Authors' accepted version.
The final published version is: F. Markatopoulou, D. Galanopoulos, C. Tzelepis, V. Mezaris, I. Patras, "Concept-Based and Event-Based Video Search in Large Video Collections", in book "Big Data Analytics for Large-Scale Multimedia Search", S. Vrochidis, B. Huet, E. Chang, I. Kompatsiaris (Eds.), ISBN: 9781119376972 (Hardcover), 9781119376996 (online), Wiley, March 2019.
DOI:10.1002/9781119376996.ch2, https://doi.org/10.1002/9781119376996.ch2

1
Concept-Based and Event-Based Video Search in Large Video Collections

Foteini Markatopoulou, Damianos Galanopoulos, Christos Tzelepis, Vasileios Mezaris, Ioannis Patras

Video content can be annotated with semantic information such as simple concept labels that may refer to objects (e.g., "car" and "chair"), activities (e.g., "running" and "dancing"), scenes (e.g., "hills" and "beach"), etc.; or more complex (or high-level) events that describe the main action that takes place in the complete video. An event may refer to complex activities, occurring at specific places and times, which involve people interacting with other people and/or object(s), such as "changing a vehicle tire", "making a cake", or "attempting a bike trick", etc. Concept-based and event-based video search refers to the retrieval of videos/video fragments (e.g., keyframes) that present specific simple concept labels or more complex events from large-scale video collections, respectively. To deal with concept-based video search, video concept detection methods have been developed that automatically annotate video-fragments with semantic labels (concepts). Then, given a specific concept a ranking component retrieves the top related video fragments for this concept. While significant progress has been made during the last years in video concept detection, it continues to be a difficult and challenging task. This is due to the diversity in form and appearance exhibited by the majority of semantic concepts and the difficulty to express them using a finite number of representations. A recent trend is to learn features directly from the raw keyframe pixels using deep convolutional neural networks (DCNNs). Other studies focus on combining many different video representations in order to capture different perspectives of the visual information. Finally, there are studies that focus on multi-task learning in order to exploit concept model sharing, and methods that look for existing semantic relations e.g., concept correlations. In contrast to concept detection, where we most often can use annotated training data for learning the detectors, in the problem of video event detection we can distinguish two different but equally important cases: when a number of positive examples, or no positive examples at all ("zero-example" case), are available for training. In the first case, a typical video event detection framework includes a feature extraction and a classification stage, where an event detector is learned by training one or more classifiers for each event class using available features (sometimes similarly to the learning of concept detectors), usually followed by a fusion approach in order to combine different modalities. In the latter case, where solely a textual description is available

for each event class, the research community has directed its efforts towards effectively combining textual and visual analysis techniques, such as using text analysis techniques, exploiting large sets of DCNN-based concept detectors and using various re-ranking methods, such as pseudo-relevance feedback, or self-paced re-ranking. In this chapter, we survey the literature and we present our research efforts towards improving concept- and event-based video search. For concept-based video search, we focus on i) feature extraction using hand-crafted and DCNN-based descriptors, ii) dimensionality reduction using accelerated generalised subclass discriminant analysis (AGSDA), iii) cascades of hand-crafted and DCNN-based descriptors, iv) multi-task learning (MTL) to exploit model sharing and v) stacking architectures to exploit concept relations. For video event detection, we focus on methods which exploit positive examples, when available, again using DCNN-based features and AGSDA, and we also develop a framework for zero-example event detection that associates the textual description of an event class with the available visual concepts in order to identify the most relevant concepts regarding the event class. Additionally, we present a pseudo-relevant feedback mechanism that relies on AGSDA.

1.1 Introduction

Video understanding is the overall problem that deals with automatically detecting what is depicted in a video sequence. This problem can be divided into two separate sub-problems that focus at different levels of video analysis. Concept-based video search that works at video fragment-level, i.e., assigning one or more semantic concepts to video fragments (e.g., video keyframes) based on a predefined concept list (e.g., "car", "running") [1], and event-based video search that works at the complete video level, i.e., detecting the main event that is presented on the overall video sequence. These two sub-problems share some video pre-processing techniques. For example, video representation and dimensionality reduction are typically the same. However, they are overall treated as two separate tasks because the different video analysis level that each of them focuses at requires different detection and recognition mechanisms.

Video concept detection is an important video understanding problem that facilitates many applications such as semantics-based video segmentation, video event detection and concept-based video search. The latter, which is the problem investigated in this section, refers to the retrieval of videos/video fragments (e.g., keyframes) that present specific simple concept labels. In a typical video concept detection process, the video is initially segmented into meaningful fragments called shots, and each shot may be represented by one or more characteristic keyframes that are subsequently annotated. This is a multi-label classification problem (one keyframe may be annotated with more than one semantic concepts), that can be treated as multiple independent binary classification problems, where for each concept a model can be learned to distinguish keyframes that the concept appears from those that the concept does not appear. Given feature-based keyframe representations that have been extracted from

different keyframes and also the ground-truth annotations for each keyframe (i.e. the concepts presented) any supervised machine learning algorithm that solves classification problems can be used in order to learn the relations between the low-level image representations and the high-level semantic concepts. It has been shown that combining many different keyframe representations (e.g. SIFT, RGB-SIFT, DCNN-based) for the same concept, instead of using a single feature (e.g. only SIFT), improves the concept detection accuracy. Multi-task learning, which refers to methods that learn many tasks together at the same time, is another category of methods that aim to improve concept detection accuracy. Finally, exploiting label relations, the relations between concepts within a video shot (e.g., the fact that *sun* and *sky* will appear in the same video shot for the most of cases) is a third category of concept detection methods towards accuracy improvement. We will focus on three general learning areas towards improved concept-based video search:

- Combination of video representations using cascades (Section 1.3.2).
- Multi-task learning for concept-based video search (Section 1.3.3).
- Exploiting label relations using a two-layer stacking architecture (Section 1.3.4).

Event detection, the second sub-problem we tackle, facilitates applications such as event-based video search, i.e. the retrieval of videos that present a specific event. As a video event we consider a complex activity involving people interacting with other people and/or objects, e.g.,"Renovating a home". A typical video event detection pipeline starts with a feature extraction stage, which usually generates a plethora of low-, intermediate-, and high-level features from the different available modalities (visual, audio, and/or textual). Then, a classification stage follows, where an event detector has been learned by training one or more classifiers for each event class using some or all of the extracted video features and positive video examples (under the assumption that such examples are available). Finally, a fusion approach may be followed in order to combine different modalities. A slightly different and more challenging problem related to video event detection is the zero-example event detection, which refers to the case that solely a textual description of the event is available without any positive training examples. Typically, a zero-example system starts by analysing the textual event description so as to transform it to a meaningful set of keywords. At the same time, a predefined set of concepts is used, on the one hand to find which of these concepts are related to the extracted keywords and consequently to the event description, and on the other hand, to train visual concept detectors (i.e., using concept detection approaches as those presented in Section 1.3) that will be used to annotate the videos with these concepts. Regarding event-based video search we will present:

- Methods for video event detection when positive training examples are available (Section 1.4.2).
- Methods for video event detection when solely a textual description of the event is given (Section 1.4.3).

1.2 Video pre-processing and machine learning essentials

1.2.1 Video representation

A variety of visual, textual and audio features can be extracted to represent each piece of visual information; a review of different types of features can be found in [1]. Despite the fact that audio features have also been shown to be useful in visual understanding problems [1], in this work we focus mostly on visual features. Visual features are typically extracted from representative keyframes or similar 2D image structures [2], or from sequences of frames (e.g. motion features). We can distinguish two main categories of (static, i.e. non-motion) visual features: hand-crafted features and features based on Deep Convolutional Networks (DCNN-based). With respect to hand-crafted features, binary (ORB [3]) and non-binary (SIFT [4], SURF [5]) local descriptors, as well as color extensions of them [6] have been examined for video concept detection. Local descriptors are aggregated into global image representations by employing feature encoding techniques such as Fisher Vector (FV) [7] and VLAD [8]. With respect to DCNN-based features, one or more hidden layers of a pre-trained DCNN are typically used as a global image representation [9]. Several DCNN software libraries are available in the literature, e.g., Caffe [10], MatConvNet, and different DCNN architectures have been proposed, e.g., CaffeNet [11], GoogLeNet [12], VGG ConvNet [9]. DCNN-based descriptors present high discriminative power and generally outperform the local descriptors [13], [14].

A DCNN can be trained on a large-scale dataset and then can be used in two different ways to annotate new test keyframes with semantic concepts. a) As standalone classifier: Each test keyframe is forward propagated by the network and the network's output is used as the final class distribution assigned to the keyframe [9, 11], a process a.k.a. direct classification. b) As feature generator: The training set is forward propagated by the network and the features extracted from one or more layers of the network are used as feature vectors to subsequently train one supervised classifier (concept detector) per concept. Then, each test keyframe is firstly described by the DCNN-based features and subsequently served as input to the trained classifiers.

DCNN training requires the learning of millions of parameters, which means that a small-sized training set, which is the case for video datasets, could easily over-fit the DCNN on the training data. It has been proven that the bottom layers of a DCNN learn rather generic features, useful for different domains, while the top layers are task-specific [15]. Transferring a pre-trained network in a new dataset by fine-tuning its parameters is a common strategy that can take advantage of the bottom generic layers and adjust the top layers to the target dataset and the new target concepts. Fine-tuning is a process where the weights of a pre-trained DCNN are used as the starting point for a new target training set and they are modified in order to adapt the pre-trained DCNN to the new target dataset [16].

1.2.2 Dimensionality reduction

Dimensionality reduction in multimedia understanding problems, such as those of concept- and event-based video search has been proven to be very useful, since in such domains the data representations are typically high-dimensional. To this end, non-linear discriminant analysis (NDA) approaches [17], which aim at identifying a discriminant subspace of the original high-dimensional input space, and GPU implementations of computationally intensive algorithms, are recently getting increasing attention in large-scale video analysis problems [18, 19, 20]. For instance, generalised subclass discriminant analysis (GSDA) combined with linear support vector machines (LSVMs) achieved excellent performance in multimedia event detection [19], while in [18, 21] GPU accelerated machine learning algorithms obtain substantial speedups.

In this work, for performing dimensionality reduction efficiently, we use a non-linear discriminant analysis technique called accelerated generalised subclass discriminant analysis (AGSDA) [17] along with its GPU implementation [22, 23]. This method identifies a discriminant subspace of the input space in three steps: a) Gram matrix computation, b) eigenvalue decomposition of the between subclass factor matrix, and c) computation of the solution of a linear matrix system with symmetric positive semidefinite (SPSD) matrix of coefficients. Based on the fact that the computationally intensive parts of AGSDA, i.e. Gram matrix computation and identification of the SPSD linear matrix system solution, are highly parallelisable, a GPU implementation of AGSDA is employed [23]. The experimental evaluation of this method (GPU-AGSDA) on large-scale datasets of TRECVID for concept and event detection show that, combined with LSVM, outperforms LSVM alone in training time, memory consumption, and detection accuracy.

1.3 Methodology for concept detection and concept-based video search

1.3.1 Related work

Video concept detection is a multi-label classification problem (one keyframe may be annotated with more than one semantic concepts), that is typically treated as multiple independent binary classification problems; one per concept. Given feature-based keyframe representations that have been extracted from different keyframes and also the ground-truth annotations for each keyframe (i.e. the concepts presented) any supervised machine learning algorithm that solves classification problems can be used in order to train a concept detector. It has been shown that combining many different keyframe representations (e.g. SIFT, RGB-SIFT, DCNN-based) for the same concept, instead of using a single feature (e.g. only SIFT), improves the concept detection accuracy. The typical way of combining multiple features is to

train several supervised classifiers for the same concept, each trained separately on a different feature. When all the classifiers give their decisions, a fusion step computes the final confidence score (e.g. by averaging); this process is known as late fusion. Hierarchical late fusion [24] is a more elaborate approach; classifiers that have been trained on more similar features (e.g. SIFT and RGB-SIFT) are firstly fused together and then, more dissimilar classifiers (e.g. DCNN-based) are sequentially fused with the previous groups. A second category of classifier combination approaches performs ensemble pruning to select a subset of the classifiers prior to their fusion. For example, [25] uses a genetic algorithm to automatically select an optimal subset of classifiers separately for each concept. Finally, there is a third group of popular ensemble-based algorithms, namely cascade architectures, that have been used in various visual classification tasks for training and combining detectors [26], [27]. In a cascade architecture, e.g., [28], the classifiers are arranged in stages, from the less computationally demanding to the most demanding ones (or may be arranged according to other criteria such as their accuracy). A keyframe is classified sequentially by each stage and the next stage is triggered only if the previous one returns a positive prediction (i.e. that the concept or object appears in the keyframe). The rationale behind this is to rapidly reject keyframes that clearly do not match the classification criteria and focus on those keyframes that are more difficult and more likely to depict the sought concept. Cascades of classifiers have been mainly used in object detection tasks, however they have also been briefly examined for video/image concept detection [26], [28].

Independently training concept detectors is a single-task learning (STL) process, where each task involves recognizing one concept. However, video concept detection can be treated as a MTL problem where the different tasks (one per concept) can be learned jointly by letting the sharing of knowledge across them. The main difference between MTL methods is the way they define task relatedness, i.e., the type of knowledge that should be shared. Some methods identify shared features between different task and use regularization to model task relatedness [29]. Others identify a shared subspace over the task parameters [30, 31]. The methods above make the strong assumption that all tasks are related; some newer methods consider the fact that some tasks may be unrelated [32, 33].

Two main types of methods that exploit label relations have been adopted in the literature: a) Stacking-based approaches that collect the scores produced by a baseline set of concept detectors (in a first layer) and introduce a second learning step in order to refine them (in a second layer), b) Inner-learning approaches that follow a single-step learning process, which jointly considers low-level visual features and concept correlation information [1]. Assuming that we firstly train a set of SVM-based or LR-based independent concept detectors either using a cascade architecture, or using other typical fusion schemes (e.g., late fusion in terms of arithmetic mean), and secondly we apply this set of concept detectors to new test keyframes, stacking approaches aim to refine the initial predictions on the test set by detecting dependencies among concepts in the last layer of the stack. For example, Discriminative Model Fusion (DMF) [34] obtains concept score predictions from the individual concept detectors in the first layer, in order to create a *model vector* for each shot. These

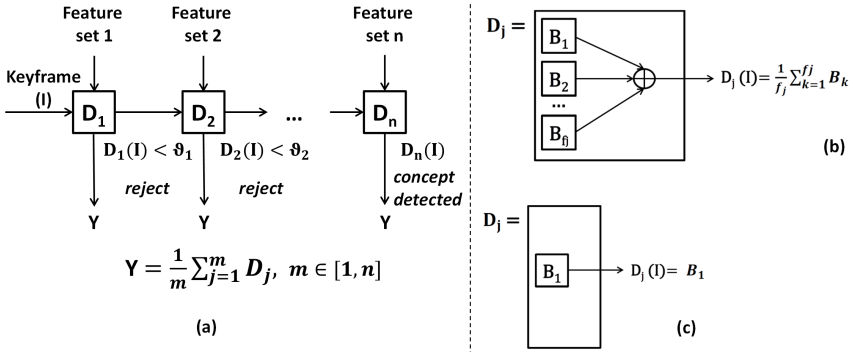

Figure 1.1 (a) Block diagram of the developed cascade architecture for one concept. (b) Stage combining many base classifiers trained on different features. (c) Stage with one base classifier trained on a single feature.

vectors form a meta-level training set, which is used to train a second layer of independent concept detectors. Correlation-Based Pruning of Stacked Binary Relevance Models (BSBRM) [35] extends the previous approach by pruning the predictions of non-correlated concept detectors before the training of each individual classifier of the second-layer models. Similarly to DMF, the Baseline CBCF (BCBCF) [36] forms model vectors, in this case using the ground truth annotation, in order to train second-layer BR models. Furthermore, the authors of [36] note that not all concepts can take advantage of CBCF, so their method refines only a subset of them. Another group of stacking approaches are the graph-based ones, which model label correlations explicitly [1]. Multi-Cue Fusion (MCF) method [37] uses the ground truth annotation to build decision trees that describe the relations among concepts, separately for each concept. Initial scores are refined by approximating these graphs. Inner-learning approaches, on the other hand, make use of contextual information from the beginning of the concept learning process. For example, the authors of [38] propose methods that simultaneously learn the relation between visual features and concepts and also the correlations among concepts. However, inner-learning approaches suffer of computational complexity. For example, [38] has complexity at least quadratic to the number of concepts, making it inapplicable to real problems where the number of concepts is large (e.g. hundreds or thousands).

1.3.2 Cascades for combining different video representations

A cascade architecture, for example the one we developed in [39], can be used to effectively combine many base classifiers that have been trained for the same concept (Figure 1.1). Each stage j of the cascade encapsulates a stage classifier D_j that either combines many base classifiers ($B_1, B_2, ..., B_{f_j}$, Fig. 1.1: (b)) that have been trained on different types of features or contains only one base classifier (B_1) that has been trained on a single type of features (Fig. 1.1: (c)). In the first case, the

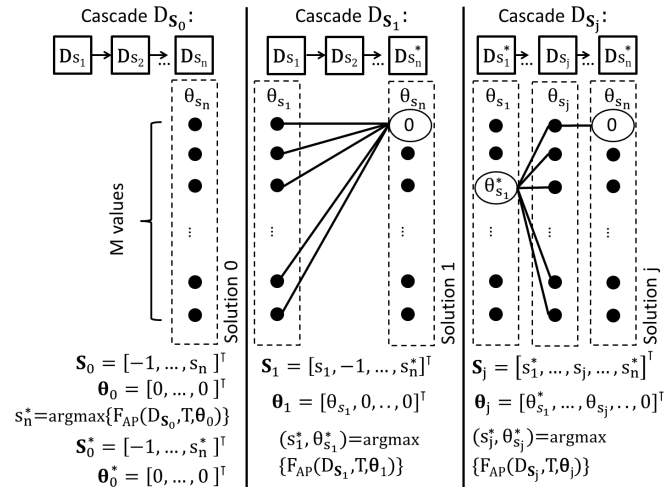

Figure 1.2 Threshold assignment and stage ordering of the proposed cascade architecture (Fig. 1.1).

output of f_j base classifiers is combined in order to return a single stage output score $D_j(I) = \frac{1}{f_j} \sum_{i=1}^{f_j} B_i(I)$, $f_j \geq 1$ in the [0,1] range. The second case is a special case where $f_j = 1$. Let I indicate an input keyframe; the classifier D_{j+1} of the cascade will be triggered for it only if the previous classifier does not reject the input keyframe I. Each stage j of the cascade is associated with a rejection threshold, while a stage classifier is said to reject an input keyframe if $D_j(I) < \theta_j$. A rejection indicates the classifier's belief that the concept does not appear in the keyframe. Let $D = \{D_1, D_2, ..., D_n\}$ be a set of n independently trained classifiers for a specific concept.

The rest of this Section presents an advanced algorithm that sets the ordering of cascade stages (i.e. the ordering of stage classifiers) and assigns thresholds to each stage in order to instantiate the proposed cascade of Fig. 1.1. The ordering of stages and the threshold assignment is performed towards the optimization of the complete cascade and not the optimization of each stage separately from the other stages.

1.3.2.1 Problem Definition and Search Space

Let $\mathbf{S} = [s_1, s_2, ..., s_n]^\top$ denote a vector of integer numbers in $[-1, 0) \cup [1, n]$. Each number represents the index of a classifier from D and appears at most once. The value -1 indicates that a classifier from D is omitted. Consequently, $\mathbf{S}$ expresses the ordering of the pre-trained classifiers $D_1, ..., D_n$. For example, given a pre-trained set of 4 classifiers $D = \{D_1, D_2, D_3, D_4\}$, the solution $\mathbf{S} = [2, 1, 3, -1]^\top$ denotes the cascade $D_{2,1,3,-1} : D_2 \rightarrow D_1 \rightarrow D_3$, where stage classifier D_4 is not used at all. In addition, let $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_n]^\top$ denote a vector of rejection thresholds for the solution $\mathbf{S}$ and let T= $\{\boldsymbol{x_i}, y_i\}_{i=1}^{M}$, where $y_i \in \{\pm 1\}$, be a set of annotated training

samples for the given concept ($\boldsymbol{x_i}$ being the feature vectors and y_i the ground-truth annotations). The problem we aim to solve is finding the pair of the index sequence $\mathbf{S}$ (that leads to the cascade $D_{\mathbf{S}} : D_{s_1} \rightarrow D_{s_2} \rightarrow ... \rightarrow D_{s_n}$) and the vector of thresholds $\boldsymbol{\theta} = [\theta_1^\star, \theta_2^\star, ..., \theta_n^\star]^\top$ that maximizes the expected ranking gain on the finite set T. The implied optimization problem is given by the following equation:

$$(\mathbf{S}^\star, \boldsymbol{\theta}^\star) = \underset{(\mathbf{S},\boldsymbol{\theta})}{\operatorname{argmax}}\{F(D_{\mathbf{S}}, T, \boldsymbol{\theta})\}, \tag{1.1}$$

where the ranking function $F(D_{\mathbf{S}}, T, \boldsymbol{\theta})$ can be defined as the expected ranking gain of $D_{\mathbf{S}}$ on T, that is

$$F_{AP}(D_{\mathbf{S}}, T, \boldsymbol{\theta}) = AP@k(rank(y), rank(D_{\mathbf{S}}(T, \boldsymbol{\theta})),$$

where, $rank(y)$ is the actual ranking of the samples in T (i.e., samples with $y_i = 1$ are ranked higher than samples with $y_i = -1$), and $rank(D_{\mathbf{S}}(T, \boldsymbol{\theta}))$ the predicted ranking of the samples of cascade $D_{\mathbf{S}}, \boldsymbol{\theta}$ on T. $AP@k$ is the average precision in the top k samples.

Let $l \leq n$ refer to the number of variables $s_j \in \mathbf{S}$ whose value is different from -1 (i.e., l is the number of cascade stages that solution $\mathbf{S}$ implies). The size of the search space related to the ordering of cascade stages is $\sum_{l=1}^{n} \binom{n}{l} l!$ (i.e. all index sequences for $l = 1$, all permutations of index sequences for $l = 2$, and similarly for all higher values of l, up to $l = n$). Furthermore, $\Theta \subset \mathbb{R}^n$ is the search space that consists of all the possible rejection thresholds for each stage of the cascade. To collect candidate threshold values, we apply each stage classifier on the training set T. Each of the M returned probability output scores constitutes a candidate threshold. The size of the search space equals to M^n. Considering that this is a large search space, exhaustive search cannot be practically applied. To solve the problem we propose the greedy search algorithm described below.

1.3.2.2 Problem Solution

Our algorithm finds the final solution by sequentially replacing at each iteration a simple solution (consisting of a cascade with a certain number of stages) with a more complex one (consisting of a cascade with one additional stage). Algorithm 1 presents the proposed greedy search algorithm that instantiates the proposed cascade (Fig. 1.1). Fig. 1.2 presents graphically the algorithm's steps. Let $\mathbf{S} = [s_1, s_2, ..., s_n]^\top$, and $\boldsymbol{\theta} = [\theta_{s_1}, \theta_{s_2}, ..., \theta_{s_n}]^\top$, represent a solution. Each variable $s_1, s_2, ..., s_n$ can take n possible values, from 1 to n or the value -1 which indicates that a stage is omitted. Each variable $\theta_{s_1}, \theta_{s_2}, ..., \theta_{s_n}$ can take M possible values. Initially we set, $s_j = -1$ for $j = 1, ..., n$ and $\boldsymbol{\theta} = [0, 0, ..., 0]^\top$ where $|\boldsymbol{\theta}| = n$. In the first step the algorithm optimizes $\mathbf{S}$ with respect to s_n (Alg. 1: States 1-3) in order to build the solution:

$$\mathbf{S}_0 = [-1, -1, ..., s_n]^\top, \boldsymbol{\theta}_0 = [0, 0, ..., 0]^\top,$$

where according to (1.1),

$$s_n^\star = \underset{s_n}{\operatorname{argmax}}\{F_{AP}(D_{\mathbf{S}_0}, T, \boldsymbol{\theta}_0)\}. \tag{1.2}$$

and $\boldsymbol{\theta}_0^\star = [0, 0, ..., \theta_{s_n}^\star]$, $\theta_{s_n}^\star = 0$. This can be interpreted as the optimal solution of $l = 1$, that maximizes (1.1). Then the algorithm, in iteration j (Alg. 1: States 4-7), assumes that it has solution with $l = j$, that is:

$$\mathbf{S}_{j-1}^\star = [s_1^\star, s_2^\star, ..., s_{j-1}^\star, -1, -1, ..., s_n^\star]^\top,$$

$$\boldsymbol{\theta}_{j-1}^\star = [\theta_{s_1}^\star, \theta_{s_2}^\star, ..., \theta_{s_{j-1}}^\star, 0, 0, ..., \theta_{s_n}^\star]^\top,$$

and finds the pair of $\mathbf{S}_j$ and $\boldsymbol{\theta}_j$ in one step as follows. It optimizes the pair of $\mathbf{S}_{j-1}^\star$ and $\boldsymbol{\theta}_{j-1}^\star$ with respect to s_j and θ_j, respectively, in order to find the solution:

$$\mathbf{S}_j = [s_1^\star, s_2^\star, ..., s_{j-1}^\star, s_j, -1, -1, ..., s_n^\star]^\top,$$

$$\boldsymbol{\theta}_j = [\theta_{s_1}^\star, \theta_{s_2}^\star, ..., \theta_{s_{j-1}}^\star, \theta_{s_j}, 0, 0..., \theta_{s_n}^\star]^\top.$$

According to (1.1):

$$(s_j^\star, \theta_{s_j}^\star) = \underset{(s_j, \theta_{s_j})}{\operatorname{argmax}} \{F_{AP}(D_{\mathbf{S}_j}, T, \boldsymbol{\theta}_j)\}. \tag{1.3}$$

The algorithm finds the pair of (s_j, θ_{s_j}) that optimizes (1.1). The complexity of this calculation equals to $(n - j) \times M$. This corresponds to $n - j$ possible values that variable s_j can take in iteration j and M possible threshold rejection values that variable θ_{s_j} can take for every different instantiation of s_j. Finally, the optimal sequence $\mathbf{S}^\star$ equals to

$$\mathbf{S}^\star = \underset{\mathbf{S} \in \{\mathbf{S}_0^\star, \mathbf{S}_1^\star, ..., \mathbf{S}_{n-1}^\star\}}{\operatorname{argmax}} \{F_{AP}(D_{\mathbf{S}}, T, \boldsymbol{\theta})\}, \tag{1.4}$$

which is the sequence that optimizes (1.1) within all the iterations of the algorithm (Alg. 1: States 6-7). The optimal threshold vector $\boldsymbol{\theta}^\star$ is the vector connected to the optimal sequence $\mathbf{S}^\star$. Our algorithm focuses on the optimization of the complete cascade and not the optimization of each stage separately from the other stages. This is expected to give a better complete solution. Furthermore, the algorithm can be slightly modified to make the search more efficient. For example, at each iteration we can keep the p best solutions. However, this would increase the computational cost.

1.3.3 Multi-task learning for concept detection and concept-based video search

Having seen in the previous section the way that different video representations can be combined in a cascade architecture to improve the accuracy of the individual concept detectors, in this section we will focus on the way that more accurate individual concept detectors can be built by sharing knowledge across the concepts. MTL focuses exactly on the sharing of knowledge across different tasks. Assuming that some groups of concepts are expected to be related through some underlying structure, we can train a MTL classifier instead of single-task learning (STL) ones (e.g., SVMs).

Algorithm 1 Cascade stage ordering and threshold search

Input: Training set T= $\{\boldsymbol{x_i}, y_i\}_{i=1}^{M}$, $y_i \in \{\pm 1\}$; n trained classifiers $D = \{D_1, D_2, ..., D_n\}$
Output: i) An index sequence $\mathbf{S}^\star$, of the ordering of cascade stages: $D^\star_{\mathbf{S}} : D^\star_{s_1} \rightarrow D^\star_{s_2} \rightarrow ... \rightarrow D^\star_{s_n}$. ii) A vector of thresholds $\boldsymbol{\theta}^\star = [\theta^\star_{s_1}, \theta^\star_{s_2}, ..., \theta^\star_{s_n}]^\top$
Initialize: $\mathbf{S} = [s_1, s_2, ..., s_n]^\top$, $s_j = -1$, $j = 1, ..., n$, $\boldsymbol{\theta} = [0, 0, ..., 0]^\top$, $|\boldsymbol{\theta}| = n$,
1. $s^\star_n = \text{argmax}_{s_n}\{F_{AP}(D_{\mathbf{S}_0}, T, \boldsymbol{\theta}_0)\}$ (1.1),
$\mathbf{S}_0 = [-1, -1, ..., s_n]^\top$, $\boldsymbol{\theta}_0 = [0, 0, .., 0]^\top$
2. $maxCost = F_{AP}(D_{\mathbf{S}^\star_0}, T, \boldsymbol{\theta}^\star_0)$,
$\mathbf{S}^\star_0 = [-1, -1, ..., s^\star_n]^\top$, $\boldsymbol{\theta}^\star_0 = [0, 0, ..., \theta^\star_{s_n}]^\top$, $\theta^\star_{s_n} = 0$
3. $\mathbf{S}^\star = \mathbf{S}^\star_0$, $\boldsymbol{\theta}^\star = \boldsymbol{\theta}^\star_0$
for $j = 1$ to $n - 1$ **do**
  4. $(s^\star_j, \theta^\star_{s_j}) = \text{argmax}_{(s_j, \theta_{s_j})}\{F_{AP}(D_{\mathbf{S}_j}, T, \boldsymbol{\theta}_j)\}$ (1.1),
  $\mathbf{S}_j = [..., s_j, -1, ..., s^\star_n]^\top$, $\boldsymbol{\theta}_j = [..., \theta_{s_j}, 0, ..., \theta^\star_{s_n}]^\top$
  5. $cost = F_{AP}(D_{\mathbf{S}^\star_j}, T, \boldsymbol{\theta}^\star_j)$,
  $\mathbf{S}^\star_j = [..., s^\star_j, -1, ..., s^\star_n]^\top$, $\boldsymbol{\theta}^\star_j = [..., \theta^\star_{s_j}, 0, ..., \theta^\star_{s_n}]^\top$
  if cost>maxCost **then**
    6. maxCost = max(cost, maxCost)
    7. $\mathbf{S}^\star = \mathbf{S}^\star_j$, $\boldsymbol{\theta}^\star = \boldsymbol{\theta}^\star_j$
  end if
end for

MTL is expected to improve the concept-based video search accuracy. Training MTL algorithms is similar to the training of the STL alternatives. Specifically, each type of features serves as input to the MTL algorithm in order to train a MTL models. The scores returned from the MTL models for the same concept are fused using any existing fusion-scheme e.g, late fusion (averaging), cascades etc.

We choose the following MTL algorithms in order to examine different assumptions of task relatedness. The first method makes the strong assumption that all tasks are related. The next two methods consider the fact that some tasks may be unrelated [32, 33].:

- The L_1-norm (or Lasso) regularized methods are widely used to introduce sparsity into the model and achieve the goal of reducing model complexity and feature learning [40]. An implementation that extends the L_1-norm regularized STL to MTL formulations is proposed in [41]. A common simplification of Lasso in MTL is that the parameter controlling the sparsity is shared among all tasks, assuming that different tasks share the same sparsity parameter. We will refer to this MTL method as Lasso-MTL in the sequel.
- The clustered MTL algorithm (CMTL) [32] uses a clustering approach to assign to the same cluster parameters of tasks that lie nearby in terms of their L2 distance
- The adaptive MTL (AMTL) [33] decomposes the task parameters into a low-rank structure that captures task relations, and a group-sparse structure that detects outlier tasks.

1.3.4
Exploiting label relations

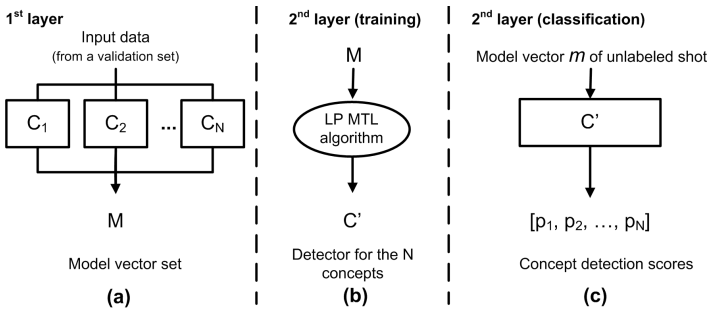

Figure 1.3 A two-layer stacking architecture instantiated with the LP [42] multi-label classification algorithm.

For further improving the accuracy of a video concept detection system we can use a stacking architecture that exploits the concept relations in the last layer of the stack. Specifically in this Section we present a two-layer stacked model, initially proposed in [43], which uses appropriate multi-label classification methods able to capture concept relations. To build concept detectors the two-layer concept detection system presented in Fig. 1.3, is employed. The first layer builds multiple independent concept detectors, either using STL or MTL approaches 1.3.3. In the second layer of the stacking architecture, the fused scores from the first layer are aggregated in model vectors and refined using a multi-label learning algorithm that incorporates concept correlations. We choose the Label powerset (LP) [42] transformation that models correlations among sets of more than two concepts. The typical stacking methods learn concept relations only by using the meta-level feature space. I.e, the learning of each concept in the second layer is still independent of the learning of the rest of the concepts. In contrast, our stacking architecture learns concept relations in the last layer of the stack both from the outputs of first-layer concept detectors and by modelling relations directly from the ground-truth annotation of the meta-level training set. This is achieved by instantiating our architecture with the LP [42] algorithm that searches for subsets of labels that appear together in the training set and consider each set as a separate class in order to solve a multi-class problem. Of course, different multi-label classification algorithms that model different types of concept relations could also be used such as Calibrated Label Ranking (CLR) algorithm that model relations between pairs of concepts and ML-kNN algorithm [44] that models multiple relations in the neighbourhood of each testing instance.

1.3.5 Experimental study

1.3.5.1 Dataset and Experimental Setup

TRECVID Semantic Indexing (SIN) task [45] is a popular benchmarking activity that provides a large-scale dataset for video concept detection. The task is as follows. Given a set of shot boundaries for the SIN test dataset and a pre-defined list of concepts, participants are asked to return for each concept, the top 2000 video shots from the test set, ranked according to the highest possibility of depicting the concept. The presence of each concept is assumed to be binary, i.e., it is either present or absent in the given standard video shot. If the concept was true for some frame within the shot, then it was true for the shot. This is a simplification adopted for the benefits it affords in pooling of results and approximating the basis for calculating recall. A list of 346 concepts is provided that has been collaboratively annotated by the participants and by Quaero annotators. In this study our experiments were performed on the TRECVID 2013 SIN dataset [45] that provides the following materials:

- a development set that contains roughly 800 hours of Internet archive videos comprising more than 500000 shots;
- a test set that contains roughly 200 hours of videos, comprising 112677 shots;
- shot boundaries (for both sets);
- a set of 346 concepts for training;
- elements of ground-truth: some shots were collaboratively annotated.

TRECVID 2013 SIN task was evaluated for 38 semantic concepts by calculating the Mean Extended Inferred Average Precision (MXinfAP) at depth 2000 [46]. MXinfAP is an approximation of the Mean Average Precision (MAP) that has been adopted by TRECVID [45] because it is suitable for the partial ground-truth that accompanies the TRECVID dataset [45].

In section 1.3.5.2 we evaluate the developed methods for each of the three general learning areas that aim to improve concept-based video search. I.e., cascades of classifiers (Section 1.3.2), multi-task learning (Section 1.3.3) and exploiting label relations (Section 1.3.4). In our experiments we extracted the following features: Three binary descriptors (ORB, RGB-ORB and OpponentORB) and six non-binary descriptors (SIFT, RGB-SIFT and OpponentSIFT; SURF, RGB SURF and OpponentSURF). All of them were compacted using PCA and were subsequently aggregated using the VLAD encoding. In addition, we used features based on the pre-trained CaffeNet-1k, GoogLeNet-1k and 16-layer deep ConvNet [9] networks trained on 1000 ImageNet [47] categories. We applied each of these networks on the TRECVID keyframes and we used as a feature i) the output of the second last fc layer of ConvNet (fc7), which resulted to a 4096-element vector, ii) the output of the last fc layer of CaffeNet-1k (fc8), which resulted to a 1000-element, iii) the output of the last fc layer of GoogLeNet-1k (loss3/classifier). We refer to these features as CONV, CAFFE and GNET in the sequel, respectively. To train our base classifiers (i.e., LSVMs), for each concept, the TRECVID training set was used.

Firstly,we compared the developed cascade (Section 1.3.2) with five different en-

semble combination approaches: i) Late-fusion with arithmetic mean [24]. ii) The ensemble pruning method proposed by [25]. iii) The simple cascade proposed by [28] with fixed ordering of the stages in terms of classifier accuracy. We refer to this method as cascade-thresholding. iv) A cascade with fixed ordering of the stages in terms of classifier accuracy, and the offline dynamic programming algorithm for threshold assignment proposed by [48]. In contrast to [48] that aims to improve the overall classification speed, we optimize the overall detection performance of the cascade in terms of AP. We refer to this method as cascade-dynamic in the sequel. v) A boosting-based approach (i.e., the multi-modal sequential SVM [49]). We refer to this method as AdaBoost. For all the methods, except for the Late-fusion that does not require this, the training set was also used as the validation set. With respect to the developed method we calculated the AP for each candidate cascade at three different levels (i.e., for k=50,100 and equal to the number of the training samples per concept) and we averaged the results. Secondly, we assessed the usefulness MTL, presented in Section 1.3.3, instantiated with different MTL algorithms and compared it with the typical single-task learning approach. The MALSAR MTL library [41] was used as the source of the MTL algorithms. Thirdly, we assessed the usefulness of the stacking architecture, presented in Section 1.3.4. For this we further used the TRECVID 2012 test set (approx. 200 hours; 145634 shots), which is a subset of the 2013 development set, as a validation set to train our multi-label classification algorithm for the second layer of the stack. Finally, we evaluate jointly all the proposed methods in a single concept detection system.

1.3.5.2 Experimental results

Table 1.1 Performance (MXinfAP, %) for each of the stage classifiers that we used in the experiments. For stage classifiers that are made of more than one base classifiers, we report in parenthesis the MXinfAP for each of these base classifiers.

Stage classifier	MXinfAP	Base classifiers
ORBx3	17.91 (12.18,13.81,14.12)	ORB, RGB-ORB, OpponentORB
SURFx3	18.68 (14.71,15.49,15.89)	SURF, OpponentSURF, RGB-SURF
SIFTx3	20.23 (16.55,16.73,16.75)	SIFT, OpponentSIFT, RGB-SIFT
CAFFE	19.80	Last fully-connected layer of CaffeNet
GNET	24.36	Last fully-connected layer of GoogLeNet
CONV	25.26	Second last fully-connected layer of ConvNet

Table 1.2 Performance (MXinfAP, %) for different classifier combination approaches.

	M1	M2	M3	M4	M5	M6
Stage classifiers	Late-fusion [24]	Ensemble pruning [25]	Cascade-thresholding [28]	Cascade-dynamic [48]	AdaBoost [49]	Cascade-proposed (Section 1.3.2)
ORBx3; SURFx3;CAFFE; SIFTx3;GNET; CONV	29.84	29.74	29.79	29.84	29.70	**29.96**

Tables 1.1, 1.2, 1.3, 1.4 and 1.2 present the results of our experiments in terms of MXinfAP [46]. Table 1.1 presents the MXinfAP for the different types of features that were used by the algorithms of this Section. Each line of this table was used as a cascade stage for the cascade-based methods (Table 1.2: M3, M4, M6). Specifically, stages that correspond to SIFT, SURF and ORB consist of three base classifiers (i.e. for the grayscale descriptor and its two color variants), while the stages of DCNN features (CAFFE, CONV, GNET) consist of one base classifier each. For the late fusion methods (Table 1.2: M1, M2) and the boosting-based method (Table 1.2: M5), the corresponding base classifiers per line of Table 1.1 were firstly combined by averaging the classifier output scores and then the combined outputs of all lines were further fused together. We adopted this grouping of similar base classifiers as this was shown to improve the performance for all the methods in our experiments, increasing the MXinfAP by $\sim 2\%$. For M2 we replaced the genetic algorithm with exhaustive search (i.e. to evaluate all $2^6 - 1$ possible classifier subsets) because this was more efficient for the examined number of classifiers.

Table 1.2 presents the performance of the developed cascade-based method (Section 1.3.2) and compares it with other classifier combination methods. The second column shows the stage classifiers that were considered, i.e., the evaluated system utilised six stage classifiers and all twelve types of features. The best results were reached by the proposed cascade, which outperforms all the other methods reaching a MXinfAP of 29.96 %. Compared to the ensemble pruning method (M2) the results show that exploring the best ordering of visual descriptors on a cascade architecture (M6), instead of just combining subsets of them (M2), can improve the accuracy of video concept detection. In comparison to the other cascade-based methods (M3, M4) that utilize fixed stage orderings and different algorithms to assign the stage thresholds, the proposed cascade (M6) also shows small improvements in MXinfAP. These can be attributed to the fact that our method simultaneously searches both for optimal stage ordering and threshold assignment. These MXinfAP improvements, of the proposed cascade, although small, are accompanied by considerable improvements in computational complexity, as discussed in Section 1.3.5.3.

Table 1.3 MXInfAP (%) for different STL and MTL methods, trained on the features of Table 1.1. Scores for the same concept were fused in terms of arithmetic mean using late-fusion.

	Single-task learning			Multi-task learning		
	LR	LSVM	KSVM	Lasso-MTL [40]	AMTL [33]	CMTL [32]
MXInfAP	27.56	29.84	29.29	**31.15**	30.3	29.51

In table 1.3 we evaluate the usefulness of using MTL, as presented in Section 1.3.3, for concept-based video search. We performed comparisons across the following methods: i) Single-task learning (STL) using a) LR, b) LSVM and c) kernel SVM with radial kernel (KSVM). ii) MTL using a) Lasso-MTL [40], b) AMTL [33], and c) CMTL [32]. Single-task learning refers to the typical training of cocnept detectors, i.e., training independent classifiers per concept. All the features of Table 1.1 were

used to train either STL or MTL detectors and late-fusion in terms of arithmetic mean was used to combine the scores from the different detectors for the same concept. According to table 1.3 the best performance is achieved when the Lasso-MTL[40] algorithm is used. Lasso-MTL can define task relatedness on the parameters of the independently trained concept detectors which shows to performs better both from all the STL approaches and also the compared MTL ones.

Table 1.4 shows the results of the two-layer stacking architecture presented in Section 1.3.4 and compares it with the typical single-layer concept detection pipeline. To train our first layer classifiers we used all the features presented in Table 1.1 and combined them using the proposed cascade. We also experimented with dimensionality reduction of these features prior to serve them as input to LSVMs. Specifically, the AGSDA method, presented in Section 1.2.2, was used to derive a lower dimensional embedding of the original feature vectors. Then, the features in the resulting subspace served as input to LSVMs. Consequently, The first layer of the employed stacking consists from the independent detectors, either trained on the raw features (Table 1.4 (a)) or trained on the AGSDA-reduced features (Table 1.4 (b)), that are subsequently combined using the proposed cascade evaluated in Table 1.2. The same detectors were applied on a meta-learning validation set in order to construct model vectors that were introduced in the second layer. The second layer was instantiated with the LP [42] algorithm, we refer to our method as P-LP.

P-LP outperforms the independent first layer detectors, reaching a MXinfAP of 25.6%. LP considers each subset of labels (label sets) presented in the training set as a class of a multi-class problem, which seems to be helpful for the stacking architecture. In [43] we evaluated many more different multi-label learning algorithms for our developed two-layer stacking architecture and we compared these instantiations against BCBCF [36], DMF [34], BSBRM [35] and MCF [37] presented in Section 1.3.4.

Table 1.4 Performance, (MXinfAP (%)), for the typical single-layer concept detection pipeline and the developed two-layer stacking architecture instantiated with the LP algorithm. For the developed method we also report CPU times that refers to mean training (in minutes) for all concepts, and application of the trained second-layer detectors on one shot of the test set (in milliseconds). Column (a) shows the results for detectors trained on raw features. Column (b) shows the results for detectors trained on AGSDA-reduced features. In parenthesis we show the relative improvement w.r.t. the typical approach.

	Dimensionality reduction		
	(a) N/A	(b) AGSDA	(c) Mean Exec. Time Training/Testing
Cascade	29.96	30.03	N/A
Proposed-LP	**30.9 (+3.1%)**	**30.35 (+1.1%)**	549.40/24.93

Table 1.5 evaluates the usefulness of sequentially adding each of the three methods presented above towards improving concept-based video search (i.e., cascades of classifiers, multi-task learning and exploitation of label relations), in accordance with the AGSDA dimensionality reduction approach. The typical concept-detection system that uses STL and late fusion to combine concept detectors trained on dif-

Table 1.5 MXinfAP for different evaluations of our concept detection approach.

Dataset	Late fusion-STL	Cascade	Cascade+AGSDA	Cascade+ AGSDA+MTL	Cascade+AGSDA+ MTL+LP
SIN 2015	23.7	23.9	24.98	25.8*	**26.03**
SIN 2013	29.84	29.96	30.03	32.47*	**32.57**

ferent features for the same concept is treated as our baseline. Then starting from a cascade architecture that more cleverly combines the different detectors, we continue by sequentially adding one more method to further improve concept detection accuracy. We present results both on the TRECVID SIN 2013 dataset but also on the TRECVID SIN 2015 dataset that consists of another test set and is evaluated on a different subset of the 346 available concepts. We observe that the proposed-cascade performs slightly better in terms of MXinfAP compared to the late fusion method, achieving 0.8% and 0.4% relative improvement for the SIN 2015 and SIN 2013 dataset, respectively. At the same time it is computationally less expensive during classification as we will see in the next section. Dimensionality reduction with AGSDA leads to more discriminative features which is indicated by an increase of the MXInfAP from 23.9% to 24.98% and 29.96% to 30.03% for the SIN 2015 and SIN 2013 dataset, respectively. Using also the MTL-Lasso method in the concept-detection pipeline, instead of STL techniques, significantly outperforms all the previous methods. Finally, exploiting label relations results to further improvement. We conclude that the methods presented in this section, i.e, cascades of detectors, MTL, two-layer stacking architecture that exploits label relations and dimensionality reduction using the AGSDA algorithm presented in section 1.2.2, are complementary and combining all of them in a concept detection system can boost concept-based video search reaching the best overall MXInfAP of 26.03% and 32.57% for the SIN 2015 and SIN 2013 dataset, respectively. These numbers are among the state of the art results for these specific datasets and, although seemingly low (since they are much lower than 100%), a qualitative analysis shows how good results they represent. For example, considering the concept "chair" that reaches an infAP of 32.84% on the SIN 2013 dataset, we observe in the resulting ranked list of concept-based retrieval results that among the top-20 retrieved keyframes all of them are positive; among the top-50 retrieved keyframes 46 are positive; and among the top-100 ones 86 of them are positive. To investigate the statistical significance of the differences between the results of the various methods/combinations reported in Table 1.5 we used a paired t-test as suggested by [50]; in Table 1.5, the absence of * suggests statistical significance. We found that differences between the complete method (Cascade+AGSDA+MTL+LP) and all the other methods are significant (at 5% significance level), except for the run that combines Cascade+AGSDA+MTL. Investigating label correlations in a system that already combines cascades, AGSDA and MTL does not lead to significant improvement. However, LP is still a useful method that consistently improves concept-based video search in both of the considered datasets.

Table 1.6 Training complexity: (a) Required number of classifier combinations during the training of different classifier combination approaches. (b) Required number of classifiers to be retrained.

		Required classifier evaluations	Number of classifiers to be retrained
M1	Late-fusion [24]	-	-
M2	Ensemble pruning [25]	$(2^n - 1)M$	-
M3	Cascade-thresholding [28]	$\sum_{j=0}^{n} M_j, M_j \subseteq Mj - 1$	-
M4	Cascade-dynamic [48]	$(n-2)Q^2$	-
M5	AdaBoost [49]	$M(n(n+1)/2)$	$n(n+1)/2$
M6	Cascade-proposed (Section 1.3.2)	$Q(n(n+1)/2)$	-

Table 1.7 Relative amount of classifier evaluations (%) for different classifier combination approaches during the classification phase.

	M1	M2	M3	M4	M5	M6
Stage classifiers	Late-fusion [24]	Ensemble pruning [25]	Cascade-thresholding [28]	Cascade-dynamic [48]	AdaBoost [49]	Cascade-proposed (Section 1.3.2)
ORBx3; SURFx3;CAFFE SIFTx3;CONV; GNET	100	66.67	74.94	92.38	100	**62.24**

1.3.5.3 **Computational Complexity**

We continue the analysis of our results with respect to the computational complexity of the different methods compared in Table 1.2 during the training and classification phase. Table 1.6 summarizes the computational complexity during the training phase. Let us assume that n stage classifiers need to be learned, M training examples are available for training the different methods and Q is the quantization value, where $Q \leq M$. The late-fusion approach [24], which builds n models (one for each set of features), is the simplest one. Cascade-thresholding [28] follows, which evaluates n cascade stages in order to calculate the appropriate thresholds per stage. Cascade-dynamic [48] works in a similar fashion as the Cascade-thresholding, requiring a little higher number of evaluations. Cascade-proposed is the next least complex algorithm, requiring $Q(n(n+1)/2)$ classifier evaluations. Ensemble pruning [25] follows, requiring the evaluation of $2^n - 1$ classifier combinations. Finally, only AdaBoost requires the retraining of different classifiers, which depends on the complexity of the base classifier, in our case the SVM, making this method the computationally most expensive.

Table 1.7 presents the computational complexity of the proposed cascade-based method for the classification phase, and compares it with other classifier combination methods. We observe that the proposed algorithm reaches good accuracy while at the same time is less computationally expensive than the other methods. Specifically, the best overall accuracy achieved 37.8% and 32.6% relative decrease in the

amount of classifier evaluations compared to the late fusion alternative (Table 1.7: M1) and the cascade-dynamic alternative (Table 1.7: M4), respectively, which are the two most accurate methods after the proposed-cascade. Finally, we should note that the training of the proposed cascade is computationally more expensive than the training of the late fusion and the cascade-dynamic methods. However, considering that training is performed offline only once, but classification will be repeated many times for any new input video, the latter is more important and this makes the reduction in the amount of classifier evaluations that is observed in Table 1.7 for the proposed cascade very important.

With respect now to the two-layer stacking architecture, according to the last column of table 1.4, one could argue that the proposed architecture requires considerably a lot of time. However, we should note here that extracting one model vector from one video shot, using the first-layer detectors for 346 concepts requires approximately 3.2 minutes in our experiments, which is about three orders of magnitude slower than the proposed second-layer methods. As a result of the inevitable computational complexity of the first layer of the stack, the execution time that P-LP requires can be considered negligible. This is in sharp contrast to building a multi-label classifier directly from the low-level visual features of video shots, where the high requirements for memory space and computation time that the latter methods exhibit make their application to our dataset practically infeasible. Specifically, the computational complexity LP when used in a single-layer architecture depends on the complexity of the base classifier, in our case the LSVMs, and on the parameters of the learning problem (e.g., number of training examples, feature video dimensionality). LP would build a multi-class model, with the number of classes being equal to the number of distinct label sets in the training set; this is in order of N^2 in our dataset. Taking into consideration the dimensionality of the utilised feature vectors, using any such multi-label learning method in a single-layer architecture would require several orders of magnitude more computations compared to the BR alternative that we employ as the first layer in our presented stacking architecture. We conclude that the major obstacle of using multi-label classification algorithms in a one-layer architecture is the computation time requirements, and this finding further stresses the merit of using a multi-label stacking architecture. Finally, the training of the MTL algorithms, compared in Table 1.3, is computationally more expensive than the training of linear STL alternatives (e.g., LR, LSVM), but less expensive than the kernel SVM. Considering that training is performed offline only once, but classification will be repeated many times for any new input video, the latter is more important and MTL methods present the same classification time with the linear STL alternatives.

1.4 Methods for event detection and event-based video search

1.4.1 Related work

There are several challenges associated with building an effective detector of video events. One of them is finding a video representation that reduces the gap between the traditional low-level audio-visual features that can be extracted from the video and the semantic-level actors and elementary actions that are by the definition the constituent parts of an event. In this direction, several works have shown the importance of using simpler visual concepts as a stepping stone for detecting complex events (e.g. [51, 19]). Another major challenge is to learn an association between the chosen video representation and the event or events of interest; for this, supervised machine learning methods are typically employed, together with suitably annotated training video corpora. While developing efficient and effective machine learning algorithms is a challenge in its own right, finding a sufficient number of videos that depict the event so as to use them as positive training samples for training any machine learning method is also not an easy feat. In fact, video event detection is even more challenging when the available positive training samples are limited, or even non-existent; that is, when one needs to train an event detector using only textual information that a human can provide about the event of interest.

Zero-example event detection is an active topic with many literature works proposing ways to build event detectors without any training samples using solely the event's textual description. Research towards this problem was mainly triggered a few years ago when the TRECVID benchmark activity introduced the 0Ex task as a subtask of the Media Event Detection (MED) task [52]. A similar to zero-example event detection problem, known as zero-shot learning (ZSL), also appears in the image recognition task. A new unseen category, for which training data is not available, is asked to be detected in images [53, 54, 55]. It should be noted that although the two problems have many common properties, zero-example event detection is a more challenging problem as it focuses on more complex queries, where multiple actions, objects and persons interact with each other compared to the simple object or animal classes that appear in ZSL [56]. The problem of zero-example event detection is typically addressed by transforming both the event textual description and the available videos into concept-based representations. Specifically, a large pool of concept detectors is used to annotate the videos with semantic concepts, the resulted vectors, a.k.a. model vectors, contain the scores indicating the degree that each of the concepts is related to the video. The query description is analysed and the most related concepts from the pool are selected. Finally, the distance between the model vectors and the event concept vectors is calculated and the most related videos are retrieved [57, 58, 59, 60].

In this work we present methods that solve the problem of event detection and event-based video search in two different cases. Firstly, when ground-truth annotated video examples are provided. In this case, we present how event detectors can be

learned using the video positive examples which are available separately for each event (Section 1.4.2). Secondly, when solely the textual description of event is given without any positive video examples. In this case, we present how event detectors can be learned using solely textual information for the examined events (Section 1.4.3).

1.4.2 Learning from positive examples

The target of an event detection system is to learn a decision function $f\colon \mathcal{F} \rightarrow \{\pm 1\}$, where $\mathcal{F}$ denotes the space where the video representations lie in. f assigns a test video to the event class (labeled with the integer 1) or to the "rest of the world" class (labeled with the integer -1). For each event class, this is typically achieved using a training set $\mathcal{X} = \{(\mathbf{x}_i, y_i)\colon \mathbf{x}_i \in \mathcal{F}, y_i \in \{0, \pm 1\},\ i = 1, \ldots, N\}$, where $\mathbf{x}_i$ denotes the representation of the i-th training video and y_i denotes the corresponding ground truth label. Labels -1, $+1$ correspond respectively to negative, positive training examples.

In order to train an event detector from positive video examples, firstly, we utilized an extended and speeded-up version of our Kernel Subclass Discriminant Analysis [61, 62] for dimensionality reduction and after that we used a fast linear SVM (AGS-DA+LSVM) in order to train an event detector.

Specifically, two types of visual information have been used for training the event detectors: motion features and DCNN-based features. We briefly describe the different visual modalities in the following:

- Each video is decoded into a set of keyframes at fixed temporal intervals (approximately 2 keyframes per second). We annotated the video frames based on 12988 ImageNet [47] concepts, 345 TRECVID SIN [63] concepts, 500 event-related concepts [64], 487 sport-related concepts [65] and 205 place-related concepts [66]. To obtain scores regarding the 12988 ImageNet concepts we used the pre-trained GoogLeNet provided by [12]. We also experimented with a subset of the 12988 concepts; in order to do that we self-trained a GoogLeNet network [67] on 5055 ImageNet concepts (gnet5k). To obtain the scores regarding the 345 TRECVID SIN concepts and the 487 sport-related concepts we fine-tuned (FT) the gnet5k network on the TRECVID AVS development dataset and on the YouTube Sports-1M dataset [65], respectively. We also used the EventNet [64] that consists of 500 events and the Places205-GoogLeNet, which was trained on 205 scene categories of Places Database [66]. All the above networks were also used as feature generators. I.e., the output of one or more hidden layers was used as a global frame representation.
- For encoding motion information we use improved dense trajectories (DT) [68]. Specifically, we employ the following four low-level feature descriptors: Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF) and Motion Boundary Histograms in both x (MBHx) and y (MBHy) directions. Hellinger kernel normalization is applied to the resulting feature vectors, followed by Fisher Vector (FV) encoding with 256 GMM codewords. Subsequently, the four feature

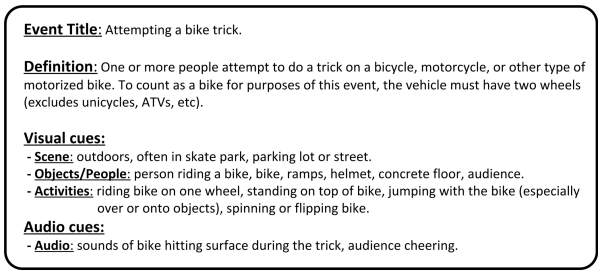

Figure 1.4 The event kit text for the event class "Attempting a bike trick"

vectors are concatenated to yield the final motion feature descriptor for each video in $\mathbb{R}^{101376}$.

The final feature vector representing a video is formed by concatenating the feature vectors derived for each visual modality (motion, model vectors), yielding a new feature vector in $\mathbb{R}^{153781}$.

1.4.3 Learning solely from textual descriptors - Zero-example learning

In this section we present a method we developed in [69] that builds a fully automatic zero-example event detection system as presented in Fig. 1.5. The developed system takes as input the event kit, i.e., a textual description of the event query, and retrieves the most related videos from the available event collection. We assume that the only knowledge available, with respect to each event class, is a textual description of it, which consists of a title, a free-form text, and a list of possible visual and audio cues, as in [70, 71]. Fig. 1.4 shows an example of such a textual description for the event class *Attempting a bike trick*. For linking this textual information with the visual content of the videos that we want to examine, similarly to [72, 73], we a) use a pool of N_c concepts along with their titles and in some cases a limited number of subtitles (e.g. concept *bicycle-built-for-two* has the subtitles *tandem bicycle* and *tandem*), and b) a pre-trained detector (based on DCNN output scores) for each concept.

Given the textual description of an event, our system first identifies N words or phrases that most closely relate to the event; this word-set is called Event Language Model (ELM). The ELM is based on the automatic extraction of word terms from the visual and audio cues of the event kit along with the title of the event. In parallel, for each of the X concepts of our concept pool, our framework similarly identifies M words or phrases: the Concept Language Model (CLM) of the corresponding concept using the top-10 articles in Wikipedia and transforming this textual information in a BoW representation.

Subsequently, for each word in ELM and each word in each one of CLMs we calculate the Explicit Semantic Analysis (ESA) similarity [74] between them. For each CLM, the resulting $N \times M$ distance matrix expresses the relation between the given

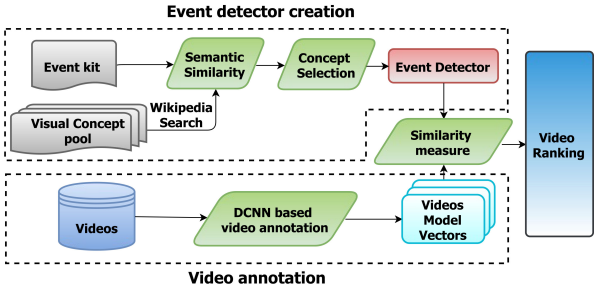

Figure 1.5 The proposed pipeline for zero-example event detection.

event and the corresponding concept. In order to compute a single score expressing this relation, we apply to this matrix the Hausdorff distance. Consequently, a score is computed for each pair of ELM and CLM. The N_c considered concepts are ordered according to these scores (in descending order) and the K-top concepts along with their scores constitute our event detector.

In contrast to [60] and [59], where the number of selected concepts is fixed across the different events and motivated by statistical methods such as PCA [75], where a fraction of components are enough to efficiently or even better describe the data, we propose a statistical strategy that decides on the appropriate number of concepts k, where $k \leq k'$, that should be kept for an event query. First, we check if the event title is semantically close to any of the available concepts from the concept pool. If so, these concepts are used as the event detector. If this is not the case, our strategy orders the vector of concepts scores $\mathbf{d}'$ in descending order, constructs an exponential curve, and then selects the first k concepts so that the corresponding area under the curve is at the $X\%$ of the total area under the curve. This procedure, consequently returns different number of selected concepts for different target events. For example for the event "*Attempting ordering the a bike trick*" the selected concepts are the following four: "*ride a dirt bike*", "*mountain biking*", "*put on a bicycle chain*", "*ride a bicycle*", while for the event "*Cleaning an appliance*" only the concept "*clean appliance*" is selected. The final event detector is a k-element vector that contains the relatedness scores of the selected concepts.

In parallel, each video is decoded into as set of keyframes at fixed temporal intervals. Then, a set of pre-trained concept-based DCNNs are applied to every keyframe and each keyframe is represented by the direct output of those networks. Finally, a video model vector is computed by averaging (in terms of arithmetic mean) the corresponding keyframe-level representations. Each element of a model vector indicates the degree that each of the predefined concepts appears in the video. The distance between an event detector and each of the video-level model vectors is calculated, and the h videos with the smallest distance are retrieved. As distance measure we choose the histogram intersection, which calculates the similarity of two discretized probability distributions and is defined as follows:
$K_{\cap}(a,b) = \sum_{i=1}^{k} min(a_i, b_i)$.

E021 - Attempting a bike trick	E031 - Beekeeping
E022 - Cleaning an appliance	E032 - Wedding shower
E023 - Dog show	E033 - Non-motorized vehicle repair
E024 - Giving directions to a location	E034 - Fixing musical instrument
E025 - Marriage proposal	E035 - Horse riding competition
E026 - Renovating a home	E036 - Felling a tree
E027 - Rock climbing	E037 - Parking a vehicle
E028 - Town hall meeting	E038 - Playing fetch
E029 - Winning a race without a vehicle	E039 - Tailgating
E030 - Working on a metal crafts project	E040 - Tuning a musical instrument

Table 1.8 MED 2016 Pre-Specified (PS) events.

1.4.4 Experimental study

1.4.4.1 Dataset and Experimental Setup

Multimedia Event Detection (MED)[76] task is part of TRECVID evaluation which is a popular benchmarking activity. The goal of MED task is to support the creation of event detection and retrieval technologies that will permit users to define their own complex events and to quickly and accurately search large collections of multimedia clips. For this reason MED provides large-scale datasets for training and evaluation purposes as well as a set of events which consists of an event name, definition, explication (textual exposition of the terms and concepts), evidential descriptions, and illustrative video exemplars.

For training purposes of learning form positive examples method we used the PS-Training video sets consisting of 2000 (80 hours) positive (or near-miss) videos as positive exemplars, and the Event-BG video set containing 5000 (200 hours) of background videos as negative ones.

To evaluate both systems (i.e. learning form positive examples and zero-example learning) a common video dataset was used. We processed the MED16–EvalSub set consisting of 32000 videos (960 hours). We evaluate all the methods on the 20 MED2016 [76] Pre-Specified events (E021-E040) for which event kits are provided. We evaluate all the methods in terms of the Mean Average Precision (MAP) and Mean Inferred Average Precision (mInfAP).

1.4.4.2 Experimental Results: Learning from positive examples

In this section, we validate the performance of the presented system for learning video event detectors from positive samples 1.4.2. We simulate two different case scenarios. In the first scenario only few (10) positive video samples per event are available for training while in the second one, an abundant of training video samples are available (100 positive videos) for each individual event. Table 1.9 illustrates how the performance of our method is affected when different amount of video samples is used.

Undoubtedly the plethora of positive video samples, leads to significant performance improvement,i.e. the relative improvement is 45.3% in terms of MAP and

Table 1.9 Learning from positive examples results

	MAP(%)	mInfAP@200(%)
10 positive videos	31.8	34.2
100 positive videos	46.2	47.5

38% in terms of mInfAP@200%.

1.4.4.3 Experimental Results: Zero-example learning

In this section we evaluate the developed method for event-based video search in the case that solely the event's textual description is given (Section 1.4.3. For our zero-example learning experiments we utilize 2 different concept pools as well as an extension of the pipeline with an extra pseudo-relevance feedback step with online training in which the top retrieved videos are used as positive video samples and the AGSDA+LSVM method, as described in 1.4.2 section, is used to train new event detectors.

- **DCNN13K**: In our first configuration we use the annotation from 2 different DCNNs: i) The pre-trained GoogLeNet provided by [67] trained on 12988 ImageNet concepts [47] and ii) the EventNet [64] that consists of 500 events.
- **DCNN14K**: In this configuration, we use the annotation from 5 different DCNNs: i) The pre-trained GoogLeNet provided by [67] trained on 12988 ImageNet concepts [47], ii) the GoogLeNet [12] self-trained on 5055 ImageNet concepts (gnet5k) and subsequently fine-tuned for 345 TRECVID SIN [63] concepts, iii) the gnet5k network fine-tuned for 487 sport-related [65] concepts, iv) the EventNet [64] that consists of 500 events and v) the Places205-GoogLeNet, trained on 205 scene categories [66].
- **Train**: In this configuration an online training stage are utilized using the top-10 retrieved videos from the first configuration as positive samples, and the learning procedure of 1.4.2 section.

Table 1.10 illustrates how the performance of the above three different approaches for zero-example learning.

Table 1.10 Zero-example learning results

	MAP(%)	mInfAP@200(%)
DCNN13K	14.6	12.2
DCNN14K	14.5	11.9
Train	**16.2**	**14.2**

It is clear that adding the pseudo-relevance feedback step by using the top retrieved videos as positive samples has a significant impact to our performance (the relative improvement is 10.96% in terms of MAP and 16.39% in terms of mInfAP@200(%)).

However, the arbitrary addition of heterogeneous different concept pools leads to a small performance decrease due to noisy information that those concept pools were

add to our system (the Percentage relative reduction in terms of mInfAP@200(%) is -2.5%).

Similarly to the concept detection case, a qualitative analysis shows that the event-based video search works satisfactorily. For example, considering the event "Horse riding competition" that reaches an InfAP@200 of 13.9% and AP of 16.4% on the MED16–EvalSub dataset, we observe that among the top-20 retrieved videos 16 of them are positive, and similarly among the top-50 retrieved ones the 32 are positive.

1.5 Conclusions

In this chapter we surveyed the literature and presented in more detail some of the methods that we have developed for concept-based and event-based video search. In terms of concept-based video search we presented methods from three machine-learning areas (cascades of classifiers, multi-task learning and exploitation of label relations), that can improve the accuracy of concept-based video search and/or reduce computational complexity, in comparison to similar methods, thus enabling effective real-time video search. Our experiments on two large-scale datasets, i.e., TRECVID SIN 2013 and TRECVID SIN 2015, show the effectiveness and scalability of the presented methods, and especially how combining cascades of detectors, MTL, a two-layer stacking architecture and dimensionality reduction with AGSDA in a concept detection system can boost concept-based video search. Concerning the event-based video search, we presented methods for learning when ground-truth video data are available for a given event, as well as when solely a textual description of event is given without any positive video examples. Our experiments on a large-scale video event dataset (MED16–EvalSub) showed that the exploitation of a larger number of video samples generally leads to better performance; but also in the more realistic scenario of zero-example event detection problem, we presented a state of the art method that achieves very promising results.

1.6 Acknowledgments

This work was supported by the EU's Horizon 2020 research and innovation programme under grant agreements H2020-693092 MOVING, H2020-687786 InVID and H2020-732665 EMMA.

References

1 Snoek, C.G.M. and Worring, M. (2009) Concept-Based Video Retrieval. *Foundations and Trends in Information Retrieval*, **2** (4), 215–322.

2 Sidiropoulos, P., Mezaris, V., and Kompatsiaris, I. (2014) Video tomographs and a base detector selection strategy for improving large-scale video concept detection. *IEEE Trans. on Circuits and Systems for Video Technology*, **24** (7), 1251–1264.

3 Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011) ORB: An efficient alternative to SIFT or SURF, in *IEEE Int. Conf. on Computer Vision*, pp. 2564–2571.

4 Lowe, D.G. (2004) Distinctive Image Features from Scale-Invariant Keypoints. *Int. Journal of Computer Vision*, **60** (2), 91–110.

5 Bay, H., Tuytelaars, T., and Van Gool, L. (2006) Surf: Speeded up robust features, in *ECCV*, *LNCS*, vol. 3951, Springer, pp. 404–417.

6 Van de Sande, K.E.A., Gevers, T., and Snoek, C.G.M. (2010) Evaluating color descriptors for object and scene recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **32** (9), 1582–1596.

7 Csurka, G. and Perronnin, F. (2011) Fisher vectors: Beyond bag-of-visual-words image representations, in *Computer Vision, Imaging and Computer Graphics. Theory and Applications*, *Communications in Computer and Information Science*, vol. 229 (eds P. Richard and J. Braz), Springer Berlin, pp. 28–42.

8 Jegou, H. and et al. (2010) Aggregating local descriptors into a compact image representation, in *IEEE on Computer Vision and Pattern Recognition (CVRP 2010)*, San Francisco, CA, pp. 3304–3311.

9 Simonyan, K. and Zisserman, A. (2014) Very deep convolutional networks for large-scale image recognition. *arXiv technical report*.

10 Jia, Y., Shelhamer, E., Donahue, J., and et al. (2014) Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

11 Krizhevsky, A., Ilya, S., and Hinton, G. (2012) Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems (NIPS 2012)*, Curran Associates, Inc., pp. 1097–1105.

12 Szegedy, C. and et al. (2015) Going deeper with convolutions, in *CVPR 2015*. URL `http://arxiv.org/abs/1409.4842`.

13 Safadi, B., Derbas, N., Hamadi, A., Budnik, M., Mulhem, P., and Qu, G. (2014) LIG at TRECVid 2014 : Semantic Indexing tion of the semantic indexing, in *TRECVID 2014 Workshop*, Gaithersburg, MD, USA.

14 Snoek, C.G.M., Sande, K.E.A.V.D., Fontijne, D., Cappallo, S., Gemert, J.V., and Habibian, A. (2014) MediaMill at TRECVID 2014 : Searching Concepts , Objects , Instances and Events in Video, in *TRECVID 2014 Workshop*, Gaithersburg, MD, USA.

15 Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014) How transferable are features in deep neural networks? *CoRR*, **abs/1411.1792**.

16 Pittaras, N., Markatopoulou, F., Mezaris, V., and et al. (2017) *Comparison of Fine-Tuning and Extension Strategies for Deep Convolutional Neural Networks*, Springer, Cham, pp. 102–114.

17 Gkalelis, N., Mezaris, V., Kompatsiaris, I., and Stathaki, T. (2013) Mixture subclass discriminant analysis link to restricted gaussian model and other generalizations. *IEEE transactions on neural networks and learning systems*, **24** (1), 8–21.

18 Agullo, E., Augonnet, C., Dongarra, J., and et. al (2010), Faster, cheaper, better–a hybridization methodology to develop linear algebra software for gpus.

19 Gkalelis, N. and Mezaris, V. (2014) Video event detection using generalized subclass discriminant analysis and linear support vector machines, in *Proceedings of international conference on multimedia retrieval*, ACM, p. 25.

20 Schölkopf, B. and Smola, A.J. (2002) *Learning with kernels: support vector machines, regularization, optimization, and beyond*, MIT press.

21 Athanasopoulos, A., Dimou, A., Mezaris, V., and Kompatsiaris, I. (2011) Gpu acceleration for support vector machines, in *WIAMIS 2011: 12th International Workshop on Image Analysis for*

Multimedia Interactive Services, Delft, The Netherlands, April 13-15, 2011, TU Delft; EWI; MM; PRB.

22 Arestis-Chartampilas, S., Gkalelis, N., and Mezaris, V. (2015) Gpu accelerated generalised subclass discriminant analysis for event and concept detection in video, in *Proceedings of the 23rd ACM international conference on Multimedia*, ACM, pp. 1219–1222.

23 Arestis-Chartampilas, S., Gkalelis, N., and Mezaris, V. (2016) Aksda-msvm: A gpu-accelerated multiclass learning framework for multimedia, in *Proceedings of the 2016 ACM on Multimedia Conference*, ACM, pp. 461–465.

24 Strat, S.T., Benoit, A., Bredin, H., Quenot, G., and Lambert, P. (2012) Hierarchical late fusion for concept detection in videos, in *European Conference on Computer Vision (ECCV) 2012. Workshops and Demonstrations*, *Lecture Notes in Computer Science*, vol. 7585, Springer, pp. 335–344.

25 Sidiropoulos, P., Mezaris, V., and Kompatsiaris, I. (2014) Video tomographs and a base detector selection strategy for improving large-scale video concept detection. *IEEE Trans. on Circuits and Systems for Video Technology*, **24** (7), 1251–1264, doi:10.1109/TCSVT.2014.2302554.

26 Nguyen, C., Vu Le, H., and Tokuyama, T. (2011) Cascade of multi-level multi-instance classifiers for image annotation, in *KDIR'11*, pp. 14–23.

27 Cheng, W.C. and Jhan, D.M. (2011) A cascade classifier using adaboost algorithm and support vector machine for pedestrian detection, in *IEEE Int. Conf. on SMC*, pp. 1430–1435, doi:10.1109/ICSMC.2011.6083870.

28 Markatopoulou, F., Mezaris, V., and Patras, I. (2015) Cascade of classifiers based on binary, non-binary and deep convolutional network descriptors for video concept detection, in *2015 IEEE International Conference on Image Processing (ICIP)*, pp. 1786–1790, doi:10.1109/ICIP.2015.7351108.

29 Mousavi, H., Srinivas, U., Monga, V., Suo, Y., and et al. (2014) Multi-task image classification via collaborative, hierarchical spike-and-slab priors, in *Proc. of the IEEE Int. Conf. on Image Processing (ICIP 2014)*, pp. 4236–4240.

30 Daumé, III, H. (2009) Bayesian multitask learning with latent hierarchies, in *Proc. of the 25th Conf. on Uncertainty in Artificial Intelligence (UAI '09)*, AUAI Press, Arlington, Virginia, US, pp. 135–142.

31 Argyriou, A., Evgeniou, T., and Pontil, M. (2008) Convex multi-task feature learning. *Machine Learning*, **73** (3), 243–272.

32 Zhou, J., Chen, J., and Ye, J. (2011) Clustered multi-task learning via alternating structure optimization. *Advances in Neural Information Processing Systems (NIPS 2011)*.

33 Sun, G., Chen, Y., Liu, X., and Wu, E. (2015) Adaptive multi-task learning for fine-grained categorization, in *Proc. of the IEEE Int. Conf. on Image Processing (ICIP 2015)*, pp. 996–1000.

34 Smith, J., Naphade, M., and Natsev, A. (2003) Multimedia semantic indexing using model vectors, in *2003 Int. Conf. on Multimedia and Expo. (ICME)*, IEEE, NY, pp. 445–448, doi:10.1109/ICME.2003.1221649.

35 Tsoumakas, G., Dimou, A., Spyromitros, E., and et al. (2009) Correlation-Based Pruning of Stacked Binary Relevance Models for Multi-Label learning, in *Proceedings of the 1st international workshop on learning from multi-label data*, pp. 101–116.

36 Jiang, W., Chang, S.F., and Loui, A.C. (2006) Active context-based concept fusion with partial user labels, in *IEEE Int. Conf. on Image Processing*, IEEE, NY.

37 Weng, M.F. and Chuang, Y.Y. (2012) Cross-Domain Multicue Fusion for Concept-Based Video Indexing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **34** (10), 1927–1941.

38 Qi, G.J. and et al. (2007) Correlative multi-label video annotation, in *Proc. of the 15th Int. Conf. on Multimedia*, ACM, NY, pp. 17–26.

39 Markatopoulou, F., Mezaris, V., and Patras, I. (2016) *Ordering of Visual Descriptors in a Classifier Cascade Towards Improved Video Concept Detection*, Springer International Publishing, pp. 874–885.

40 Tibshirani, R. (1996) Regression shrinkage

and selection via the lasso. *Journal of the Royal Statistical Society*, **Series B**, 267–288.

41 Zhou, J., Chen, J., and Ye, J. (2011) *MALSAR: Multi-tAsk Learning via StructurAl Regularization*, Arizona State University.

42 Tsoumakas, G., Katakis, I., and Vlahavas, I. (2010) Mining multi-label data, in *Data Mining and Knowledge Discovery Handbook*, Springer, Berlin, pp. 667–686.

43 Markatopoulou, F., Mezaris, V., Pittaras, N., and Patras, I. (2015) Local features and a two-layer stacking architecture for semantic concept detection in video. *IEEE Transactions on Emerging Topics in Computing*, **3** (2), 193–204, doi:10.1109/TETC.2015.2418714.

44 Zhang, M.L. and Zhou, Z.H. (2007) ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, **40** (7), 2038–2048, doi:10.1016/j.patcog.2006.12.019.

45 Over, P., Awad, G., Fiscus, J., Sanders, G., and Shaw, B. (2013) Trecvid 2013 – an overview of the goals, tasks, data, evaluation mechanisms and metrics, in *Proc. of TRECVID 2013*, NIST, USA.

46 Yilmaz, E., Kanoulas, E., and Aslam, J.A. (2008) A simple and efficient sampling method for estimating ap and ndcg, in *31st ACM SIGIR Int. Conf. on Research and Development in Information Retrieval*, ACM, USA, pp. 603–610.

47 Russakovsky, O., Deng, J., Su, H., and et al. (2015) ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, **115** (3), 211–252, doi:10.1007/s11263-015-0816-y.

48 Chellapilla, K., Shilman, M., and Simard, P. (2006) Combining multiple classifiers for faster optical character recognition, in *7th Int. Conf. on Document Analysis Systems*, Springer, Berlin, pp. 358–367.

49 Bao, L., Yu, S.I., and Hauptmann, A. (2011) Cmu-informedia @ trecvid 2011 semantic indexing, in *TRECVID 2011 Workshop*, Gaithersburg,MD, USA.

50 Blanken, H.M., de Vries, A.P., Blok, H.E., and Feng, L. (2005) *Multimedia Retrieval*, Springer Berlin Heidelberg, NY.

51 Gkalelis, N., Mezaris, V., Dimopoulos, M., Kompatsiaris, I., and Stathaki, T. (2013) Video event detection using a subclass recoding error-correcting output codes framework, in *Multimedia and Expo (ICME), IEEE Int. Conf. on*, IEEE, pp. 1–6.

52 Over, P., Fiscus, J., Sanders, G., and et al. (2015) Trecvid 2015 – an overview of the goals, tasks, data, evaluation mechanisms and metrics, in *TRECVID 2015 Workshop*, NIST, USA.

53 Elhoseiny, M., Saleh, B., and Elgammal, A. (2013) Write a classifier: Zero-shot learning using purely textual descriptions, in *Computer Vision (ICCV), IEEE Int. Conf. on*, IEEE, pp. 2584–2591.

54 Fu, Z., Xiang, T., Kodirov, E., and Gong, S. (2015) Zero-shot object recognition by semantic manifold distance, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2635–2644.

55 Norouzi, M., Mikolov, T., Bengio, S., and et al. (2013) Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*.

56 Jiang, Y.G., Bhattacharya, S., Chang, S.F., and Shah, M. (2012) High-level event recognition in unconstrained videos. *International Journal of Multimedia Information Retrieval*, pp. 1–29.

57 Habibian, A., Mensink, T., and Snoek, C.G. (2014) Videostory: A new multimedia embedding for few-example recognition and translation of events, in *Proc. of the ACM Int. Conf. on Multimedia*, ACM, pp. 17–26.

58 Elhoseiny, M., Liu, J., Cheng, H., Sawhney, H., and Elgammal, A. (2015) Zero-shot event detection by multimodal distributional semantic embedding of videos. *arXiv preprint arXiv:1512.00818*.

59 Lu, Y.J., Zhang, H., de Boer, M., and Ngo, C.W. (2016) Event detection with zero example: select the right and suppress the wrong concepts, in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, ACM, pp. 127–134.

60 Tzelepis, C., Galanopoulos, D., Mezaris, V., and Patras, I. (2016) Learning to detect video events from zero or very few video examples. *Image and vision Computing*, **53**, 35–44.

61 Gkalelis, N., Mezaris, V., Kompatsiaris, I.,

and Stathaki, T. (2013) Mixture subclass discriminant analysis link to restricted Gaussian model and other generalizations. *IEEE Trans. Neural Netw. Learn. Syst.*, **24** (1), 8–21.

62 Gkalelis, N. and Mezaris, V. (2014) Video event detection using generalized subclass discriminant analysis and linear support vector machines, in *International Conference on Multimedia Retrieval, ICMR '14, Glasgow, United Kingdom - April 01 - 04, 2014*, p. 25.

63 Smeaton, A.F., Over, P., and Kraaij, W. (2009) High-Level Feature Detection from Video in TRECVid: a 5-Year Retrospective of Achievements, in *Multimedia Content Analysis, Theory and Applications* (ed. A. Divakaran), Springer Verlag, Berlin, pp. 151–174.

64 Ye, G., Li, Y., Xu, H., Liu, D., and Chang, S.F. (2015) Eventnet: A large scale structured concept library for complex event detection in video, in *ACM MM*.

65 Karpathy, A., Toderici, G., Shetty, S., and et al. (2014) Large-scale video classification with convolutional neural networks, in *CVPR*.

66 Zhou, B., Lapedriza, A., Xiao, J., and et al. (2014) Learning deep features for scene recognition using places database, in *Advances in neural information processing systems*, pp. 487–495.

67 Mettes, P., Koelma, D., and Snoek, C. (2016) The imagenet shuffle: Reorganized pre-training for video event detection. *arXiv preprint arXiv:1602.07119*.

68 Wang, H. and Schmid, C. (2013) Action recognition with improved trajectories, in *IEEE International Conference on Computer Vision*, Sydney, Australia.

69 Galanopoulos, D., Markatopoulou, F., Mezaris, V., and Patras, I. (2017) Concept language models and event-based concept number selection for zero-example event detection, in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, ACM.

70 Younessian, E., Mitamura, T., and Hauptmann, A. (2012) Multimodal knowledge-based analysis in multimedia event detection, in *Proc. of the 2nd ACM Int. Conf. on Multimedia Retrieval*, ACM, pp. 51:1–51:8.

71 Jiang, L., Mitamura, T., Yu, S.I., and Hauptmann, A.G. (2014) Zero-example event search using multimodal pseudo relevance feedback, in *Proceedings of International Conference on Multimedia Retrieval*, ACM, p. 297.

72 Wu, S., Bondugula, S., Luisier, F., Zhuang, X., and Natarajan, P. (2014) Zero-shot event detection using multi-modal fusion of weakly supervised concepts, in *Computer Vision and Pattern Recognition (CVPR), IEEE Conf. on*, IEEE, pp. 2665–2672.

73 Yu, S.I., Jiang, L., Mao, Z., Chang, X., and et al. (2014) Informedia at TRECVID 2014 MED and MER, in *NIST TRECVID Video Retrieval Evaluation Workshop*.

74 Gabrilovich, E. and Markovitch, S. (2007) Computing semantic relatedness using wikipedia-based explicit semantic analysis., in *IJCAI*, vol. 7, vol. 7, pp. 1606–1611.

75 Jolliffe, I. (2002) *Principal component analysis*, Wiley Online Library.

76 Awad, G., Fiscus, J., Michel, M., and et al. (2016) Trecvid 2016: Evaluating video search, video event detection, localization, and hyperlinking, in *Proceedings of TRECVID*, vol. 2016, vol. 2016.