J. Vis. Commun. Image R. 72 (2020) 102908

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Full length article

Scalable Hash From Triplet Loss Feature Aggregation For Video De-duplication

Wei Jia [a], Li Li [a], Zhu Li [a,*], Shuai Zhao [b], Shan Liu [b]

[a] Department of Computer Science and Electrical Engineering, University of Missouri-Kansas City, 5110 Rockhill Rd, Kansas City, MO, 64110-2446, USA
[b] Tencent Media Lab, 2747 Park Blvd, Palo Alto, CA 94306, USA

ARTICLE INFO

MSC:
41A05
41A10
65D05
65D17

Keywords:
Binary hash
Binary tree
Fisher vector
Triplet loss
Video de-duplication

ABSTRACT

The producing, sharing and consuming life cycle of video content creates massive amount of duplicates in video segments due to variable bit rate representation and fragmentation in the playbacks. The inefficiency of this duplicates to storage and communication motivate researchers in both academia and industry to come up with computationally efficient video deduplication solutions for storage and CDN providers. Moreover, the increasing demands of high resolution and quality aggravate the status of heavy burden of cluster storage side and restricted bandwidth resources. Hence, video de-duplication in storage and transmission is becoming an important feature for video cloud storage and Content Delivery Network (CDN) service providers. Despite of the necessity of optimizing the multimedia data de-duplication approach, it is a challenging task because we should match as many as possible duplicated videos under not removing videos by mistake. The current video de-duplication schemes mostly relies on the URL based solution, which is not able to deal with non-cacheable content like video, which the same piece of content may have totally different URL identification and fragmentation and different quality representations further complicate the problem. In this paper, we propose a novel content based video segmentation identification scheme that is invariant to the underlying codec and operational bit rates, it computes robust features from a triplet loss deep learning network that captures the invariance of the same content under different coding tools and strategy, while a scalable hashing solution is developed based on Fisher Vector aggregation of the convolutional features from the Triplet loss network. Our simulation results demonstrate the great improvement in terms of large scale video repository de-duplication compared with state-of-the-art methods.

1. Introduction

Modern dynamic adaptive video streaming methods such as MPEG-DASH [1], Apple HLS [2] and Microsoft Smooth Streaming [3] have a great impact on how content providers store and serve the media contents in the cloud, such as a content delivery network (CDN). OTT (over the top) content providers are also pushing subscription-based video on demand (VoD) services that offer streaming services on television. The media content creation, sharing and consumption process generate many duplicates but are not necessarily identical in bit stream. There is a de-duplication of media content use case for example. If a content identification scheme can support identification of duplicates in network caches in core networks and edge nodes, then traffic can be localized and bandwidth saved. This creates challenges to the existing Content Delivery Network (CDN) and storage de-duplication schemes like those based on MD5 [4] hashing of file chunks. New compact rate agnostic and coding scheme agnostic content identification and hashing solution are needed, to characterize media segments across different representations and with totally different bit streams. Scalable and robust signatures for media content to support de-duplication at fine granular spatio-temporal segments granularity, are important to rip the full benefits of storage de-duplication.

Therefore the massive multimedia data is pushing forward the paradigm of effective storage on cluster servers. Fig. 1 depicts that various contents of resolutions and quantized parameters(REQP) are consumed by very diversified consumers' platform. In current media content storage scheme, the storage side has to hold all of the REQP media content, which is error-prone and not cost-effective. We define the version as the combination of resolution and quantization parameter namely REQP in this paper. If users apply the same version (REQP) of videos from the server ignoring the identical ones in the

* Corresponding author.
E-mail addresses: wj3wr@umsystem.edu (W. Jia), lil1@umkc.edu (L. Li), zhu.li@ieee.org (Z. Li), shuai.zhao@ieee.org (S. Zhao), shanl@tencent.com (S. Liu).

https://doi.org/10.1016/j.jvcir.2020.102908
Received 17 June 2019; Received in revised form 1 February 2020; Accepted 6 September 2020
Available online 8 September 2020

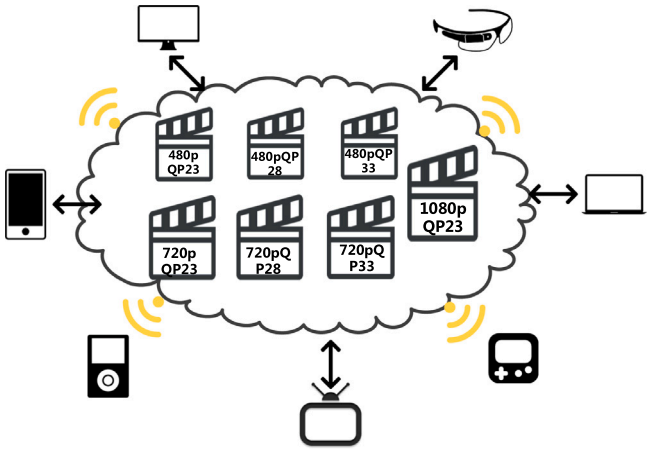


Fig. 1. CDN Content Cache. There are multiple versions of videos with different resolutions and qualities on the CDN. Users visit which version of videos corresponding to the conditions of their devices.

content delivery network (CDN), the pressure on the network from video delivery and storage will be quite large. Hence, how to retrieve and remove the duplicated versions of videos is an essential task for researchers.

In spite of that leveraging the video de-duplication [5] scheme is quite necessary and promising, the micro improvement of its performance exhibits it is difficult to develop. First of all, multimedia data on the cloud cluster and CDN is all the cherish product from industry and user, so it is extremely strict to remove any videos. This results in that we should derive the system of high accuracy and recall. Especially, we focus on the true positive rate (TPR) under false positive rate equals 0 because we cannot allow the judgment is error and the product is deleted accidentally. Secondly, a tremendous quantity of videos cost the system much time to recognize and match the video identity if the algorithm is not precise and efficient. A high-delay method cannot satisfy the real time requirement in the social media time.

To alleviate the stated problems above, there are two groups of methods on video de-duplication depending on the comparison domain. The first group tries to perform video de-duplication directly in the pixel or frequency domain. They make use of the geometry correlation in a frame or the time correlation in a sequence to decide based on comparing the pixels information. The second group tries to use the hashing representation to replace to pixels. The most representative work is to use the deep learning features to derive the hash. Though this method utilized deep learning method to obtain some performance improvements, the cross-entropy loss function is in essence unsuitable for the video de-duplication task. And the lack of dataset is not convincing enough to claim good video de-duplication results.

Therefore, we propose a novel deep learning based scheme to de-duplicate the replicated videos in the cluster. Our method comprise two parts: off-line training model and on-line aggregating model. The off-line train model means that we employ triplets dataset to train out triplet loss function embedded VGG11 network. To acquire the hard and valuable training triplets, we apply the binary-tree partitioning the samples according to their attributions. Afterward, we perform the mature triplets VGG11 [6] model to train a variety of Primary Components Analysis (PCA) [7] models and Gaussian mixture models (GMM) [8]. For the on-line aggregation model, we first aggregate the fisher vector [9] by the trained triplets VGG11, PCA and GMM models above. Then we binary hash the fisher vector with different bits to get the scalable hash code which is a brief and effective representation for video de-duplication.

We proposed a deduplication method in our previous work [10]. In this paper, we propose a novel deep learning based scheme for deduplications. We provide more motivation, analysis, experimental results and comparison of related works on our proposed method. Additionally, in order to validate the efficiency of our algorithm, we implement more ablation studies for comparison. Our method comprises both a offline training and online aggregation model:

- Offline training model: employ triplets dataset to train out triplet loss function embedded Visual Geometry Group (VGG) network and acquire the hard and valuable training triplets by applying the binary tree partitioning the samples according to their attributions. Then mature triplets VGG11 model is performed to train a variety of Primary Components Analysis (PCA) models and Gaussian mixture models (GMM)
- Online aggregation model: binary hashing the fisher vector (FV) with aggregated trained triplets VGG11, PCA and GMM models obtained from offline training.

Our contribution towards video deduplications are summarized as below:

(1) We consider combining triplet loss with Visual Geometry Group (VGG) deep learning network which is trained of outstanding performance by huge media dataset to derive the features. Triplet loss function based network can learn convolutional features which is invariant to coding method and bit rates.
(2) We propose applying fisher vector to the features for feature aggregation. We utilize proposed algorithm to extract fisher vectors from outputs of VGG with triplet loss function. Fisher vector exhibits the powerful expression ability of main features for a video frame.
(3) Particularly, we propose employing binary tree to obtain the triplets to boost the performance of the triplet-loss based VGG network.
(4) We also utilize the extracting algorithm generating the scalable binary hash. The scalable binary hash can obtain different trade-offs according to different bitrate requirements.

The experimental results show that the proposed binary-tree embedded triplet loss network combining with scalable hash from fisher vector (BTF) algorithm outperforms cross-entropy [11] loss function with PCA (CP) approach in various scalable hashes.

The remaining of this paper is organized as follows. The instruction of related work will be in Section 2. We elaborate the principle of triplet loss function embedding into VGG network and the integral network structure in Section 3. In Section 4, the Binary-Tree algorithm to produce the triplets with similar variance attributions is introduced in detail. We experiment on large-scale video dataset and give the whole process and results of this in Section 5. We conclude the whole paper in Section 6.

2. Related work

As mentioned in Section 1, we can divide the video de-duplication work into two categories. The first type is traditional methods using the comparison information of pixel or frequency domain. The other one is deep learning based approaches extracting the convolutional features as the match evidences.

For conventional ways, Katiyar et al. [13] used a 2-phase video comparing scheme which is for localizing a short frames clip in a long video. Paisitkriangkrai et al. [14] defined a new heuristic rule to measure the degree of resemblance between two clip sequences based on sequence Shape Similarity method. The work of Greene et al. [15] recognizes patterns of video content with a first intermediate device and sends a communication to another one which transmits a cached version of the video. These works [16–21] roughly present a type of secure system architecture design which bridges together the advantages of video compression and encrypted data de-duplication. They exploit clip or layer-level de-duplication, which treats each clip as a unit for

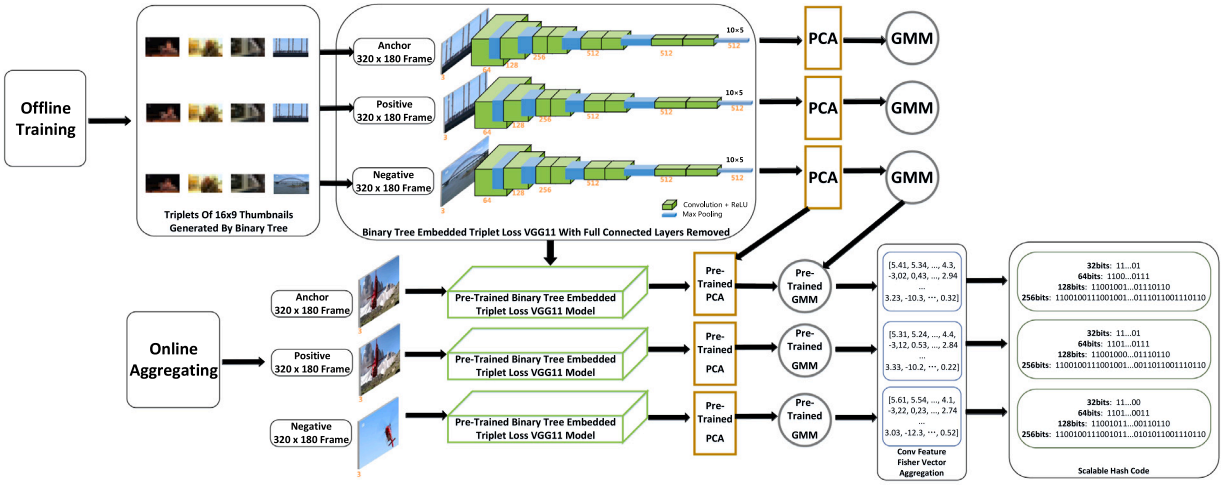


Fig. 2. Scalable hashing framework. Offline training: At first of all we develop the binary-tree [12] generating valuable triplets including anchors, positive samples and negative samples; Then we input the 320 × 180 frames of triplets into VGG11 of full connected layers removed respectively to train the network, PCA and GMM models. Online aggregating: we utilize the trained network, PCA and GMM models to calculate the fisher vector aggregation from convolutional features; then we quantize the fisher vectors to scalable binary hash code involving 32, 64, 128 and 256 bits in this paper. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

de-duplication. [22,23] basically generate a hash for each data block. After determining a processing status for the hash by a model, they perform another model discarding the duplicate hashes and their data blocks. As [24] analyzed, secure hash such as fingerprint [25] shows much more computationally efficient than the traditional compression approaches in large-scale storage systems.

Their rules incorporate both spatial and temporal information. However, using the pixel or frequency domain instead of the feature domain is quite inefficient in real applications. The second category tries to use the hashing representation with deep learning networks to replace pixels. The most representative work is to use the convolutional features to derive the hash. For example, supervised deep hash approach constructs binary hash codes from labeled data for large-scale image search [26]. Radford et al. [27] develop a class of conventional neural networks (CNNs) called deep convolutional generative adversarial networks (DCGANs) learning hierarchy of features to represent general image. Wu et al. [28] proposed the Procrustean Approach addressing the problem of learning similarity-preserving binary codes [29] for efficient similarity search in large-scale image collections [30]. Compact Scalable Hash [30] from Deep Learning Features [31] Aggregation by Feng et al. developed a novel hash scheme which is scalable and robust to typical CDN induced transcoding and manipulations [32]. Though this method utilized deep learning Visual Geometry Group (VGG) network method to obtain some performance improvements, the cross-entropy loss function is in essence unsuitable for the video de-duplication task. And due to that there are no comparisons and subjective frame shown, it is not convincing enough to claim good video de-duplication results. Xu et al. [33] present a cache design namely DeepCache with deep learning inference. It exploits temporal locality in videos to reuse the cached information. Although these work obtain a good performance by CNN, they do not mine the potential of more suitable loss function fitting the de-duplication case. We explain our binary-tree embedded triplet loss CNN comprehensively in next Section.

3. Triplet loss network for binary hashing model

The overall framework of the proposed scalable hash scheme is illustrated in Fig. 2. It consists of two components: (a) Triplet loss network feature representation generation in Section 3.1; (b) The fisher vector (FV) feature aggregation using fisher vector for generating scalable hash in Section 3.2.

3.1. Triplet loss network

In our work, VGG11, as the key method in general to derive accurate feature representation, is used to generate our convolutional features. The main novelty of the network is to use the triplet loss [34] to replace the cross-entropy loss to make the feature more distinguishable and more suitable for the video de-duplication application. In order to obtain a reliable triplet loss embedded network, three basic constraints shall be applied to choose the triplets: First, this loss function should make sure that an anchor feature x_j^a is as close with the same type samples x_j^p as possible. In addition, what is also relatively critical is that anchor feature x_j^a is as far with other types samples x_j^n. Moreover, the distance between anchor feature x_j^a with positive feature x_i^p should be less than the one between anchor feature x_i^a and negative feature x_j^n at least margin distance.

$$\|F(x_j^a) - F(x_j^p)\|_2^2 + m < \|F(x_j^a) - F(x_j^n)\|_2^2, \tag{1}$$

where F defines the VGG11 network. According to these conditions, we express the triplet loss in (1).

To monitor the training process effectively, we keep the fully connected (FC) layers [35] during the training of the VGG11 network. We employ not only the triplet loss function but also the accuracy indicator computed in (2) observing the features from FC layers.

The accuracy definition:

$$\alpha_{\iota_{ap},\iota_{an}}(i) = \frac{\sum_i^N \Phi(\iota_{ap}(i) - \iota_{an}(i) - \xi)}{\Psi_{a,p,n}} \tag{2}$$

where ι_{ap} is the L_2 distance between positive pairs which mean anchors and positive samples pairs. ι_{an} is the L_2 distance between negative pairs which mean anchors and negative samples pairs. ξ is the least margin distance between ι_{ap} and ι_{an} which makes sure the base judgment is correct. Φ defines the amount of number which is greater than 0. $\Psi_{a,p,n}$ is the total number of triplet pairs in this batch. Here we set ξ equal 1.0.

Besides of the network training, it is also very important to choose the appropriate layer to generate the features for aggregation. Hence, we drop out the full connected layers behind the last max-pooling layer as the last blue cube shown in Fig. 2 so that we can carry out the multiple dimensions features as the PCA [36] input data directly. We train our triplets dataset stated in Section 4 for 81 epochs. And the best accuracy of validating checkpoint is 92.08% occurring on the 31*st* epoch.

3.2. FV aggregation for scalable hash

After obtaining the triplet loss VGG11 model, we trained the PCA and GMM model respectively with 1000 frames in 1000 iterations as well. Then the fisher vector is used to aggregate the features to generate the scalable hash. The specific steps we process are stated as follows.

(1) Derive features whose dimensions are $512 \times 10 \times 5$ from the VGG triplet loss network;
(2) Apply principle component analysis (PCA) to the reshaped features for each frame to generate a kd dimensional vector;
(3) The Gaussian mixture model (GMM) is used to extract the nc main component.

where nc is the number of main components in GMM. Now we get a $2 \times kd \times nc$ fisher vector [37] (involving reshape process).

Note that the fisher vector selection is relatively vital for scalable hash here. We already have the eigenvalues from trained PCA models and covariance weights from trained GMM models. We first normalize the eigenvalues of PCA, then apply a method of extracting main features with the equation below:

$$\Xi^{\tau}(m,n) = \alpha \times \varepsilon(m) + (1-\alpha) \times W(n) \tag{3}$$

Here the $\varepsilon(m)$ is the normalized eigenvalues matrix [38] of PCA with m components. The W(n) represents the normalized covariance weights matrix of GMM with n components. α means the tuning coefficient ranging from 0 to 1. We choose the first τ high value of Ξ as the main features constituting scalable hash code. τ is the length of hash code, such as $32, 64, 128$ and 256 bits.

To achieve the binary hash, we apply 0 as the threshold. Specifically, assign the value of the fisher vector which is greater than 0 to be 1, otherwise define the value of the fisher vector which is less than 0 to be 0. Now we acquire the quantized binary hash code of scalable length.

Meanwhile we calculate the triplet pair distances between anchor fisher vector and positive fisher vector or negative fisher vector respectively as follows.

$$\begin{aligned} D^{\tau}_{a,p}(a,p) &= \|B^{\tau}(\eta_a(a)), \quad B^{\tau}(\eta_p(p))\|_2^2, \\ D^{\tau}_{a,n}(a,n) &= \|B^{\tau}(\eta_a(a)), \quad B^{\tau}(\eta_n(n))\|_2^2. \end{aligned} \tag{4}$$

where τ represents the bits of scalable binary hash. Then we compute the TPR under False Positive Rate(FPR)= 0 with $D^{\tau}_{a,p}$ and $D^{\tau}_{a,n}$, because this condition reflects the real recall when there is no error positive sample judgment.

We utilize scalable binary hash because of its three benefits. First, binary hashing is helpful for simplifying the calculation process through computing the hamming distance between binary hash codes. Second, scalable binary hash decreases the calculation complexity for features matching process due to its less length representation comparing to original fisher vector. This can accelerate the video de-duplication dramatically. If it is still a $2 \times kd \times nc$ original fisher vector which is represented by float numbers, we should spend many calculating resources on every frame feature. Obviously, this is too expensive to afford for time and computational burden. Third, the scalable binary hash very is flexible. When the calculation resource is limited, we can choose shorter binary hash. When there is powerful computing clusters, we can select longer binary hash code to gain better performance. This will be verified by experiments shown in Section 5.

4. Triplets generation

In this section, we will introduce how binary tree divides the dataset and generates the triplets. The training samples will be elaborated in Section 4.1. The binary-tree based generation process will be introduced in Section 4.2.

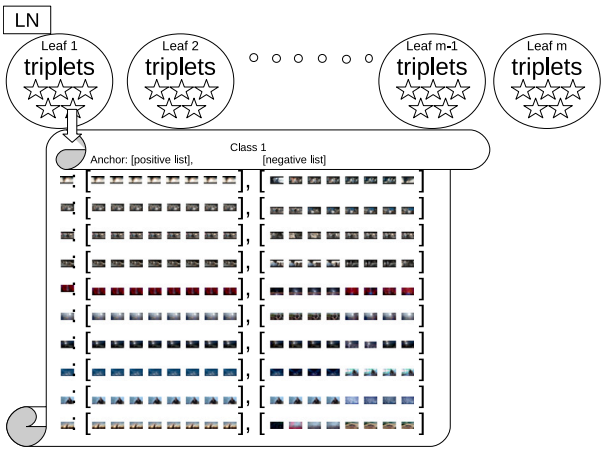


Fig. 3. Binary-Tree generates hard valuable triplets. LN represents the leaf node. Each thumbnail represents its coded frame. We employ the Binary-Tree diving the thumbnail samples according to their attributions of components. Thus we select the thumbnails of similar features as the hard triplet comprising positive samples and negative samples for anchor.

4.1. Train sequence selection

Since we should maximize the robust and learning capability of the deep learning network and models, there are 3 requirements for the dataset. First, the types of scenes from selected videos should cover widely. This principle guarantees us not to constrain the learning ability of the deep learning network and GMM Fisher Vector model. Second, large dataset size is also important to add the data source diversity for training the whole network and models. Last but not least, the quality of these videos data should be high so that we can convert them to different video versions from high quality to low quality, from high resolution to low resolution.

Owing to these principles, first of all, we collected 177.7 h original documentary [39] videos which can provide varieties of scenarios. They are all $1080p$ resolutions with $QP23$. Actually, refer to all kinds of videos (involving different resolutions and contents) uploaded from different users, our target is to de-duplicate the same contents though they may be with different resolutions on CDN. To simulate this real application, we transcode the original video to other 6 versions of REQP videos including $720p$ $QP23$, $720p$ $QP28$, $720p$ $QP33$, $480p$ $QP23$, $480p$ $QP28$, $480p$ $QP33$ with the ffmpeg that has been built in GPU acceleration. In this way, for the same video content, we can have 7 versions in total. We then sample the 7 versions videos to 320×180 frames and 16×9 thumbnails. The 320×180 frames will be used for training the triplet network, while the thumbnails will be used for training the GMM model. We process them with the rate of 2 frames (or thumbnails) per second such that 1-h video exchanges to 7200 frames (or thumbnails).

4.2. Binary tree based triplets generation

The essential problem for triplet loss function is that how to obtain the hard and valuable triplets samples. This influences the robust and efficiency of the network we construct directly. In this work, as drawn in Fig. 3, we choose Binary-Tree [40] to split the huge video frames dataset to analyze out valuable and hard triplets. Since binary-tree can assign the frames with similar textures to the same leaf node, deriving negative pairs from the same leaf node is beneficial for generating hard triplets. In the following, we use two steps to explain how to generate triplets in detail.

We use thumbnails proposed in Section 4.1 as input data for producing triplets with Binary-Tree. We first convert thumbnails to grayscale images. Then, we reshape thumbnails from 16×9 to 1×144. Next, we

Table 1
Scalable hash code TPR comparison between CP and our BTF when FPR = 0.

CP TPR | our BTF TPR	32 bits	64 bits	128 bits	256 bits
kd 16 - - — kd 16, nc 24, α 0.01	0.3216|**0.7454**	0.5171|**0.8835**	0.7140|**0.9190**	0.8073|**0.9445**
kd 16 - - — kd 16, nc 24, α 0.2	0.3216|**0.7290**	0.5171|**0.8607**	0.7140|**0.9069**	0.8073|**0.9461**
kd 16 - - — kd 16, nc 24, α 0.4	0.3216|**0.5407**	0.5171|**0.7802**	0.7140|**0.9095**	0.8073|**0.9399**
kd 16 - - — kd 16, nc 24, α 0.8	**0.3216**|0.2473	0.5171|**0.6904**	0.7140|**0.8785**	0.8073|**0.9302**
kd 16 - - — kd 16, nc 24, α 0.99	**0.3216**|0.2614	0.5171|**0.6797**	0.7140|**0.8630**	0.8073|**0.9188**
kd 16 - - — kd 16, nc 48, α 0.01	0.3216|**0.6928**	0.5171|**0.8342**	0.7140|**0.9188**	0.8073|**0.9409**
kd 16 - - — kd 16, nc 48, α 0.2	0.3216|**0.7590**	0.5171|**0.8685**	0.7140|**0.9180**	0.8073|**0.9421**
kd 16 - - — kd 16, nc 48, α 0.4	0.3216|**0.7316**	0.5171|**0.8457**	0.7140|**0.9090**	0.8073|**0.9338**
kd 16 - - — kd 16, nc 48, α 0.8	0.3216|**0.7371**	0.5171|**0.8516**	0.7140|**0.8904**	0.8073|**0.9295**
kd 16 - - — kd 16, nc 48, α 0.99	0.3216|**0.7369**	0.5171|**0.8519**	0.7140|**0.8878**	0.8073|**0.9278**
kd 24 - - — kd 24, nc 24, α 0.01	0.1319|**0.6440**	0.3359|**0.7007**	0.5569|**0.8392**	0.7323|**0.9459**
kd 24 - - — kd 24, nc 24, α 0.2	0.1319|**0.5809**	0.3359|**0.7497**	0.5569|**0.8697**	0.7323|**0.9557**
kd 24 - - — kd 24, nc 24, α 0.4	0.1319|**0.4935**	0.3359|**0.7559**	0.5569|**0.8721**	0.7323|**0.9590**
kd 24 - - — kd 24, nc 24, α 0.8	0.1319|**0.4916**	0.3359|**0.7288**	0.5569|**0.8859**	0.7323|**0.9595**
kd 24 - - — kd 24, nc 24, α 0.99	0.1319|**0.4883**	0.3359|**0.7347**	0.5569|**0.8478**	0.7323|**0.9569**
kd 24 - - — kd 24, nc 48, α 0.01	0.1319|**0.6592**	0.3359|**0.7835**	0.5569|**0.9092**	0.7323|**0.94**
kd 24 - - — kd 24, nc 48, α 0.2	0.1319|**0.5526**	0.3359|**0.7673**	0.5569|**0.8845**	0.7323|**0.9516**
kd 24 - - — kd 24, nc 48, α 0.4	0.1319|**0.2169**	0.3359|**0.5669**	0.5569|**0.8447**	0.7323|**0.9442**
kd 24 - - — kd 24, nc 48, α 0.8	**0.1319**|0.1049	0.3359|**0.5921**	0.5569|**0.8259**	0.7323|**0.9333**
kd 24 - - — kd 24, nc 48, α 0.99	**0.1319**|0.0966	0.3359|**0.5923**	0.5569|**0.8202**	0.7323|**0.9333**

concatenate resized thumbnails to become a large thumbnail block with $N \times 144$ in memory. N is the total number of thumbnails. Since every thumbnail has 144 dimensions after reshaping, the number of dimensions is too large to calculate for the next step. PCA is used to perform dimension reduction. Assume that K is the amount of main feature dimensions we want to keep. We can now get the $N \times K$ thumbnails matrix as Binary-Tree input data matrix. During the construction of the binary tree, we also assign the thumbnail with an index to indicate which video it comes from.

Since then, we can get the valuable and hard triplets from the generated Binary-Tree. Initially, we classify nodes in a leaf through different content videos. So we aggregate all the frames from an identical content video into a class together. After sorting frames in one class by ascending display order, for one frame, we pick out all other frames with the same display order as its positive samples. However, the quality and difficulty of negative pairs determines the triplets' value for the training. On the one hand, Binary-Tree assembles the near feature frames together. On the other hand, due to the strong correlation on time dimension, we should consider it into the negative pair's decision. So we should choose the neighbors as near as possible to ensure the difficulty. Whereas they may be actually identical frames if it is too close. Therefore here we set a least base time threshold which can assist to avoid the above problem. Then we search bi-directions involving front and back basing on the threshold to find the nearest frames. Once we collect enough negative samples we will stop the search. Fig. 4 compares the normal triplet with Binary-Tree splitting triplet, which obvious shows that the triplets generated by the binary-tree are more valuable and harder to learn.

5. Experimental results

In this section, we will first introduce the experimental results of the overall framework in Section 5.1. Then we will show the influences of the various aggregation parameters in Section 5.2. In Section 5.3, we will illustrate the improvements of the proposed algorithm with a few subjective samples.

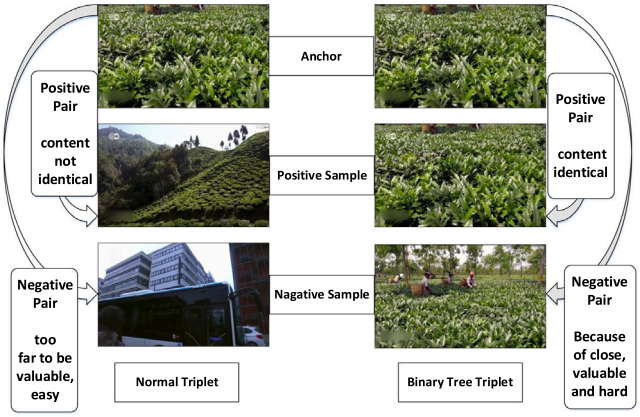
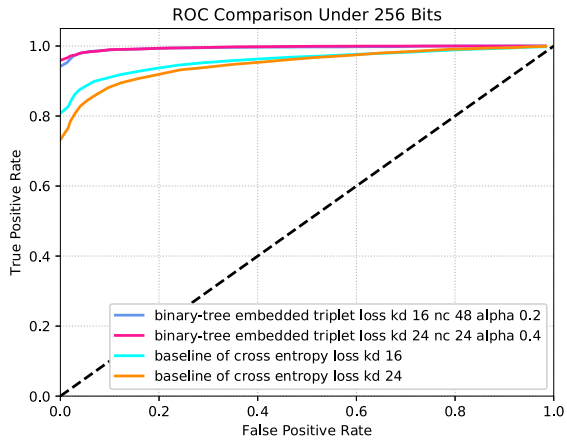


Fig. 4. Comparison between Normal triplet and Binary-Tree triplet. The binary tree triplet exhibits similar features on its positive sample and negative sample. Especially for the negative sample, due to the close distance with anchor, it provides the network with the difficult case and improve its robustness. But normal triplet shows far distance with anchor. Therefore, it is easy to be learned by VGG11.

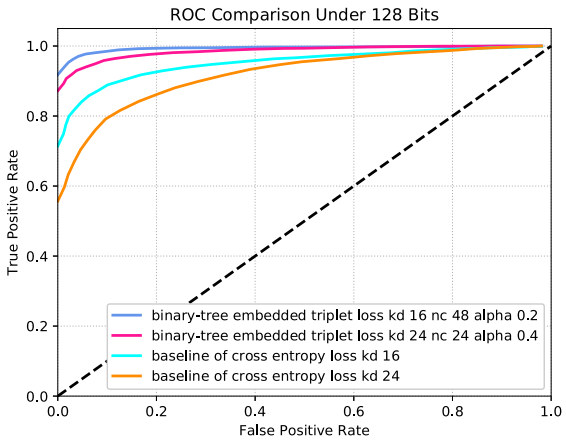
5.1. Overall framework test

To demonstrate the effectiveness of the proposed BTF approach, we compare it with the cross-entropy loss network combined with PCA (CP). To be more specific, the VGG11 model pre-trained on ImageNet using cross-entropy loss combined with PCA is used as the anchor. The VGG11 model trained on our training set using triplet less combined with scalable hash from fisher vector is used as the test. We test two cases with $kd = 16$ and $kd = 24$ for both the anchor and the proposed algorithm. We also test two cases with $nc = 24$ and $nc = 48$ for GMM. The α is set as 0.2 in this experiment. We test $32, 64, 128$ and 256 as the numbers of scalable hash bits.

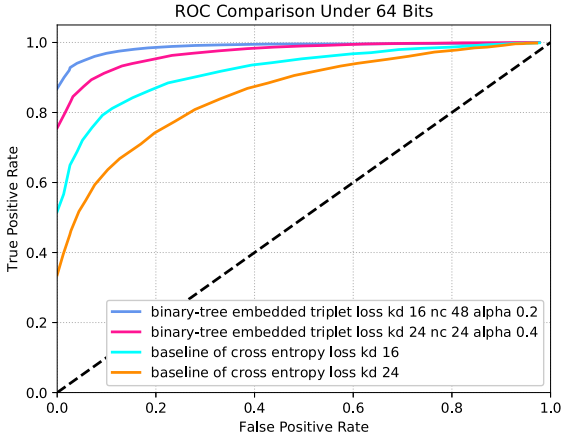
Table 1 shows the comparison between the proposed BTF approach and the CP method. From Table 1, we can see that the proposed BTF algorithm outperforms the CP method significantly in all the test cases. When kd equals 16, our BTF exhibits the best performance of 0.9461 TPR under nc 24, α 0.2 and 256 bits. It is 0.1388 higher than the



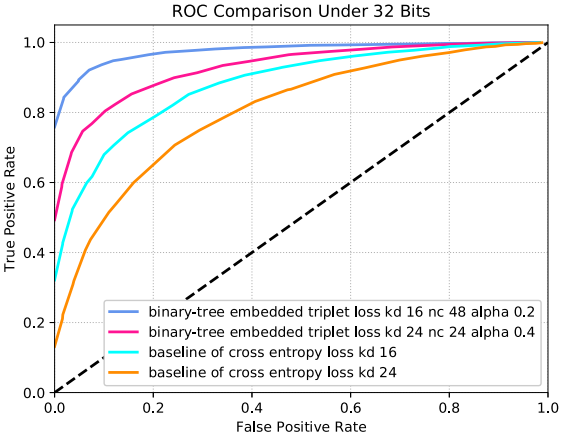
(a) ROC comparison results under hash code of 256 bits



(b) ROC comparison results under hash code of 128 bits



(c) ROC comparison results under hash code of 64 bits



(d) ROC comparison results under hash code of 32 bits

Fig. 5. Comparison results of scalable hash ROC between CP method and proposed BTF approach. We test the ROC experiments under scalable bits of 32, 64, 128 and 256 individually corresponding to (d), (c), (b) and (a). Our BTF scheme shows an overwhelming advantage of ROC compared to CP method on each scalable hash bits. Typically, the ROC curves of BTF hold a high level momentum of TPR under 128 bits and 256 bits. They are above TPR of 0.8721 and 0.9421 respectively.

Table 2
Scalable hash code TPR of BTF comparison between kd 16 and kd 24 when FPR = 0.

kd 16 | kd 24	32 bits	64 bits	128 bits	256 bits
nc 24, α 0.01	**0.75**|0.64	**0.88**|0.70	**0.92**|0.84	0.94|**0.95**
nc 24, α 0.2	**0.73**|0.58	**0.86**|0.75	**0.91**|0.87	0.95|**0.96**
nc 24, α 0.4	**0.54**|0.49	**0.78**|0.76	**0.91**|0.87	0.94|**0.96**
nc 24, α 0.8	0.25|**0.49**	0.69|**0.73**	0.88|**0.89**	0.93|**0.96**
nc 24, α 0.99	0.26|**0.49**	0.68|**0.73**	**0.86**|0.85	0.92|**0.96**
nc 48, α 0.01	**0.69**|0.66	**0.83**|0.78	**0.92**|0.91	0.94|0.94
nc 48, α 0.2	**0.76**|0.55	**0.87**|0.77	**0.92**|0.88	0.94|**0.95**
nc 48, α 0.4	**0.73**|0.22	**0.85**|0.57	**0.91**|0.84	0.93|**0.94**
nc 48, α 0.8	**0.74**|0.10	**0.85**|0.59	**0.89**|0.83	0.93|0.93
nc 48, α 0.99	**0.74**|0.10	**0.85**|0.59	**0.89**|0.82	0.93|0.93

best result 0.8073 of CP. When kd equals 24, our BTF shows the best performance of 0.9595 TPR under nc 24, α 0.8 and 256 bits. It is 0.2272 higher than the best result 0.7323 of CP. The experimental results obviously demonstrate the effectiveness of the proposed algorithm.

In addition, as we can see from Table 1, the proposed algorithm shows consistently better results along with the increase of the bits spent on the hash representation since more bits can keep more information. It should be noted that the TPR of the proposed BTF approach is always as high as over 0.9 under FRP equals 0. While the performance is better, it will also lead to high complexity in deduplication compared with the 32 bits case. In the 32 bits case, the TPR of our BTF framework is significantly 0.4374 and 0.5273 higher than the one of CP under kd 16 and kd 24 respectively. Different bits show different trade-offs between the performance and the complexity. We can choose the suitable case according to our requirement in various applications.

We also compare the integrated TPR trend as FPR increases in the receiver operating characteristic (ROC) [41] curves, as described in Fig. 5. Due to the limited space, we select nc 48α 0.2 and nc 24α 0.4

Table 3
Scalable hash code TPR of BTF comparison between nc 24 and nc 48 when FPR = 0.

nc 24 | nc 48	32 bits	64 bits	128 bits	256 bits
kd 16, α 0.01	**0.75**|0.69	**0.88**|0.83	0.92|0.92	0.94|0.94
kd 16, α 0.2	0.73|**0.76**	0.86|**0.87**	0.91|**0.92**	**0.95**|0.94
kd 16, α 0.4	0.54|**0.73**	0.78|**0.85**	0.91|0.91	**0.94**|0.93
kd 16, α 0.8	0.25|**0.74**	0.69|**0.85**	0.88|**0.89**	0.93|0.93
kd 16, α 0.99	0.26|**0.74**	0.68|**0.85**	0.86|**0.89**	0.92|**0.93**
kd 24, α 0.01	0.64|**0.66**	0.70|**0.78**	0.84|**0.91**	**0.95**|0.94
kd 24, α 0.2	**0.58**|0.55	0.75|**0.77**	0.87|**0.88**	**0.96**|0.95
kd 24, α 0.4	**0.49**|0.22	**0.76**|0.57	**0.87**|0.84	**0.96**|0.94
kd 24, α 0.8	**0.49**|0.10	**0.73**|0.59	**0.89**|0.83	**0.96**|0.93
kd 24, α 0.99	**0.49**|0.10	**0.73**|0.59	**0.85**|0.82	**0.96**|0.93

Table 4
Scalable hash code TPR of BTF comparison in different α of 0.01, 0.2, 0.4, 0.8 and 0.99 when FPR = 0.

	32 bits	64 bits	128 bits	256 bits
kd 16, nc 24	**0.75**|0.73|0.54|0.25|0.26	**0.88**|0.86|0.78|0.69|0.68	**0.92**|0.91|0.91|0.88|0.86	0.94|**0.95**|0.94|0.93|0.92
kd 16, nc 48	0.69|**0.76**|0.73|0.74|0.74	0.83|**0.87**|0.85|0.85|0.85	**0.92**|**0.92**|0.91|0.89|0.89	**0.94**|**0.94**|0.93|0.93|0.93
kd 24, nc 24	**0.64**|0.58|0.49|0.49|0.49	0.70|0.75|**0.76**|0.73|0.73	0.84|0.87|0.87|**0.89**|0.85	0.95|**0.96**|**0.96**|**0.96**|**0.96**
kd 24, nc 48	**0.66**|0.55|0.22|0.10|0.10	**0.78**|0.77|0.57|0.59|0.59	**0.91**|0.88|0.84|0.83|0.82	0.94|**0.95**|0.94|0.93|0.93

as representations of kd 16 and 24 respectively for the proposed BTF approach. The ROC curves also show that the proposed BTF approach outperforms the CP method significantly.

5.2. The influences of the various aggregation parameters

Table 2 illustrates the compared results of our BTF between kd 16 and kd 24. The TPR of kd 16 shows a better TPR under all nc and α combinations of 32, 64 and 128 bits except nc 24 with α 0.8 or 0.99. Especially, the TPR difference can be as high as 0.64 under nc 48 and α 0.8. However there are 7 TPR cases where kd 24 in 256 bits shows slightly better performance compared with kd 16. The results account for that TPR does not follow a linear relationship with kd. The larger kd means that there are more primary components extracted from $512 \times 10 \times 5$ convolutional features. It can represent larger amount of information more precisely. However, it also requires the hash code supply bigger capacity loading the abundant convolutional feature information. If the hash code only provide a few bits for expression, the hash from the fisher vector of smaller kd (kd 16) fits the features better than the one from the fisher vector of larger kd (kd 24). As shown in Table 2, the hash from kd 16 adapts to the features better under bits of 32, 64 and 128 whereas the hash from kd 24 fits better under bits of 256. The performance is in accordance with the analysis.

The exceptions of individual cases under nc 24 with α 0.8 or 0.99 demonstrate the sensitivity of kd on different convolutional features. Not all features have similar amount of information. The high variance or dense textures of features involves more details than flat features. The kd selection can be flexible on these cases. Meanwhile we find out that the nc is also vital for hash representation. For instance, the nc 48α 0.8 is in accordance with the analyzed theory.

The comparison results of using different ncs are shown in Table 3. The best TPR from nc 24 achieving 0.96 is slightly higher than 0.95 from nc 48. The hashes from nc 24 outperform ones from nc 48 on 7 cases under 256 bits. The TPR from nc 48 exceeds the one from nc 24 on 6 and 5 rows under 64 and 128 bits, respectively. They execute even performance under 32 bits. Indeed, the smaller value of nc aggregates the components into less classifications while the larger one has the reversal effect in fisher vector. TPR of nc 48 performing better under 64 and 128 bits compared with nc 24 demonstrates that 64 and 128 bits are the appropriate quantity for playing a role on the cluster effect from nc 48. More bits like 256 may include more redundancy information from unimportant features giving rise to inference for matching hash in de-duplication. Opposed to this, less bits like 64 loses a few main clusters of features aggregated by nc 48 leading to lack of representation. Hence, it is not definite larger nc or smaller one performing better. The TPR does not monotonically increases with nc so that it is not linear as well. We should select an adaptive nc united with kd consisting of fisher vector with $2 \times kd \times nc$ dimensions fitting the convolutional features appropriately.

Table 4 compares the TPRs of the proposed BTF method in a variety of α values when FPR equals 0. The TPRs from α 0.01 and α 0.2 surpass the one from other α values on 9 and 7 cases individually. The best hash from α 0.01 obtains TPR 0.94 as the ones from α 0.2, 0.4, 0.8 and 0.99 all reach TPR 0.96 under 256 bits. These results demonstrate that different α values impact the TPR dramatically as well. As stated in (3), α is a tunable coefficient for choosing the most representative features from fisher vector. The smaller α lowers the scanning priority of kd components from PCA while improves the one of nc clusters from GMM in a fisher vector. α 0.01 and α 0.2 assistant tuning the priority of selecting τ bits from main features extracted out by fisher vector applicably.

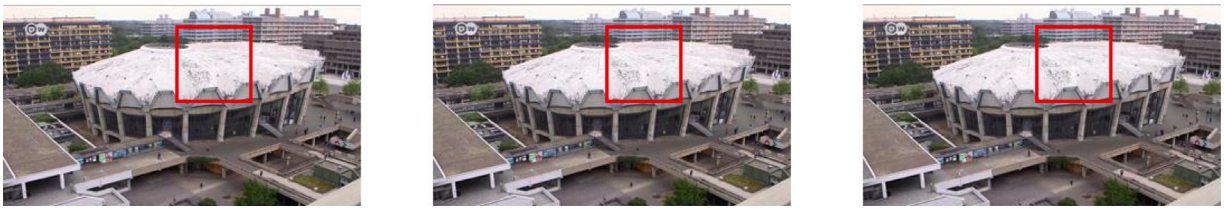
5.3. Analysis of subjective samples

We display the comparison of subjective samples between BFT and CP approaches in Fig. 6 and Fig. 7, respectively. The samples with the same background are hard for CNN to learn. For instance, as shown in Fig. 6, under kd 16, we test the anchor with REQP of $1920 \times 1080qp23$, positive sample with REQP of $1280 \times 720qp23$ and negative sample with REQP of $854 \times 480qp33$. Since the environment of the samples is identical, it is difficult to learn the difference. But the area of red block moves an angle, BFT figures out the variation depending on the base trained by triplets loss. And it applies the configuration of nc 24, α 0.01 and 32 bits generating hash which represents the features excellently. Finally BFT discards the positive sample as duplication and save the negative sample correctly. However, CP recognizes the negative sample as the replication of anchor and deletes it by mistake.

The same situation occurs in Fig. 7 under kd 24 as well. The anchor, positive sample, and negative sample are with REQP of $1920 \times 1080qp23$, $1280 \times 720qp23$ and $854 \times 480qp28$, respectively. The negative sample just adjusts a bit of textures and micro angle in red block as illustrated in Fig. 7. BFT captures the slight difference with the 128 bits hash produced by model of nc 48, α 0.2. Then BFT de-duplicates the positive

(a) anchor of $1920 \times 1080qp23$ (b) positive sample of $1280 \times 720qp23$ (c) negative sample of $854 \times 480qp33$

Fig. 6. Comparison between BTF and CP of the samples under kd 16. Our BTF de-duplicates the positive sample and reserves the negative sample successfully with setting of nc 24, α 0.01 and 32 bits hash while CP deletes the negative one by mistake.



(a) anchor of $1920 \times 1080qp23$ (b) positive sample of $1280 \times 720qp23$ (c) negative sample of $854 \times 480qp28$

Fig. 7. Comparison between BTF and CP of the samples under kd 24. Our BTF de-duplicates the positive sample and reserves the negative sample successfully with setting of nc 48, α 0.2 and 128 bits hash while CP removes the negative sample accidentally.

sample and reserves the negative one while CP matches the negative sample with anchor and removes it. According to these analysis above, we consider BFT outperforms CP significantly.

6. Conclusion

Prosperous development on multiple media big data producing, transmission and depleting have occupied the massive memory and storage in all kinds of devices, network systems, and data clusters of clouds. Improving the theory and algorithm to recognize the duplications of multiple media on every layer is an essential and urgent topic for transmitting and caching media big data quickly and efficiently. In this paper, we propose a distinct video de-duplication framework involving a triplet loss network learning convolutional features that do not vary as codec and rates. Furthermore, we generate the scalable hash from FV aggregation of the convolutional features. Especially, we design a novel binary tree embedded algorithm to generate hard triplet samples for triplet loss function feeding VGG network more robustly. Experimental results show that this embedding triplet loss function framework offers a strong and stable network system to process video de-duplication efficiently.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The work is partially supported by a grant from NSF under award 1747751.

References

[1] Iraj Sodagar, The mpeg-dash standard for multimedia streaming over the internet, IEEE MultiMedia 18 (4) (2011) 62–67.
[2] M Christopher, Mpeg-dash vs. apple hls vs. microsoft smooth streaming vs. adobe hds, 2015.
[3] Alex Zambelli, IIS smooth streaming technical overview, Microsoft Corp. 3 (2009) 40.
[4] Ronald Rivest, The MD5 Message-Digest Algorithm, Technical Report, 1992.
[5] Bo Liu, Zhu Li, Linjun Yang, Meng Wang, et al., Real-time video copy-location detection in large-scale repositories, IEEE MultiMedia 18 (3) (2011) 22–31.
[6] Karen Simonyan, Andrew Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.
[7] Thierry Bouwmans, El Hadi Zahzah, Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance, Comput. Vis. Image Underst. 122 (2014) 22–34.
[8] Douglas Reynolds, Gaussian mixture models, Encycl. Biom. (2015) 827–832.
[9] Philippe-Henri Gosselin, Naila Murray, Hervé Jégou, Florent Perronnin, Revisiting the fisher vector for fine-grained classification, Pattern Recognit. Lett. 49 (2014) 92–98.
[10] W. Jia, L. Li, Z. Li, S. Zhao, S. Liu, Triplet loss feature aggregation for scalable hash, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 1918–1922.
[11] Reuven Y Rubinstein, Dirk P Kroese, The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning, Springer Science & Business Media, 2013.
[12] Julio Alejandro Di Rienzo, Adolfo Washington Guzmán, Fernando Casanoves, A multiple-comparisons method based on the distribution of the root node distance of a binary tree, J. Agric. Biol. Environ. Stat. 7 (2) (2002) 129–142.
[13] Atul Katiyar, Jon B Weissman, Videdup: An application-aware framework for video de-duplication, in: HotStorage, 2011.
[14] Sakrapee Paisitkriangkrai, Tao Mei, Jian Zhang, Xian-Sheng Hua, Scalable clip-based near-duplicate video detection with ordinal measure, in: Proceedings of the ACM International Conference on Image and Video Retrieval, ACM, 2010, pp. 121–128.
[15] Spencer Greene, Transparent caching of repeated video content in a network, in: Google Patents, US Patent 7, 770, 198.
[16] Yifeng Zheng, Xingliang Yuan, Xinyu Wang, Jinghua Jiang, Cong Wang, Xiaolin Gui, Enabling encrypted cloud media center with secure deduplication, in: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ACM, 2015, pp. 63–72.
[17] Yifeng Zheng, Xingliang Yuan, Xinyu Wang, Jinghua Jiang, Cong Wang, Xiaolin Gui, Toward encrypted cloud media center with secure deduplication, IEEE Trans. Multimedia 19 (2) (2017) 251–265.
[18] Fatema Rashid, Ali Miri, Isaac Woungang, Proof of storage for video deduplication in the cloud, in: 2015 IEEE International Congress on Big Data, IEEE, 2015, pp. 499–505.
[19] Fatema Rashid, Ali Miri, Isaac Woungang, A secure video deduplication scheme in cloud storage environments using H. 264 compression, in: 2015 IEEE First International Conference on Big Data Computing Service and Applications, IEEE, 2015, pp. 138–146.
[20] Hongyang Yan, Xuan Li, Yu Wang, Chunfu Jia, Centralized duplicate removal video storage system with privacy preservation in IoT, Sensors 18 (6) (2018) 1814.

[21] Xuan Li, Jie Lin, Jin Li, Biao Jin, A video deduplication scheme with privacy preservation in IoT, in: International Symposium on Computational Intelligence and Intelligent Systems, Springer, 2015, pp. 409–417.
[22] John Edward Gerard Matze, System and method for data deduplication, Google Patents, US Patent 8, 205, 065, 2012.
[23] Emmanuel Barajas Gonzalez, Shaun E Harrington, David C Reed, Max D Smith, Efficient video data deduplication, Google Patents, US Patent 9, 646, 017, 2017.
[24] Wen Xia, Hong Jiang, Dan Feng, Fred Douglis, Philip Shilane, Yu Hua, Min Fu, Yucheng Zhang, Yukun Zhou, A comprehensive study of the past, present, and future of data deduplication, Proc. IEEE 104 (9) (2016) 1681–1710.
[25] Jaehong Min, Daeyoung Yoon, Youjip Won, Efficient deduplication techniques for modern backup operation, IEEE Trans. Comput. 60 (6) (2011) 824–840.
[26] Xiaofang Wang, Yi Shi, Kris M Kitani, Deep supervised hashing with triplet labels, in: Asian Conference on Computer Vision, Springer, 2016, pp. 70–84.
[27] Alec Radford, Luke Metz, Soumith Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 2015, arXiv preprint arXiv:1511.06434.
[28] Zuxuan Wu, Ting Yao, Yanwei Fu, Yu-Gang Jiang, Deep learning for video classification and captioning, 2016, arXiv preprint arXiv:1609.06782.
[29] Mohammad Norouzi, David M Blei, Minimal loss hashing for compact binary codes, in: Proceedings of the 28th International Conference on Machine Learning, ICML-11, Citeseer, 2011, pp. 353–360.
[30] Ruimao Zhang, Liang Lin, Rui Zhang, Wangmeng Zuo, Lei Zhang, Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification, IEEE Trans. Image Process. 24 (12) (2015) 4766–4779.
[31] Refik Can Malli, Mehmet Aygun, Hazim Kemal Ekenel, Apparent age estimation using ensemble of deep learning models, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016, pp. 9–16.
[32] Shan Feng, Zhu Li, Yiling Xu, Jun Sun, Compact scalable hash from deep learning features aggregation for content de-duplication, in: Multimedia Signal Processing (MMSP), 2017 IEEE 19th International Workshop on, IEEE, 2017, pp. 1–5.
[33] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, Xuanzhe Liu, DeepCache: Principled cache for mobile deep vision, in: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, ACM, 2018, pp. 129–144.
[34] De Cheng, Yihong Gong, Sanping Zhou, Jinjun Wang, Nanning Zheng, Person re-identification by multi-channel parts-based cnn with improved triplet loss function, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1335–1344.
[35] Tara N Sainath, Oriol Vinyals, Andrew Senior, Haşim Sak, Convolutional, long short-term memory, fully connected deep neural networks, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2015, pp. 4580–4584.
[36] Ian Jolliffe, Principal component analysis, in: International Encyclopedia of Statistical Science, Springer, 2011, pp. 1094–1096.
[37] Florent Perronnin, Jorge Sánchez, Thomas Mensink, Improving the fisher kernel for large-scale image classification, in: European Conference on Computer Vision, Springer, 2010, pp. 143–156.
[38] J. Li, L. Ji, Adjusting multiple testing in multilocus analyses using the eigenvalues of a correlation matrix, Heredity 95 (3) (2005) 221.
[39] DW Documentary, DW documentary, 2018, https://www.youtube.com/channel/UCW39zufHfsuGgpLviKh297Q, (Accessed Oct 2018).
[40] Michael Greenspan, Mike Yurick, Approximate kd tree search for efficient ICP, in: Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings, IEEE, 2003, pp. 442–448.
[41] Caren M Rotello, Evan Heit, Chad Dubé, When more data steer us wrong: Replications with the wrong dependent measure perpetuate erroneous conclusions, Psychon. Bull. Rev. 22 (4) (2015) 944–954.