf{s}
 Normalize \mathbf{B} ; center \mathbf{x} ;
 $s_1, \dots, s_m = 0$;
 $\mathbf{c} = \mathbf{B}^T \mathbf{x}$; $C = \max_j \{c_j\}$; $j = \operatorname{argmax}_j \{c_j\}$; $AS = \{j\}$;
repeat
 $\mathbf{G}_A = (\mathbf{B}^T \mathbf{B})^{-1}$; $\mathbf{A}_A = (\mathbf{1}_A^T \mathbf{G}_A \mathbf{1}_A)^{-\frac{1}{2}}$; $\mathbf{u}_A = \mathbf{B}(\mathbf{A}_A \mathbf{G}_A \mathbf{1}_A)$;
 $\mathbf{a} = \mathbf{B}^T \mathbf{u}_A$; $\mathbf{c} = \mathbf{B}^T (\mathbf{x} - \mathbf{u}_A)$; $C = \max_j \{c_j\}$;
 if $|AS| < k$ **then**
 $\gamma = \min_{j \notin AS} \{\frac{C - c_j}{A_A - a_j}\}$
 else
 $\gamma = \frac{C}{A_A}$. *// γ : the maximal length of the next step updated*
 end
 Lasso Modification *//same as in [Efron et al. 2004];*
 Update AS *//same as in [Efron et al. 2004];*
until meeting the pre-defined conditions;
return \mathbf{s} ;

Fig. 43: SH_LARS algorithm

$$\begin{aligned} \Leftrightarrow \min_s s^T (B^T B + \alpha l_{ii} I) s - 2s^T (B^T x + \alpha S w_i) \\ \Leftrightarrow \min_s \|\tilde{x} - \tilde{B}s\|_2^2 \end{aligned}$$

So the bases B and the sparse codes S of training data can be get, follow the algorithm shown in Fig.43.

Binarization rule: In this step, SH converts the low-dimensional data into compact Hamming space by a simple binarization rule: encoding each positive value in low-dimensional nonnegative real-valued space into 1, and zero into 0. The proposed binarization rule can be naturally interpreted for real-world data. In low-dimensional real-valued space, each data point is sparsely represented by the bases. The positive weight (i.e., sparse codes) in the i -th dimension means that the data point is composed by the i -th basis. The value of the weight means the important degree of the i -th basis to the data point. The zero weight in the i -th dimension indicates that the data point does not contain the i -th basis.

25.2 Learning Hash Functions

Although the proposed SH can generate explicit mapping functions for encoding unseen data, that is via the overall objective function. But due to the exist of Laplacian matrix L , it can not be used in reality. In this paper, the author utilize the Elastic Net (EN) modal as the regression modal. Although it is a regression modal, it is still unsupervised because the label information (i.e, the binary codes of training data) is learned

rather than given in advance. EN modal can simultaneously achieve accuracy and sparsity and encourages grouping effect, where strongly correlated predictors (or encoders) tend to be in or out of the model together. In addition, EN modal is particularly useful when the number of predictors (or encoders) (d) is much bigger than the number of observations (n).

Given d -dimensional training data and the learned m -dimensional binary codes, build m explicit hash functions. For any fixed nonnegative λ_1 and λ_2 , denoting the learned m -dimensional binary codes as Y , where $Y \in R^{n \times m}$ is centered, that is, $\sum_{i=1}^n y_i = 0$, denoting d -dimensional standardized training data as $X \in R^{d \times n}$, that is, $\sum_{i=1}^n x_{j,i} = 0$, for each column $y \in R^n$ in Y , the EN model is defined as:

$$\min_{\beta_i} \|y_i - X^T \beta_i\|_2^2 + \lambda_1 \|\beta_i\|_1 + \lambda_2 \|\beta_i\|_2^2$$

where $\beta_i (\beta_i \in R^d, i = 1, \dots, m)$ is the elastic net estimator (or coefficient), that is, the transformation coefficient of the i -th hash function (or classifier). So, given unseen data point x_t , obtain its binary codes of the i -th hash function by $y_i^t = \text{sgn}(x_t \beta_i)$. Denoting $\beta = (\beta_1, \dots, \beta_m)$, m binary codes of x_t can be encoded by $\text{sgn}(x_t^T \beta)$.

For improving the encoding of the learned hash function via the EN model, the CW method is used, which searches for an optimal matrix $C (C \in R^{m \times m}$ and m is the number of hash functions) to make the use of the disparity of m classifiers learned from the EN model.

CW method performs CCA between Y and X to obtain $r_i = \max \text{corr}(Y u_i, X^T v_i), i = 1, \dots, m (m = \min(m, n), \text{ actually, } m \ll n)$, $U = \{u_1, \dots, u_m\}$ and $V = \{v_1, \dots, v_m\}$. u_i and v_i are the vectors such that the correlation between the linear combinations of the response $Y u_i$ and $X^T v_i$ is maximized.

Second, set $W = \text{diag}\{w_1, \dots, w_m\}$ as an $m \times m$ diagonal matrix of shrinkage factors with

$$w_i = \max \frac{(1-f)(r_i^2 - f)}{(1-f)^2 r_i^2 + f^2(1-r_i^2)}, 0, i = 1, \dots, m$$

where r_i is the canonical correlations between y_i and X , and $f = d/n$.

Third, the optimal matrix C is obtained by:

$$C = U^{-1} W U$$

Finally, given unseen data point x_t and $\beta = (\beta_1, \dots, \beta_m)$ learned from the EN model, m binary codes of x_t can be encoded by $\text{sgn}(x_t \beta C^T)$.

The pseudo of ENCW method is presented in Algorithm 2, shown in Fig.44.

ALGORITHM 2: ENCW algorithm

Input: $\mathbf{X}, \mathbf{Y}, \mathbf{x}_t, \lambda_1, \lambda_2$
Output: $\hat{\mathbf{y}}_t$
Perform CCA on \mathbf{X} and \mathbf{Y} ;
Calculate matrix \mathbf{W} by Equation (9);
for each y_i **in** \mathbf{Y} **do**
| Compute vector β_i by Equation (7),
end
Compute $\hat{\mathbf{y}}_t = \mathbf{x}_t \beta \mathbf{C}^T$;

Fig. 44: ENCW algorithm

SH generates the binary codes of training data by SH-LARS algorithm, and hash functions by ENCW algorithm. Given a test data point x_t , SH maps it into Hamming space and obtains its m -bit binary codes by $\text{sgn}(x_t^T \beta (U^{-1} W U)^T)$.

25.3 Summarization

This article is an earlier article. From the model point of view, it is to change the previous thinking about $H = TX$. The previous idea was to transform X into B using a random linear change, which is a process of forward projection. Here, sparse coding is introduced so that $X = BH$ is a process of back projection, regression and linear representation. Such a process avoids the constraint of binarization, and directly thresholds the coefficients of the linear representation, which reduces the difficulty of the entire solution process. For the process of solving the hash function, there is no SVM for regression, but the Elastic Net model is used for regression. There is nothing particularly need to be discussed here. As for the CW method, I understand it as a compensation for the regression model, because for regression, many times we think the given data is independent and identically distributed, this assumption is difficult to establish in reality. The CW method is a method for analyzing the correlation between data, thereby making the regression model more accurate.

In general, from the time when this article was published at that time, it should have opened up a new idea of hashing problems and other mature models, so the reference value is greater. This article gives a different perspective on the meaning of hash code, which is still very interesting.

References

- [1] Weng L , Preneel B . From Image Hashing to Video Hashing[C]// Advances in Multimedia Modeling, International Multimedia Modeling Conference, Mmm, Chongqing, China, January. Springer-Verlag, 2010.
- [2] Ye G , Liu D , Wang J , et al. Large-Scale Video Hashing via Structure Learning[C]// IEEE International Conference on Computer Vision. IEEE, 2014.
- [3] V. E. Liong, J. Lu, Y. Tan and J. Zhou, "Deep Video Hashing," in IEEE Transactions on Multimedia, vol. 19, no. 6, pp. 1209-1219, June 2017.
- [4] Coskun B , Member S , IEEE, et al. Spatio - Temporal Transform Based Video Hashing[J]. IEEE Transactions on Multimedia, 2007, 8(6):1190-1208.
- [5] Z. Wang, L. Lu, A. C. Bovik, and J. Kouloheris, "Video quality assessment based on structural distortion measurement," Signal Process.: Image Commun., vol. 19, no. 1, 2004.
- [6] Liu X , Sun J , Liu J . Visual Attention Based Temporally Weighting Method for Video Hashing[J]. IEEE Signal Processing Letters, 2013, 20(12):1253-1256.
- [7] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," CVPR 1999, vol. II, pp. 246 - 252, 1999.
- [8] J. Zhang, J. D. Sun, and H. Yan et al., "Visual attention model with cross-layer saliency optimization," in IEEE Int. Conf. Intelligent Information Hiding and Multimedia Signal Processing, 2011, pp. 240 - 243.
- [9] P. Jiang and X. L. Qin, "Keyframe-Based video summary using visual attention clues," IEEE Trans. Multimedia, vol. 17, no. 2, pp. 64 - 73, 2010.
- [10] Bilibili,<https://www.bilibili.com/video/BV13b411w7Xj>
- [11] Zhihu,<https://zhuanlan.zhihu.com/p/30483076>
- [12] Itti, Laurent. Feature combination strategies for saliency-based visual attention systems[J]. Journal of Electronic Imaging, 2001, 10(1):161.
- [13] CSDN,<https://blog.csdn.net/qq-40855366/article/details/81177174>
- [14] Wang J , Sun J , Liu J , et al. A visual saliency based video hashing algorithm[C]// IEEE International Conference on Image Processing. IEEE, 2012.
- [15] M. M. Esmaeili, M. Fatourechi and R. K. Ward, A robust and fast video copy detection system using content-based fingerprinting, IEEE Transactions on Information Forensics and Security, 6(1), pp.213-226, 2011.
- [16] Sun J , Wang W , Li J , et al. Hash length prediction for video hashing[C]// 2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW). IEEE, 2016.
- [17] Yang Z , Ian Raymond O , Sun W , et al. Deep Attention-Guided Hashing[J]. IEEE Access, 2019, 7:11209-11221.
- [18] Borji, Ali, Cheng, Ming-Ming, Hou, Qibin,et al. Salient Object Detection: A Survey[J]. Eprint Arxiv, 2014, 16(7):3118.
- [19] Borji A , Itti L . State-of-the-Art in Visual Attention Modeling[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2013, 35(1):p.185-207.

- [20] Zhang J , Wang M , Lin L , et al. Saliency Detection on Light Field: A Multi-Cue Approach[J]. ACM transactions on multimedia computing communications and applications, 2017, 13(3):32.1-32.22.
- [21] Li N , Sun B , Yu J . A Weighted Sparse Coding Framework for Saliency Detection[C]// CVPR 2015. IEEE Computer Society, 2015.
- [22] Xiaohui Shen, Ying Wu. A Unified Approach to Salient Object Detection via Low Rank Matrix Recovery[J]. 2012.
- [23] Li X , Lu H , Zhang L , et al. Saliency Detection via Dense and Sparse Reconstruction[C]// IEEE International Conference on Computer Vision. IEEE, 2013.
- [24] CSDN, <https://blog.csdn.net/dayenglish/article/details/51275128#commentsedit>
- [25] Peng H , Li B , Ling H , et al. Salient Object Detection via Structured Matrix Decomposition[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, 39(4):818-832.
- [26] Shen F , Xu Y , Liu L , et al. Unsupervised Deep Hashing with Similarity-Adaptive and Discrete Optimization[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018:1-1.
- [27] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In Proc. ICML, pages 1 – 8, 2011.
- [28] B. Wu and B. Ghanem, " ℓ_p -Box ADMM: A Versatile Framework for Integer Programming," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 7, pp. 1695-1708, 1 July 2019, doi: 10.1109/TPAMI.2018.2845842.
- [29] Cui H , Zhu L , Li J , et al. Scalable Deep Hashing for Large-Scale Social Image Retrieval[J]. IEEE Transactions on Image Processing, 2019, PP(99):1-1.
- [30] Cao Z , Long M , Wang J , et al. HashNet: Deep Learning to Hash by Continuation[J]. 2017.
- [31] Hu D , Nie F , Li X . Deep Binary Reconstruction for Cross-Modal Hashing[J]. IEEE Transactions on Multimedia, 2019, 21(4):973-985.
- [32] Luo X , Zhang P F , Huang Z , et al. Discrete Hashing with Multiple Supervision[J]. IEEE Transactions on Image Processing, 2019:1-1.
- [33] Luo X , Nie L , He X , et al. Fast Scalable Supervised Hashing[C]// The 41st International ACM SIGIR Conference. ACM, 2018.
- [34] Wang J , Zhang T , Song J , et al. A Survey on Learning to Hash[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016, PP(99):1-1.
- [35] Shen F , Zhou X , Yang Y , et al. A Fast Optimization Method for General Binary Code Learning[J]. IEEE Transactions on Image Processing, 2016:1-1.
- [36] Attouch H , Jér?me Bolte, Svaiter B F . Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward – backward splitting, and regularized Gauss – Seidel methods[J]. Mathematical Programming, 2013, 137(1-2):91-129.
- [37] F. Shen, C. Shen, Q. Shi, A. van den Hengel, Z. Tang and H. T. Shen, "Hashing on Nonlinear Manifolds," in IEEE Transactions on Image Processing, vol. 24, no. 6, pp. 1839-1851, June 2015, doi: 10.1109/TIP.2015.2405340.

- [38] Xiaofeng Zhu, Zi Huang, Hong Cheng, Jiangtao Cui, and Heng Tao Shen. 2013. Sparse hashing for fast multimedia search. *ACM Trans. Inf. Syst.* 31, 2, Article 9 (May 2013), 24 pages. DOI:<https://doi.org/10.1145/2457465.2457469>