J. Vis. Commun. Image R. 85 (2022) 103501

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

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

Full length article

Deep multi-query video retrieval ☆

Enver Akbacak [a], Cabir Vural [b,*]

[a] Engineering Faculty, Computer Engineering, Haliç University, Güzeltepe Mahallesi, 15 Temmuz Şehitler Caddesi, No: 15 34060 Eyüp/Istanbul, Turkey
[b] Engineering Faculty, Department of Electrical and Electronics Engineering, Marmara University, Aydinevler Mah., 34854 Maltepe/Istanbul, Turkey

ARTICLE INFO

Keywords:
Video hashing
Pareto optimization
Multi-query video retrieval

ABSTRACT

Video retrieval methods have been developed for a single query. Multi-query video retrieval problem has not been investigated yet. In this study, an efficient and fast multi-query video retrieval framework is developed. Query videos are assumed to be related to more than one semantic. The framework supports an arbitrary number of video queries. The method is built upon using binary video hash codes. As a result, it is fast and requires a lower storage space. Database and query hash codes are generated by a deep hashing method that not only generates hash codes but also predicts query labels when they are chosen outside the database. The retrieval is based on the Pareto front multi-objective optimization method. Re-ranking performed on the retrieved videos by using non-binary deep features increases the retrieval accuracy considerably. Simulations carried out on two multi-label video databases show that the proposed method is efficient and fast in terms of retrieval accuracy and time.

1. Introduction

Fast and efficient search from a large-scale video database has become an important issue. Video search by text queries in search engines is quite common, but it cannot be as effective as content-based video retrieval methods since video signals contain more information compared to text. In the video retrieval studies conducted so far, single videos have always been used as queries. However, by nature, a video signal is usually related to more than one semantic. Consequently, annotated videos usually have a multi-label character. In other words, videos are usually comprised of multi-video clips.

The purpose of this study is to develop a multi-query video retrieval framework that retrieves videos containing spatio-temporal features in all multiple input queries. This requires that the search database be multi-label. The study is totally different from other studies. The queries can be a single clip video or videos consisting of multi clips. That is, the queries can be single or multi label videos. In fact, this purpose can be achieved with a single video query. However, using multi queries allows the user to express his or her intention better. In addition, a query video consisting of multiple clips can be difficult to obtain and it may not be enough to express the intention of the user. Multi-query video retrieval (MQVR) remains a problem to be solved when searching for multiple semantics from an underlined video database. No retrieval study has been conducted by multiple video queries each containing different concepts.

Hash-based multimedia retrieval methods were shown to be very successful in terms of efficiency and speed. Although many hash-based image retrieval studies exist, a limited number of hash-based single-query video retrieval algorithms have been developed [1–8]. On the other hand, MQVR has not been studied yet. In this study, we develop a novel MQVR framework for the case in which query videos are related to more than one semantic. In the proposed framework, video hash codes are generated by a deep multi-label video hashing (DMVH) method that does not require the category (or equivalently class) information for queries. If the category of a query is not known, the deep learning structure obtains a category by performing prediction. Thus, query videos can be arbitrary videos as well as they can be chosen from a search database formed by using known category information for its items. The retrieval is based on the Pareto front method. Since the proposed retrieval method uses the trained DMVH for the prediction of the query labels and for the generation of the query features, it is called deep multi-query video retrieval framework whose block diagram is illustrated in Fig. 1.

In the framework, one frame is extracted from queries and database items per second. The convolution blocks of the pre-trained VGG-16 network are used for feature extraction [9]. Extracted features are then used as input for trainable layers. Bidirectional long short term memory (BLSTM) layers capture temporal dependencies across frames and contain information about whole frame sequences. Hash codes are

☆ This paper has been recommended for acceptance by Zicheng Liu.

* Corresponding author.
E-mail addresses: enverakbacak@halic.edu.tr (E. Akbacak), cabir.vural@marmara.edu.tr (C. Vural).

https://doi.org/10.1016/j.jvcir.2022.103501
Received 2 April 2020; Received in revised form 21 November 2021; Accepted 26 March 2022
Available online 11 April 2022

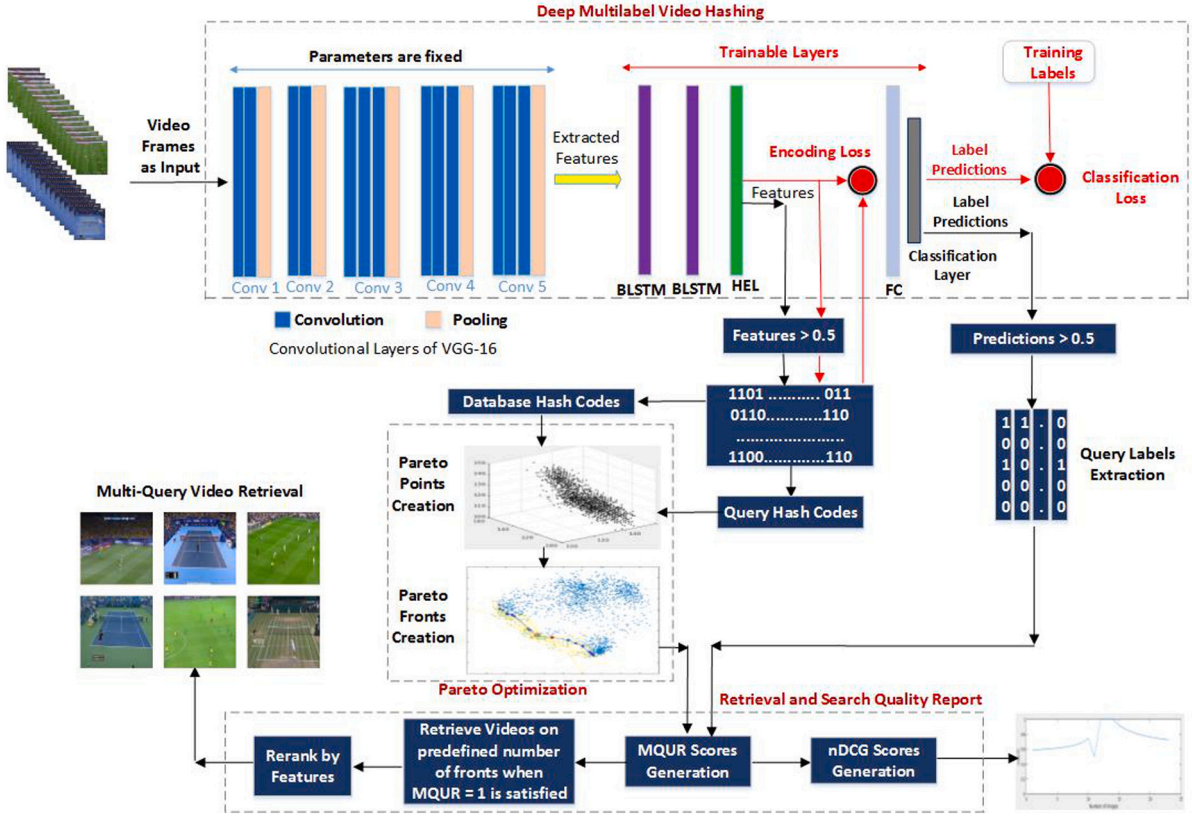


Fig. 1. Overview of the proposed DMQVR framework.

extracted at the hash encoding layer (HEL), which is a dense layer, with a sigmoid output. Two loss functions are used. The first one is the classification loss, which is a binary-cross entropy loss. The second one is defined over the HEL layer. It minimizes the squared difference between the HEL output and learned binary codes during training. In the retrieval block, Pareto points, Pareto fronts are generated and multiple-query unique relevance (MQUR) is calculated. Additionally, if query labels are not known, they are predicted at the classification layer. A video on a Pareto front with an MQUR score of one is a candidate for the retrieval. This step is followed by a reranking step in which retrieved videos are sorted in another Pareto space based on non-binary features extracted at the HEL layer.

We also prepared a graphical user interface (GUI) that combines the trained DMVH algorithm and the Pareto front method. To our best knowledge, MQVR has been investigated for the first time in this study. Simulations show that DMQVR is efficient and fast when the queries are related to different semantics.

Contributions of the study can be listed as follows:

- If labels of queries are known, DMQVR accurately retrieves videos that meet the retrieval conditions on the Pareto fronts. If the labels are not known, DMQVR predicts them.
- DMQVR supports an arbitrary number of queries.
- The proposed video hashing method is a new video hashing method.
- A deep learning-based hash codes and the Hamming distance are used instead of non-binary features and the Euclidean distance in DMQVR. Consequently, it is fast and requires lower storage space.
- Re-ranking performed on the retrieved videos by using non-binary deep features increases the retrieval accuracy considerably.

The rest of the study is organized as follows. Details of content-based MQVR, hashing, and Pareto optimality are given in Section 2. In Section 3, performance metrics used in simulations are defined. Section 4 introduces the DMQVR method. Section 5 is devoted to simulation results. This section also summarizes the GUI developed. Finally, Section 6 concludes the study highlighting the main observations.

2. Background information

Two basic tools lie at the heart of the proposed DMQVR framework: video hashing and Pareto optimality. Consequently, these basic concepts will be introduced first before presenting the DMQVR method.

2.1. Video hashing

Hashing is a transformation that maps non-binary features extracted from a signal into binary features called hash codes. The transformed space consisting of hash codes is referred to as the hash space. Hashing approximates the nearest neighbor (NN) search problem in a low dimensional space. As a result, retrieval time and memory requirements are significantly reduced.

Let the video database containing N items be denoted by $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N] \in \mathbb{R}^{M\times N}$, where $\mathbf{x}_n \in \mathbb{R}^M$ is the nth item in the database. Hashing maps each item $\mathbf{x}_n$ into a K-bit binary vector by hash functions. It actually converts high-dimensional $\mathbb{R}^M$ feature space to a lower-dimensional $\mathbb{B}^K$ hash space, where $\mathbb{B} \in \{0, 1\}$. The database after transformation is given by $\mathbf{Y} = H(\mathbf{X}) = [h_1(\mathbf{X}), h_2(\mathbf{X}), \dots, h_N(\mathbf{X})]$, where $\mathbf{Y} \in \mathbb{B}^{K\times N}$ is the hash space. Similarity distance in the initial real-valued $\mathbb{R}^M$ space should be preserved as much as possible after the mapping so that similar hash codes are assigned to semantically similar items. Similarity measure in the hash space is the Hamming distance calculated by using bitwise XOR operation between the two codes. One can use several alternatives such as cosine or Euclidean distance for the distance metric in the original feature space. Since hash codes are low dimensional and XOR operation is very fast, video retrieval in the hash space is efficient in terms of retrieval time and memory requirement.

The discriminative power of hash codes depends on the features extracted from the video signal. If a feature represents a video well, the corresponding hash code is likely to be efficient. Some methods use hand-crafted features such as Speeded-Up Robust Features (SURF) [10]. Generating appropriate hand-crafted features is a challenging task. On the contrary, deep convolutional neural networks (CNN) can generate features automatically by using frames directly as inputs. CNN based features were shown to be more efficient than hand-crafted features [11,12].

Unlike linear hashing methods that aim to learn a single projection matrix, deep hashing methods try to learn multiple non-linear projection matrices. As a result, the non-linear structure of a given data leads to efficient hash codes. The proposed hashing approach in this study generates hash codes over a classification architecture whose details are given in Section 4. Transfer learning is another powerful tool heavily used in deep learning structures. Thanks to the transfer learning, efficient hash codes can be produced with a small size training data. A complete discussion on hashing can be found in recently published surveys [13–15].

Deep networks have recently been used to generate video hash codes. Hash codes for videos were extracted by using a set of successive frames with a specific stride in [1]. Hash codes for each successive frame are then averaged to get one hash code for each video. A CNN based Siamese network was used to maximize interclass distances and minimize intraclass distances. To reveal temporal dependencies, different spatial–temporal feature pooling models such as early fusion, slow fusion and late fusion between fully connected layers by using average pooling were investigated. To exploit the structural relation between frames in a video and non-linear information among videos, a multilayer network was developed in [3]. The so-called subspace clustering was used to cluster frames according to frame semantics. Each subspace stands for a scene in a video. Thus, a binary matrix is obtained for each video. The binary matrices were then clustered into subspaces based on scenes. To measure the distance between two videos, average Hamming distances among subspaces were obtained. A self-supervised hierarchical binary encoder was proposed to generate video hash codes in [2]. The method takes both temporal and spatial information into account. Video hash codes were extracted in the binary encoder block. To improve retrieval accuracy, binary codes are forced to rebuild frames and videos in a decoder model. A two-stream CNN for video feature modeling (one for spatial information and one for temporal information) was developed in [4]. Then, these features were fused averagely in a dense layer and hash codes were produced. As the distribution of video features is more dispersed than image features, the variances between the dimensions of the video features are balanced by the balanced rotation method to enhance binary code extraction. Thus, balanced data have the same effect on Hamming distances. A supervised two-stream CNN by weights sharing was proposed in [6]. Intra-pairs that are a pair of frame sets extracted from the same video were used for training. Classification and intra-pair losses were used in optimization. Video features were extracted at the latent dense layer. To filter out hash bits that degrade classification performance, a category mask filter was added to the hash code generation module. An unsupervised one-dimensional (1-D) CNN-LSTM based encoder–decoder model was developed in [8]. A variational 1-D CNN was used as an encoder and a 1-D CNN-LSTM was used as a decoder. The encoder model takes frame-level features, and a posterior distribution is learnt for each video. Thus, the binary codes are encoded in a probabilistic way. The decoder module reconstructs the frame-level features from the hash codes.

2.2. Pareto optimization

In the single query video retrieval case, there is only objective that is to find the minimum distance between hash codes of all items in the search database and query. In the multiple-query case, on the other hand, there exist multiple objectives. Hence, MQVR can be transformed into a multi-objective optimization problem. In a multi-objective optimization problem, a single solution that simultaneously optimizes each objective usually does not exist because optimizing one objective may have an adverse effect on the other one. Consequently, a solution is based on the resolutions between the objectives. Instead of a single solution, one can talk about a set of solutions. The set of solutions that satisfy multiple objectives is called Pareto optimal solutions [16].

Suppose that we have T objectives that we want to minimize. Optimizing them simultaneously in a database containing N items is formulated as

$$\min_{\mathbf{x}} \ \mathbf{d}(\mathbf{x}) = \min_{\mathbf{x}} \ [d_1(\mathbf{x}), d_2(\mathbf{x}), \dots, d_T(\mathbf{x})] \tag{1}$$

where $\mathbf{x} = [x_1, x_2, \dots x_N] \in \Omega$ is the decision vector in the parameter space Ω and $\mathbf{d}$ is the objective vector. Components of the decision vector are obtained by calculating the dissimilarities between queries and $\mathbf{x}$ (an item in the database) by means of a suitable distance measure. Objective vectors form the objective space. Its subspace that satisfies the constraints is called a feasible space and is denoted by S. Points forming the feasible space are called Pareto points. Fig. 2(a) illustrates Pareto points in the case of two objectives for $N = 120$. Each Pareto point is created based on the dissimilarities between two queries and a sample in the search database.

The following definitions facilitate comprehension of the Pareto optimal solution. Complete details are available in [16].

Definition 1. If a point in S is not dominated by any other points for all objectives, it is called a Pareto optimal solution.

Definition 2. $\mathbf{x}$ dominates $\mathbf{y}$ if it is equal to or better in every objective, that is $d_i(\mathbf{x}) \leq d_i(\mathbf{y})$ for all i and better in at least one objective, that is $d_j(\mathbf{x}) < d_j(\mathbf{y})$ for some j.

Definition 3. Pareto optimal solutions denoted by F_1 is the first Pareto front. Similarly, the second Pareto front, F_2, is formed by removing F_1 from S and obtaining Pareto optimal solutions of the remaining set. Continuing in this manner, kth Pareto front is constructed from

$$F_k = S - (U_{j=1}^{k-1} F_j) \tag{2}$$

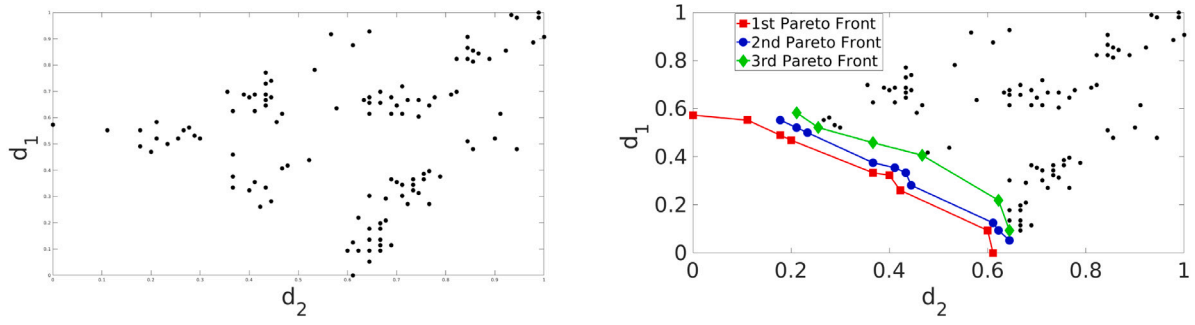
The first three Pareto fronts illustrated in Fig. 2(b) are obtained from the set of Pareto points shown in Fig. 2(a). The points on the first front are not dominated by any other points for each objective. After F_1 is removed, F_2 is created among the remaining Pareto points. Consequently, points on the second front are not dominated by any other remaining points for each objective. Continuing in this manner, the other fronts can be determined.

In this study, a deep video hashing algorithm is developed for generating hash codes. Thus, Pareto optimal solutions are obtained by finding the best trade-off between multiple Hamming distance vectors that are related to each query.

3. Performance metrics

Metrics used to evaluate retrieval results are multiple-query unique relevance (MQUR) [17] and normalized discounted cumulative gain (nDCG) scores [18]. MQUR gives the ratio of the labels of a retrieved item to the labels contained in the queries. For example, MQUR is zero if the retrieved item's label is not related to the query's labels. Let C denote the total number of classes in the database. Additionally, assume that $\boldsymbol{l}$ and $\boldsymbol{y}^i$ denote the label vectors of a retrieved item and a query q_i, where $\boldsymbol{l}, \boldsymbol{y}^i \in [0,1]^C$. MQUR of a retrieved item is obtained from Eq. (3) whose calculation is provided below.

For two binary label vectors $\boldsymbol{l}^1$ and $\boldsymbol{l}^2$, their logical conjunction and disjunction are denoted by the notations $\boldsymbol{l}^1 \cup \boldsymbol{l}^2$ and $\boldsymbol{l}^1 \cap \boldsymbol{l}^2$. jth entry of the resulting binary label vector is determined from $\max(\boldsymbol{l}_j^1, \boldsymbol{l}_j^2)$



(a) (b)

Fig. 2. Illustration of Pareto points and fronts for two objectives for $N = 120$. (a) Pareto points, (b) Pareto fronts. The axes are normalized Hamming distances.

for conjunction and $\min(l_j^i, l_j^2)$ for disjunction. Then, MQUR score of a retrieved item $\mathbf{x}$ for a given a set of queries $[q_1, \ldots, q_T]$ is defined as

$$\text{MQUR}(\mathbf{x}) = \frac{|\boldsymbol{l} \cap \boldsymbol{b}|}{|\boldsymbol{b}|} \tag{3}$$

where $\boldsymbol{b} = \boldsymbol{y}^1 \cup \boldsymbol{y}^2 \cup \cdots \cup \boldsymbol{y}^T$ and $|\boldsymbol{l}|$ is the number of non-zero elements in $\boldsymbol{l}$. Note that MQUR score can be calculated if the labels of both the queries and items in the search database are known. The database is constructed with items having labels. Label for an item outside of the database must be generated if that information is not available. One solution is to assign labels manually to such items before the retrieval process. Alternatively, the whole retrieval process can be automatized by predicting the required label with DMVH approach described in Section 4.

The quality of the search results is measured by nDCG score. It emphasizes the fact that relevant items appearing earlier in search results are more useful than non relevant items. In our case, nDCG score should be computed in a different way since more relevant results appear in the middle of the fronts in the Pareto front method [17]. Thus, we calculate nDCG scores beginning from the middle of the fronts and moving towards the right and left tails. For K retrieved items in a front, DCG score is calculated from

$$\text{DCG} = \text{MQUR}(1) + \sum_{i=2}^{K} \frac{\text{MQUR}(i)}{\log_2(i)} \tag{4}$$

where MQUR(i) denotes the MQUR score of the ith retrieved item in the front. When all the MQUR scores are equal to one, the corresponding DCG is referred to as the ideal DCG (iDCG). That is, iDCG is the best possible score. iDCG is determined by substituting 1 for all MQUR(i) in Eq. (4). Then, it is given by

$$\text{iDCG} = 1 + \sum_{i=2}^{K} \frac{1}{\log_2(i)}. \tag{5}$$

Finally, normalized DCG (nDCG) is the ratio of the DCG to the best possible score and it is computed from

$$\text{nDCG} = \frac{\text{DCG}}{\text{iDCG}}. \tag{6}$$

4. Proposed approach

The main components of the DMQVR framework are discussed in this section. The first component is DMVH that is a CNN-BLSTM network with multiple loss functions. It generates hash codes for all database items and label vectors for queries outside the database. Its training and its hash code generation are different tasks but on the same architecture. The second component is the Pareto optimization block responsible for forming Pareto points and generating Pareto fronts. The third component provides a report for search results by calculating MQUR and nDCG scores. Please refer to Fig. 1 for the following discussion.

4.1. DMVH component

The first part of the DMVH component comprises five convolutional blocks with a corresponding architecture and a set of parameters taken from the VGG-16 pre-trained network. This part is not trained and spatial features from video frames are extracted from it. Its output is the input for the second part that is trainable, which is composed of a two-layer stacked BLSTMs, and the HEL layer that is a dense layer with sigmoid activation having K outputs. The trainable part generates hash codes. K represents the size of the hash code for each video. BLSTM layers capture temporal dependencies across frames and contain information about whole frame sequences. Each BLSTM layer creates two LSTM instances, each having 1024 outputs. The first LSTM instance processes the frame sequence in the forward order and the second one processes them in the backward order in order to capture a richer representation of the frames. While the first BLSTM layer outputs a vector for each frame, the second layer outputs a single vector for each video. The reason is to train the network in a supervised way based on video labels. The purpose of using two BLSTM layers is to make the network deeper. The outputs of the two LSTM instances are then concatenated in the HEL. A classification layer exists at the end of the network. The classification layer has a sigmoid output with binary cross-entropy loss since the search database is multi-label.

In Fig. 1, let $\mathbf{a}^{\mathbf{F}} \in \mathbb{R}^{2048}$ denote the output of the second BLSTM layer, and $\mathbf{W}^{\mathbf{H}} \in \mathbb{R}^{2048 \times K}$ denote the weight matrix between the HEL and the second BLSTM layer. Since HEL has sigmoid activation, its output will be $\mathbf{a}^{\mathbf{H}} = \sigma(\mathbf{a}^{\mathbf{F}}\mathbf{W}^{\mathbf{H}} + \mathbf{b}^{\mathbf{H}})$, where $\mathbf{b}^{\mathbf{H}} \in \mathbb{R}^K$ is the bias term. The learned binary code extracted from the HEL is given by

$$\mathbf{B} = \text{sgn}(\mathbf{a}^{\mathbf{H}}) > 0.5. \tag{7}$$

In the training stage, two loss functions are used. The first one is defined over the HEL and minimizes the mean-square difference between the output of the HEL and the learned binary codes. It forces the output of the HEL to be closer to 0 or 1 so that the HEL output is closer to the learned binary codes during network training. This corresponding loss function is defined as

$$L_1 = \frac{1}{K} \sum_{n=1}^{K} (\mathbf{B} - \mathbf{a}^{\mathbf{H}})^2. \tag{8}$$

The second loss function defined in Eq. (9) is the binary cross-entropy loss at the classification layer:

$$L_2(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{C} \sum_{c=1}^{C} [\boldsymbol{y}_c \ln(\hat{\boldsymbol{y}}_c) + (1 - \boldsymbol{y}_c) \ln(1 - \hat{\boldsymbol{y}}_c)] \tag{9}$$

where y and $\hat{y}$ denote the true and predicted labels, respectively and C is the number of classes (in this study, $C = 4$ since database videos are related to four different semantics). The second loss forces semantically similar videos to have the same class labels. The overall loss function

denoted by L is equal to the sum of the respective two losses and is given by

$$L = L_1 + L_2. \tag{10}$$

The deep network was retrained by assigning different weights to L_1 and L_2 losses and hash codes were re-generated. However, using different weights did not improve retrieval performance. For this reason, equal weights (specifically 1) for L_1 and L_2 were used.

Note that hash code generation is implemented over a classification architecture. As a result, similar hash codes are expected to be generated for semantically similar videos. All codes related to hash code generation exist on the web.[1]

4.2. Pareto optimization component

4.2.1. Pareto points

The distance between the two hash codes is measured by the Hamming distance. The deep structure was designed such that Hamming distance between hash codes is low for semantically similar videos. Hence, Hamming distance was used as a measure of dissimilarity between two videos in this study.

Pareto points are formed by calculating the Hamming distances between each item in the search database and each query. Let $\mathbf{q}_{\mathbf{1}\times K}$ and $\mathbf{L}_{N\times K}$ denote the hash code of a query and hash codes of all items in the database, where N is the number of database items and K is the hash code length. N Hamming distances between $\mathbf{q}$ and each row of $\mathbf{L}$ must be computed to obtain the Pareto points. This poses a bottleneck from a computational point of view for large scale video databases. One solution to speed up the distance calculations is as follows. A $N \times K$ matrix whose each row is identical to $\mathbf{q}$ is constructed. Let us denote the matrix by $\mathbf{Q}$. If the $\oplus$ operator stands for the row-wise Hamming distance vector between two matrices of the same size, the Hamming distances between the whole search database and a query can be represented as

$$\mathbf{d} = \mathbf{Q}_{N\times K} \oplus \mathbf{L}_{N\times K} \tag{11}$$

where $\mathbf{d}$ is a $N \times 1$ vector. The process explained for one query can be repeated for the other queries. Let the distance vector for the ith query be denoted by $\mathbf{d}^i$. Then, the set of all Pareto points denoted by $\mathbf{X}_{N\times T}$ is obtained as

$$\mathbf{X}_{N\times T} = [\mathbf{d}^1, \mathbf{d}^2, \ldots, \mathbf{d}^T] \tag{12}$$

where T is the number of queries. Each row of $\mathbf{X}$ represents a Pareto point.

4.2.2. Forming Pareto fronts

By definition of Pareto front, a chosen Pareto point is included in the first Pareto front if it ranks better than the other ones for all objectives. Otherwise, it is left as a candidate for the remaining fronts. To construct the second front, points on the first Pareto front are deleted from the set of Pareto points and the process for the first Pareto front is repeated. The process is repeatedly applied to construct the remaining fronts until all Pareto points are processed. The corresponding Pareto front forming codes are available on the web.[2] After forming Pareto fronts, a predefined number of fronts, points in each front and the related video indexes in the search database are obtained.

4.3. Multi-query video retrieval component

The retrieval process can be summarized as follows. Hamming distances between each query and the dataset are calculated with Eq. (11) and the Pareto space is formed from the distances by using Eq. (12). Depending on the Hash code size, a Point in the Pareto space may represent a single video or multiple videos. When the hash code size is low, different videos might have the same hash code. In this case, the number of Pareto points is limited and a Pareto point usually corresponds to more than one video. When the hash code size increases, only visually similar videos will have similar hash codes. For this case, the number of Pareto points increases and a Pareto point is related to only one video. Then, Pareto fronts that are Pareto optimal solutions are formed in the Pareto space. Videos represented by Pareto points comprising the Pareto fronts are the candidate videos for the retrieval. Retrieved videos in the first several fronts are likely to be the items semantically related to all queries. However, all videos in a front are not semantically related to all queries. In a 2-query case, for example, some videos in the middle of a front may have semantic similarity to all queries and their MQUR scores are one while videos on the tails of the front may have similarity to only one query and their MQUR scores are 1/2. The DMQVR method retrieves a video from a front when it has an MQUR score of one.

One question still needs to be answered. How are the retrieved items having a MQUR of 1 sorted? A re-ranking on the retrieved items is performed for sorting them based on the non-binary features. There are two main motivations for applying re-ranking. First, recall that information loss occurs during the hash code generation. Sorting the retrieved items based on a proper distance measure computed from their original real-valued features avoids the information loss resulting from the hash code generation. Second, a given Pareto point may correspond to more than one database item especially when the hash code length is small. However, the probability that two or more database items have the same real-valued feature vector is lower even if a small size feature vector is used. Hence, multiple database items are not likely to share a single real-valued feature vector.

Implementation of re-ranking is as follows. A new set of Pareto points are formed for only the retrieved items based on the non-binary features extracted from the HEL and using the Euclidean distance as the distance measure. The items are then re-ranked by their Euclidean distances to the origin in the real-valued feature space.

5. Simulation results

We have found out that a publicly available database for MQVR research is not available. Consequently, we have created two multi-label video datasets to test the retrieval performance of the proposed MQVR framework. They are Pareto available on the web.[3]

The first dataset contains 120 sport videos that are related to four semantics including “football”, “tennis”, “running” and “ski”. 16 videos belong to the “football” class, the “tennis” category contains 16 videos, 20 videos exist in the“running” class and the “ski” group consists of 12 videos. Multi-label video clips were obtained by concatenating single concept video clips. 51 videos have labels consisting of two classes and 5 videos are related to three classes. Each video duration is about 15 s. For videos related two concepts, the duration of each concept is around 8 s and for those containing three concepts, the duration of each concept is 5 s. The second dataset contains 90 scene-based videos which are “traffic of Istanbul”, “satellite tour around the world”, “driving in the suburban area” and “walk in the beach”. 15 videos exist for each group. Multi-label videos were obtained in the same way as in the first dataset. 20 videos have labels consisting of two classes and 10 videos are related to three classes. The duration of the videos and the concepts in the videos are the same as the first dataset.

The trainable part was trained by the back-propagation algorithm using the stochastic gradient descent algorithm with momentum optimization by 120 epochs. The related training parameters are: batch

[1] https://github.com/akbacak/DMLVH2

[2] https://github.com/akbacak/DMQVR/blob/master/pareto_fronts.m

[3] https://github.com/akbacak/DMQVR

size is 32, the validation/train ratio is 0.2, the learning rate is 0.001. The deep learning architecture was implemented in the Keras deep learning library by TensorFlow backend. 32, 64 and 128-bit hash codes were extracted after the network was trained. One frame per second is extracted from videos for the training stage.

In order to show that the DMQVR method is efficient and fast for query pairs and triplets in terms of nDCG scores, 250 query pairs were chosen randomly from the first dataset. Among them, 50 query pairs belong to “football-tennis”, 80 pairs are from “football-running”, 80 pairs contain “running-tennis” and 40 pairs are related to “ski-running” classes. Similarly, 150 random triple queries were chosen from the second dataset. In this case, 75 query triples are from “traffic of Istanbul - satellite tour around the world - walk in the beach” and 75 triples are related to “satellite tour around the world - driving in the suburban area - walk in the beach” classes. Performance metrics results for query pairs and triplets given below were obtained without applying re-ranking. Hence, performance results obtained before the re-ranking process show the video retrieval accuracy. Since the re-ranking just ranks the true candidates, results satisfy the conditions of retrieval.

nDCG scores were obtained beginning from the middle of the Pareto fronts and moving towards both tails. The reason is that the most relevant videos semantically similar to queries are in the middle of the fronts. Since the front sizes may not be equal, their middle points do not align. Front sizes are made equal by zero padding. Then, the center points of the fronts are aligned. Across different Pareto fronts, average nDCG scores of the videos in the same position are obtained.

Simulations were divided into three groups and the following issues were investigated.

- How does retrieval accuracy depend on the hash code length?
- How is retrieval accuracy affected by the number of queries?
- How is retrieval time related to the hash code length and the number of queries?

Also, the designed GUI will be briefly explained.

5.1. Retrieval accuracy versus the hash code length

The dependency of retrieval accuracy on the hash code length is provided in Fig. 3. Even though the cumulative retrieval result from the first 5 Pareto fronts is satisfying for low-length hash codes, more selective and efficient results are obtained for high-length hash codes. The total number of Pareto points is around 60–80 for 32-bit hash codes. As a result, a Pareto point corresponds to 2 database videos. Thus, the number of videos on the fronts is high but the nDCG scores are low. The number of Pareto points increases up to 115 and a Pareto point almost represents one video in the database if 128-bit hash codes are used. Consequently, the number of videos on the fronts decreases but nDCG scores increase. This is why high-length hash codes are more selective and efficient. In conclusion, for the cumulative video retrieval, low-length hash codes are ideal. One should prefer high-length hash codes if more selective and efficient results are required. Fig. 3(a) clearly illustrates this observation.

As can be seen in Fig. 3(b), a similar behavior exists in the 3-query case. However, dependency of retrieval accuracy on the hash code length for the 3-query case is not strongly reflected because only 10 videos are related to three labels.

5.2. Retrieval accuracy versus the number of queries

Retrieved items on the Pareto fronts have MQUR scores of zero, 0.5 or 1 for query pairs while MQUR scores may be zero, 1/3, 2/3 or 1 when there are three queries. In other words, using more queries tends to increase the number of videos in a front. As a result, average nDCG curves in Fig. 3(b) are wider than those in Fig. 3(a). On the other hand, since the number of videos related to three concepts is less than those related to one and two concepts, the retrieval accuracies in Fig. 3(b) are lower compared to those in Fig. 3(a).

Table 1
Effect of hash code length on pareto points creation time.

	Pareto points creation time (s)	
Hash code length (bit)	2-query case	3-query case
32	0.00011	0.00043
64	0.00015	0.00050
128	0.00022	0.00058
256	0.00037	0.00076
512	0.00053	0.0011
1024	0.0011	0.0018

5.3. Retrieval time versus the hash code length and the number of queries

The dependency is investigated by using randomly chosen 150 query pairs and 150 query triples from the second dataset for different hash code lengths. Simulations were carried out with a personal computer having an Intel i7-4700HQ 2.4 GHz processor and 16 GB RAM. Results are summarized in Table 1. When the number of queries is kept constant, the time it takes to create the Pareto points increases as the hash code length increases. However, the increase is not dramatic. On the contrary, if the hash code length is fixed, the Pareto points forming time increases around three or four times when an additional query is used. In conclusion, retrieval time significantly increases by the number of queries while it rises slowly when the hash code length is doubled.

5.4. Designed GUI

A GUI that is publicly available was designed.[4] MATLAB 2015a/2018a on the Ubuntu 18 operating system was the development environment. Fig. 4 provides a screenshot of the designed GUI. The GUI supports only 2-query and 3-query cases even though the DMQVR framework is valid for an arbitrary number of queries. The reason is that we do not have tools to visualize Pareto points in a four or higher-dimensional space. In Fig. 4, three videos from the first dataset were used as queries. The hash code length was set to 512 bits. Although the labels of the queries are actually known, they are predicted by the DMVH algorithm and the predicted labels were used in the retrieval process. The total number of Pareto points, which are 3-dimensional, is 120. On the middle-right portion of the figure, retrieved videos on the first Pareto front are shown. nDCG scores of the first Pareto front can be seen on the upper right side. Even though there are 35 items in the first front, only two of them are related to all queries. The button at the bottom left implements re-ranking by means of which videos in the first 5 fronts that are relevant to all queries were obtained. Then, Pareto points were recreated with their non-binary features by using Euclidean distances. Retrieved Videos were sorted according to their Euclidean distances to the origin. The final retrieved videos are shown at the bottom left. There are exactly two videos related to all query videos in the first dataset. Hence, the proposed method did a perfect job of finding the database videos that are semantically related to queries. Only one screenshot was provided because of the page limit.

6. Conclusion

MQVR problem for which each query is related to different video semantics was investigated for the first time in this study. The proposed MQVR framework supports an arbitrary number of queries. It is based on video hash codes generated by a deep CNN-BLSTM network. Retrieval is carried out in the hash space with the Pareto front optimization method. Deep CNN-BLSTM was used for not only generating hash codes of the dataset items but also for calculating query hash codes and predicting their labels when they are chosen outside the

[4] https://github.com/akbacak/DMQVR

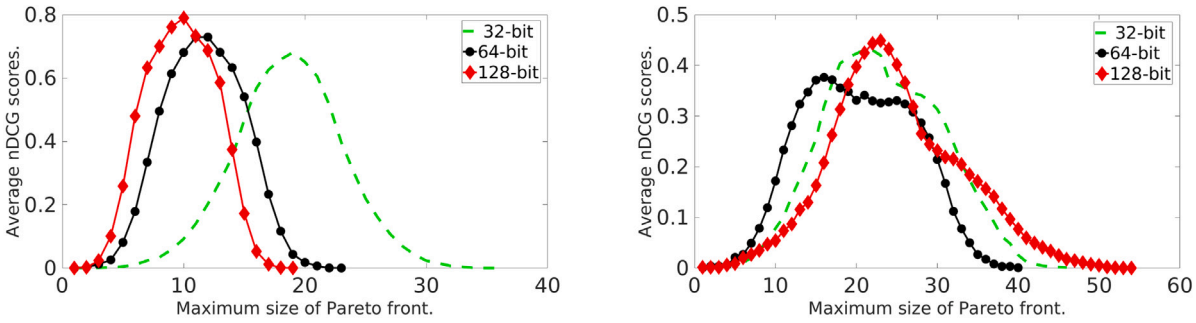


Fig. 3. Across top 5 Pareto fronts, average nDCG scores of the videos retrieved by the DMQVR method for (a) 2-query case, (b) 3-query case.

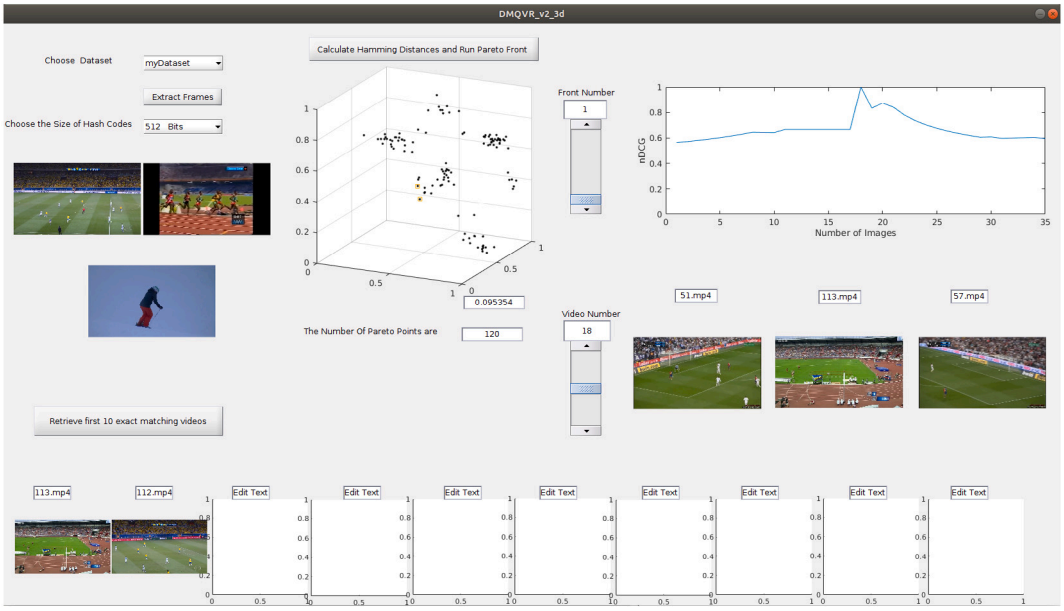


Fig. 4. A screenshot from GUI performing MQVR by a 3-query.

database. Retrieval time and retrieval accuracy provided by the MQVR framework were shown to be satisfactory for real-time applications. Also, the functionality of the DMQVR method was investigated by using the designed GUI that is publicly available on the web.

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.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jvcir.2022.103501.

References

[1] V.E. Liong, J. Lu, Y.-P. Tan, J. Zhou, Deep video hashing, IEEE Trans. Multimed. 19 (6) (2016) 1209–1219.

[2] J. Song, H. Zhang, X. Li, L. Gao, M. Wang, R. Hong, Self-supervised video hashing with hierarchical binary auto-encoder, IEEE Trans. Image Process. 27 (7) (2018) 3210–3221.

[3] Z. Chen, J. Lu, J. Feng, J. Zhou, Nonlinear structural hashing for scalable video search, IEEE Trans. Circuits Syst. Video Technol. 28 (6) (2017) 1421–1433.

[4] G. Wu, J. Han, Y. Guo, L. Liu, G. Ding, Q. Ni, L. Shao, Unsupervised deep video hashing via balanced code for large-scale video retrieval, IEEE Trans. Image Process. 28 (4) (2018) 1993–2007.

[5] Y. Hao, T. Mu, J.Y. Goulermas, J. Jiang, R. Hong, M. Wang, Unsupervised t-distributed video hashing and its deep hashing extension, IEEE Trans. Image Process. 26 (11) (2017) 5531–5544.

[6] X. Liu, L. Zhao, D. Ding, Y. Dong, Deep hashing with category mask for fast video retrieval, 2017, arXiv preprint arXiv:1712.08315.

[7] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, S.W. Baik, Action recognition in video sequences using deep bi-directional lstm with CNN features, IEEE Access 6 (2017) 1155–1166.

[8] S. Li, Z. Chen, X. Li, J. Lu, J. Zhou, Unsupervised variational video hashing with 1D-CNN-LSTM networks, IEEE Trans. Multimed. (2019).

[9] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.

[10] H. Bay, T. Tuytelaars, L. Van Gool, Surf: Speeded up robust features, in: European Conference on Computer Vision, Springer, 2006, pp. 404–417.

[11] L. Zheng, Y. Yang, Q. Tian, Sift meets CNN: A decade survey of instance retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 40 (5) (2017) 1224–1244.

[12] H. Kim, E. Ahn, M. Shin, S.-H. Sim, Crack and noncrack classification from concrete surface images using machine learning, Struct. Health Monit. 18 (3) (2019) 725–738.

[13] Y. Cao, H. Qi, W. Zhou, J. Kato, K. Li, X. Liu, J. Gui, Binary hashing for approximate nearest neighbor search on big data: A survey, IEEE Access 6 (2017) 2039–2054.

[14] J. Wang, T. Zhang, N. Sebe, H.T. Shen, et al., A survey on learning to hash, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2017) 769–790.
[15] J. Wang, W. Liu, S. Kumar, S.-F. Chang, Learning to hash for indexing big data—A survey, Proc. IEEE 104 (1) (2015) 34–57.
[16] X. Zhang, Y. Tian, R. Cheng, Y. Jin, An efficient approach to nondominated sorting for evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 19 (2) (2014) 201–213.
[17] K.-J. Hsiao, J. Calder, A.O. Hero, Pareto-depth for multiple-query image retrieval, IEEE Trans. Image Process. 24 (2) (2014) 583–594.
[18] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of IR techniques, ACM Transactions on Information Systems (TOIS) 20 (4) (2002) 422–446.