Asymmetric Spatio-Temporal Embeddings for Large-Scale Image-to-Video Retrieval

Noa Garcia
garciadn@aston.ac.uk
George Vogiatzis
g.vogiatzis@aston.ac.uk

Aston University
Birmingham
United Kingdom

Abstract

We address the problem of image-to-video retrieval. Given a query image, the aim is to identify the frame or scene within a collection of videos that best matches the visual input. Matching images to videos is an asymmetric task in which specific features for capturing the visual information in images and, at the same time, compacting the temporal correlation from videos are needed. Methods proposed so far are based on the temporal aggregation of hand-crafted features. In this work, we propose a deep learning architecture for learning specific asymmetric spatio-temporal embeddings for image-to-video retrieval. Our method learns non-linear projections from training data for both images and videos and projects their visual content into a common latent space, where they can be easily compared with a standard similarity function. Experiments conducted here show that our proposed asymmetric spatio-temporal embeddings outperform state-of-the-art in standard image-to-video retrieval datasets.

1 Introduction

In this work, we address the task of image-to-video retrieval. With billions of images and videos generated online every day, it is essential to build specific tools for indexing and accessing multimedia content efficiently. In the context of multimedia retrieval, image-to-video retrieval is the task for which a specific frame or scene within a video collection is identified from a static query image. Identifying a certain frame from a collection of videos has been successfully exploited in many different applications, such as video search [5], content augmentation [14] or video bookmark [10].

Image-to-video retrieval is an asymmetric problem. Whereas images contain static content, richer visual information can be extracted from videos, such as optical flow or temporal motion. Standard techniques for extracting video descriptors [20, 23, 28, 32] cannot be directly used on static images, due to the lack of temporal information. On the other hand, standard features for image retrieval [2, 19, 25, 31] are applied to video data by processing each frame as an independent image. When frames are processed as independent images, the temporal correlation within a video is not being considered and potentially unnecessary visual data is computed. This becomes extremely inefficient in large-scale datasets. Thus, specific asymmetric embeddings for extracting the visual content of images and compressing the temporal information of videos are needed.

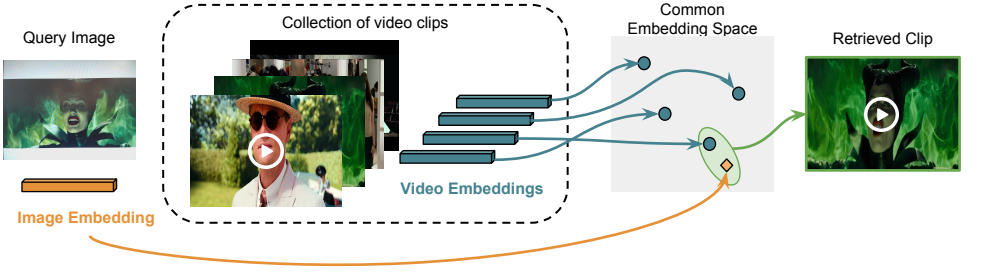

Figure 1: **Image-to-video retrieval**. Image and video embeddings obtained from query images and video clips, respectively, are projected into a common embedding space to compute similarities and find a specific video clip.

In image-to-video retrieval, temporal information is usually compressed either by reducing the number of local features or by encoding multiple frames into a single global representation. Methods based on the aggregation of local features obtain better performance than methods that encode several frames into a single global representation, at the expense of a lower data compression and an increased search time. In large-scale datasets, memory requirements and search time are crucial, so techniques based on global representations are preferred. Traditionally, image-to-video retrieval methods [1, 2, 14] are based on handcrafted features (SIFT [22], BRIEF [9], etc.) and not much effort has been put so far into the adaptation of deep learning techniques, such as convolutional neural networks (CNN) or recurrent neural networks (RNN). In [3] there was an attempt to use pre-trained CNN architectures to extract features for image-to-video retrieval, but the reported performance was systematically worse than the one obtained with SIFT and Fisher Vectors [27] features.

This paper tries to fill this gap by proposing a deep learning architecture for learning asymmetric spatio-temporal visual embeddings specifically for the image-to-video retrieval task. Our proposed architecture generates an image embedding with a CNN model for images and a video embedding based on RNN for short video clips. Video embeddings compress the spatio-temporal visual information on videos into a global vector and image embeddings extracts the meaningful visual information from static images. The model learns from training data to project images and video clips into a common embedding space where they can be easily compared, as shown in Figure 1. In this common space, image and video embeddings are matched by using a standard similarity function. Videos, then, can be ranked according to their similarity score with respect a given query image. Our proposed model outperforms state-of-the-art methods based on global representations on several image-to-video retrieval datasets.

2 Related Work

Image-to-video retrieval started to attract attention since the early 2000s. With relatively small datasets, early work in the field [24, 30] processed frames as independent images by applying image retrieval techniques. For example, Sivic and Zisserman [30] indexed frames from two different movies by using a bag-of-words (BoW) and Nister *et al.* [24] introduced

vocabulary trees to index BoW from video frames. These early methods did not consider the temporal structure of videos to compress video data, which makes them unsuitable for scaling over hundreds of hours of video in large-scale datasets.

To perform large-scale image-to-video retrieval and reduce the amount of data to be processed, the temporal redundancy in similar looking frames needs to be exploited. Methods for aggregating and exploiting temporal features are classified into two different groups: techniques that aggregate local features into a smaller number of features, and techniques that compacts multiple frames into a global vector representation.

In the first group, local features (commonly a few hundred) are extracted from each frame. Each of these local features is tracked along time and aggregated into a reduced number of vectors. For example, Anjulan and Nishan [1] averaged SIFT [22] descriptors within the same track to get a single vector per track, Araujo *et al.* [2] explored different methods of aggregation such as averaging, keeping just one or computing the minimum distance and Garcia and Vogiatzis [14] aggregated binary features using majorities. At query time, all these methods perform a matching between local query features and aggregated video features, which involves performing multiple searches per query.

On the other hand, global aggregation methods compact the visual information of a short video segment into a single vector. For example, Zhu and Satoh [34] aggregate all the SIFT local features in a video clip into a single high-dimensional BoW vector and Araujo and Girod [3] compute compact Fisher Vectors [27] per frame and aggregate them into a single vector using Bloom Filters [8]. At query time, a single search between the query vector and the video descriptors is conducted, which requires less memory than the search involved when using local features.

Most of the image-to-video techniques are based on hand-crafted features, such as SIFT [22], BRIEF [4], etc. Araujo and Girod [3] compared pre-trained CNN architectures as feature extraction models against systems based on SIFT and Fisher Vectors, obtaining worse performance with the deep learning methods. However, considering the outstanding results of deep learning in many retrieval tasks, such as image-to-image [15] or text-to-image retrieval [33], we suspect that the lack of success of deep learning models in [3] might be due the model architecture and the lack of training. In a preliminary work [13], we show that pre-trained deep learning features for image retrieval such as RMAC [31] are more convenient for image-to-video retrieval than standard neural network outputs. In this work, we propose a model based on CNN and RNN to learn asymmetric spatio-temporal embeddings so that images and videos are projected into a common embedding space, in where the matching is performed with a simple similarity function. Unlike [3, 13], we train our architecture using related video data, which allows us to learn specific non-linear functions for the image-to-video retrieval problem.

3 Asymmetric Spatio-Temporal Embeddings

In this section, we present an architecture for learning asymmetric spatio-temporal embeddings for image-to-video retrieval. We project images and videos into a common embedding space where a standard similarity function is used to rank videos according to their similarity with respect a query image. Our model is described in Figure 2. With a spatial encoder based on convolutional neural networks (CNN), query images and video frames are independently mapped into image embeddings. Image embeddings from frames within the same video clip are then input into a temporal encoder, which is based on recurrent neural networks (RNN),

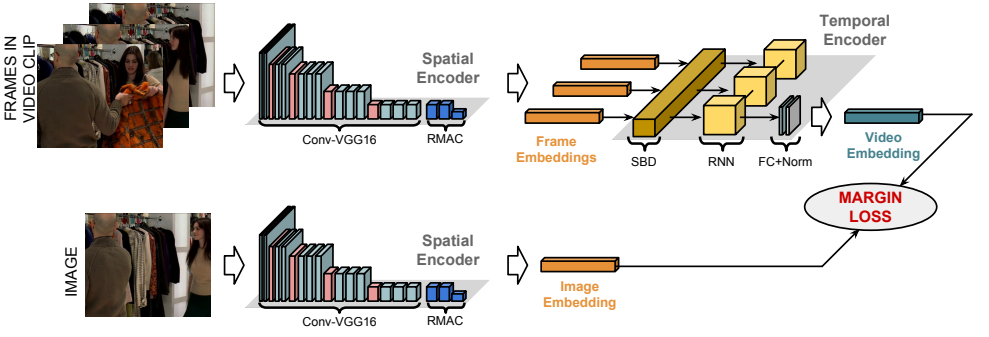

Figure 2: **Proposed model**. Images and frames are mapped into image embeddings with a Spatial Encoder. Videos are mapped into video embeddings with a Temporal Encoder. A margin loss function between image and video embeddings is computed to learn the weights.

to obtain a set of video or shot embeddings describing the visual content of the scene. Image and shot embeddings are compared in a common embedding space by using cosine similarity. During training, a contrastive or margin loss is computed between pairs of images and video clips to learn the network parameters. A detailed description of each of the components of our model is provided in the following subsections.

3.1 Spatial Encoder

To capture the spatial visual content of query images and frames and compute meaningful image embeddings, we use the RMAC descriptor proposed in [41]. RMAC is an image representation method obtained from the last convolutional layer of a CNN pre-trained for image classification. When an image is fed into the neural network, the last convolutional layer outputs a $W \times H \times K$ dimensional response, where K is the number of filters and W and H are the spatial width and height of the output that depend on the network architecture and on the size of the input image. The response of the k-th filter of the last convolutional layer can be represented by Ω_k, a 2D tensor of size $W \times H$. If $\Omega_k(p)$ is the response at a particular position p, and R is a spatial region within the feature map, the regional feature vector f_R is defined as:

$$f_R = [f_{R,1} \dots f_{R,k} \dots f_{R,K}]^\top \tag{1}$$

where $f_{R,k} = \max_{p \in R} \Omega_k(p)$. Thus, f_R consists of the maximum activation of each filter inside the region R. Several regional features are extracted at different multi-scale overlapping regions. Each of these regional vectors is independently post-processed with $\ell 2$-normalization, PCA-whitening and $\ell 2$-normalization. Regional vectors are summed up and $\ell 2$-normalized once again to obtain the final compact vector, whose dimensionality is K (i.e. the number of filters in the last convolutional layer) and it is independent of the size of the input image, its aspect ratio or the number of regions used.

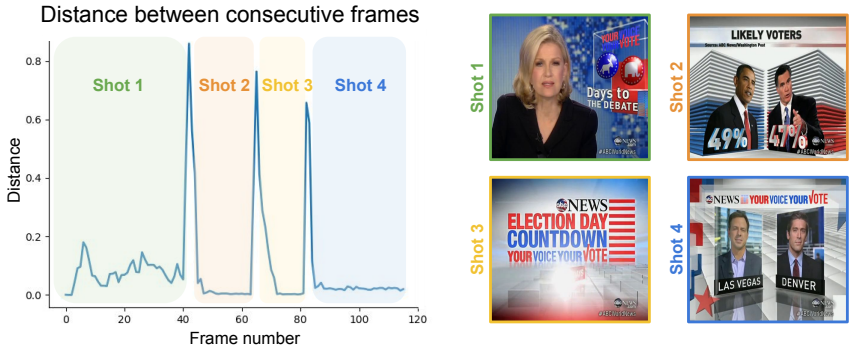

Figure 3: **Shot boundary detection algorithm**. *Left*: Shot boundaries detected when the distance between consecutive frames is high. *Right*: Frames from each detected shot.

3.2 Temporal Encoder

We capture the temporal visual information using a temporal encoder model. Videos are composed by a set of frames, shots and scenes. Shots are sequences of consecutive frames captured by the same camera without interruption. Commonly, frames belonging to the same shot are highly correlated. On the other hand, scenes are groups of shots which share a common topic or theme. Frames within different shots but in the same scene may not share many visual similarities. With our temporal encoder, we take advantage of this inner temporal structure of videos to first, identify shots within a video and then, encode the visual information within a shot with a recurrent neural network.

Shot Boundary Detection: We split up each video into a collection of shots by using a shot boundary detection (SBD) algorithm. The aim of the SBD algorithm is to detect groups of similar looking frames to encode them into a single video embedding. Although there exist many SBD algorithms [16, 17] to detect different kinds of transitions between shots, such as fade in, fade out, wipes or dissolves, to the purposes of this work we are only interested in *hard cuts*, i.e. when two shots are put one after the other without any transition effect. To detect hard cuts, we use Algorithm 1. We compute the distance between each pair of consecutive frame embeddings along the duration of a video and assign a shot boundary when the distance is higher than a predefined threshold, Th. An example of the frame distances computed with the SBD algorithm along with sample frames from each detected shot can be seen in Figure 3.

Recurrent Neural Networks: To summarize the temporal visual information of frames within a shot, we use a recurrent neural network (RNN). We explore different RNN models, such as long short-term memory networks (LSTM) [18] and gated recurrent units (GRU) [11]. At each state of the RNN, we input each of the frame embeddings belonging to the shot. The RNN captures the salient temporal information in the sequence by using at each state both the current frame embedding and the output of the previous frame embeddings. The output of the last state is further processed with a fully connected layer, a *tanh* non-linearity and a $\ell 2$-normalization to obtain the video or shot embedding. Formally, let $h_{|\mathbf{x}|} = \text{RNN}(\mathbf{x})$ be the output of the last state of a recurrent neural network that processes the sequence of frame embeddings $\mathbf{x} = [x_1, x_2, \cdots, x_{|\mathbf{x}|}]$. The shot embedding is computed as

Algorithm 1 Shot Boundary Detection

```
procedure SBD(video)
    SB ← []
    i ← 0
    img ← getFrame(video, i)                    ▷ Get first frame and compute image embedding
    u ← RMAC(img)
    while hasFrame(video, i+1) do               ▷ Get new frame and compute image embedding
        img ← getFrame(video, i+1)
        v ← RMAC(img)
        dist ← 1 − (u·v)/(‖u‖2‖v‖2)             ▷ Compute similarity distance between consecutive frames
        if dist > Th then
            append(SB, i)                       ▷ Frame is a boundary if distance is higher than a threshold
        u ← v
        i ← i+1
    return SB
```

$v_{\mathbf{x}} = \text{norm}(\tanh(W \cdot h_{|\mathbf{x}|} + b))$, where $\text{norm}(z) = \frac{z}{\|z\|_2}$, $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ and W and b are the weight matrix and the bias vector of the last fully connected layer, respectively. The shot embedding is set to have the same dimensionality, K, as the image embedding.

3.3 Image-to-Video Retrieval

To perform image-to-video retrieval, we rank videos according to query images. We compute image-shot similarity as the cosine similarity, $\cos(a,b) = \frac{a \cdot b}{\|a\|\|b\|}$, between image embeddings and shot embeddings. The visual similarity between a query and a specific video is the max-pooled image-shot similarity between the query and all the shot embeddings in the video.

3.4 Margin Loss Function

We train our model using pairs of images and video shots, where images are frames from the same video collection as shots. For the i-th training pair, we denote as $\mathcal{F}_i$ to the image embedding representing the frame and as ϑ_i to the shot embedding representing the shot. For each pair, we automatically assign a positive or a negative label, y_i, as:

$$y_i = \begin{cases} 1 & \text{if } \mathcal{F}_i \text{ and } \vartheta_i \text{ belong to the same shot} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

We compute the loss of a pair as the cosine similarity with a margin, Δ, between the image and shot embeddings:

$$\text{Loss}(\mathcal{F}_i, \vartheta_i) = y_i(1 - \cos(\mathcal{F}_i, \vartheta_i)) + (1 - y_i)(\max(0, \cos(\mathcal{F}_i, \vartheta_i) - \Delta)) \tag{3}$$

4 Data

We evaluate our model in standard image-to-video retrieval datasets. To learn the parameters of our model, we use a related training dataset.

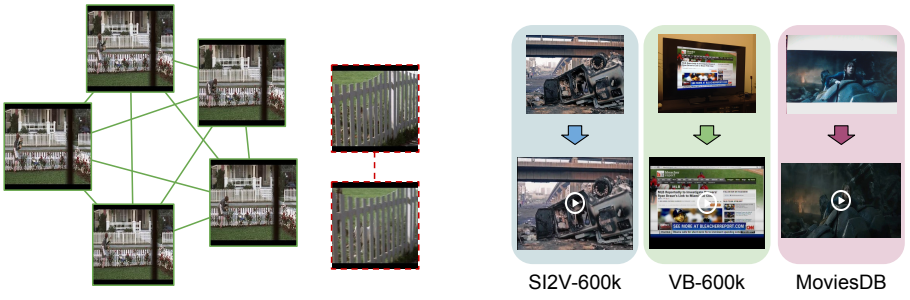

Figure 4: *Left*: **Data graph**. Nodes are frames and connections are matches. We keep frames in the strongest component (solid green lines) and remove the rest (dashed red lines). *Right*: **Test data**. Query images and video clip examples for each of the evaluation datasets.

4.1 Training Dataset

To train our model we use the data from the LSMDC dataset [26]. The LSMDC dataset contains 202 movies split into 128,118 short video clips of about 5 seconds. We remove the movies that overlap with our evaluation datasets and select a subset of 40 movies with 26,495 clips for training and 10 movies with 7,440 clips for validation, to speed up the training. We use clips provided in LSMDC as training shots. As clips in LSMDC do not exactly correspond to video shots (i.e. each clip is a short sequence which may contain frames from one or more shots), we conduct a cleaning process [15] to keep only frames from the longest shot for each clip. We extract SIFT [22] features from all the frames in the clip and perform an all-versus-all feature matching between all possible pairs of frames in the video clip. For each pair of frames, we assign a score as the number of shared descriptors over the total number of descriptors, keeping only scores greater than 0.25. Next, we build a graph where each node corresponds to a frame and each connection corresponds to their assigned score. We extract the strongest component of the graph and remove the nodes (i.e. frames) that do not belong to it. See Figure 4 (left) for an example of graph.

4.2 Evaluation Datasets

We evaluate our proposed model in the following datasets:

- SI2V-600k [5]: a dataset with newscast videos. It contains 164 hours of video with 3,401 clips corresponding to news stories. Queries are a set of 229 images from news websites. For each query image, a list of all clips where it is found is provided. Results are reported as mean average precission (mAP) and recall at 1 (R@1).
- VB-600k [5]: same videos as in SI2V-600k with 282 queries captured with an external camera while videos are played in a screen. Results are reported as mAP and R@1.
- MoviesDB [14]: a collection of movies where queries images are captured with a webcam while videos are being displayed in a screen. We use the lighter version, which consists on a single movie with about 2 hours duration and 615 query images.

Examples of each of the evaluation datasets are shown in Figure 4 (right).

Table 1: **Spatial Encoder Results**. Comparison between different image representations methods when frames are processed independently without temporal aggregation (i.e. image-to-image retrieval).

		SI2V-600k		VB-600k	
Method	**dim**	**mAP**	**R@1**	**mAP**	**R@1**
Edge Histogram [12]	-	0.154	0.372	-	-
JDC [12]	-	0.174	0.384	-	-
PHOG [12]	-	0.223	0.450	-	-
AlexNet FC6 [3]	4096	0.484	-	0.182	-
AlexNet FC7 [3]	4096	0.363	-	0.157	-
VGG16 FC6 [3]	4096	0.344	-	0.070	-
VGG16 FC7 [3]	4096	0.316	-	0.048	-
FV [3]	4096	0.715	-	**0.704**	-
RMAC	512	**0.718**	**0.834**	0.643	**0.592**

5 Experimental Evaluation

5.1 Experimental Details

Frames are extracted at three frames per second rate and resized to 1024 pixels width. In the spatial encoder, RMAC representations are obtained with a VGG16 network [29] pre-trained for image classification, without the last fully connected layers. The dimensionality of the image embeddings is $K = 512$. PCA-whitening is implemented as a fully connected layer and its weights are computed using American Beauty movie from the training collection. In the temporal encoder, the dimensionality of the hidden state of the RNN is 512, as well as the number of filters in the last fully connected layer. The maximum number of frames per shot used in the RNN is 50. At training time, $\Delta = 0.1$. We assign 20% of training pairs as positive and 80% as negative. The spatial encoder is frozen. We optimize the parameters of the temporal encoder using Adam [21] with backpropagation, batch size of 512 and learning rate of 0.0001. In the SBD algorithm, Th is experimentally chosen as 0.5. Query images are resized to 960 pixels width.

5.2 Spatial Encoder Results

We first evaluate the performance of the RMAC representation as our spatial encoder. We compare results obtained with different image representation methods when frames are processed as independent images (i.e. image-to-image retrieval). For a fair comparison, none of the networks are retrained or fine-tunned. Results are summarized in Table 1. RMAC obtains comparable performance to the best reported results (FV in [3]) by using 8 times less memory. When compared against other deep learning features (AlexNet FC6, AlexNet FC7, VGG16 FC6 and VGG FC7) RMAC is, by far, superior.

5.3 Temporal Encoder Results

Next, we evaluate our proposed temporal encoder. RMAC image representation is used to obtain frame embeddings. We compare our temporal encoder model based on RNN,

Table 2: **Temporal Encoder Results**. Comparison between different techniques to aggregate image embeddings (RMAC) from multiple frames (i.e. image-to-video retrieval).

	SI2V-600k		VB-600k		MoviesDB
Method	**mAP**	**R@1**	**mAP**	**R@1**	**R@1**
Max-Pooling	0.038	0.066	0.033	0.011	-
Sum-Pooling	0.152	0.275	0.316	0.262	-
Temporal Encoder (LSTM)	0.602	0.773	**0.580**	**0.525**	**0.833**
Temporal Encoder (GRU)	**0.606**	**0.777**	0.572	0.514	**0.833**

against simple aggregation baselines, such as Max-Pooling and Sum-Pooling. We evaluate two versions of the temporal encoder, one based on LSTM [13] and another one based on GRU [11]. Results are shown in Table 2. Max-Pooling and Sum-Pooling are not evaluated on MoviesDB as there is only one video in that dataset. Our temporal encoder is considerably superior to both baselines. In SI2V-600k, the temporal encoder based on GRU obtains the best performance. In VB-600k, LSTM is superior. With respect to MoviesDB, both LSTM and GRU perform equally well.

Shot Boundary Detection: We evaluate the contribution of the shot boundary detection algorithm to the system by considering three different scenarios: no shot boundary detection (No SBD), shot boundaries are selected randomly (Rand. SBD), and our proposed shot boundary detection algorithm based on RMAC distances (Our SBD). For a more extended comparison, we use both Sum-Pooling and LSTM as temporal aggregation methods. Results are reported in Table 3. Detecting shots is always beneficial, even if the shots are detected randomly, as the number of visual embeddings per video is increased. When we use our SBD, the performance of the overall system is improved considerably.

5.4 Comparison with State-of-the-art

Finally, we compare our asymmetric spatio-temporal embeddings with state-of-the-art compact methods in image-to-video retrieval. For a fair comparison, we only report results of compact aggregation methods, that is, methods that encode multiple frames into a single compact vector rather than using multiple local vectors per query or super high-dimensional vectors. This is because local aggregation methods use several local vectors per query image, performing multiple searches at test time for a single query. This increases performance with respect to global aggregation methods, but also increases the search time and the memory requirements. Also, as previous work is mostly based on non-trainable architectures, for a fair comparison we keep our spatial encoder with the original pre-trained weights (i.e. no additional training).

Accuracy: Results are detailed in Table 4. Our techniques based on LSTM and GRU outperform reported methods in terms of both memory and accuracy. When compared with previous state-of-the-art (Scene FV*), our embeddings are superior on the SI2V-600k dataset and obtain comparable results in VB-600k. This performance is remarkable considering that our methods are using embeddings 128 times smaller than Scene FV* (512 versus 65,536 dimensions).

Table 3: **Shot Boundary Detector**. Analysis of the SBD algorithm in the SI2V-600k dataset.

Method	mAP	R@1	Method	mAP	R@1
Sum-Pool, No SBD	0.152	0.275	LSTM, No SBD	0.121	0.319
Sum-Pool, Rand. SBD	0.322	0.589	LSTM, Rand. SBD	0.390	0.616
Sum-Pool, Our SBD	0.596	0.764	LSTM, Our SBD	**0.602**	**0.773**

Table 4: **Comparison with State-of-the-art**. Results provided as mAP.

Method	dim	SI2V-600k	VB-600k
Scene FV* (DoG) [4]	65,536	0.473	-
Scene FV* [3]	65,536	0.500	**0.622**
Sum-Pool AlexNet FC6 [3]	4096	0.071	0.012
Sum-Pool AlexNet FC7 [3]	4096	0.065	0.013
Sum-Pool VGG16 FC6 [3]	4096	0.067	0.013
Sum-Pool VGG16 FC7 [3]	4096	0.069	0.011
Spatio-Temporal-LSTM (Ours)	**512**	0.602	0.580
Spatio-Temporal-GRU (Ours)	**512**	**0.606**	0.572

Computational Cost: In [3], memory requirements are reported in SI2V-4M and VB-4M datasets, which are larger versions of SI2V-600k and VB-600k with 1080 hours of video. Their simplest baseline with FV* and no temporal aggregation needed 20.59 GB of memory to encode the whole dataset, whereas their Scene FV* required 3.01 GB. Their best performing method based on super high-dimensional Bloom Filters (33.5 million binary dimensional vectors) needed 10.76 GB of memory. With our approach, we are able to encode 160 hours of video in SI2V-600k and VB-600k datasets in only 0.15GB. With a simple conversion, for the larger versions (1080 hours of video), we would need just about 1GB of memory, which is a compression of 20 times with respect to the baseline, 10 times with respect to the Bloom Filters approach and 3 times with respect to Scene FV*.

6 Conclusions

In this paper, we proposed a model to obtain asymmetric spatio-temporal embeddings for large-scale image-to-video retrieval. The aim was to find a video clip within a collection of videos that best matches a given input image. We proposed an asymmetric architecture to project images and videos into a common embedding space, so that they can be easily matched with a cosine similarity. We computed image embeddings with a spatial encoder based on convolutional neural netwroks. Video embeddings were obtained by using a temporal encoder based on recurrent neural networks. We trained our model with relevant pairs of images and videos. Experiments in different image-to-video retrieval datasets showed that our spatial-temporal encoder is superior to the state-of-the-art methods both in terms of accuracy and computational cost.

References

[1] Arasanathan Anjulan and Nishan Canagarajah. Object based video retrieval with local region tracking. *Signal Processing: Image Communication*, 22(7), 2007.

[2] A Araujo, Mina Makar, Vijay Chandrasekhar, D Chen, S Tsai, Huizhong Chen, Roland Angst, and Bernd Girod. Efficient video search using image queries. In *ICIP*, 2014.

[3] Andre Araujo and Bernd Girod. Large-scale video retrieval using image queries. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[4] André Araujo, Jason Chaves, Roland Angst, and Bernd Girod. Temporal aggregation for large-scale query-by-image video retrieval. In *ICIP*, 2015.

[5] André Araujo, Jason Chaves, David Chen, Roland Angst, and Bernd Girod. Stanford i2v: a news video dataset for query-by-image experiments. In *ACM Multimedia Systems*, 2015.

[6] André Araujo, Jason Chaves, Haricharan Lakshman, Roland Angst, and Bernd Girod. Large-scale query-by-image video retrieval using bloom filters. *arXiv preprint arXiv:1604.07939*, 2016.

[7] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014.

[8] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13, 1970.

[9] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. *ECCV*, 2010.

[10] David Chen, Ngai-Man Cheung, Sam Tsai, Vijay Chandrasekhar, Gabriel Takacs, Ramakrishna Vedantham, Radek Grzeszczuk, and Bernd Girod. Dynamic selection of a feature-rich query frame for mobile video retrieval. In *ICIP*, 2010.

[11] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, 2014.

[12] Gabriel de Oliveira Barra, Mathias Lux, and Xavier Giro-i Nieto. Large scale content-based video retrieval with livre. In *International Workshop on Content-Based Multimedia Indexing*. IEEE, 2016.

[13] Noa Garcia. Temporal aggregation of visual features for large-scale image-to-video retrieval. In *ICMR*, 2018.

[14] Noa Garcia and George Vogiatzis. Dress like a star: Retrieving fashion products from videos. In *ICCV Workshops*, 2017.

[15] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *IJCV*, 124(2), 2017.

[16] Michael Gygli. Ridiculously fast shot boundary detection with fully convolutional neural networks. *arXiv preprint arXiv:1705.08214*, 2017.

[17] Ahmed Hassanien, Mohamed Elgharib, Ahmed Selim, Mohamed Hefeeda, and Wojciech Matusik. Large-scale, fast and accurate shot boundary detection through spatio-temporal convolutional neural networks. *arXiv preprint arXiv:1705.03281*, 2017.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8), 1997.

[19] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *ECCV*, 2016.

[20] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004.

[23] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[24] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. *CVPR*, 2006.

[25] Ali Sharif Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. Visual instance retrieval with deep convolutional networks. *Transactions on Media Technology and Applications*, 4(3), 2016.

[26] Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Chris Pal, Hugo Larochelle, Aaron Courville, and Bernt Schiele. Movie description. *IJCV*, 2017.

[27] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 105(3), 2013.

[28] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576, 2014.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[30] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[31] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. In *ICLR*, 2016.

[32] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[33] Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning deep structure-preserving image-text embeddings. In *CVPR*, 2016.

[34] Cai-Zhi Zhu and Shin'ichi Satoh. Large vocabulary quantization for searching instances from videos. In *ICMR*, 2012.