Recognizing and Presenting the Storytelling Video Structure with Deep Multimodal Networks

Lorenzo Baraldi, Costantino Grana, *Member, IEEE,* and Rita Cucchiara, *Member, IEEE*

***Abstract*—In this paper, we propose a novel scene detection algorithm which employs semantic, visual, textual and audio cues. We also show how the hierarchical decomposition of the storytelling video structure can improve retrieval results presentation with semantically and aesthetically effective thumbnails. Our method is built upon two advancements of the state of the art: 1) semantic feature extraction which builds video specific concept detectors; 2) multimodal feature embedding learning, that maps the feature vector of a shot to a space in which the Euclidean distance has task specific semantic properties. The proposed method is able to decompose the video in annotated temporal segments which allow for a query specific thumbnail extraction. Extensive experiments are performed on different data sets to demonstrate the effectiveness of our algorithm. An in-depth discussion on how to deal with the subjectivity of the task is conducted and a strategy to overcome the problem is suggested.**

***Index Terms*—Temporal Video Segmentation, Scene Detection, Deep Networks, Performance Evaluation**

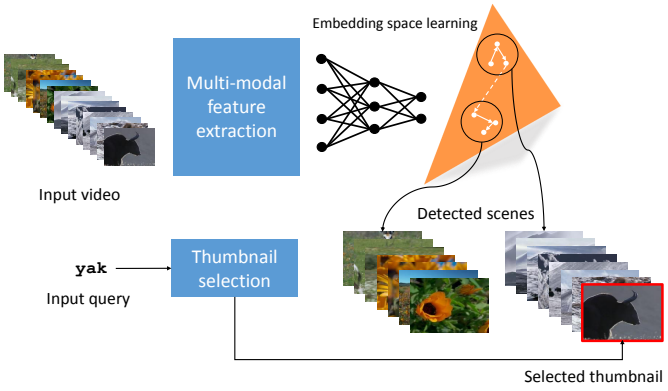


Fig. 1. We propose a scene detection algorithm which exploits multimodal features and a deeply-learned embedding space. By means of a novel thumbnail selection strategy, we also show that the decomposition of a video into scenes can enhance retrieval.

I. Introduction

REAL-TIME Entertainment is currently the dominant traffic category on the web, and video accounts for most of it. In the first half of 2016, 71% of downstream bytes during peak period were due to this category, and the top 3 applications were Netflix (35%), YouTube (18%) and Amazon Video (4%) [1]. While User Generated Videos are popular, in a recent survey, people of ages 13-34 indicated that the primary type of video content they viewed was TV shows, full-length movies, music videos, sports, and clips of TV shows for a total of 78% of the respondents, while another 8% was "Videos of people playing video games", and the rest was "Other user-generated content" [2].

Browsing video content is not as easy as searching other media, e.g. images. The returned result page in most search engines presents videos through their thumbnails, so assessing if the content is indeed pertinent to our query requires further playing, possibly with fast forward and backward operations. Professionally edited videos have a well-defined storytelling structure that we could leverage for improving the user experience. This structure may be described by a hierarchical decomposition (see Figure 1): at the lowest level we have frames, which are in turn grouped in shots, sequences of consecutive frames taken by a single camera act [3]. Temporal video segments on the level above shots are usually defined as *scenes*. The term borrows from stage production, focuses on the location of the action, and is mainly used in fictional narrative-driven videos. Given the broader sense that the term scene has in many contexts, the usual assumption that a scene is a set of shots with visually similar content [4], is clearly unsatisfactory. If we want to describe not only the location but also the topic of the video sequence, audio, speech and semantics also share an important role in the definition of a scene.

In this paper, we address the problem of automatically extracting the storytelling structure of an edited video, by grouping shots in scenes with a multimodal deep network approach, which employs semantic, visual, textual and audio cues. We show how the hierarchical decomposition can improve retrieval results in the form of semantically and aesthetically effective thumbnails.

The main novelties of our work are:

- We propose a strategy for extracting semantic features from the video transcript which are incorporated with perceptual cues into a multimodal embedding space, thanks to a Triplet Deep Network. Using these features, we are able to provide a state-of-the-art scene detection algorithm.
- We leverage the extracted storytelling structure to provide improved query dependent thumbnails, combining semantic and aesthetic information.
- We discuss the problem of evaluating scene detection and provide a dynamic programming algorithm for managing the subjectivity in presence of different contradicting annotations.

Both the source code of the algorithm and the datasets used are available at http://imagelab.ing.unimore.it/imagelab/page.asp?IdPage=12.

L. Baraldi, C. Grana and R. Cucchiara are with the Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Italy (e-mail: lorenzo.baraldi@unimore.it; costantino.grana@unimore.it; rita.cucchiara@unimore.it).

II. Related Work

In this section, we review the literature related to scene detection, video retrieval, and thumbnail selection techniques.

A. Scene detection and video decomposition

Existing works in the field of automatic scene detection can be roughly categorized into three groups [5]: *rule-based methods*, that consider the way a video is structured in professional movie production, *graph-based methods*, where shots are arranged in a graph representation, and *clustering-based methods*.

The drawback of rule-based methods is that they tend to fail in videos where film-editing rules are not followed strictly, or when two adjacent scenes are similar and follow the same rules. The method proposed by Liu *et al.* in [6] falls into this category: they propose a visual based probabilistic framework that imitates the authoring process. In [7], shots are represented by means of key-frames, clustered using spectral clustering and low-level color features, and then labeled according to the clusters they belong to. Since video editing tends to follow repetitive patterns, boundaries are detected from the alignment score of the symbolic sequences, using the Needleman-Wunsch algorithm.

In graph-based methods, instead, shots are arranged in a graph representation and then clustered by partitioning the graph. The Shot Transition Graph (STG) [8] is one of the most used models in this category: here each node represents a shot and the edges between the shots are weighted by shot similarity. In [9], color and motion features are used to represent shot similarity, and the STG is then split into sub-graphs by applying the normalized cuts for graph partitioning. Sidiropoulos *et al.* [10] introduced an STG approximation that exploits features from the visual and the auditory channel.

Clustering-based solutions assume that similarity of shots can be used to group them into meaningful clusters, thus directly providing the final temporal boundaries. In [11], for instance, a Siamese Network is used together with features extracted from a CNN and time features to learn distances between shots. Spectral clustering is then applied to detect coherent sequences.

Our work belongs to this latter class, but overcomes the limitations of the previous approaches incorporating audio and video in two flavors: they are used to extract perceptual features (e.g. CNN activations and MFCC) and semantic features (e.g. concepts and transcript words). We employ a temporal aware clustering algorithm which, by construction, generates contiguous segments: temporal coherence is not an additional requirement forced later, but is optimized during the clustering itself.

While the aim of our work is that of recovering the storytelling structure of a professionally edited video, a related research direction is that of generating a storytelling video from photos or video clips. For example, the method proposed in [12] generates stories from personal photo collections, by mimicking cinematic knowledge with a set of predesigned editing styles. After a summarization step, in which the best photos are selected, each photo is converted to a video clip by applying a virtual camera with appropriate motions and a set of video effects.

Our research is also related to the task of video summarization, which often has video decomposition as one of its fundamental elements. Indeed, the goal of video summarization is to produce a compact visual summary that encapsulates the most informative parts of a video, and which can be represented by key-frames [13], image montages [14], or video synopses [15]. Usually a shot detection step is followed by an interestingness prediction method, which can rely on low-level, high-level, or spatiotemporal features [16]. In [17] a deep ranking model was proposed to identify important segments in life logging first-person videos. In contrast, we focus on edited videos and, instead of ranking video segment according to their importance, we learn a model to identify homogeneous segments inside the video.

B. Video Retrieval

A lot of work has also been proposed for video retrieval: with the explosive growth of online videos, this has become a hot topic in computer vision. In their seminal work, Sivic *et al.* proposed Video Google [18], a system that retrieves videos from a database via bag-of-words matching. Lew *et al.* [19] reviewed earlier efforts in video retrieval, which mostly relied on feature-based relevance feedback or similar methods.

More recently, concept-based methods have emerged as a popular approach to video retrieval. Snoek *et al.* [20] proposed a method based on a set of concept detectors, with the aim to bridge the semantic gap between visual features and high level-concepts. In [21], authors proposed a video retrieval approach based on tag propagation: given an input video with user-defined tags, Flickr, Google Images and Bing are mined to collect images with similar tags: these are used to label each temporal segment of the video, so that the method increases the number of tags originally proposed by the users, and localizes them temporally. In [22], the problem of retrieving videos using complex natural language queries is tackled, by first parsing the sentential descriptions into a semantic graph, which is then matched to visual concepts using a generalized bipartite matching algorithm. This also allows to retrieve the relevant video segment given a text query. Our method, in contrast to [21], does not need any kind of initial manual annotation, and, thanks to the availability of the video structure, is able to return specific scenes related to the user query. This provides the retrieved result with a context that allows to better understand the video content.

Retrieved results need eventually to be presented to the user, but previewing many videos playing simultaneously is not something feasible. The usual approach is to present a set of video thumbnails. Thumbnails are basically surrogates for videos [23], as they take the place of a video in search results. Therefore, they may not accurately represent the content of the video, and create an *intention gap*, i.e. a discrepancy between the information sought by the user and the actual content of the video. Most conventional methods aim at selecting the "best" thumbnail and have focused on learning visual representativeness purely from visual content [24], [25].

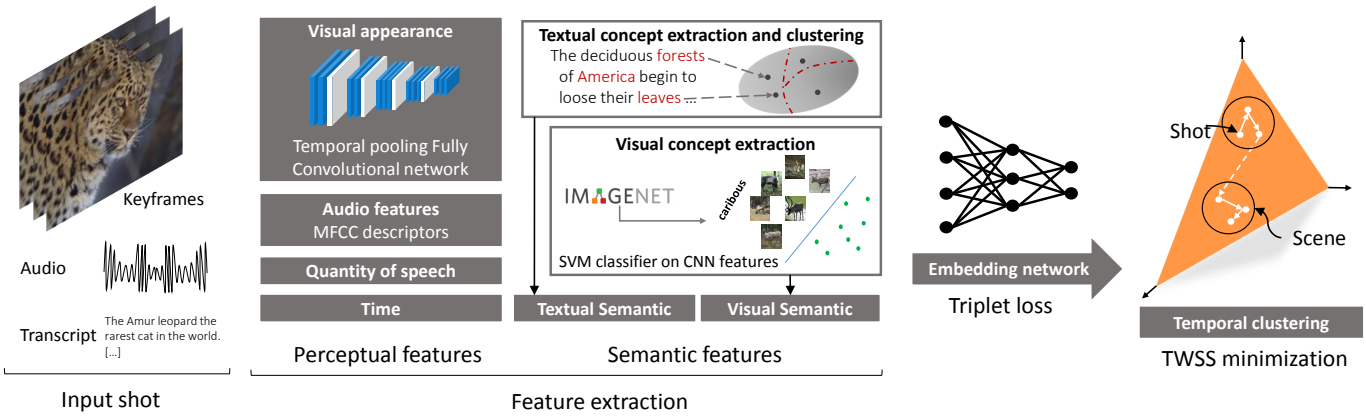


Fig. 2. Overview of the proposed approach. A semantic embedding is learned through a triplet deep network, which combines perceptual and semantic features.

However, more recent researches have focused on choosing query-dependent thumbnails to supply specific thumbnails for different queries. To reduce the intention gap, [23] proposes a new kind of animated preview, constructed of frames taken from a full video, and a crowdsourced tagging process which enables the matching between query terms and videos. Their system, while going in the right direction, suffers from the need of manual annotations, which are often expensive and difficult to obtain.

In [26], instead, authors proposed a method to enforce the representativeness of a selected thumbnail given a user query, by using a reinforcement algorithm to rank frames in each video and a relevance model to calculate the similarity between the video frames and the query keywords. Recently, Liu *et al.* [27] trained a deep visual-semantic embedding to retrieve query-dependent video thumbnails. Their method employs a deeply-learned model to directly compute the similarity between a query and video thumbnails, by mapping them into a common latent semantic space.

Our work can push things further, because we already retrieved a video scene for which the query is relevant, thus we just need to pick a keyframe within a very limited set of candidates. All possible thumbnails are thus ranked according to their relevance to the query and to their aesthetic value, providing the best presentation of the result for the specific user request.

III. Perceptual-Semantic Feature Embedding and Clustering for Scene Detection

We tackle the task of detecting scenes in edited videos as a supervised temporally constrained clustering problem. We firstly extract a rich set of perceptual and semantic features from each shot. In order to obtain a significant measure of similarity between shots features, we learn an embedding of these features in a Euclidean space. Finally, we detect the optimal scene boundaries by minimizing the sum of squared distances inside temporal segments (candidate scenes), using a penalty term to automatically select the number of scenes. A summary of our approach is depicted in Fig. 2.

In the following, we present a set of perceptual features based on visual appearance, audio, speech and time. Then, we propose two semantic features which rely on a joint conceptual analysis of the visual content and of the transcript, and which account for scene changes which are not recognizable using purely perceptual cues. Eventually, we present the embedding and clustering strategies.

A. Perceptual features

Visual appearance: A shot in an edited video is usually uniform from the visual content point of view, and it is, therefore, reasonable to rely on keyframes to describe visual appearance. At the same time, using a single keyframe could result in a poor description of both short and long shots, since the visual quality could be unsatisfactory, or its content may be insufficient to describe the temporal evolution of a shot. For this reason, we propose a solution which preserves the ability of Convolutional Neural Networks (CNNs) to extract high-level features, while accounting for the temporal evolution of a shot.

Specifically, we build a Temporal Pooling Fully Convolutional Neural Network, which can encode the visual appearance of a variable number of keyframes into a descriptor with fixed size. The proposed network is Fully Convolutional in that it contains only convolutional and pooling stages, and does not include fully connected layers. Moreover, the last stage of the network performs a temporal pooling operation, thus reducing a variable number of keyframes to a fixed dimension.

The architecture of the network follows that of the 16 layers model from VGG [28]. To keep a fully convolutional architecture, the last fully connected layers are removed, and a temporal pooling layer is added at the end. Parameters of the network are initialized with those pre-trained on the ILSVRC-12 dataset [29].

Given a set of keyframes $\{I_1, ..., I_t\}$ with size $l \times l$, each of them is independently processed by the convolutional and spatial pooling layers of the network, thus obtaining a three-dimensional tensor $CNN(I_i)$ for each keyframe I_i with shape $\left\lfloor \frac{l}{f} \right\rfloor \times \left\lfloor \frac{l}{f} \right\rfloor \times k$, where f is the factor by which the input image is resized by the spatial pooling layers of the network, and k is the number of convolutional filters of the last layer. Each of these k activation maps intuitively contains

the spatial response of a specific high-level feature detector over the input image. The temporal pooling layer performs a max-pooling operation over time: the output of this layer, therefore, has the same shape of $CNN(I_i)$, and contains, at each position (x, y, j), the element-wise maximum along the time dimension, $\max_{i\in\{1,\dots,t\}} CNN(I_i)(x, y, j)$. For the VGG-16 model, the input shape is 224×224, the resize factor f is 32, and k is 512.

Based on a preliminary evaluation, we chose to extract three keyframes per shot, with uniform sampling. More sophisticated sampling techniques were also tested: we encoded all the frames in a shot using color histogram and selected the t most different keyframes. However, no significant improvement with respect to uniform sampling was observed. Average-pooling in the temporal layer was also tested, but it led to worse performance than max-pooling.

Audio features: The audio of an edited video is another meaningful cue for detecting scene boundaries, since audio effects and soundtracks are often used in professional video production to underline the development of a scene, and a change in soundtrack usually highlights a change of content. For this reason, a standard audio descriptor based on short-term power spectrum is employed.

Following recent works in the field [30], we extract MFCCs descriptors [31] over a 10ms window. The MFCC descriptors are aggregated by Fisher vectors using a Gaussian Mixture Model with 256 components, where we retrain only components with weight greater than $1/256$.

Quantity of speech: Sometimes a pause in the speaker discourse can be enough to identify a change of scene: for this reason, we turn to the video transcript and build a quantity of speech feature, which computes the amount of words being said inside a shot. Notice that, when the video transcript is not directly provided by the video producer, it can be obtained with standard speech-to-text techniques.

For each shot, the quantity of speech is defined as the number of words which appear in that shot, normalized with respect to the maximum number of words found in a shot for the full video.

Time features: We also include the timestamp and length of each shot. The rationale behind this choice is that since scenes need to be temporally consecutive, shots having similar semantic content which are temporally distant should be distinguishable. Moreover, the average length of scenes can be a useful prior to be learned.

Notice that a shot-based representation has been kept in all the proposed features. For each shot, indeed, the concatenation of its feature vectors will be the input of the Triplet Deep Network, which will learn the embedding.

B. Semantic features

Perceptual features can be sufficient to perform scene detection on videos which have a simple storyline; however, it is often the case that scene boundaries correspond to changes in the topic which can not be detected by simply looking at appearance and sound. In the following we extract concepts from the video transcript, and project them into a semantic space; each concept is then validated by looking at the visual content of a shot.

To collect candidate concepts, sentences in the transcript are firstly parsed and unigrams which are annotated as *noun*, *proper noun*, and *foreign word* are collected with the Stanford CoreNLP [32] part-of-speech tagger. Selected unigrams contain terms which may be present in the video, and may be helpful to visually detect a change in topic. On the contrary, there are also terms which do not have concrete visual patterns, but that can still be important to infer a change in topic from the transcript. We will describe two features to account for both these situations.

Concept clustering: The resulting set of terms can be quite redundant and contain lots of synonyms, therefore we cluster it according to the pairwise similarities of terms, in order to obtain a set of semantically non-related clusters. In particular, we train a Word2Vec model [33] on the dump of the English Wikipedia. The basic idea of this model is to fit a word embedding such that the words in the corpus can predict their context with high probability. Semantically similar words lie close to each other in the embedded space.

In our case, each word is mapped to a 1000-dimensional feature vector, and the semantic similarity of two terms is defined as the cosine similarity between their embeddings. The resulting similarity matrix is then used together with spectral clustering to cluster the mined terms into K concept groups. K was set to 50 in all our experiments.

Due to the huge variety of concepts which can be found in the video collection, the video corpus itself may not be sufficient to train detectors for the visual concepts. Therefore, we mine images from the Imagenet database [34], which contains images from more than 40.000 categories from the WordNet [35] hierarchy. Our method, in principle, is applicable to any visual corpus, provided that it contains a sufficiently large number of categories.

Each concept in Imagenet is described by a set of words or word phrases (called *synset*). We match each unigram extracted from the text with the most similar synset in the aforementioned semantic space, and call $M(u)$ the synset resulting from this matching process for a unigram u. For synsets containing more than one word, we take the average of the vectors from each word and L_2-normalize the resulting vector.

Visual semantic features: Having mapped each concept from the video transcript to an external corpus, a classifier can be built to detect the presence of a visual concept in a shot. Since the number of terms mined from text data is large, the classifier needs to be efficient. Images from the external corpus are represented using feature activations from pre-trained deep convolutional neural networks. Then, a linear probabilistic SVM is trained for each concept, using randomly sampled negative training data; the probability output of each classifier is then used as an indicator of the presence of a concept in a shot. Again the 16-layers model from VGG [28] is employed, pretrained on the ILSVRC-2012 [29] dataset. We use the activations from layer `fc6`.

We build a feature vector which encodes the influence of each concept group on the considered shot. Given the temporal

coherency of a video, it is unlikely for a visual concept to appear in a shot which is far from the point in which the concept was found in the transcript. At the same time, concepts expressed in the transcript are not only related to the single shot they appear in, but also to its neighborhood. For this reason, we apply a normalized Gaussian weight to each term based on the temporal distance. Formally, the probability that a term u is present in a shot s is defined as:

$$P(s,u) = f_{M(u)}(s)\ e^{-\frac{(t_u - t_s)^2}{2\sigma_a^2}} \tag{1}$$

where M is the mapping function to the external corpus, and $f_{M(u)}(s)$ is the probability given by the SVM classifier trained on concept $M(u)$ and tested on shot s. t_u and t_s are the timestamps of term u and shot s (expressed as frame indexes). Parameter σ_a was set as 20 times the frame rate in all experiments, so to have a full width at half maximum of the Gaussian equal to $2\sqrt{2\ln(2)} \cdot 20 \approx 47$ seconds.

Given the definition of $P(s,u)$ the visual concept feature of a shot is a K-dimensional vector, defined as

$$\mathbf{v}(s) = \left[\sum_{u \in \mathcal{T}} \delta_{u,i} P(s,u) \right]_{i=1,\ldots,K} \tag{2}$$

where $\mathcal{T}$ is the set of all terms inside a video, $\delta_{u,i} \in \{0,1\}$ indicates whether term u belongs to the i-th concept group.

Textual semantic features: Textual concepts are as important as visual concepts to detect scene changes, and detected concept groups provide an ideal mean to describe topic changes in the text. Therefore, a textual concept feature vector, $\mathbf{t}(s)$, is built as the textual counterpart of $\mathbf{v}(s)$

$$\mathbf{t}(s) = \left[\sum_{u \in \mathcal{T}} \delta_{u,i} e^{-\frac{(t_u - t_s)^2}{2\sigma_a^2}} \right]_{i=1,\ldots,K} \tag{3}$$

We thus get a representation of how much each concept group is present in the transcript of a shot and in its neighborhood.

The overall feature vector $\mathbf{x}$ of a shot s is the concatenation of all the perceptual and conceptual features.

C. Embedding network

Given an input video, we would like to partition it into a set of sequences with the goal of maximizing the semantic coherence of the resulting segments. To this end, we would need a distance between shots feature vectors $\mathbf{x}$, which reflects the semantic similarity. Instead of explicitly defining this hypothetical distance, we learn an embedding function $\phi(\mathbf{x})$ that maps the feature vector of a shot to a space in which the Euclidean distance has the required semantic properties.

The ideal pairwise distance matrix would be $[\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2]_{i,j=1,\ldots,n} = [1 - \delta_{i,j}]_{i,j=1,\ldots,n}$, where $\delta_{i,j}$ is a binary function that indicates whether shot i and shot j belong to the same scene. For this reason, $\phi(\cdot)$ is learned such that a shot $\mathbf{x}_i$ of a specific scene should be closer to all the shots $\mathbf{x}_i^+$ of the same scene than to any shot $\mathbf{x}_i^-$ of any other scene, thus enforcing $\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_i^+)\|^2 < \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_i^-)\|^2$.

To this end, a Triplet Deep Network is designed. It consists of three base networks which share the same parameters, each taking the descriptor of a shot as input, and computing the desired embedding function $\phi(\cdot)$. The loss of the network for a training triplet $(\mathbf{x}_i, \mathbf{x}_i^+, \mathbf{x}_i^-)$ is defined by the Hinge loss as

$$L_i(\mathbf{w},\theta) = \max\big(0, \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_i^+)\|^2 + \\ + \big(1 - \| \phi(\mathbf{x}_i) - \phi(\mathbf{x}_i^-) \|^2\big)\big) \tag{4}$$

where $\mathbf{w}$ are the network weights, and θ are biases. The overall loss for a batch of N triplets is given by the average of the losses for each triplet, plus a L_2 regularization term on network weights to reduce over-fitting

$$L(\mathbf{w},\theta) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} L_i(\mathbf{w},\theta). \tag{5}$$

During learning, we perform mini-batch Stochastic Gradient Descent (SGD). At each iteration, we randomly sample N training triplets. For every triplet, we calculate the gradients over its components and perform back propagation according to Eq. 5. Details of the learning procedure are given in Algorithm 1.

The embedding network computes the projection $\phi(\mathbf{x})$ of a shot in the embedding space by means of three fully connected layers having, respectively, 500, 125 and 30 neurons, with ReLU activation. These are interleaved with Dropout layers [37], with retain probability 0.5, to reduce over-fitting. Since the embedding network is replicated three times to compute the final Triplet loss, Dropout is synchronized among the three branches, so that the same neurons are deactivated when computing $\phi(\mathbf{x}_i)$, $\phi(\mathbf{x}_i^+)$ and $\phi(\mathbf{x}_i^-)$.

The overall network is trained with momentum $\gamma = 0.9$ and regularization strength $\lambda = 0.0005$. The learning rate η is initially set to 0.01 and then scaled to 0.001 after 50 iterations. Training is performed in mini-batches containing $N = 500$ triplets. The amount of regularization and number of neurons were selected with a grid search on the BBC Planet Earth dataset, the most challenging we used.

Algorithm 1: Embedding space learning through Gradient Descent

Input : Number of iterations T; mini-batch size N; regularization strength λ; learning rate η; momentum γ; training triplets $(\mathbf{x}_i, \mathbf{x}_i^+, \mathbf{x}_i^-)_i$
Output: Optimized parameters $\mathbf{w}$ and θ
Initialize $\mathbf{w}$ according to [36] and θ to $\vec{0}$.
for $1 \le t \le T$ **do**
  Randomly select N training triplets
  for $1 \le i \le N$ **do**
    if $\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_i^+)\|^2 + \big(1 - \| \phi(\mathbf{x}_i) - \phi(\mathbf{x}_i^-) \|^2\big) > 0$ **then**
      $\mathbf{v}_w \leftarrow \gamma \mathbf{v}_w + \eta \left(\lambda \mathbf{w} + \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L_i}{\partial \mathbf{w}} \right)$
      $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{v}_w$
      $\mathbf{v}_\theta \leftarrow \gamma \mathbf{v}_\theta + \eta \left(\frac{1}{N} \sum_{i=1}^{N} \frac{\partial L_i}{\partial \theta} \right)$
      $\theta \leftarrow \theta - \mathbf{v}_\theta$
    end
  end
end

D. Temporal Aware Clustering

To obtain a temporal segmentation of the video we require segments to be as semantically homogeneous as possible. Inspired by k-means, a cluster homogeneity may be described by the sum of squared distances between cluster elements and its centroid, called within-group sum of squares (WSS). A reasonable objective is thus minimizing the total within-group sum of squares (TWSS), i.e. the sum of the WSS for all clusters. Differently from k-means, we would also like to find the number of clusters, with the additional constraint of them being temporally continuous intervals. Minimizing the TWSS alone would lead to the trivial solution of having a single shot in each sequence, so a penalty term needs to be added to avoid over-segmentation.

The problem we need to solve is thus

$$\min_{m,t_1,\ldots,t_m} \sum_{i=0}^{m} WSS_{t_i,t_{i+1}} + Cg(m,n) \tag{6}$$

where m is the number of change points by which the input video is segmented, t_i is the position of i-th change point (t_0 and t_{m+1} are the beginning and the end of the video respectively), and $WSS_{t_i,t_{i+1}}$ is the within-group sum of squares of the i-th segment in the embedding space. The term $g(m,n) = m(\log(n/m)+1)$ is a Bayesian information criterion penalty [38] parametrized with the number of segments m and the number of shots in the video n, which aims to reduce the over-segmentation effect. Parameter C tunes the relative importance of the penalty: higher values of C penalize segmentations with too many segments.

The sum of squared distances between a set of points and their mean can be expressed as a function of the pairwise squared distances between the points alone. Therefore, the within-group sum of squares can be written as

$$\begin{aligned} WSS_{t_i,t_{i+1}} &\triangleq \sum_{t=t_i}^{t_{i+1}-1} \|\phi(\mathbf{x}_t) - \mu_i\|^2 \\ &= \frac{1}{2(t_{i+1}-t_i)} \sum_{i,j=t_i}^{t_{i+1}-1} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 \end{aligned} \tag{7}$$

where μ_i is the mean of each scene, defined as:

$$\mu_i = \frac{1}{t_{i+1}-t_i} \sum_{t=t_i}^{t_{i+1}-1} \phi(\mathbf{x}_t) \tag{8}$$

The temporal clustering objective (Eq. 6) can, in this way, be minimized using a Dynamic Programming approach. First, $WSS_{k,k+d}$ is computed for each possible starting point k and segment duration d. Then, the objective is minimized by iteratively computing the best objective value for the first $j \in [1,n]$ shots and $m \in [0,n-1]$ change points

$$D_{m,j} = \min_{k=m,\ldots,j-1} (D_{m-1,k} + WSS_{k,j}) \tag{9}$$

having set $D_{0,j} = WSS_{0,j}$.

The optimal number of change points is then selected as $m^* = \arg\min_m D_{m,n} + Cg(m,n)$, and the best segmentation into scenes is reconstructed by backtracking.

IV. Scene presentation with aesthetically pleasing thumbnails

The availability of the video structure, i.e. its layered decomposition in scenes, shots, and keyframes, is not just an indexing tool for easier navigation or section selection but may be employed as an extremely effective presentation aid. Given a set of videos relevant to a query term q, we can leverage the scene structure to point to the most relevant part of the video and use the two lower layers (shots and keyframes) to cheaply select an aesthetically pleasing and semantically significant presentation.

For each relevant video, we build a ranking function which returns an ordered set of (video, scene, thumbnail) triplets. In each triplet, the retrieved scene must belong to the retrieved video and should be as consistent as possible with the given query. Moreover, the returned thumbnail must belong to the given scene and should be representative of the query as well as aesthetically remarkable.

Given a query q, we first match q with the most similar detected concept u, using the Word2Vec embedding. If the query q is composed of more than one word, the mean of the embedded vectors is used. The probability function $P(s,u)$, defined in Eq. 1, accounts for the presence of a particular unigram in one shot and is, therefore, useful to rank scenes given a user query. Each scene a inside the relevant set is then assigned a score according to the following function:

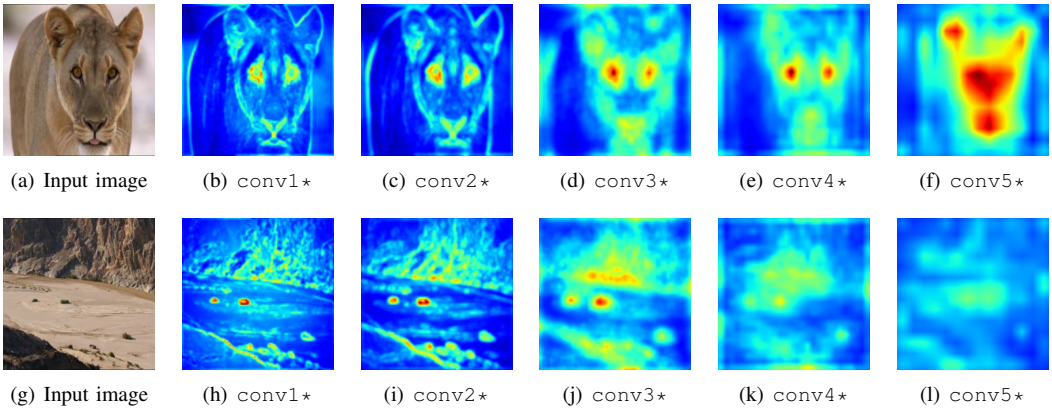
$$R_a(q) = \max_s \left(\alpha P(s,u) + (1-\alpha) \max_{d \in s} A(d) \right) \tag{10}$$

where s is a shot inside the given scene, and d represents a keyframe extracted from a given shot. Parameter α tunes the relative importance of semantic representativeness with respect to function $A(d)$, which is a measure of the aesthetic beauty. The final retrieval results is a collection of scenes, ranked according to $R_a(q)$, each one represented with the keyframe that maximizes the second term of the score.

A. Thumbnail selection

In order to evaluate how much aesthetically pleasing a thumbnail is, we should account for low-level characteristics, like color, edges, and sharpness, as well as high-level features, such as the presence of a clearly visible and easily recognizable object. We claim that the need for low and high-level features is an excellent match with the hierarchical nature of CNNs: convolutional filters, indeed, are known to capture low level as well as high-level characteristics of the input image. This has also been proved by visualization and inversion techniques, like [39] and [40].

Being activations from convolutional filters discriminative for visual representativeness, a ranking strategy could be set up to learn their relative importance given a dataset of user preferences. However, medium sized CNNs, like the VGG-16 model, contain more than 4000 convolutional filters: this makes the use of raw activations infeasible with small datasets. Moreover, maps from different layers have different sizes, due to the presence of pooling layers. To overcome this issue, we resize each activation map to fixed size with bilinear interpolation,



(a) Input image (b) conv1⋆ (c) conv2⋆ (d) conv3⋆ (e) conv4⋆ (f) conv5⋆

(g) Input image (h) conv1⋆ (i) conv2⋆ (j) conv3⋆ (k) conv4⋆ (l) conv5⋆

Fig. 3. Hypercolumn features extracted from two sample images. Each map represents the mean activation map over a set of layers: (b) and (h) are built using layers conv1_1 and conv1_2, (c) and (i) with layers conv2_1 and conv2_2; (d) and (j) with conv3_1, conv3_2 and conv3_3; (e) and (k) with conv4_1, conv4_2, and conv4_3. Finally, (f) and (i) are built using layers conv5_1, conv5_2 and conv5_3. Best viewed in color.

and average feature maps coming from the different layers, inspired by the Hypercolumn approach presented in [41]. Since the user usually focuses on the center of the thumbnail rather than its exterior, each map is multiplied by a normalized gaussian density map, centered on the center of the image and with horizontal and vertical standard deviations equal to $\sigma_b \cdot l$, where $l \times l$ is the size of the CNN input. Parameter σ_b was set to 0.3 in all our experiments.

Following the VGG-16 architecture [28], we build five hypercolumn maps, each one summarizing convolutional layers before each pooling layer: the first one is computed with activation maps from layers conv1_1 and conv1_2; the second one with conv2_1 and conv2_2; the third with conv3_1, conv3_2 and conv3_3; the fourth with conv4_1, conv4_2 and conv4_3; the last with conv5_1, conv5_2 and conv5_3. An example of the resulting activation maps is presented in Fig. 3: as it can be seen, both low level and high level layers are useful to distinguish between a significant and non significant thumbnail.

To learn the relative contribution of each hypercolumn map, we rank thumbnails from each scene according to their visual representativeness and learn a linear ranking model. Given a dataset of scenes $\{a_i\}_{i=0}^{m}$, each with a ranking r_i^*, expressed as a set of pairs (d_i, d_j), where thumbnail d_i is annotated as more relevant than thumbnail d_j, we solve the following problem:

$$\min_{\mathbf{w_r},\epsilon} \frac{1}{2}\|\mathbf{w_r}\|^2 + C_r \sum_{i,j,k} \epsilon_{i,j,k} \tag{11}$$

subject to

$$\begin{aligned} &\forall (d_i, d_j) \in r_1^* : \mathbf{w_r}\tau(d_i) \geq \mathbf{w_r}\tau(d_j) + 1 - \epsilon_{i,j,1} \\ &\dots \\ &\forall (d_i, d_j) \in r_m^* : \mathbf{w_r}\tau(d_i) \geq \mathbf{w_r}\tau(d_j) + 1 - \epsilon_{i,j,m} \\ &\forall i,j,k : \epsilon_{i,j,k} \geq 0 \end{aligned} \tag{12}$$

where $\tau(d_i)$ is the feature vector of thumbnail d_i, which is composed by the mean and standard deviation of each hypercolumn map extracted from the thumbnail itself. C_r allows trading-off the margin size with respect to the training error. The objective stated in Eq. 11 is convex and equivalent to that of a linear SVM on pairwise difference vectors $\tau(d_i) - \tau(d_j)$ [42]. The final aestethic score for keyframe d is given by $A(d) = \mathbf{w_r}\tau(d)$.

V. Dealing with Subjectivity

A. Evaluation protocol

Measuring scene detection performance is significantly different from measuring shot detection performance. Indeed, classical boundary detection scores, such as Precision and Recall, fail to convey the true perception of an error, which is different for an off-by-one shot or for a completely missed scene boundary.

Better fitting measures were proposed in [43]: Coverage measures the quantity of shots belonging to the same scene correctly grouped together, while Overflow evaluates to what extent shots not belonging to the same scene are erroneously grouped together. An F-Score measure, F_{co}, can be defined to combine Coverage and Overflow in a single measure, by taking the harmonic mean of Coverage and 1-Overflow. These measures are nevertheless known to have some drawbacks, which may affect the evaluation. As also noted in [44], F_{co} is not symmetric, leading to unusual phenomena in which an early or late positioning of the scene boundary, of the same amount of shots, may lead to strongly different results. Moreover, the relation of Overflow with the previous and next scenes creates unreasonable dependencies between an error and the length of a scene observed many shots before it.

An alternative symmetric measure, based on intersection over union, was proposed in [11] and was proved to be more effective. Here, a scene in a video is represented as a closed interval, where the left bound of the interval is the starting frame of the scene, and the right bound is the ending frame of the sequence. The intersection over union of two scenes a and b, IoU(a, b), can therefore be written as

$$\text{IoU}(a,b) = \frac{a \cap b}{a \cup b} \tag{13}$$

A segmentation of a video into scenes can be seen as a set of non-overlapping scenes, whose union is the set of frames of the

video. By exploiting this relation, [11] defines the intersection over union of two segmentations $\mathcal{A}$ and $\mathcal{B}$ as

$$\overline{\text{IoU}}(\mathcal{A},\mathcal{B}) = \frac{1}{2}\left(\frac{1}{\#\mathcal{A}}\sum_{a\in\mathcal{A}}\max_{b\in\mathcal{B}}\text{IoU}(a,b) + \right. \\ \left. + \frac{1}{\#\mathcal{B}}\sum_{b\in\mathcal{B}}\max_{a\in\mathcal{A}}\text{IoU}(a,b)\right) \quad (14)$$

It is easy to see that, considering the particular case of $\mathcal{A}$ being the ground-truth annotation and $\mathcal{B}$ being the segmentation produced by an algorithm, Eq. 14 computes, for each ground-truth scene, the maximum intersection over union with the detected scenes. Then, the same is done for detected scenes against ground-truth ones, and the two quantities are averaged.

B. Finding an agreement between annotations

Being scene detection a considerably subjective task, it is often the case that the same video is annotated differently by more than one annotator. This results in a set of annotations for each video, while an automatic model should produce a single segmentation, as consistent as possible with all given human annotations. Therefore, the prediction of a model should not be compared with each given annotation, but with the segmentation which is most similar to all given annotations.

In this paragraph, we investigate the problem of finding the segmentation which maximizes the agreement with respect to a set of annotations. Formally, given a set of m annotations $\mathbb{S}$, we aim at finding the segmentation $\mathcal{A}^*$ which maximizes the average intersection over union with respect to $\mathbb{S}$

$$\mathcal{A}^* = \arg\max_{\mathcal{A}} \frac{1}{m}\sum_{\mathcal{S}\in\mathbb{S}}\overline{\text{IoU}}(\mathcal{A},\mathcal{S}) \quad (15)$$

Algebraic manipulation reveals that given Eq. 14, this maximization is equivalent to find $\mathcal{A}$ to maximize

$$J(\mathcal{A}) = \underbrace{\frac{1}{\#\mathcal{A}}\sum_{a_i\in\mathcal{A}}\sum_{\mathcal{S}\in\mathbb{S}}\max_{s_j\in\mathcal{S}}(\text{IoU}(a_i,s_j))+}_{J_1(\mathcal{A})} \\ \underbrace{+\sum_{\mathcal{S}\in\mathbb{S}}\frac{1}{\#\mathcal{S}}\sum_{s_j\in\mathcal{S}}\max_{a_i\in\mathcal{A}}(\text{IoU}(a_i,s_j))}_{J_2(\mathcal{A})} \quad (16)$$

Given a video with n shots, a brute force resolution of Eq. 15 would require testing every possible annotation compatible with the shots, thus leading to a time complexity $O(2^n)$.

We approximate the maximization of Eq. 16 as a longest-path problem in a graph. Let $\mathcal{G}(V,E)$ be a weighted directed acyclic graph with vertices representing shots of the video, $V=\{1,2,...,n\}$, and edges connecting each vertex to all the other nodes with greater values, $E=\{(i,j): i,j\in V, i<j\}$. A valid segmentation of the video can be seen as a path $\mathcal{P}$ in $\mathcal{G}$ having source node 1 and target node n. In this configuration, each edge (i,j) in the path corresponds to a scene in the segmentation.

If the number of scenes in the segmentation (i.e. its cardinality) is known in advance, then $J_1(\mathcal{A})$ can be decomposed into a sum of edge weights, as follows:

$$J_1(\mathcal{A}) = \sum_{(i,j)\in\mathcal{P}} w_1(i,j) \quad (17)$$

where the weight of an edge (i,j) is

$$w_1(i,j) = \frac{1}{l}\sum_{\mathcal{S}\in\mathbb{S}}\max_{s_k\in\mathcal{S}}(\text{IoU}([i,j],s_k)) \quad (18)$$

$[i,j]$ is the scene corresponding to edge (i,j), and l is the length of the segmentation (indicated as $\#\mathcal{A}$ in Eq. 16).

Unfortunately, $J_2(\mathcal{A})$ cannot be factored in the same way. However, we notice that it can be rewritten as follows:

$$J_2(\mathcal{A}) = \sum_{t=1}^{\#\mathcal{P}} w_2(\mathcal{P},t) \quad (19)$$

where

$$w_2(\mathcal{P},t) = \left(\sum_{\mathcal{S}\in\mathbb{S}}\frac{1}{\#\mathcal{S}}\sum_{s_j\in\mathcal{S}}\max_{a_i\in\mathcal{A}_t}(\text{IoU}(a_i,s_j)) + \right. \\ \left. - \sum_{\mathcal{S}\in\mathbb{S}}\frac{1}{\#\mathcal{S}}\sum_{s_j\in\mathcal{S}}\max_{a_i\in\mathcal{A}_{t-1}}(\text{IoU}(a_i,s_j))\right) \quad (20)$$

where $\mathcal{A}_t$, at each step t of the path, is the set of scenes corresponding to already visited nodes.

The maximization of $J_1 + J_2$ can be addressed as the problem of finding the longest path of length l in $\mathcal{G}$, and approximately solved through a Dynamic Programming strategy, by pretending that $J_2(\mathcal{A})$ is a sum of edge weights (even though $w_2(\mathcal{P},t)$ actually depend on the specific path).

Having chosen a path length l, For each $1\le i\le l$, and every vertex v, we compute $D[i,v]$ where $D[i,v]$ is the weight of the longest walk of length exactly i starting at vertex 1 and ending at vertex v. To compute $D[l,n]$, we use the following relation:

$$D[i+1,v] = \max_{x\in Pred(v)}(D[i,x] + w_1(\mathcal{P},t) + w_2(x,v)) \quad (21)$$

where $Pred(v)$ is the predecessor set of vertex v, and $w_1(\mathcal{P},t)$ is computed by considering the path used in $D[i,x]$, plus node v. The best path from vertex 1 to vertex n with length l is then reconstructed by backpropagation, and the same procedure is repeated for $1\le l\le n$. $\mathcal{A}^*$ is then selected as the path of maximum cost.

Since for each l the Dynamic Programming algorithm has time complexity $O(l\cdot n)$, the overall complexity is $O(n^3)$, being n the number of shots in the video.

It is worth mentioning that to assess the quality of the proposed approximation, we tested it on 11.000 randomly generated sequences for which $\mathcal{A}^*$ has been computed with brute-force, with length $n=100$, a number of scenes varying from 2 and 7, and with a number of annotations m ranging from 2 to 10. 98.4% of the generated segmentations were correct, while the mean absolute error, in terms of $\overline{\text{IoU}}$, was $1.16\cdot 10^{-5}$.

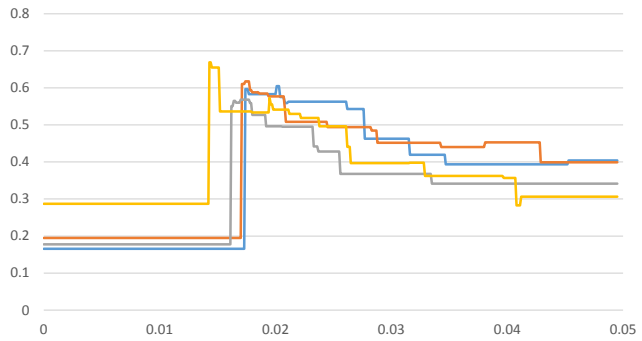


Fig. 4. Parameter C influence. Variation of $\overline{\text{IoU}}$ with respect to C for different videos of BBC Planet Earth.

VI. Experimental evaluation

We compare our scene detection approach against state-of-the-art algorithms from the literature which are applicable and perform experiments to assess the role of the proposed features and embedding. In addition, we address the subjective nature of scene detection by using our embedding to learn the style of different annotators, and the segmentation provided by the algorithm described in Section V-B. Finally, we evaluate the effectiveness of the proposed video retrieval strategy, both quantitatively and qualitatively.

To perform shot detection, we use an off-the-shelf shot detector [45] which relies on SURF descriptors and HSV color histograms. Abrupt transitions are detected by thresholding a distance measure between frames, while longer gradual transitions are detected by means of the derivative of the moving average of the aforesaid distance.

A. Datasets

To test the temporal segmentation capabilities of our model, we run a series of experimental tests on the Ally McBeal dataset released in [46], which contains the temporal segmentation into scenes of four episodes of the first season of Ally McBeal. The dataset contains 2660 shots and 160 scenes, which correspond to, on average, 61 million training triplets and 6.7 million test triplets. Closed captions were used as a transcript.

We also employ the BBC Planet Earth dataset [11], which contains the segmentation into scenes of eleven episodes from the BBC documentary series *Planet Earth* [47]. Each episode is approximately 50 minutes long, and the whole dataset contains around 4900 shots and 670 scenes. This corresponds, on average, to roughly 125 million training triplets and 1.2 million test triplets for each video. Each video is also provided with the corresponding transcript. To augment the dataset, and test the proposed way to deal with different annotations, we asked four more annotators to segment each video in the dataset.

It is worth to mention that the aforementioned datasets are considerably different, both because of the nature of the videos they contain, and because of the kind of annotation. Indeed, the annotation in Ally McBeal reproduces the partitioning of a TV series into scenes, which is mainly based on the dialogues and the location of the scenes, while the annotation of the BBC Planet Earth episodes is far more difficult to reproduce, since it relies on the semantics of the video and of the speaker transcript.

TABLE I
Prior methods vs. our approach for Scene detection on the Ally McBeal dataset.

Episode	STG [10]	NW [7]	SDN [11]	Ours
Ep. 1	0.65	0.38	0.37	**0.98**
Ep. 2	0.70	0.36	0.34	**0.86**
Ep. 3	0.72	0.40	0.36	**0.94**
Ep. 4	0.63	0.36	0.30	**0.96**
Average	0.68	0.38	0.34	**0.94**

TABLE II
Prior methods vs. our approach for Scene detection on the BBC Planet Earth dataset.

Episode	STG [10]	NW [7]	SDN [11]	Ours
From Pole to Pole	0.42	0.35	0.50	**0.72**
Mountains	0.40	0.31	0.53	**0.75**
Fresh Water	0.39	0.34	0.52	**0.67**
Caves	0.37	0.33	0.55	**0.62**
Deserts	0.36	0.33	0.36	**0.62**
Ice Worlds	0.39	0.37	0.51	**0.73**
Great Plains	0.46	0.37	0.47	**0.63**
Jungles	0.45	0.38	0.51	**0.62**
Shallow Seas	0.46	0.32	0.51	**0.74**
Seasonal Forests	0.42	0.20	0.38	**0.65**
Ocean Deep	0.34	0.36	0.48	**0.65**
Average	0.41	0.33	0.48	**0.67**

B. Comparison with the State of the art

The performance of our method depends on the selection of hyperparameter C in the temporal clustering objective (Eq. 6), which yields a trade-off between over- and under-segmentation. Figure 4 reports an example of the variation of intersection over union with respect to C for different videos of the BBC Planet Earth dataset. Clearly, each chart presents a global maximum, but the optimal C value changes from video to video. This would lead to a sub-optimal choice of C if selected with cross-validation. The temporal clustering selects a scene for each shot for low values of C and as soon as the parameter goes over a certain value, the clustering begins to provide very significant groupings. For this reason, our choice of C is video dependent and, using a step of 0.001, we increase the C value until the number of clusters is lower than the number of shots in the video. This may be sub-optimal, but the results are totally independent of the training phase and do not require assumptions on the specific video.

Our model is compared against three recent proposals for video decomposition: [10], which uses a variety of visual and audio features merged in a Shot Transition Graph (STG); [7], that combines low-level color features with the Needleman-Wunsch (NW) algorithm, and [11], which exploits visual features extracted with a CNN and Bag-of-Words histograms extracted from the transcript, which are merged in a Siamese Deep Network (SDN).

TABLE III
EVALUATION ON THE ALLY MCBEAL DATASET, WHEN TRAINING ON BBC PLANET EARTH AND ON ALLY MCBEAL.

Episode	Train on BBC PE	Train on AMB
Ep. 1	0.87	0.98
Ep. 2	0.81	0.86
Ep. 3	0.93	0.94
Ep. 4	0.91	0.96
Average	**0.88**	**0.94**

TABLE IV
SCENE DETECTION PERFORMANCE WITH VARIOUS FEATURES.

Features/Embedding	Ally McBeal	BBC Planet Earth
VA	0.898	0.638
VA+A	0.915	0.654
VA+A+QoS	0.914	0.656
VA+A+QoS+T	0.921	0.657
VA+A+QoS+T+VS	0.925	0.660
VA+A+QoS+T+VS+TS	0.935	0.672

We use a web service made by the authors of [10] and the source code of [11] provided by its authors, and re-implement the method in [7]. Parameters of all methods were selected to maximize the performance on the training set. The shot detector we use is the same as [10], so performance results are not affected by differences in the shot detection phase.

In Tables I and II we compare the performance of our method with the aforementioned methods, on Ally McBeal and BBC Planet Earth, using annotations provided in [11]. All experiments were conducted in a leave-one-out setup, using one video for testing and all other videos from the same dataset as training. Reported results suggest that our embedding strategy is able to deal effectively with different kinds of videos and of annotations, learning the specific annotation style of each dataset. On all datasets, indeed, our method outperforms all the approaches it has been compared to.

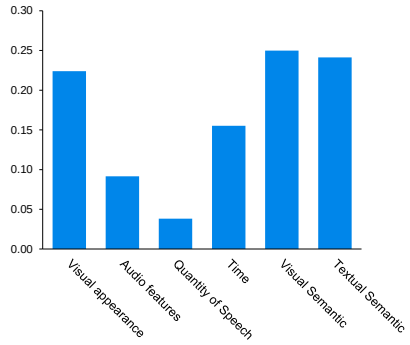
To test the generality of the learned embedding, we also perform a second experiment, in which we train a model on the entire BBC Planet Earth dataset, and test it on the Ally McBeal series. The objective of the experiment is, therefore, to investigate how a model learned on a particular kind of videos can generalize to another category. Results are shown in Table III: even if the embedding has been learned on documentaries, and even if in this case visual semantic features are less effective, the model is still able to generalize to unseen kinds of videos.

C. Feature and embedding comparisons

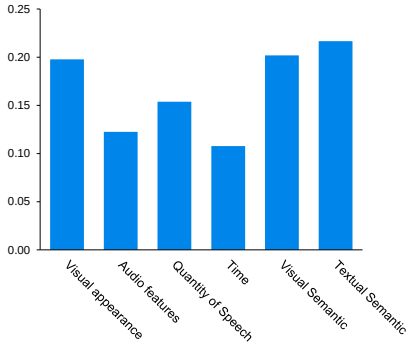
To test the role of the proposed features and embedding, we conducted two additional tests. In the first one, whose results are reported in Table IV the triplet embedding is trained using an increasing set of features: visual appearance (VA), Audio (A), Quantity of Speech (QoS), Time (T), Visual and Textual semantic (VS, TS). Results are reported in terms of mean $\overline{\text{IoU}}$. Each feature, when added, resulted in a performance improvement.

TABLE V
SCENE DETECTION PERFORMANCE WITH DIFFERENT EMBEDDINGS.

Embedding	Ally McBeal	BBC Planet Earth
LSTM	0.82	0.58
Siamese	0.87	0.49
Triplet	0.94	0.67



(a) Ally McBeal [46]



(b) BBC Planet Earth [11]

Fig. 5. Feature importance analysis. For each dataset, the relative importance of each feature is reported. See Section VI-D for details.

In the second experiment, we use all features and test different embeddings. We test a Siamese network with the same architecture and the same number of neurons of the Triplet network. We also train an LSTM network: the descriptor of each shot is fed to a fully connected network with the same structure of the embedding network, and then to an LSTM layer with memory size 10 and output size 1. The network is trained to predict, at each time step, the presence of a scene boundary, with a binary cross entropy loss. Results, reported in Table V show that the proposed Triplet strategy is superior both to the Siamese and the LSTM approach. In conclusion, all features are important but the embedding architecture boosts performances.

D. Feature importance analysis

We evaluate the relative importance and effectiveness of each of the proposed features in the final embedding. In the following, we will define the importance of a feature as the extent to which a variation of the feature can affect the embedding. Consider, for example, a linear embedding model,

in which each dimension of the embedding, ϕ_i, is given by the following equation

$$\phi_i(\mathbf{x}) = w_i^T \mathbf{x} + \theta_i \tag{22}$$

where w_i and θ_i are respectively the weight vector and the bias for the i-th dimension of the embedding, while $\mathbf{x}$ is the concatenation of the proposed features. In this case, it is easy to see that the magnitude of elements in w_i defines the importance of the corresponding features. Each feature is indeed multiplied by a subset of the w_i vector, and the absolute values in w_i encode the importance of each of those features. In the extreme case of a feature which is always multiplied by 0, it is straightforward to see that that feature is ignored by the i-th dimension of the embedding and has, therefore, no importance, while a feature with high absolute values in w will have a considerable effect.

In our case, $\phi_i(\cdot)$ is a highly non-linear function of the input, thus the above reasoning is not directly applicable. Instead, given an shot $\mathbf{x}_j$, we can approximate $\phi_i(\mathbf{x}_j)$ in the neighborhood of $\mathbf{x}_j$ as follows

$$\phi_i(\mathbf{x}_j) \approx \nabla\phi_i(\mathbf{x}_j)^T \mathbf{x} + \theta_i \tag{23}$$

An intuitive explanation of this approximation is that the magnitude of the partial derivatives indicates which features need to be changed to affect the embedding. Also notice that Eq. 23 is equivalent to a first-order Taylor expansion.

To get an estimation of the importance of each feature regardless of the choice of $\mathbf{x}_j$, we can average the element-wise absolute values of the gradient computed in the neighborhood of each test sample

$$w_i = \frac{1}{N}\sum_{j=1}^{N}\left[\left|\frac{\partial\phi_i}{\partial x^1}(\mathbf{x}_j)\right|, \left|\frac{\partial\phi_i}{\partial x^2}(\mathbf{x}_j)\right|, \cdots, \left|\frac{\partial\phi_i}{\partial x^d}(\mathbf{x}_j)\right|\right] \tag{24}$$

where d is the dimensionality of $\mathbf{x}_j$. Then, to get the relative importance of each proposed feature, we average the values of w_i corresponding to that feature. The same is done for each of the dimensions of the embedding, and results are then averaged. The resulting importances for each features are finally L_1 normalized.

Figure 5 reports the relative importance of our features on Ally Mc Beal and BBC Planet Earth. It is easy to notice that all features give a valuable contribution to the final result. In TV-series and documentaries visual appearance and semantic features are the most relevant cues. The quantity of speech plays an important role in documentaries, confirming that in this kind of videos a pause in the speaker discourse is often related to a scene boundary, while in TV series appearance and conceptual features are often enough to perform scene detection. It is also worth to notice that when the annotation to be learned is challenging, like in the BBC Planet Earth dataset, every feature becomes relevant, thus confirming the effectiveness of the proposed features.

E. Qualitative results

To give a qualitative indication of the results, in Figure 6 we report the temporal segmentation provided by our method and all the methods we compare to, as well as the ground truth annotation, on a part of the first episode of BBC *Planet Earth*. Each thumbnail represents the middle frame of a shot, and the first row is the ground truth segmentation. A change in color underlines a change of scene.

Compared to the human annotation, our method identifies the exact change point in four cases, and merges together adjacent ground truth scenes in one case. On the other hand, the STG method in [10] is able to identify some scene changes correctly but creates short scenes with just one shot. The NW method in [7] does not show this over segmentation phenomena, but creates unreasonable scene changes. Finally, the Siamese approach of [11] can actually identify correct scene boundaries in some cases, still the segmentation provided by our method looks more consistent with the human annotation.

Finally, we also investigate the execution times of the compared methods. Given a one hour video, our implementations of NW and SDN require, respectively, 8 and 33 minutes to compute the final scene boundaries. For the STG approach, instead, we consider the running time of a sample run on their web-service, which was roughly half the duration of the video. Our method, instead, is definitely the most time-consuming, as it requires more than two times the video duration, due to the complex feature extraction pipeline which requires the creation, on the fly, of a set of visual classifiers. The time required to download Imagenet images has not been taken into account.

F. Evaluation with multiple annotators

As stated at the beginning of this section, we extended the BBC Planet Earth dataset by collecting four more annotations. This, along with that provided in [11], results in a set of five different annotations, which are used to investigate both the role of subjectivity in scene detection and the capabilities of our embedding to learn a particular annotation style. The choice of this particular dataset is motivated by the fact that in documentaries scene boundaries are less objective than in movies and TV-shows, and are also related to changes in topic. Collected annotations differ in terms of granularity (with some annotators putting scene boundaries for minor topic or place changes, and others building longer scenes) and also in terms of localization (given that sometimes the exact change point is not easy to identify).

We first run the algorithm described in Section V-B to get the the segmentation which maximally agrees with all the given annotations. The resulting segmentation presents a mean $\overline{\text{IoU}}$ with the five annotators of 0.762. This represents an upper-bound for scene detection algorithms trained on this set of annotations, given that no segmentation could achieve a better result (ignoring the approximation introduced by our algorithm, which is negligible).

The proposed embedding is then trained and tested on all annotations, as well as on the agreement given by the Dynamic Programming algorithm, always keeping a leave-one-out setup among the eleven videos. Results are reported in Table VI: clearly, higher $\overline{\text{IoU}}$ values are obtained when training and testing on the same annotator, and this suggests that our model

Fig. 6. Qualitative results on the first episode of BBC *Planet Earth*. Each row represents the segmentation generated by a method, and a change in color represents a change of scene. First row (blue) is the ground truth, second row (green) is our method, remaining (red) are, respectively, [10], [7] and [11] (best viewed in color).

TABLE VI
SCENE DETECTION PERFORMANCE ON THE BBC PLANET EARTH DATASET, TRAINING AND TESTING ON DIFFERENT ANNOTATORS AND THE MAXIMUM AGREEMENT SEGMENTATION.

		Test on					
		Annotator 1	**Annotator 2**	**Annotator 3**	**Annotator 4**	**Annotator 5**	**Agreement**
Train on	**Annotator 1**	0.669	0.528	0.428	0.474	0.416	0.475
	Annotator 2	0.474	0.654	0.437	0.505	0.418	0.541
	Annotator 3	0.455	0.546	0.572	0.481	0.404	0.420
	Annotator 4	0.481	0.536	0.436	0.606	0.436	0.433
	Annotator 5	0.468	0.538	0.432	0.492	0.545	0.411
	Agreement	0.605	0.585	0.547	0.580	0.454	0.556

was indeed able to capture some features of the segmentation style of an annotator, such as the level of granularity. At the same time, training on the maximum agreement annotation leads, on average, to better $\overline{\text{IoU}}$ scores when testing on the five human annotators.

G. Thumbnail selection evaluation

On a different note, we conducted a series of experiments regarding the proposed retrieval strategy. Since aesthetic quality is subjective, three different users were asked to mark all keyframes either as aesthetically relevant or nonrelevant for the scene they belong to. For each shot, the middle frame was selected as the keyframe. Annotators were instructed to consider the relevance of the visual content as well as the quality of the keyframe in terms of color, sharpness, and blurriness. Each keyframe was then labeled with the number of times it was selected, and a set of (d_i, d_j) training pairs was built according to the given ranking, to train our aesthetic ranking model.

For comparison, an end-to-end deep learning approach (*Ranking CNN*) was also tested. In this case, the last layer of a pre-trained VGG-16 network was replaced with just one neuron, and the network was trained to predict the score of each shot, with a Mean Square Error loss. Both the Ranking CNN model and the proposed Hypercolumn-based ranking were trained in a leave-one-out setup, using ten videos for training and one for test from the BBC Planet Earth collection.

TABLE VII
AESTHETIC RANKING: AVERAGE PERCENT OF SWAPPED PAIRS ON THE BBC PLANET EARTH DATASET (LOWER IS BETTER).

Episode	**Ranking CNN**	**Hypercolumns Ranking**
From Pole to Pole	8.23	4.10
Mountains	12.08	7.94
Fresh Water	12.36	8.11
Caves	9.98	8.76
Deserts	13.90	9.35
Ice Worlds	6.62	4.33
Great Plains	10.92	9.63
Jungles	12.28	7.43
Shallow Seas	10.91	6.22
Seasonal Forests	9.47	4.82
Ocean Deep	10.73	5.75
Average	**10.68**	**6.95**

Table VII reports the average percent of swapped pairs: as it can be seen, our ranking strategy is able to overcome the Ranking CNN baseline and features a considerably reduced error percentage. This confirms that low and high-level features can be successfully combined together and that high

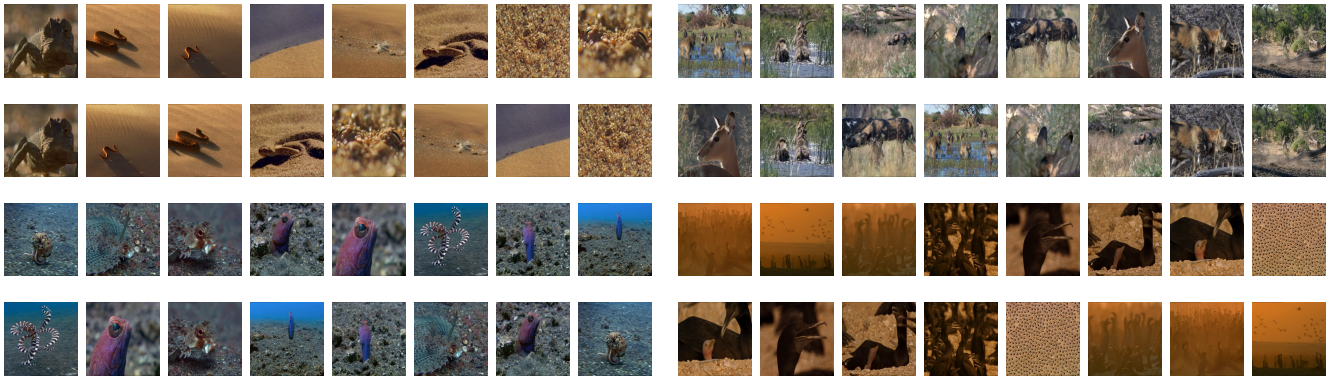


Fig. 7. Ranking of four sample shot sequences. First and third row report all shots in temporal order, while second and fourth row show the produced aesthetic ranking in descending order (with leftmost thumbnails predicted to be more aesthetically pleasing than rightmost ones). Thumbnails with a centered and clearly visible object are preferred against blurred and low-quality frames (best viewed in color).

features alone, such as the ones the Ranking CNN is able to extract from its final layers, are not sufficient. Figure 7 shows the ranking results of four shot sequences: as requested in the annotation, the SVM model preferred thumbnails with good quality and a clearly visible object in the middle. Qualitative results are also available in the demo interface hosted at http://imagelab.ing.unimore.it/neuralstory, where the reader can test the proposed retrieval system on textual queries.

VII. Conclusion

This paper presented a new approach for scene detection in broadcast videos. Our proposal builds a set of domain specific concept classifiers and learns an embedding space via a Triplet Deep Network, which considers visual as well as textual concepts extracted from the video corpus. We showed the effectiveness of our approach compared to different techniques via quantitative experiments and demonstrated the effectiveness of the proposed features. The subjectivity of the task was also taken into account, by demonstrating that the proposed embedding can adapt to different annotators, and by providing an algorithm to maximize the agreement between a set of annotators. As a potential application of scene detection, we also introduced its use in retrieval results presentation, allowing the simultaneous use of semantic and aesthetic criteria.

References

[1] "Sandvine Global Internet Phenomena Report," Jun. 2016.

[2] "TiVo Third Annual Millennial Video Entertainment Survey," Dec. 2015.

[3] C. Petersohn, *Temporal Video Segmentation*. Jrg Vogt Verlag, 2010.

[4] S. Zhu and Y. Liu, "Automatic scene detection for advanced story retrieval," *Expert Systems with Applications*, vol. 36, no. 3, Part 2, pp. 5976–5986, 2009.

[5] A. Hanjalic, R. L. Lagendijk, and J. Biemond, "Automated high-level movie segmentation for advanced video-retrieval systems," *IEEE TCSVT*, vol. 9, no. 4, pp. 580–588, 1999.

[6] C. Liu, D. Wang, J. Zhu, and B. Zhang, "Learning a Contextual Multi-Thread Model for Movie/TV Scene Segmentation," *IEEE TMM*, vol. 15, no. 4, pp. 884–897, 2013.

[7] V. T. Chasanis, C. Likas, and N. P. Galatsanos, "Scene detection in videos using shot clustering and sequence alignment," *IEEE TMM*, vol. 11, no. 1, pp. 89–100, 2009.

[8] M. M. Yeung, B.-L. Yeo, W. H. Wolf, and B. Liu, "Video browsing using clustering and scene transitions on compressed sequences," in *IS&T/SPIE Symp. Electron. Imaging*, 1995, pp. 399–413.

[9] Z. Rasheed and M. Shah, "Detection and representation of scenes in videos," *IEEE TMM*, vol. 7, no. 6, pp. 1097–1105, 2005.

[10] P. Sidiropoulos, V. Mezaris, I. Kompatsiaris, H. Meinedo, M. Bugalho, and I. Trancoso, "Temporal video segmentation to scenes using high-level audiovisual features," *IEEE TCSVT*, vol. 21, no. 8, pp. 1163–1177, 2011.

[11] L. Baraldi, C. Grana, and R. Cucchiara, "A deep siamese network for scene detection in broadcast videos," in *ACM MM*. ACM, 2015, pp. 1199–1202.

[12] Y. Wu, X. Shen, T. Mei, X. Tian, N. Yu, and Y. Rui, "Monet: A system for reliving your memories by theme-based photo storytelling," *IEEE TMM*, 2016.

[13] G. Kim, L. Sigal, and E. P. Xing, "Joint summarization of large-scale collections of web images and videos for storyline reconstruction," in *CVPR*. IEEE, 2014, pp. 4225–4232.

[14] A. Aner and J. R. Kender, "Video summaries through mosaic-based shot and scene clustering," in *ECCV*. Springer, 2002, pp. 388–402.

[15] Y. Pritch, A. Rav-Acha, and S. Peleg, "Nonchronological video synopsis and indexing," *IEEE TPAMI*, vol. 30, no. 11, pp. 1971–1984, 2008.

[16] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool, "Creating summaries from user videos," in *ECCV*. Springer, 2014, pp. 505–520.

[17] T. Yao, T. Mei, and Y. Rui, "Highlight detection with pairwise deep ranking for first-person video summarization," in *CVPR*, 2016.

[18] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *ICCV*. IEEE, 2003, pp. 1470–1477.

[19] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain, "Content-based multimedia information retrieval: State of the art and challenges," *ACM Trans. Multimedia Comput. Commun. and Appl.*, vol. 2, no. 1, pp. 1–19, 2006.

[20] C. G. Snoek, B. Huurnink, L. Hollink, M. De Rijke, G. Schreiber, and M. Worring, "Adding semantics to detectors for video retrieval," *IEEE TMM*, vol. 9, no. 5, pp. 975–986, 2007.

[21] L. Ballan, M. Bertini, G. Serra, and A. Del Bimbo, "A data-driven approach for tag refinement and localization in web videos," *Comput. Vis. Image Und.*, vol. 140, pp. 58–67, 2015.

[22] D. Lin, S. Fidler, C. Kong, and R. Urtasun, "Visual Semantic Search: Retrieving Videos via Complex Textual Queries," in *CVPR*, Jun. 2014, pp. 2657–2664.

[23] B. Craggs, M. Kilgallon Scott, and J. Alexander, "Thumbreels: query sensitive web video previews based on temporal, crowdsourced, semantic tagging," in *SIGCHI*. ACM, 2014, pp. 1217–1220.

[24] H.-W. Kang and X.-S. Hua, "To learn representativeness of video frames," in *ACM MM*. ACM, 2005, pp. 423–426.

[25] A. Rav-Acha, Y. Pritch, and S. Peleg, "Making a long video short: Dynamic video synopsis," in *CVPR*, vol. 1. IEEE, 2006, pp. 435–441.

[26] C. Liu, Q. Huang, and S. Jiang, "Query sensitive dynamic web video thumbnail generation," in *ICIP*. IEEE, 2011, pp. 2449–2452.

[27] W. Liu, T. Mei, Y. Zhang, C. Che, and J. Luo, "Multi-task deep visual-semantic embedding for video thumbnail selection," in *CVPR*, 2015, pp. 3707–3715.

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

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and

L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vision*, pp. 1–42, April 2015.
[30] A. Habibian, T. Mensink, and C. G. Snoek, "VideoStory Embeddings Recognize Events when Examples are Scarce," *arXiv preprint arXiv:1511.02492*, 2015.
[31] B. Logan *et al.*, "Mel frequency cepstral coefficients for music modeling." in *ISMIR*, 2000.
[32] M.-C. De Marneffe, B. MacCartney, C. D. Manning *et al.*, "Generating typed dependency parses from phrase structure parses," in *Proceedings of LREC*, vol. 6, 2006, pp. 449–454.
[33] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *ANIPS*, 2013, pp. 3111–3119.
[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*. IEEE, 2009, pp. 248–255.
[35] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[36] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.
[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[38] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.
[39] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV*. Springer, 2014, pp. 818–833.
[40] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *CVPR*. IEEE, 2015, pp. 5188–5196.
[41] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *CVPR*, 2015, pp. 447–456.
[42] T. Joachims, "Optimizing search engines using clickthrough data," in *ACM SIGKDD Int. Conf. Knowl. Discov. and Data Min.* ACM, 2002, pp. 133–142.
[43] J. Vendrig and M. Worring, "Systematic evaluation of logical story unit segmentation," *IEEE TMM*, vol. 4, no. 4, pp. 492–499, 2002.
[44] P. Sidiropoulos, V. Mezaris, I. Kompatsiaris, and J. Kittler, "Differential edit distance: A metric for scene segmentation evaluation," *IEEE TCSVT*, vol. 22, no. 6, pp. 904–914, 2012.
[45] E. Apostolidis and V. Mezaris, "Fast Shot Segmentation Combining Global and Local Visual Descriptors," in *IEEE Int. Conf. Acoustics, Speech and Signal Process.*, 2014, pp. 6583–6587.
[46] P. Ercolessi, H. Bredin, C. Sénac, and P. Joly, "Segmenting tv series into scenes using speaker diarization," in *Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2011), Delft-Pays bas*, 2011, pp. 13–15.
[47] "BBC Planet Earth series," http://www.bbc.co.uk/programmes/b006mywy, [Online; accessed 18-July-2016].

Lorenzo Baraldi received the B.Sc. and M.Sc. degrees in Computer Engineering from the University of Modena and Reggio Emilia, Italy. He is currently pursuing the Ph.D. degree at the Imagelab Laboratory at the Department of Engineering Enzo Ferrari of the University of Modena, Italy. His research interests include video understanding, Deep Learning and Multimedia.

Costantino Grana graduated at the University of Modena, Italy in 2000 and achieved the Ph.D. in Computer Science and Engineering in 2004. He is currently Associate Professor at the Department of Engineering Enzo Ferrari of the University of Modena, Italy. His research interests are mainly in Computer Vision and Multimedia and include analysis and search of digital images of historical manuscripts and other cultural heritage resources, multimedia image and video retrieval, medical imaging, color based applications, motion analysis for tracking and surveillance. He published 5 book chapters, 25 papers on international peer-reviewed journals and 75 papers on international conferences.

Rita Cucchiara (M.S. degree in Electronics Engineering, in 1989, and Ph.D. degree in Computer Engineering from the University of Bologna, in 1992) is currently a Full Professor of Computer Engineering within the University of Modena and Reggio Emilia. She is the Director of the Research Center Softech-ICT, and heads the Imagelab Laboratory. She is President of the Gruppo Italiano Ricercatori in Pattern Recognition (affilated with IAPR), member of the advisory board of the Computer Vision Foundation, member of the IEEE Computer Society, the ACM and Fellow of IAPR. She has been a Coordinator of several projects in computer vision and pattern recognition, and in particular on video surveillance, human behavior analysis and video understanding. In the field of multimedia, she works on annotation, retrieval, and human-centered searching in images and video big data for cultural heritage. She has authored over 350 papers in journals and international proceedings, and is a reviewer for several international journals.