Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

Ordering of Visual Descriptors in a Classifier Cascade Towards Improved Video Concept Detection

Foteini Markatopoulou[1,2], Vasileios Mezaris[1], and Ioannis Patras[2]

[1] Information Technologies Institute (ITI), CERTH, Thermi 57001, Greece
{markatopoulou,bmezaris}@iti.gr
[2] Queen Mary University of London, Mile end Campus, UK, E14NS
i.patras@qmul.ac.uk

Abstract. Concept detection for semantic annotation of video fragments (e.g. keyframes) is a popular and challenging problem. A variety of visual features is typically extracted and combined in order to learn the relation between feature-based keyframe representations and semantic concepts. In recent years the available pool of features has increased rapidly, and features based on deep convolutional neural networks in combination with other visual descriptors have significantly contributed to improved concept detection accuracy. This work proposes an algorithm that dynamically selects, orders and combines many base classifiers, trained independently with different feature-based keyframe representations, in a cascade architecture for video concept detection. The proposed cascade is more accurate and computationally more efficient, in terms of classifier evaluations, than state-of-the-art classifier combination approaches.

Keywords: Concept detection; video analysis; cascade architecture; classifier ordering.

1 Introduction

Video concept detection is a popular research topic that aims to annotate video fragments (e.g. keyframes) with semantic concept labels (e.g. sky, people etc.). Large-scale semantic concept detection systems mainly follow a process where a video is initially segmented into meaningful fragments, called shots; each shot is represented by e.g. one or more characteristic keyframes; and, several features (e.g. different local visual descriptors) are extracted from the keyframes (or any other chosen representation) of each shot. Given a ground-truth annotated video training set, supervised machine learning algorithms are used for building multiple independent base classifiers (concept detectors), using different types of features, for the same concept; the outputs of them are combined by means of late fusion. This ensemble-based approach is more accurate than using a single base classifier, trained on a single type of features (e.g. SIFT only). In this work

Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

we propose an improved way of ordering and combining independently trained concept detectors using a cascade. The proposed cascade combines handcrafted (e.g. SIFT) and deep convolutional neural network (DCNN) features, and is computationally more efficient and more accurate than other combination approaches by adjusting the required processing (i.e. evaluate fewer classifiers) based on the input video fragment.

The rest of this paper is organized as follows: Section 2 reviews related work on learning and combining concept detectors. Section 3 introduces the proposed cascade architecture. Section 4 presents the experimental results and, finally, Section 5 presents conclusions.

2 Related Work

State of the art feature extraction methods are often based on local descriptors (e.g. SIFT [7], SURF [2]) in combination with encoding approaches (e.g. VLAD [5], FV [12]) in order to extract global, feature-based keyframe representations. In the last few years, features extracted with the use of pre-trained deep convolutional neural networks (DCNN) have also shown excellent results [15]. Using any of the above features, concept detection is typically treated as multiple independent binary classification problems (one per concept). That is, given the feature-based keyframe representations and also the ground-truth concept annotations for each keyframe, any supervised machine learning algorithm that solves classification problems, such as Support Vector Machine (SVM), can be trained in order to learn the relations between the low-level keyframe representations and the high-level semantic concepts.

It has been shown that combining many different keyframe representations (e.g. SIFT, RGB-SIFT, DCNN) for the same concept, instead of using a single feature (e.g. only SIFT), improves the concept detection accuracy [8]. 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 [16] 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, [14] 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 [18], [3], [10], [9], [4]. In a cascade architecture 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

Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

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 or object. Cascades of classifiers have been mainly used in object detection tasks [18], however they have also been briefly examined for video/image concept detection [10], [9].

Cascades developed for object and face detection are mainly boosting-based [18], [3], [10], [4], [1]. Each stage of the cascade is build using a boosting algorithm such as AdaBoost. Such approaches require the presence of a big pool of weak features (e.g. Haar-like features) in order to combine them and build a strong classifier. In contrast, video concept detection systems utilize a different kind of features, visual local descriptors encoded into global image representations or DCNN-based features that alone can build strong classifiers without boosting. For example, [9] presents a cascade with fixed ordering of the stages in terms of classifier accuracy and a simple threshold selection strategy that selects one rejection threshold per stage on the probability output of a classifier. The authors of [9] use the above cascade to combine binary, non-binary and DCNN-based features. In the present work, the proposed cascade is also developed so as to combine similar types of features, however in contrast to [9] which is based on a fixed ordering of the cascade stages, the proposed algorithm dynamically selects, orders, and combines a larger number of pre-trained base classifiers. This leads to both concept detection effectiveness and computational efficiency gains.

3 Cascade Construction with Re-trained Classifiers

3.1 Cascade Architecture Overview

Figure 1 shows a cascade architecture suitable for combining many base classifiers that have been trained for the same concept [9]. Each stage j of the cascade encapsulates a stage classifier D_j that either combines many base classifiers $(B_1, B_2, ..., B_{f_j})$ 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. In the first case, the 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. Given a set of pre-trained classifiers, we will present an 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 above cascade.

Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

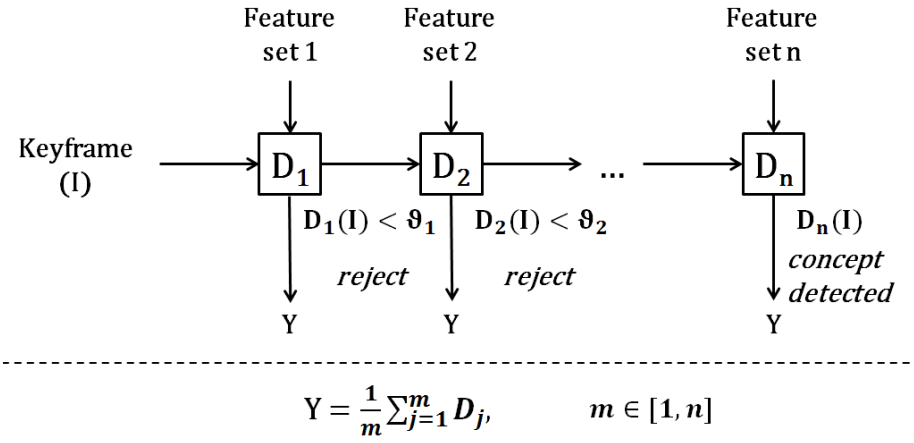

Fig. 1. Block diagram of a cascade architecture for one concept.

3.2 Problem Definition and Search Space

Let $D = \{D_1, D_2, ..., D_n\}$ be a set of n independently trained classifiers for a specific concept. 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 finite 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:

$$(\mathbf{S}^\star, \boldsymbol{\theta}^\star) = \underset{(\mathbf{S}, \boldsymbol{\theta})}{\operatorname{argmax}} \{F(D_{\mathbf{S}}, T, \boldsymbol{\theta})\}, \tag{1}$$

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.

Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

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 finite 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.

3.3 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). 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),

$$s_n^\star = \underset{s_n}{\operatorname{argmax}} \{F_{AP}(D_{\mathbf{S}_0}, T, \boldsymbol{\theta}_0)\}. \tag{2}$$

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). 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):

$$(s_j^\star, \theta_{s_j}^\star) = \underset{(s_j, \theta_{s_j})}{\operatorname{argmax}} \{F_{AP}(D_{\mathbf{S}_j}, T, \boldsymbol{\theta}_j)\}. \tag{3}$$

Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

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),
$\mathbf{S}_0 = [-1, -1, ..., s_n]^\top$, $\boldsymbol{\theta}_0 = [0, 0, .., 0]^\top$
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),
  $\mathbf{S}_j = [..., s_j, -1, ..., s^\star_n]^\top$, $\boldsymbol{\theta}_j = [..., \theta_{s_j}, 0, ..., \theta^\star_{s_n}]^\top$
  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

The algorithm finds the pair of (s_j, θ_{s_j}) that optimizes (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

$$\mathbf{S}^\star = \underset{\mathbf{S}\in\{\mathbf{S}^\star_0, \mathbf{S}^\star_1, ..., \mathbf{S}^\star_{n-1}\}}{\text{argmax}} \{F_{AP}(D_{\mathbf{S}}, T, \boldsymbol{\theta})\}, \tag{4}$$

which is the sequence that optimizes (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.

4 Experiments

4.1 Dataset and Experimental Setup

Our experiments were performed on the TRECVID 2013 Semantic Indexing (SIN) dataset [11], which consists of a development set and a test set (approximately 800 and 200 hours of internet archive videos for training and testing,

Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

respectively). We evaluated our system on the test set using the 38 concepts that were evaluated as part of the TRECVID 2013 SIN Task [11]. The video indexing problem was examined; that is, given a concept, we measure how well the top retrieved video shots for this concept truly relate to it. For experimenting with all methods, one keyframe was extracted for each video shot. Regarding local feature extraction, we followed the experimental setup of [8]. More specifically, we extracted nine local descriptors, presented in Table 1. All the local descriptors were compacted using PCA and were subsequently aggregated using the VLAD encoding. The VLAD vectors were reduced to 4000 dimensions. In addition, we used features based on three different pre-trained convolutional neural networks: i) The 16-layer deep ConvNet network provided by [15], ii) the 22-layer GoogLeNet network provided by [17], and iii) the 8-layer CaffeNet network described in [6]. We applied each of these networks on the TRECVID keyframes and we used as a feature i) the output of the last hidden layer of ConvNet (fc7), which resulted to a 4096-element vector, ii) the output of the last fully-connected layer of CaffeNet (fc8), which resulted to a 1000-element, iii) the output of the last fully-connected layer of GoogLeNet (loss3). We refer to these features as CONV, CAFFE and GNET in the sequel, respectively.

To train our base classifiers, for each concept, a training set was assembled that included all negative annotated training examples for the given concept and three copies of each positive training sample (in order to account for the most often limited number of the latter). Then the positive and negative ratio of training examples was fixed by randomly rejecting any excess negative samples, to achieve an 1:6 ratio. This is important for building a balanced classifier. Given the twelve different types of feature vectors described above, for each concept we trained twelve different base-classifiers, using linear SVMs. In all cases, the final step of concept detection was to refine the calculated detection scores by employing the re-ranking method proposed in [13].

We compared the proposed cascade (Section 3) with five different ensemble combination approaches: i) Late-fusion with arithmetic mean [16]. ii) The ensemble pruning method proposed by [14]. iii) The cascade proposed by [9]. In this case we only applied the thresholding strategy of [9], that is, we did not perform any retraining, which appeared to be less accurate and computationally more expensive. 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 [3]. In contrast to [3] 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 [1]). We refer to this method as AdaBoost. Both for the proposed and also for the cascade-dynamic method we used quantization to ensure that the optimized cascade generalizes well to unseen samples. In these lines, instead of searching for candidate thresholds on all the M examples of a validation set, we sorted the score values in descending order and split at every M/Q example (Q was set to 32). For all the methods, except for the Late-fusion

Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

Table 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	Last hidden layer of ConvNet

Table 2. Performance (MXinfAP, %) for different classifier combination approaches.

		M1	M2	M3	M4	M5	M6
RunID	Stage classifiers	Late-fusion [16]	Ensemble pruning [14]	Cascade-thresholding [9]	Cascade-dynamic [3]	AdaBoost [1]	Cascade-proposed
R1	ORBx3; SURFx3;CAFFE SIFTx3	**24.97**	23.63	24.96	**24.97**	24.14	23.68
R2	ORBx3; SURFx3; SIFTx3;GNET	27.72	28.47	27.69	27.7	27.69	**28.52**
R3	ORBx3; SURFx3; SIFTx3;CONV	28.14	28.6	28.25	28.08	28.08	**28.84**
R4	ORBx3; SURFx3;CAFFE; SIFTx3;GNET; CONV	29.84	29.74	29.79	29.84	29.70	**29.96**

that does not require this, the training set was also used as the validation set. With respect to the proposed 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.

4.2 Experimental Results

Tables 1 and 2 present the results of our experiments in terms of the Mean Extended Inferred Average Precision (MXinfAP) [19], which is an approximation of the Mean Average Precision suitable for the partial ground-truth that accompanies the TRECVID dataset [11]. Table 1 presents the MXinfAP for the different types of features that were used by the algorithms of this study. Each line of this table was used as a cascade stage for the cascade-based methods (Table 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

Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

variants), while the stages of DCNN features (CAFFE, CONV, GNET) consist of one base classifier each. For the late fusion methods (Table 2: M1, M2) and the boosting-based method (Table 2: M5), the corresponding base classifiers per line of Table 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 2 presents the performance of the proposed cascade-based method and compares it with other classifier combination methods. The second column shows the stage classifiers that were considered in each run. Runs R1-R3 encapsulate nine types of features from local descriptors and only one type of DCNN features; ultimately, R4 refers to the systems that utilize six stage classifiers and all twelve types of features. The best results were reached by the proposed cascade in R4, where it 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 the following section.

4.3 Computational Complexity

We continue the analysis of our results with respect to the computational complexity of the different methods compared in this study during the training and classification phase. Table 3 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 [16], which builds n models (one for each set of features), is the simplest one. Cascade-thresholding [9] follows, which evaluates n cascade stages in order to calculate the appropriate thresholds per stage. Cascade-dynamic [3] 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 [14] 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.

Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

Table 3. 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 [16]	-	-
M2	Ensemble pruning [14]	$(2^n - 1)M$	-
M3	Cascade-thresholding [9]	$\sum_{j=0}^{n} M_j,\ M_j \subseteq Mj - 1$	-
M4	Cascade-dynamic [3]	$(n-2)Q^2$	-
M5	AdaBoost [1]	$M(n(n+1)/2)$	$n(n+1)/2$
M6	Cascade-proposed	$Q(n(n+1)/2)$	-

Table 4. Relative amount of classifier evaluations (%) for different classifier combination approaches during the classification phase.

		M1	M2	M3	M4	M5	M6
RunID	Stage classifiers	Late-fusion [16]	Ensemble pruning [14]	Cascade-thresholding [9]	Cascade-dynamic [3]	AdaBoost [1]	Cascade-proposed
R1	ORBx3; SURFx3;CAFFE SIFTx3	83.33	55.92	66.17	77.69	83.33	**53.50**
R2	ORBx3; SURFx3; SIFTx3;GNET	83.33	55.70	66.98	77.95	83.33	**52.74**
R3	ORBx3; SURFx3; SIFTx3;CONV	83.33	57.68	66.98	78.54	83.33	**54.32**
R4	ORBx3; SURFx3;CAFFE SIFTx3;CONV; GNET	100	66.67	74.94	92.38	100	**62.24**

Table 4 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 reached in R4 achieved 37.8% and 32.6% relative decrease in the amount of classifier evaluations compared to the late fusion alternative (Table 4: R4-M1) and the cascade-dynamic alternative (Table 4: R4-M4), respectively, which are the two most accurate methods after the proposed-cascade. Figure 4 presents the computational complexity of the proposed cascade-based method and compares it with other classifier combination methods, separately for each target concept. We can observe that the proposed method is computationally less expensive for 26 out of the 38 concepts.

Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

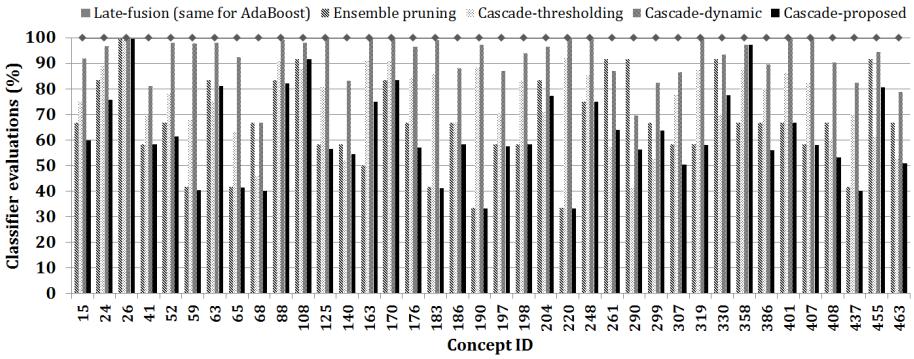

Fig. 2. Relative amount of classifier evaluations (%) per concept for R4 of Table 4.

To sum up, according to Tables 2 and 4, the three best-performing methods are the proposed-cascade, the late fusion [16] and the cascade-dynamic [3]. With respect to runs R2-R3 the proposed-cascade outperforms the two other methods, while it is always computationally more efficient during classification. When the number of features/stage classifiers increases (R4) the proposed-cascade performs slightly better in terms of MXinfAP compared to the late fusion and cascade-dynamic method, achieving 0.4% relative improvement, for both cases. At the same time it is computationally less expensive during classification. Only for R1, which uses a small number of stage classifiers, the proposed-cascade presents lower accuracy than the other two best performing methods; however, it remains computationally less expensive. 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 4 for the proposed cascade very important.

5 Conclusions

In this work we presented an improved way of ordering and combining independently trained base concept detectors using a cascade. A search-based algorithm that finds the optimal stage ordering and rejection thresholds was presented and evaluated. The resulting cascade-based concept detection method is computationally more efficient, in terms of classifier evaluations, and more accurate than other state-of-the-art approaches.

Acknowledgements This work was supported by the European Commission under contract FP7-600826 ForgetIT.

Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, Jan. 2016.
Author's accepted version. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-27671-7_73

References

1. Bao, L., et al.: CMU-Informedia@TRECVID 2011 semantic indexing. In: TRECVID 2011 Workshop. Gaithersburg, MD, USA (2011)
2. Bay, H., et al.: Speeded-up robust features (surf). Computer Vision and Image Understanding 110(3), 346–359 (2008)
3. Chellapilla, K., Shilman, M., Simard, P.: Combining multiple classifiers for faster optical character recognition. In: 7th Int. Conf. on Document Analysis Systems. pp. 358–367. Springer, Berlin (2006)
4. Cheng, W.C., Jhan, D.M.: A cascade classifier using adaboost algorithm and support vector machine for pedestrian detection. In: IEEE Int. Conf. on SMC. pp. 1430–1435 (2011)
5. Jegou, H., et al.: Aggregating local descriptors into a compact image representation. In: IEEE on Computer Vision and Pattern Recognition (CVRP 2010). pp. 3304–3311. San Francisco, CA (2010)
6. Krizhevsky, A., Ilya, S., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012)
7. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. Int. Journal of Computer Vision 60(2), 91–110 (2004)
8. Markatopoulou, F., et al.: A study on the use of a binary local descriptor and color extensions of local descriptors for video concept detection. In: MultiMedia Modeling Conf., LNCS, vol. 8935, pp. 282–293. Springer (2015)
9. Markatopoulou, F., Mezaris, V., Patras, I.: Cascade of classifiers based on binary, non-binary and deep convolutional network descriptors for video concept detection. In: IEEE Int. Conf. on Image Processing (ICIP 2015). IEEE, Canada (2015)
10. Nguyen, C., Vu Le, H., Tokuyama, T.: Cascade of multi-level multi-instance classifiers for image annotation. In: KDIR'11. pp. 14–23 (2011)
11. Over, P., et al.: Trecvid 2013 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In: Proceedings of TRECVID 2013. NIST, USA (2013)
12. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: 11th Eur. Conf. on Computer Vision: Part IV. pp. 143–156. Springer-Verlag (2010)
13. Safadi, B., Quénot, G.: Re-ranking by local re-scoring for video indexing and retrieval. In: 20th ACM Int. Conf. on Information and Knowledge Management. pp. 2081–2084. ACM, NY (2011)
14. Sidiropoulos, P., Mezaris, V., Kompatsiaris, I.: 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 (2014)
15. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv technical report (2014)
16. Strat, S.T., et al.: Hierarchical late fusion for concept detection in videos. In: ECCV 2012, LNCS, vol. 7585, pp. 335–344. Springer (2012)
17. Szegedy, C., et al.: Going deeper with convolutions. In: CVPR 2015 (2015), `http://arxiv.org/abs/1409.4842`
18. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: CVPR (2001). vol. 1, pp. 511–518 (2001)
19. Yilmaz, E., Kanoulas, E., Aslam, J.A.: A simple and efficient sampling method for estimating ap and ndcg. In: 31st ACM SIGIR Int. Conf. on Research and Development in Information Retrieval. pp. 603–610. ACM, USA (2008)