Proc. IEEE Int. Conf. on Image Processing (ICIP 2015), Quebec City, Canada, Sept. 2015.

CASCADE OF CLASSIFIERS BASED ON BINARY, NON-BINARY AND DEEP CONVOLUTIONAL NETWORK DESCRIPTORS FOR VIDEO CONCEPT DETECTION

Foteini Markatopoulou[⋆†] *Vasileios Mezaris*[⋆] *Ioannis Patras*[†]

[⋆] Information Technologies Institute (ITI), CERTH, Thermi 57001, Greece
[†]Queen Mary University of London, Mile end Campus, UK, E14NS

ABSTRACT

In this paper we propose a cascade architecture that can be used to train and combine different visual descriptors (local binary, local non-binary and Deep Convolutional Neural Network-based) for video concept detection. The proposed architecture is computationally more efficient than typical state-of-the-art video concept detection systems, without affecting the detection accuracy. In addition, this work presents a detailed study on combining descriptors based on Deep Convolutional Neural Networks with other popular local descriptors, both within a cascade and when using different late-fusion schemes. We evaluate our methods on the extensive video dataset of the 2013 TRECVID Semantic Indexing Task.

Index Terms— Concept detection, Concept-based video retrieval, Cascade architecture, Video feature extraction

1. INTRODUCTION

Semantic concept detection in video aims to annotate video fragments with one or more semantic concepts (e.g. hand, sky, running), chosen from a predefined concept list [1]. This process facilitates many applications such as semantics-based video segmentation and retrieval, video event detection, video hyperlinking [1, 2, 3]. 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/images; 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 then trained separately for each concept, using the extracted features and ground truth annotations. The trained classifiers (concept detectors) can subsequently be applied to an unlabeled video shot, following feature extraction, and return a set of confidence scores for the appearence of the different concepts in the shot.

In this work we focus on the learning stage of video concept detection, by making two contributions. Firstly, we present a cascade architecture of classifiers that utilizes different visual descriptors and improves computational complexity in training and classification, without sacrificing the concept detection accuracy. Secondly, we show optimal ways to combine features based on Deep Convolutional Networks (Deep CNN) with other popular local descriptors, both within a cascade and in two other late-fusion architectures.

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

This work was supported by the EC under contracts FP7-600826 ForgetIT and FP7-287911 LinkedTV.

2. RELATED WORK

A variety of visual features has been proposed to facilitate concept detection [1] [4]. We can distinguish two main categories: handcrafted features and features based on Deep Convolutional Networks (Deep CNN). With respect to handcrafted features, binary (ORB [5]) and non-binary (SIFT [6], SURF [7]) local descriptors, as well as color extensions of them ([8], [4]) 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) [9] and VLAD [10]. With respect to features based on Deep CNN, one or more hidden layers of a Deep CNN are typically used as a global image representation [11]. Deep CNN-based descriptors present high discriminative power and generally outperform the local descriptors [12], [13]. There is a lot of recent research on the complementarity of different features, focusing mainly on local descriptors [4]. However, the combination of Deep CNN-based features with other state-of-the-art local descriptors has not been thoroughly examined [12].

The typical way of combining multiple features for concept detection is to separately train supervised classifiers for the same concept and each different feature. When all the classifiers give their decisions, a fusion step computes the final confidence score (e.g. in terms of averaging); this is known as late fusion. Hierarchical late fusion [14] 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. SURF) are sequentially fused with the previous groups. A problem when training supervised classifiers is the large-scale and imbalanced training sets, where for many concepts negative samples are significantly more than positives. The simplest way to deal with this is to randomly select a subset of the negative examples in order to have a reasonable ratio of positive-negative examples [15]. Random Under-Sampling is a different technique, which can somewhat balance the loss of negative samples of random selection [15] [2]. Different subsets of the negative examples are given as input to train different classifiers that are finally combined by late fusion. To achieve fast detection and reduce the computational requirements of the above process, linear supervised classifiers such as linear SVMs or Logistic Regression (LR) models, are typically preferred.

While late fusion is one reasonable solution, there are other ways that these classifiers can be trained and combined in order to accelerate the learning and detection process [16],[17]. Cascading is a learning and fusion technique that is useful in large-scale datasets and also accelerates training and classification. In a cascading archi-

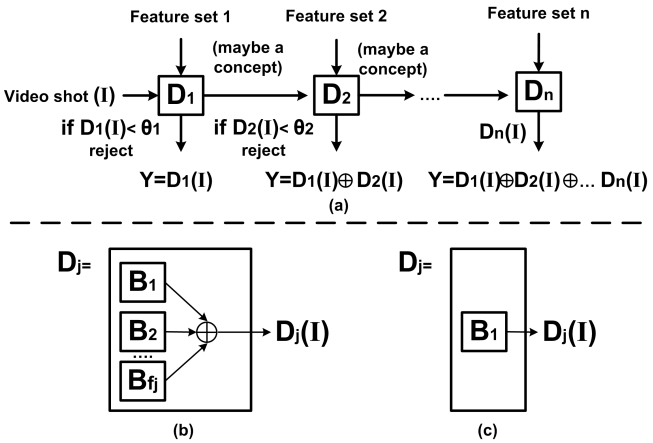

Fig. 1. (a) Block diagram of the proposed cascade architecture. (b) Stage combining many base classifiers trained on different features. (c) Stage with one base classifier trained on a single feature.

tecture, the classifiers are arranged in stages, from the less computationally demanding to the most demanding ones (or may be arranged according to other criteria e.g. their accuracy). A video shot 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 appears in the shot). The rationale behind this is to rapidly reject shots (i.e. keyframes) that clearly do not present a specific concept and focus on those shots that are more difficult and more likely to depict the concept. Cascades of classifiers have been mainly used in object detection tasks, where one of the most popular methods for face detection is the cascade of Viola & Jones [18]. Extensions of this have been proposed, focusing either on the design of each stage of the cascade ([19],[20], [21]) or the optimization of the whole cascade [22]. These architectures are trained on specific types of features (e.g. Haar-like features), which are suitable for object or face detection, and are not directly applicable to video concept detection using local or Deep CNN-based descriptors. Cascade architectures have been only briefly examined for video/image concept detection [23], [24].

3. CASCADE ARCHITECTURE FOR VIDEO CONCEPT DETECTION

3.1. Cascade architecture overview

Figure 1 shows the proposed cascade architecture that can be used for combining local and Deep CNN-based descriptors. Each stage j of the cascade encapsulates a stage classifier D_j. The stages of the cascade are sorted such that the classifier D_{j+1} is more accurate/expensive than D_j. Let I indicate an input image, the classifier D_{j+1} will be triggered only if the previous classifier does not reject the input image I. Each stage of the cascade is associated with a rejection threshold θ_j, and a stage output score $D_j(I)$ is associated with each stage classifier. A classifier is said to reject an input image if $D_j(I) < \theta_j$. A rejection indicates the classifier's belief that the concept does not appear in the image. Each stage classifier either combines many base classifiers trained on different types of features (Fig. 1:(b)) or contains only one base classifier trained on a single type of features (Fig. 1:(c)). In the first case, f_j base classifiers are still trained independently from each other (e.g. one classifier is trained on SIFT features, the other on RGB-SIFT features etc.), but their output is combined in order to return a single stage output score ($D_j(I) = B_1(I) \bigoplus ... \bigoplus B_{f_j}(I)$) in the [0,1] range that indicates the belief of the combined classifier that a concept appears into the shot. Symbol $\bigoplus$ indicates the function used to combine the output of base classifiers (e.g. arithmetic mean). In the second case, the stage output score is just the probability output of the single base classifier ($D_j(I) = B_1(I)$) in the [0,1] range. Any supervised classifier that solves a binary classification problem and outputs scores in the [0,1] range can be used as a base classifier.

Algorithm 1 presents the training and the classification process of the proposed cascade. During the training phase, the learning goal of the n-stage cascade is the construction of a set of n stage classifiers D_j and n thresholds θ_j, one for each stage ($j = 1 : n$). Stages are trained independently from each other, but every next stage focuses on a subset of the training images that pass from the previous stage (Alg. 1 Training: Step 5). This guarantees that fewer computations will be performed, because some images will be rejected by earlier stages and subsequently will not be used to train later stages. During the classification phase every next stage is triggered only if the stage output score is higher than the rejection threshold; similarly to training, this guarantees reduced number of classifier evaluations. The final score assigned to each image is the average of all stage output scores of all the previous stages (Alg. 1 Classification: Step 3). As a result, even images that did not reach stage n have a confidence score.

In order to be effective, the proposed architecture needs to allow as many as possible positive images to reach stage n. To achieve this, a rejection threshold should be selected for each stage that maximizes the true positive rate. Minimizing the false positive rate is not as important, because false positives that pass to next stages will be processed by more accurate stages, which will correct any mistakes that the earlier weaker stages have made. We automatically adjust the rejection threshold associated with each stage using a simple threshold selection strategy, which selects a threshold on the probability output of a classifier. More specifically, we apply each stage classifier on a validation set, in order to collect probability output scores, and we select the score that maximizes the required perfor-

Algorithm 1 Algorithm for the training and classification process of the proposed cascade

Training
Input: Training set T= $\{x_i, y_i\}_{i=1}^{M}$, $y_i \in \{\pm 1\}$; Number of stages: n; Sets of types of features to be used from each stage, each set consists of f_j different types of features for $j = 1 : n$
Output: A trained n-stage cascade that consists of $D_1, ...D_n$ stage classifiers, and $\theta_1, ...\theta_n$ stage thresholds.
for $j = 1$ to n **do**
1. Train f_j base classifiers B_j, one for each type of features assigned to stage j, using T
2. Save the stage classifier $D_j = B_1 \bigoplus ... \bigoplus B_{f_j}$.
3. Find the threshold θ_j, using the threshold selection algorithm that optimizes the recall of D_j
4. Find $T_1 \subset T$ of x_i for which $D_j(x_i) > \theta_j$
5. T=T_1
end for
Classification
Input: Unlabeled sample = x', n-stage trained cascade of $D_1, ..., D_n$ classifiers, and $\theta_1, ..., \theta_n$ stage thresholds
Output: Concept confidence score Y
$j = 1, Y = 0$
while $D_j(x') > \theta_j$ and $j \leq n$ **do**
1. $Y = Y + D_j(x')$
2. $j = j + 1$
end while
3. $Y = \frac{Y}{j-1}$

mance measure. Given that high true positive rate is crucial for the success of the proposed cascade, the employed thresholding strategy optimizes recall. Recall, which is a different name for the true positive rate, guarantees that the detected rejection threshold will be as small as possible in order allow all positive images to pass to the next stage.

3.2. Arrangement of features on the stages of the cascade

Arranging different features on cascade stages is a complex optimization problem that requires to select the best features, group them and sort them on stages. Finding the globally optimal arrangement requires retraining the full cascade from scratch (for every candidate arrangement), which is ineffective in large-scale datasets. In this work the arrangement of features on the cascade is fixed and chosen based on a previous study that evaluates individual local descriptors and also combinations of them [4]. Specifically, fast, more compact but less accurate descriptors are introduced on early stages (e.g. binary descriptors), while more complex and more accurate descriptors are used on later stages (e.g. non-binary, Deep CNN-based).

3.3. Offline cascade optimization

In the above, the thresholds of the proposed cascade are calculated separately for each stage. This can be suboptimal. In this section we use the offline method proposed by [25] for optical character recognition, to refine the rejection thresholds of each stage in order to improve the overall performance of the cascade. In contrast to [25] that refines the thresholds to improve the overall classification speed, we adjust their method in order to optimize the overall detection performance of the cascade. Following the dynamic programming algorithm we start with a two stage cascade, containing only the first and last stages of a pre-trained n-stage cascade. We apply this cascade on a validation set of M examples from which we collect M possible threshold values. Then the second stage D_2 is added between the first and last stage. The initial M thresholds can be extended into M possible ways through D_2. The full cascade is evaluated on all these M^2 extensions and the best M threshold paths are collected. This process of adding a stage is repeated $n-2$ times and the threshold path that leads to the best overall performance of the cascade is selected to refine the initial thresholds. In [25] the refined thresholds are selected by minimizing the overall classification cost, subject to a low error rate. To adjust this method for video concept detection we choose to maximize the Average Precision (AP). More specifically, the algorithm finds those thresholds that optimize the overall AP of the cascade, subject to the constraint that the overall recall (the measure that optimizes each node) equals to 1. This constraint guarantees that the refined thresholds will always lead to higher overall AP than the initial thresholds, without sacrificing the recall.

4. EXPERIMENTS

4.1. Datasets and experimental setup

Our experiments were performed on the TRECVID 2013 Semantic Indexing (SIN) dataset [26], which consists of a development set and a test set (approximately 800 and 200 hours of internet archive videos for training and testing, respectively). We evaluate our system on the test set using the 38 concepts that were evaluated as part of the TRECVID 2013 SIN Task, and we follow the TRECVID methodology for the evaluation of the results [26].

For experimenting with all methods, one keyframe was extracted for each video shot and was scaled to 320×240 pixels. Regarding feature extraction, we followed the experimental setup of [4]. More specifically, we extracted three binary descriptors (ORB, RGB-ORB and OpponentORB) and six non-binary descriptors (SIFT, RGB-SIFT and OpponentSIFT; SURF, RGB SURF and OpponentSURF). All the local descriptors, were compacted using PCA following [4] and were subsequently aggregated using the VLAD encoding. The VLAD vectors were reduced to 4000 dimensions and served as input to LR classifiers, used as base classifiers on the cascade or trained independently as described in the next paragraph. In all cases, the final step of concept detection was to refine the calculated detection scores by employing the re-ranking method proposed in [27]. In addition, we used features based on the last hidden layer of a pre-trained Deep CNN. Specifically, to extract these features we used the 16-layer pre-trained deep ConvNet network provided by [11]. The network has been trained on the ImageNet data only, and provides scores for 1000 concepts. We applied the network on the TRECVID keyframes and similar to other studies [12] we used as a feature the output of the last hidden layer (fc7), which resulted to a 4096-element vector. We refer to these features as DCNN in the sequel.

To train our algorithms, for each concept, 70% of the negative annotated training examples was randomly chosen for training and the rest 30% was chosen as a validation set for the offline cascade optimization method. Each of the two negative sets was merged with all positive annotated samples, by adding in every case three copies of each such positive sample (in order to account for their, most often, limited number). Then the positive and negative ratio of training examples was fixed on each of these sets by randomly rejecting any excess negative samples, to achieve a 1:6 ratio (which is important for building a balanced classifier). To train the proposed cascade architecture, the full training set was given as input to the first stage, while each later stage was trained with the subset of it that passed from its previous stages, as discussed in section 3. We compared the proposed cascade with a typical late-fusion scheme, where one LR classifier was trained for each type of features on the same full training set (that included three copies of each positive sample), denoted in the sequel as overtraining scheme. We also compared with another late-fusion scheme, where one LR classifier was trained on different subsets of the training set for each type of features. To construct different subsets of training sets we followed the Random Under-Sampling technique of Section 2; for each classifier, trained on a different type of features, a different subset of negatives was merged with all the positives (just one copy of each positive sample, in this case) and was given as input. The ratio of positive/negative samples was also fixed to 1:6. This scheme is denoted in the sequel as undersampling. For the offline cascade optimization 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 values by confidence and split at every M/Q example ($Q = 200$).

4.2. Results and discussion

Tables 1 and 2 present the results of our experiments in terms of Mean Extended Inferred Average Precision (MXinfAP) [28], which is an approximation of the Mean Average Precision suitable for the partial ground truth that accompanies the TRECVID dataset [26]. Table 1 presents the ten types of features that have been extracted and used by the algorithms of this study. For brevity, for SIFT, ORB and SURF we only show the MXinfAP when the original grayscale descriptor is combined with other two corresponding color variants

Table 2. Performance (MXinfAP. %) and relative computational complexity for different architectures/schemes: (a) cascade architecture (in parenthesis we show results for the offline cascade optimization which was presented in Section 3.3), (c) overtraining, (d) undersampling.

			Cascade (Offline cascade optimization)			Late fusion-overtraining		Late fusion-undersampling		
Run ID	Features	# of Base detectors/ Stages	MXinfAP (%)	amount of training data (%)	amount of classifier evaluations (%)	MXinfAP (%)	amount of training data (%)	MXinfAP (%)	amount of training data (%)	amount of classifier evaluations (%); same for both late fusion schemes
R1	ORBx3;DCNN	4 / 2	27.25 (27.31)	39.3	37.3 (38.4)	27.28	40.0	25.31	15.6	40.0
R2	SIFTx3;DCNN	4 / 2	27.42 (27.47)	38.8	36.8 (38.1)	27.41	40.0	25.89	15.6	40.0
R3	SURFx3;DCNN	4 / 2	26.9 (27.3)	39.3	37.6 (38.3)	27.01	40.0	25.98	15.6	40.0
R4	ORBx3;SIFTx3;DCNN	7 / 3	**27.82 (27.88)**	65.3	57.9 (61.3)	27.76	70.0	26.42	27.3	70.0
R5	ORBx3;SURFx3;DCNN	7 / 3	27.16 (27.66)	65.8	58.4 (61.3)	27.63	70.0	26.5	27.3	70.0
R6	SIFTx3;SURFx3;DCNN	7 / 3	27.69 (27.71)	64.5	56.7 (61.4)	27.7	70.0	26.8	27.3	70.0
R7	ORBx3;SIFTx3; SURFx3;DCNN	10 / 4	27.52 (27.52)	90.8	75.4 (83.0)	27.61	100.0	26.62	39.0	100.0
R8	ORBx3;DCNN	4 / 4	24.43 (24.53)	36.6	30.4 (33.7)	24.55	40.0	22.46	15.6	40.0
R9	SIFTx3;DCNN	4 / 4	24.42 (24.43)	35.6	29.1 (32.8)	24.5	40.0	23.05	15.6	40.0
R10	SURFx3;DCNN	4 / 4	24.49 (24.49)	36.9	31.7 (34.0)	24.46	40.0	23.66	15.6	40.0
R11	ORBx3;SIFTx3;DCNN	7 / 7	24.66 (24.79)	60.5	44.7 (51.5)	24.82	70.0	23.47	27.3	70.0
R12	ORBx3;SURFx3;DCNN	7 / 7	23.02 (24.77)	61.8	46.9 (54.0)	24.72	70.0	23.96	27.3	70.0
R13	SIFTx3;SURFx3;DCNN	7 / 7	23.53 (25.24)	60.0	44.1 (53.1)	25.16	70.0	24.32	27.3	70.0
R14	ORBx3;SIFTx3; SURFx3;DCNN	10 / 10	23.55 (25.06)	82.5	57.0 (67.3)	25.09	100.0	24.28	39.0	100.0

Table 1. Performance (MXinfAP. %) for base classifiers or combinations of them trained on different features.

Descriptor	MXinfAP	Base classifiers (ordered in terms of accuracy)
ORBx3	18.31	ORB, OpponentORB, RGB-ORB
SIFTx3	18.98	SIFT, OpponentSIFT, RGB-SIFT
SURFx3	19.34	SURF, OpponentSURF, RGB-SURF
DCNN	23.84	Last hidden layer of Deep CNN

by means of late fusion (averaging) (see [4] for fine-grained results).

Table 2 presents the performance and computational complexity of the proposed cascade architecture and the overtraining and undersampling schemes. The second column shows the features on which base classifiers have been trained for each run, and the number of stages (column three) indicates how the features have been grouped in stages. Runs R1 to R7 use stages that combine many base classifiers, in terms of late fusion (averaging), as presented in Fig. 1:(b). Specifically, stages that correspond to SIFT, SURF and ORB consist of three base classifiers (i.e. the grayscale descriptor and the two color variants), while the last stage of DCNN features contains only one base classifier. Runs R8 to R14 use stages made of a single base classifier (trained on a single type of features) as in Fig. 1:(c). We sort the stages on each cascade according to the accuracy of the employed individual base classifiers or combinations of them (according to Table 1) from the less accurate to the most accurate. Stages do not refer only to the cascade but also affect the way that late fusion has been performed by the overtraining and undersampling schemes. For example, for stages that consist of many features, the corresponding base classifiers per stage were firstly combined by averaging the classifier output scores and then the combined outputs of all stages were further fused together.

Table 2 shows that both cascade and late fusion-overtraining outperform the most commonly used approach of late fusion-undersampling, which uses less negative training examples to train each base classifier. The best results for cascade and overtaining are achieved by R4, reaching a MXinfAP of 27.82 and 27.76 respectively. The cascade reaches this good accuracy while at the same time is less computationally expensive than overtraining, both during training and during classification. Specifically, the cascade employed for R4 achieves 17.3% relative decrease in the number of classifier evaluations. 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 observed reduction in the amount of classifier evaluations significant. Table 2 also presents the results of the cascade when the offline cascade optimization technique of Section 3.3 for threshold refinement is employed. We observe that in many cases MXinfAP increases. The amount of necessary classifier evaluations also increases in this case, but even so the cascade is more computationally efficient than the two late fusion schemes.

The second fold of this work is to examine how we can effectively combine Deep CNN-based features with handcrafted local descriptors. According to Table 1, the DCNN performs better than the combinations of SIFT, SURF and ORB with their color variants. It should be noted that each of the base classifiers of these groups (e.g. RGB-SIFT) is rather weak, achieving MXinfAP that ranges from 11.68 to 15.04 (depending on which descriptor is used). Table 2 shows that the way that the stages of a cascade are constructed but also the way that late fusion is performed on the overtraining and undersampling schemes affects the combination of DCNN with the other descriptors. Generally, it is better to merge weaker base classifiers to a more robust one (e.g. grouping grayscale SIFT with its color variants) in order to either use them as a cascade stage or to combine their decisions in terms of late fusion (R1-R7: MXinfAP ranges from 22.6 to 25.09), than treating each of them independently from the others (i.e., using one weak classifier per stage or fusing all of them with equal weight; R8-R14: MXinfAP ranges from 25.31 to 27.82). The best way to combine DCNN with other local descriptors is R4, where ORBx3, SIFTx3 and DCNN are arranged in a cascade, increasing the MXinfAP from 23.84 (for DCNN alone) to 27.82.

5. CONCLUSIONS

In this work we presented a cascade architecture for video concept detection that improved the computational complexity compared to typical state-of-the-art late fusion architectures, without affecting the concept detection accuracy. Furthermore, we presented effective ways to fuse Deep CNN-based features with other popular local descriptors, both within a cascade and in late fusion schemes.

6. REFERENCES

[1] C. G. M. Snoek and M. Worring, "Concept-Based Video Retrieval," *Foundations and Trends in Information Retrieval*, vol. 2, no. 4, pp. 215–322, 2009.

[2] F. Markatopoulou et al., "ITI-CERTH participation to TRECVID 2013," in *TRECVID 2013 Workshop*, Gaithersburg, MD, USA, 2013.

[3] V. Mezaris, P. Sidiropoulos, and I Kompatsiaris, "Improving interactive video retrieval by exploiting automatically-extracted video structural semantics," in *IEEE Int. Conf. on Semantic Computing (ICSC)*, 2011, pp. 224–227.

[4] F. Markatopoulou, N. Pittaras, O. Papadopoulou, V. Mezaris, and I. Patras, "A study on the use of a binary local descriptor and color extensions of local descriptors for video concept detection," in *2015 MultiMedia Modeling Conference*, vol. 8935 of *Lecture Notes in Computer Science*, pp. 282–293. Springer, 2015.

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

[6] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[7] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *ECCV*, vol. 3951 of *LNCS*, pp. 404–417. Springer, 2006.

[8] K. E. A. Van de Sande, T. Gevers, and C. G. M. Snoek, "Evaluating color descriptors for object and scene recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1582–1596, 2010.

[9] G. Csurka and F. Perronnin, "Fisher vectors: Beyond bag-of-visual-words image representations," in *Computer Vision, Imaging and Computer Graphics. Theory and Applications*, vol. 229 of *Communications in Computer and Information Science*, pp. 28–42. Springer Berlin, 2011.

[10] H. Jegou, M. Douze, C. Schmid, and P. Perez, "Aggregating local descriptors into a compact image representation," in *IEEE Int. Conf. on CVRP 2010*, SF, CA, 2010, pp. 3304–3311.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv technical report*, 2014.

[12] B. Safadi, N. Derbas, A. Hamadi, M. Budnik, P. Mulhem, and G. Qu, "LIG at TRECVid 2014 : Semantic Indexing," in *TRECVID 2014 Workshop*, Gaithersburg, MD, USA, 2014.

[13] C. G. M. Snoek, K. E. A. Van De Sande, D. Fontijne, S. Cappallo, J. Van Gemert, and A. Habibian, "MediaMill at TRECVID 2014 : Searching Concepts, Objects, Instances and Events in Video," in *TRECVID 2014 Workshop*, Gaithersburg, MD, USA, 2014.

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

[15] B. Safadi and G. Quénot, "Evaluations of multi-learner approaches for concept indexing in video documents," in *RIAO'10*, 2010, pp. 88–91.

[16] H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik, "Parallel support vector machines: The cascade SVM," in *Neural Information Processing Systems*, 2004, pp. 13–18.

[17] L. Bao et al., "Informedia@TRECVID 2011," in *TRECVID 2011 Workshop*, Gaithersburg, MD, USA, 2011.

[18] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *2001 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2001, vol. 1, pp. 511–518.

[19] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR'05)*, USA, 2005, vol. 2 of *CVPR '05*, pp. 236–243, IEEE Computer Society.

[20] C. Shen, P. Wang, and H. Li, "Lacboost and fisherboost: Optimally building cascade classifiers," in *European Conference on Computer Vision (ECCV) 2010*, vol. 6312 of *Lecture Notes in Computer Science*, pp. 608–621. Springer, 2010.

[21] L. Jianguo and Z. Yimin, "Learning surf cascade for fast and accurate object detection," in *2013 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3468–3475.

[22] L. Lefakis and F. Fleuret, "Joint cascade optimization using a product of boosted classifiers," in *NIPS'10*, 2010, pp. 1315–1323.

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

[24] H. Harzallah, F. Jurie, and C. Schmid, "Combining efficient object localization and image classification," in *IEEE 12th Int. Conf. on Computer Vision*, 2009, pp. 237–244.

[25] K. Chellapilla, M. Shilman, and P. Simard, "Combining multiple classifiers for faster optical character recognition," in *7th International Conference on Document Analysis Systems*, Berlin, 2006, DAS'06, pp. 358–367, Springer.

[26] P. Over et al., "Trecvid 2013 – an overview of the goals, tasks, data, evaluation mechanisms and metrics," in *Proceedings of TRECVID 2013*. NIST, USA, 2013.

[27] B. Safadi and G. Quénot, "Re-ranking by local re-scoring for video indexing and retrieval," in *20th ACM Int. Conf. on Information and Knowledge Management*, NY, 2011, pp. 2081–2084, ACM.

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