Machine Learning Architectures for Video Annotation and Retrieval

Foteini Markatopoulou

Submitted in partial fulfillment of the requirements of the Degree of Doctor of Philosophy

Co-supervisors: Dr. Ioannis Patras & Dr. Vasileios Mezaris

School of Electronic Engineering and Computer Science

Queen Mary University of London

United Kingdom

April 2018

Statement of originality

I, Foteini Markatopoulou, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature: Foteini Markatopoulou

Date: April 20, 2018

Details of collaboration and publications:

Publications related to the thesis:

- F. Markatopoulou, V. Mezaris, I. Patras, "Implicit and Explicit Concept Relations in Deep Neural Networks for Multi-Label Video/Image Annotation", IEEE Transactions on Circuits and Systems for Video Technology, DOI:10.1109/ TCSVT.2018.2848458, 2018.

- F. Markatopoulou, D. Galanopoulos, V. Mezaris, I. Patras, "Query and Keyframe Representations for Ad-hoc Video Search", Proc. ACM ICMR 2017, Bucharest, Romania, DOI: 10.1145/3078971.3079041, June 2017.

- F. Markatopoulou, V. Mezaris, I. Patras, "Deep Multi-task Learning with Label Correlation Constraint for Video Concept Detection", Proc. ACM Multimedia 2016, Amsterdam, The Netherlands, DOI: 10.1145/2964284.2967271, Oct. 2016.

- F. Markatopoulou, V. Mezaris, I. Patras, "Online Multi-Task Learning for Semantic Concept Detection in Video", Proc. IEEE Int. Conf. on Image Processing (ICIP 2016), Phoenix, AZ, USA, Sept. 2016.

- F. Markatopoulou, V. Mezaris, I. Patras, "Ordering of Visual Descriptors in a Classifier Cascade Towards Improved Video Concept Detection", Proc. 22nd Int. Conf. on MultiMedia Modeling (MMM'16), Miami, FL, USA, Springer LNCS vol. 9516, pp. 874-885, DOI:10.1007/978-3-319-27671-7_73, Jan. 2016.

- F. Markatopoulou, V. Mezaris, I. Patras, "Cascade of classifiers based on Binary, Non-binary and Deep Convolutional Network descriptors for video concept detection", Proc. IEEE Int. Conf. on Image Processing (ICIP 2015), Quebec City, Canada, DOI: 10.1109/ICIP.2015.7351108, Sept. 2015.

- F. Markatopoulou, V. Mezaris, N. Pittaras, I. Patras, "Local Features and a Two-Layer Stacking Architecture for Semantic Concept Detection in Video",

IEEE Trans. on Emerging Topics in Computing, vol. 3, no. 2, pp. 193-204, DOI: 10.1109/TETC.2015.2418714, June 2015.

- F. Markatopoulou, N. Pittaras, O. Papadopoulou, V. Mezaris, I. Patras, "A Study on the Use of a Binary Local Descriptor and Color Extensions of Local Descriptors for Video Concept Detection", Proc. 21st Int. Conf. on MultiMedia Modeling (MMM'15), Sydney, Australia, Springer LNCS vol. 8935, pp. 282-293, DOI: 10.1007/978-3-319-14445-0_25, Jan. 2015.

Benchmarking publications:

- A. Moumtzidou, S. Andreadis, F. Markatopoulou, D. Galanopoulos, I. Gialampoukidis, S. Vrochidis, V. Mezaris, I. Kompatsiaris, I. Patras, "VERGE in VBS 2018", Proc. 24th Int. Conf. on Multimedia Modeling (MMM2018), Bangkok, Thailand, Feb. 2018.

- F. Markatopoulou, A. Moumtzidou, D. Galanopoulos, K. Avgerinakis, S. Andreadis, I. Gialampoukidis, S. Tachos, S. Vrochidis, V. Mezaris, I. Kompatsiaris, I. Patras, "ITI-CERTH participation in TRECVID 2017", Proc. TRECVID 2017 Workshop, Gaithersburg, MD, USA, Nov. 2017.

- A. Moumtzidou, T. Mironidis, F. Markatopoulou, S. Andreadis, I. Gialampoukidis, D. Galanopoulos, A. Ioannidou, S. Vrochidis, V. Mezaris, I. Kompatsiaris, I. Patras, "VERGE in VBS 2017", Proc. 23rd Int. Conf. on MultiMedia Modeling (MMM'17), Reykjavik, Iceland, Springer LNCS vol. 10133, pp. 486-492, Jan. 2017.

- F. Markatopoulou, A. Moumtzidou, D. Galanopoulos, T. Mironidis, V. Kaltsa, A. Ioannidou, S. Symeonidis, K. Avgerinakis, S. Andreadis, I. Gialampoukidis, S. Vrochidis, A. Briassouli, V. Mezaris, I. Kompatsiaris, I. Patras, "ITI-CERTH participation in TRECVID 2016", Proc. TRECVID 2016 Workshop, Gaithersburg, MD, USA, Nov. 2016.

- F. Markatopoulou, A. Ioannidou, C. Tzelepis, T. Mironidis, D. Galanopoulos, S. Arestis-Chartampilas, N. Pittaras, K. Avgerinakis, N. Gkalelis, A. Moumtzidou, S. Vrochidis, V. Mezaris, I. Kompatsiaris, I. Patras, "ITI-CERTH participation to TRECVID 2015", Proc. TRECVID 2015 Workshop, Gaithersburg, MD, USA, Nov. 2015.

- A. Moumtzidou, K. Avgerinakis, E. Apostolidis, F. Markatopoulou, K. Apostolidis, T. Mironidis, S. Vrochidis, V. Mezaris, Y. Kompatsiaris, I. Patras, "VERGE: A Multimodal Interactive Video Search Engine", Proc. 21st Int. Conf. on Multi-Media Modeling (MMM'15), Sydney, Australia, Springer LNCS vol. 8936, pp. 249-254, DOI: 10.1007/978-3-319-14442-9_23, Jan. 2015.

- N. Gkalelis, F. Markatopoulou, A. Moumtzidou, D. Galanopoulos, K. Avgerinakis, N. Pittaras, S. Vrochidis, V. Mezaris, I. Kompatsiaris, I. Patras, "ITI-CERTH participation to TRECVID 2014", Proc. TRECVID 2014 Workshop, Orlando, FL, USA, November 2014.

- A. Moumtzidou, K. Avgerinakis, E. Apostolidis, V. Aleksic, F. Markatopoulou, C. Papagiannopoulou, S. Vrochidis, V. Mezaris, R. Busch, I. Kompatsiaris, "VERGE: An Interactive Search Engine for Browsing Video Collections", Proc. Video Browser Showdown (VBS'14) at the 20th Int. Conf. on MultiMedia Modeling (MMM'14), Dublin, Ireland, Springer LNCS vol. 8326, pp. 411-414, DOI: 10.1007/978-3-319-04117-9_48, January 2014.

- F. Markatopoulou, A. Moumtzidou, C. Tzelepis, K. Avgerinakis, N. Gkalelis, S. Vrochidis, V. Mezaris, I. Kompatsiaris, "ITI-CERTH participation to TRECVID 2013", Proc. TRECVID 2013 Workshop, Gaithersburg, MD, USA, November 2013.

Other publications:

- K. Apostolidis, F. Markatopoulou, C. Tzelepis, V. Mezaris, I. Patras, "Multimedia Processing Essentials", in book "Personal Multimedia Preservation: Remembering or Forgetting Images and Video", V. Mezaris, C. Niederee, R.H. Logie (Eds.), Springer, ISBN 978-3-319-73465-1, pp. 47-98, 2018. DOI:10.1007/978-3-319-73465-1_3.

- D. Galanopoulos, F. Markatopoulou, V. Mezaris, I. Patras, "Concept Language Models and Event-based Concept Number Selection for Zero-example Event Detection", Proc. ACM ICMR 2017, Bucharest, Romania, DOI: 10.1145/3078971.3079043, June 2017. (Best Poster Award)

- C. Collyda, E. Apostolidis, A. Pournaras, F. Markatopoulou, V. Mezaris, I. Patras, "VideoAnalysis4ALL: An on-line tool for the automatic fragmentation and concept-based annotation, and the interactive exploration of videos", Proc. ACM ICMR 2017, Bucharest, Romania, DOI: 10.1145/3078971.3079015, June 2017

- N. Pittaras, F. Markatopoulou, V. Mezaris, I. Patras, "Comparison of Fine-tuning and Extension Strategies for Deep Convolutional Neural Networks", Proc. 23rd Int. Conf. on MultiMedia Modeling (MMM'17), Reykjavik, Iceland, Springer LNCS vol. 10132, pp. 102-114, Jan. 2017.

- G. Kalpakis, T. Tsikrika, F. Markatopoulou, N. Pittaras, S. Vrochidis, V. Mezaris, I. Patras, I. Kompatsiaris, "Concept Detection on Multimedia Web Resources about Home Made Explosives", Proc. Int. Workshop on Multimedia Forensics and Security (MFSec), held in conjunction with the 10th Int. Conf. on Availability, Reliability and Security (ARES), Toulouse, France, DOI: 10.1109/ARES.2015.85, Aug. 2015.

- F. Markatopoulou, V. Mezaris, I. Kompatsiaris, "A Comparative Study on the Use of Multi-Label Classification Techniques for Concept-Based Video Indexing and Annotation", Proc. 20th Int. Conf. on MultiMedia Modeling (MMM'14), Dublin, Ireland, Springer LNCS vol. 8325, pp. 1-12, DOI: 10.1007/978-3-319-04114-8_1, January 2014. (Best Paper Award)

Abstract

In this thesis we are designing machine learning methodologies for solving the problem of video annotation and retrieval using either pre-defined semantic concepts or ad-hoc queries. Concept-based video annotation refers to the annotation of video fragments with one or more semantic concepts (e.g. hand, sky, running), chosen from a pre-defined concept list. Ad-hoc queries refer to textual descriptions that may contain objects, activities, locations etc., and combinations of the former. Our contributions are: i) A thorough analysis on extending and using different local descriptors towards improved concept-based video annotation and a stacking architecture that uses in the first layer, concept classifiers trained on local descriptors and improves their prediction accuracy by implicitly capturing concept relations, in the last layer of the stack. ii) A cascade architecture that orders and combines many classifiers, trained on different visual descriptors, for the same concept. iii) A deep learning architecture that exploits concept relations at two different levels. At the first level, we build on ideas from multi-task learning, and propose an approach to learn concept-specific representations that are sparse, linear combinations of representations of latent concepts. At a second level, we build on ideas from structured output learning, and propose the introduction, at training time, of a new cost term that explicitly models the correlations between the concepts. By doing so, we explicitly model the structure in the output space (i.e., the concept labels). iv) A fully-automatic ad-hoc video search architecture that combines concept-based video annotation and textual query analysis, and transforms concept-based keyframe and query representations into a common semantic embedding space. Our architectures have been extensively evaluated on the TRECVID SIN 2013, the TRECVID AVS 2016, and other large-scale datasets presenting their effectiveness compared to other similar approaches.

Acknowledgments

First of all, I would like to thank my supervisors Dr. Ioannis Patras and Dr. Vasileios Mezaris for the continuous support, encouragement, advice and motivation during my studies, but also for the great opportunity they gave me to actually start my postgraduate studies. Besides my main supervisors, I would like to thank the rest of the members of my supervisory team, Dr. Timothy Hospedales and Dr. Shaogang Gong, for their insightful guidance. Secondly, I would like to thank my fellow workers Damianos, Nikiforos and Christos for their help in parts of the technical and scientific work of this thesis. Thirdly, I would like to thank my family for supporting me all these years. And finally, a special thank goes to Marinos; this PhD would have never started if it was not his motivation.

Contents

List of Figures

List of Tables

List of abbreviations

AVS	Ad-hoc Video Search
BR	Binary Relevance
CCE-LC	Cost Sigmoid Cross-entropy with Label Constraint
DCNN	Deep Convolutional Neural Network
DL	Deep Learning
FV	Fisher Vector
FV-MTL	Shared Latent Feature Vectors using Multi-task Learning
MDL	Multi Domain Learning
MLC	Multi Label Classification
MTL	Multi-task Learning
MXinfAP	Mean Extended Inferred Average Precision
NLP	Natural Language Processing
PCA	Principal Components Analysis
SGD	Stochastic Gradient Descent
SIN	Semantic Indexing
STL	Single Task Learning
SVM	Support Vector Machine
VLAD	Vector of Locally Aggregated Descriptors

Chapter 1

Introduction

Contents

1.1 Video annotation and retrieval

1.1.1 Problem definition

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. Annotating videos with concepts is a very important task that facilitates many applications such as semantics-based video segmentation and retrieval, video event detection, video hyperlinking, concept-based video search and ad-hoc video search [98, 72, 73, 74, 42]. Concept-based video search refers to the retrieval of video fragments (e.g., keyframes) that present specific simple concept labels from large-scale video collections. Ad-hoc video search is another related problem. Ad-hoc queries refer to textual descriptions that aim to model the end user's need of retrieving video fragments containing persons, objects, activities,

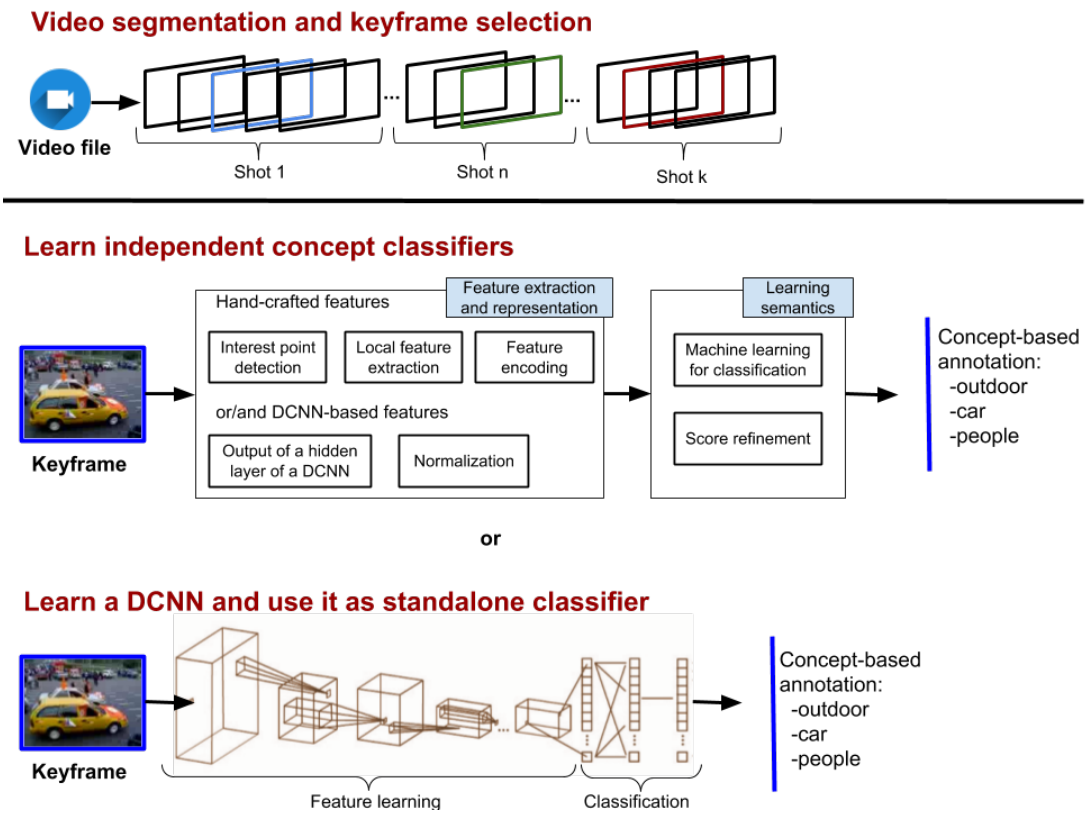

Figure 1.1: Video concept annotation pipelines: After temporal video segmentation, e.g., using automatic video shot detection and extracting one representative keyframe from the video shot, the upper part shows a typical concept-based video annotation pipeline that is based on hand-crafted or DCNN-based features and supervised classifiers trained separately for each concept. The lower part is based on features that can be learned directly from the raw keyframe pixels using a DCNN, and subsequently using the DCNN as standalone classifier to perform the final class label prediction.

locations etc., and combinations of the former (e.g., "Find shots of a person playing guitar outdoors").

To deal with concept-based video search, concept-based video annotation methods have been developed that automatically annotate video-fragments, e.g., keyframes extracted from video shots, with semantic labels (concepts), chosen from a pre-defined concept list [98]. A typical concept-based video annotation system mainly follows the process presented in Fig. 1.1. 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 hand-crafted visual, DCNN-based (DCNN stands for Deep Convolutional Neural Network), textual or audio features 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 used to train classifiers (concept classifiers) independently for each concept, using the extracted features and ground-truth annotations. The trained classifiers can subsequently be applied to an unlabeled video shot, following feature extraction, and return a set of confidence scores for the appearance of the different concepts in the shot. A recent trend in video annotation is to learn features directly from the raw keyframe pixels using deep convolutional neural networks (DCNNs). DCNNs consist of many layers of feature extractors, and are thus able to model more complex structures in comparison to handcrafted representations. DCNN layers can learn different type of features without requiring feature engineering compared to the hand-crafted features that are designed and developed such as capturing specific properties of video frames, e.g., edges and corners. DCNNs can be used both as standalone classifiers (Fig. 1.1, bottom), i.e., unlabelled keyframes are passed through a pre-trained DCNN that performs the final class label prediction directly, using typically a softmax or a hinge loss layer [94, 57], and also as generators of video keyframe features (Fig. 1.1, top), i.e., the output of a hidden layer of the pre-trained DCNN is used as a global keyframe representation [94], this latter type of features is referred as DCNN-based.

The challenge of ad-hoc video search (Fig. 1.2) is typically solved by analysing both the video fragments and the ad-hoc query to a set of pre-defined concept labels. Specifically, each video-fragment is annotated with concepts using a pre-trained concept-based video annotation system, then for each ad-hoc query the nearest concepts are retrieved from the same pool of concepts that was used to annotate the video fragments. Natural language processing (NLP) and keyword extraction is typically used in order to transform the textual query to a concept-based representation. The video fragments with the smaller distance from the target query are retrieved as the most

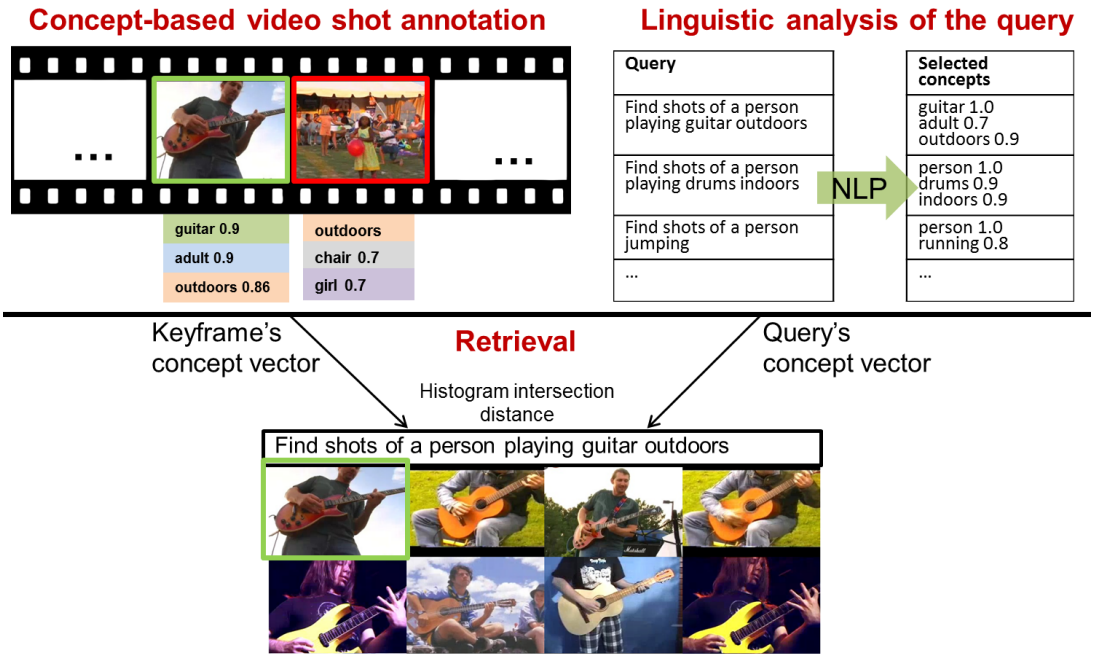

Figure 1.2: Typical ad-hoc video search pipeline. Each video fragment is annotated with concepts taken from a predefined concept pool. The text query is analysed for concepts taken from the same concept pool. The distance between the concept-based video fragment representations and the concept-based query representation is calculated and the video fragments close in terms of the distance metric are retrieved.

related by calculating the distance between the concept-based query representation and each concept-based video fragment representation.

1.1.2 Challenges and assumptions

While significant progress has been made during the last years in the task of video annotation and retrieval, 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. The system needs to learn hundreds or thousands of concepts that belong to different categories (e.g. landscapes, faces, actions). As a result, generality is an important property that a concept-based video annotation system should present in order to generalize its performance across many different heterogeneous concepts. Finally, computational

requirements are another major challenge. The large number of concepts that a video annotation system should learn is computationally expensive requiring lightweight and fast methods.

It has been shown that combining many different features for the same concept, instead of using a single feature, improves concept annotation accuracy. The subset of features that will be used and the way the classifiers trained on these features will be combined is a challenging problem that will affect the accuracy and computational complexity of the complete concept-based video annotation system. Other methods also improve the overall video annotation accuracy by looking for existing semantic relations e.g., concept correlations.

As discussed in the previous subsection the dominant approach for performing concept-based video annotation is to train DCNNs whereby concepts share features within the architectures up to the very last layer, and then branch off to T different classification branches (using typically one layer), where T is the number of concepts [83]. However, in this way, the implicit feature-level relations between concepts, e.g. the way in which concepts such as a *car* and *motorcycle* share lower-level features modeling things like their wheels, are not directly considered. Also, in such architectures, the relations or interdependencies of the concepts at a semantic level, i.e. the fact that two specific concepts may often appear together or, inversely, the presence of the one may exclude the other, are also not directly taken into consideration. While some methods have been proposed for exploiting in a more elaborate way one of these two different types of concept relations, there is no single method that jointly exploits visual- and semantic-level concept relations in a unified DCNN architecture.

Ad-hoc video search is a very recent research topic. Existing methods have been mainly focused on retrieving images for a single unknown concept label [79], [38], [99], which is a simpler problem compared to the one that we investigate, i.e., retrieving video shots given a complex textual query. Extending and improving such methods

for ad-hoc video search is a very challenging and timely topic that to date has not been thoroughly investigated.

1.2 Aims and objectives

In this thesis we are using machine learning methodologies for solving the problem of video annotation and retrieval using either pre-defined semantic concepts or ad-hoc queries. While the task of video annotation and retrieval has presented a lot of progress during the last years, it remains a timely topic with open research questions and practical application. In addition, the accuracy of state-of-the-art video annotation systems cannot be considered as satisfactory, which shows the difficulty of this problem and the need of developing novel approaches in this field. As briefly discussed in the previous subsections, one way to achieve improved concept-based video annotation accuracy is to train the concept classifiers in a rich pool of visual features (either engineered or learned). The pool of these features and the training of concept classifiers on them, the way that concept classifiers trained on different features can be combined, the way that feature sharing across the concept classifiers can be achieved and the way that concept relations can be modeled are the major topics of investigation within this thesis.

Our first aim is to design a set of effective and accurate concept-based video annotation architectures for solving the problem of concept-based video fragment retrieval. With respect to this first direction, we collect a big pool of visual features. Specifically, we examine how local binary descriptors can facilitate concept-based video annotation, we propose color extensions of them inspired by previously proposed color extensions of SIFT, and we show that the latter color extension paradigm is generally applicable to both binary and non-binary local descriptors. In order to use them in conjunction with a state-of-the-art feature encoding, we compact the above color extensions using PCA and we compare two alternatives for doing this. Then, we present an improved

way of ordering and combining independently trained concept detectors using a cascade. The proposed cascade combines hand-crafted (e.g. SIFT [62]) and DCNN-based 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. Finally, following the recent advances in this field that solve the problem using deep learning techniques, we propose a deep learning method that jointly exploits visual- and semantic-level concept relations in a unified DCNN architecture.

Our second aim is to develop a fully-automatic AVS method that uses solely a natural-language textual query to retrieve related video shots from a video collection. With respect to this direction we investigate two different ways of representing video keyframes and textual queries in order to transfer them into a common representation space. Our pre-trained concept-based video annotation system is used to annotate the video fragments with concept labels and a query analysis strategy has been developed to translate the textual query into the same pool of concepts. More advanced methods that use semantic embeddings of words and sentences, e.g., word2vec neural network models, have also been investigated in order to achieve a second approach to query/keyframe representations.

1.3 Thesis contributions

Our contributions are:

- A thorough analysis on extending and using different local descriptors towards improved concept-based video annotation and a stacking architecture that uses in the first layer, concept classifiers trained on the above local descriptors and improves their prediction accuracy by implicitly capturing concept relations, in the last layer of the stack.

- A cascade architecture that orders and combines many classifiers, trained on different visual descriptors, for the same concept. This method is computationally more efficient, in terms of classifier evaluations, and more accurate than other state-of-the-art approaches.

- A DCNN architecture that addresses the problem of video/image concept annotation by exploiting concept relations at two different levels. At the first level, we build on ideas from multi-task learning, and propose an approach to learn concept-specific representations that are sparse, linear combinations of representations of latent concepts. By enforcing the sharing of the latent concept representations, we exploit the implicit relations between the target concepts. At a second level, we build on ideas from structured output learning, and propose the introduction, at training time, of a new cost term that explicitly models the correlations between the concepts. By doing so, we explicitly model the structure in the output space (i.e., the concept labels). Both of the above are implemented using standard convolutional layers and are incorporated in a single DCNN architecture that can then be trained end-to-end with standard back-propagation.

- A fully-automatic ad-hoc video search architecture that combines concept-based video annotation and textual query analysis. We propose a new method for transforming concept-based keyframe and query representations into a common semantic embedding space, and we show that our proposed combination of concept-based representations with their corresponding semantic embeddings results in improved video search accuracy.

Our architectures have been extensively evaluated on the TRECVID SIN 2013, the TRECVID AVS 2016, and other large-scale datasets where we present their effectiveness compared to other similar approaches.

1.4 Structure of the thesis

In Chapter 2 we review the related literature in the field of video annotation and retrieval using either pre-defined semantic concepts or ad-hoc queries. Specifically, we provide an outline of the major trends in the field. We cover the bibliography related to the feature extraction process, classifier learning and classifier fusion techniques, subsequently we present methods that use multi-task learning and exploit semantic concept relations. Then existing zero-shot learning methods are reviewed. Finally, we summarize the TRECVID Semantic Indexing and Ad-hoc Video Search benchmarking activities and datasets, and other related large-scale datasets some of which have been used in this thesis.

In Chapter 3 we present how we extend existing feature extraction approaches in order to create a large pool of discriminative visual features. Then we show how these features can be used to train concept classifiers and how the predictions of these independently trained concept classifiers can be improved by exploiting concept relations in the last layer of a stacking architecture.

Chapter 4 describes the cascade architecture developed to combine many pre-trained concept detectors on different visual features for the same concept.

Chapter 5 presents the proposed deep learning method that jointly exploits visual- and semantic-level concept relations in a unified DCNN architecture for concept-based video annotation.

Chapter 6 presents a fully-automatic ad-hoc video search architecture that combines concept-based video annotation and textual query analysis.

The thesis concludes with Chapter 7 where we summarize the findings of our experimental studies, and present our general conclusions on using each of the proposed machine learning architectures for video annotation and retrieval. Finally, we identify

some areas where there is still space for further research and outline the avenues of our future work.

Chapter 2

Related Work

Contents

In this chapter we review some of the most representative works in the literature of concept-based video annotation and retrieval and ad-hoc video search. Our aim is to cover state-of-the-art studies in this field, highlight their limitations and provide a comprehensive view of the research areas for the topics addressed in this thesis. The chapter has been divided into five sections aiming to highlight the weaknesses of existing approaches and to show how the proposed approaches in this thesis can go beyond the state-of-the-art. Section 2.1 covers the bibliography related to feature extraction and representation. Section 2.2 presents related work on classifier learning, transfer learning and classifier combination techniques. Then, Section 2.3, presents related work, focusing on MTL and structured output prediction, while Section 2.4 reviews

zero-shot learning techniques that are used for ad-hoc video search. Finally, Section 2.5 summarizes the TRECVID Semantic Indexing (SIN) and Ad-hoc Video Search (AVS) benchmarking activities and the provided large-scale datasets for concept-based video annotation and ad-hoc video search, respectively, that have been used by this study, and also other related benchmarking datasets.

2.1 Feature extraction and 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 [98]. In large-scale concept-based video annotation, typically visual features are utilized, being extracted from representative keyframes or similar 2D image structures [93]. We can distinguish two main categories of visual features: hand-crafted features and features based on Deep Convolutional Networks (DCNN-based). With respect to hand-crafted features, binary (ORB [88]) and non-binary (SIFT [62], SURF [9]) local descriptors, as well as color extensions of them [113] have been examined for concept-based video annotation. Local descriptors are aggregated into global image representations by employing feature encoding techniques such as Fisher Vector (FV) [22] and VLAD [48]. With respect to DCNN-based features, one or more hidden layers of a pre-trained DCNN are typically used as a global keyframe representation [94]. Several DCNN software libraries are available in the literature, e.g., Caffe [50], MatConvNet, TensorFlow [32] and different DCNN architectures have been proposed, e.g., ResNeXt [125], ResNet [46], GoogLeNet [106], VGG ConvNet [94], CaffeNet [57]. DCNN-based descriptors present high discriminative power and generally outperform local descriptors [90], [97].

Two of the most popular local descriptors are SIFT [62] and SURF [9]. Both of them extract features that are invariant to rotation, scale and illumination variations, while SURF extraction is somewhat less computationally-demanding (SURF is two times

faster than SIFT according to [9]). SIFT and SURF construct vectors of floating-point values (which are often quantized to integers in the range [0,255]). For many modern applications, though, e.g. video annotation on mobile devices, small-sized yet discriminative descriptors are very important in order to extract, store and transmit them efficiently (e.g. send local descriptors to a server for performing concept-based video annotation). Binary local descriptors are an attractive alternative to non-binary descriptors such as SIFT and SURF, generating binary strings which can be computed efficiently while also requiring lower storage space. ORB [88], BRISK [59], and FREAK [2] are some examples of binary local descriptors that have been proposed for similarity matching between local image patches. They are all based on calculating the differences between pairs of pixel intensity values within an image patch; what distinguishes them is the pattern they follow in order to perform these pair-wise pixel comparisons. Studies show that ORB [88] and BRISK [59] are among the most accurate binary descriptors for image matching [15]. The possibility of using ORB in image classification was also briefly examined in [43].

The above mentioned non-binary and binary local descriptors are intensity-based: they are applied to grayscale images (e.g. an RGB image is firstly converted to grayscale), and the extracted features are calculated from the pixel intensity values. Two color variants of SIFT, namely RGB-SIFT and OpponentSIFT, that increase the descriptor's discriminative power were proposed in [113]. Methods that consider the color information in order to improve the SURF descriptor have also been proposed. Most of them were examined only on the image matching problem [39], [36], [20], while others, such as OpponentSURF and similar extensions of other descriptors, have also been used for concept-based annotation [101], [103]. In [43], the extraction of ORB from all three color channels of the RGB color space was considered.

For the purpose of visual concept annotation, local descriptors extracted from different patches of one image are subsequently aggregated into a global image representa-

tion, a process known as feature encoding. The most popular encoding in the last years has been the Bag-of-Words (BoW) [85]. Fisher vector (FV) [22] and VLAD (Vector of Locally Aggregated Descriptors) [48] are two state-of-the-art encodings that significantly outperform the BoW [114] [16]. FV encoding describes the difference between the distribution of features for an image and the distribution fitted to the features of all the training data. VLAD [48] is a fast approximation of FV that performs somewhat worse but is more compact and faster to compute [49], which makes it a good compromise. The two latter encodings are high-dimensional and their dimensionality is affected by the dimensionality of the local descriptors they encode, thus dimensionality reduction approaches such as PCA [123] are widely used for making the image representation more compact prior to learning/classification. Dimensionality reduction can be performed at two stages: local descriptors can be reduced prior to the encoding, and then the final encoding can also be further compacted [49].

2.2 Supervised learning, transfer learning and classifier combination techniques

Concept-based video annotation is a multi-label classification (MLC) 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. The most commonly used machine learning algorithms are Support Vector Machines (SVM) and Logistic Regression (LR). Chi-square kernels, that were originally considered to

be optimal for use in SVMs [134], [52] are now often replaced by Histogram Intersection kernels [66] or even Linear SVMs. Other machine learning algorithms have also been presented such as Random Forest and Lazy style algorithms (e.g. $k$nn); however, achieving lower performance or presenting higher computational complexity. A recent trend in video annotation is to learn features directly from the raw keyframe pixels using DCNNs. DCNNs consist of many layers of feature extractors which makes them having a more complex structure than hand-crafted representations. DCNNs can be used either as feature generators as described in Section 2.1 but also as standalone classifiers, i.e., unlabeled keyframes are forward propagated by a DCNN that performs the final class label prediction directly, using typically a softmax or a hinge loss layer [94, 57].

The small number of labeled training examples is a common problem in video datasets, making it difficult to train a deep network from scratch without over-fitting its parameters on the training set [96]. For this reason, it is common to use transfer learning that uses the knowledge captured in a source domain in order to learn a target domain without caring about the improvement in the source domain. When a small-sized dataset is available for training a DCNN then a transfer learning technique is followed, where a conventional DCNN, e.g. [46], is firstly trained on a large-scale dataset and then the classification layer is removed, the DCNN is extended by one or more fully-connected layers that are shared across all of the tasks, and a new classification layer is placed on the top of the last extension layer (having size equal to the number of concepts that will be learned in the target domain). Then, the extended network is fine-tuned in the target domain [83].

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 accuracy of concept-based video annotation. 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 [102] is a more elaborate approach; classifiers that have been trained on similar features (e.g. SIFT and RGB-SIFT) are firstly fused together and then, 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, [93] 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 [116], [17], [78], [19]. 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 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 [116], however they have also been briefly examined for video concept-based annotation [78]. Cascades developed for object and face detection are mainly boosting-based [116], [17], [78], [19], [7]. 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, concept-based video annotation systems utilize a different kind of features, visual local descriptors encoded into global image representations or DCNN-based features. These global image representation features are robust enough to be used for training strong classifiers without the need of the boosing technique.

Boosting is useful when weak features, such as pixel-level features, are available that alone cannot build a strong classifier. In addition, existing boosting-based techniques are based on pixel-level features, however, as described above concept-based video annotation is typically based on global image representations. As a result, boosting approaches for object detection could not be used without appropriate modifications.

2.3 Multi-task learning and structured outputs

As described in Section2.2, video concept annotation is a challenging multi-label classification problem that in recent years is typically addressed using DCNN models that choose a specific DCNN architecture [94, 46] and put a multi-label cost function on the top of it [121, 119, 11]. As is the case in other multi-label problems, there exist relations between the different concepts, and several methods attempt to utilise/model them so as to improve the performance or reduce the complexity of classification models that treat each concept independently. These methods can be roughly divided in two main categories. In the first category, methods that fall under the framework of multi-task learning (MTL), attempt to learn representations or classification models that, at some level, are shared between the different concepts (tasks) [3, 80, 76, 35, 23, 4, 141, 105, 70, 58, 137, 69, 127]. In the second category, methods that fall under the framework of structured-output prediction attempt to learn models that make multi-dimensional predictions that respect the structure of the output space using either label constraints or post-processing techniques [95, 122, 24, 27, 71, 84, 129, 84, 118, 117, 135, 63, 8, 65, 14, 107, 25, 104, 92, 26, 139, 69]. Label constraints refer to regularizations that are imposed into the learning system in order to exploit label relations (e.g., correlations) [84, 129, 138, 92, 26, 139, 69]. Post-processing techniques refer to re-calculating the concept prediction results using either meta-learning classifiers or other re-weighting schemes [95, 122, 24, 27, 71]. In what follows, we review works in those two broad categories.

2.3.1 Multi-task learning

Multi-task Learning (MTL) refers to jointly learning classifiers for many tasks by sharing knowledge across them so as to improve their accuracy, instead of learning individual models for each task. Video/image concept annotation can be treated as a MTL problem, where each task is about recognizing one concept. MTL methods can be divided into two broad categories: i) Shallow MTL methods that focus on shallow linear models and typically require pre-computed features as input, for example local descriptors or DCNN-based pre-computed features and ii) MTL methods that are an integral part of deep network architectures.

The MTL methods belonging to the first category extend typical linear models (e.g., Support vector machines (SVMs)) in order to incorporate task relatedness, i.e., the type of knowledge that should be shared. In a single-task learning (STL) concept annotation scenario, a supervised classifier is trained per concept on positive/negative keyframes/images of this concept. If the classifier is linear (e.g., SVM) its goal is to minimize the empirical cost: $\min_{(\boldsymbol{w}_j)} \mathcal{L}(\boldsymbol{w}_j) + \Theta(\boldsymbol{w}_j)$, where $\boldsymbol{w}_j \in \mathbb{R}^{d_1 \times 1}$ is the task parameter vector to be estimated from the training samples, $\mathcal{L}(\boldsymbol{w}_j)$ is the empirical cost on the training set, $\Theta(\boldsymbol{w}_j)$ is a regularization term and d_1 is the dimensionality of the input feature representation. MTL methods learn the parameters of all of the tasks together at the same time. As a result, assuming T tasks, all the task parameter vectors $\boldsymbol{w}_j$ for $j = 1...T$ are concatenated in a single parameter matrix $\boldsymbol{W} \in \mathbb{R}^{d_1 \times T}$ and the classifier's goal is to minimize the empirical cost: $\min_{(\boldsymbol{W})} \mathcal{L}(\boldsymbol{W}) + \Theta(\boldsymbol{W})$, where $\Theta(\boldsymbol{W})$ now encodes task relatedness. The main difference between MTL methods is the way they define task relatedness. Some methods identify shared feature representations between different tasks and use regularization over $\boldsymbol{W}$ to model task relatedness [3, 80, 76]. Others identify a shared subspace over the task parameter vectors [35, 23, 4]. The methods above make the strong assumption that all tasks are related; some newer methods consider the fact that some tasks may be unrelated.

For example, the clustered MTL algorithm (CMTL) [141] uses a clustering approach to assign to the same cluster parameters of tasks that lie nearby in terms of their L2 distance. Adaptive MTL (AMTL) [105] decomposes the task parameters into a low-rank structure that captures task relations, and a group-sparse structure that detects outlier tasks. The GO-MTL (Grouping and Overlap in Multi-Task Learning) algorithm [58] and the online version of it [70] use a matrix factorization method, e.g., $\boldsymbol{w}_j = \boldsymbol{V}\boldsymbol{s}_j^\top$, that allows two tasks from different groups to overlap by having one or more bases in common. $\boldsymbol{V}$ corresponds to the parameter vectors of k latent tasks, while $\boldsymbol{s}_j \in \mathbb{R}^{1\times k}$ is a task-specific weight vector that contains the coefficients of the linear combination of the latent tasks.

With respect to the second category of MTL methods, DCNNs themselves are MTL models that consist of many layers of feature extractors, with the bottom layers learning more generic features that are shared across all of the tasks and the top-most layers being more concept-specific [131]. Typical DCNN architectures follow a *hard* feature/parameter sharing, i.e., each task uses exactly the same weight matrix for the corresponding layer; and similarly a *hard* feature/parameter separation, i.e., the last layer (a.k.a. the classification layer) takes as input the output of the second-last layer and translates it into a set of concept annotation scores learning weight matrices independently for each task [128, 46, 94]. However, more elaborate MTL methods that introduce *soft* feature/parameter sharing, i.e., adjusting how much information and across which tasks should be shared, have been presented. Such methods mainly focus on reformulating existing shallow linear MTL methods in order to be incorporated in DCNNs. For example, [127] proposes a two-sided neural network that unifies several shallow linear MTL methods that use a predictor matrix factorization approach, e.g., $\boldsymbol{w}_j = \boldsymbol{V}\boldsymbol{s}_j^\top$ [58]. MTL in deep learning architectures has also been proposed for facial landmark detection [137] and human pose estimation [81]. In [137] the single task of facial landmark detection is optimized with the assistance of an arbitrary number of related tasks. This is a special case of the conventional MTL that typically aims

to maximize the performance of all tasks. In [81], the task of human detection is learned jointly with the task of body locations estimation, which results in improved human pose estimation. In [69] the two-sided neural-network of [127] is modified and extended, for transferring a network that has been originally trained on a source image dataset for concept annotation, to a target video dataset and a corresponding new set of target concepts.

Sometimes the terms MTL and multi-domain learning (MDL) are used interchangeably. However, the problems that each of them aims to solve are different. In MDL shared knowledge is exploited across different domains for the same tasks. For example, in [132] a cross-media (text, image, audio etc.) MDL retrieval method is proposed, where across the domains, the same set of concepts need to be learned. Similarly, an asymmetric MDL approach for person re-identification is proposed in [119] and a multiple-scene surveillance video understanding approach in [126], where in these works different domains refer to different video capture conditions. Asymmetric learning, AKA domain adaptation [60, 124], refers to the fact that the method of [119] utilizes information from different source domains in order to improve the performance on the target domain, without considering potential improvement to the source domains as well. Transfer learning, also discussed in Section 2.2, is another related problem that uses the knowledge captured in a source domain in order to learn a target domain without caring about the improvement in the source domain. MDL is also referred in the bibliography as multi-view learning, where different "views" refer to different feature representations of the same example such as, image and text, audio and video etc. The recent years, a very popular topic is multi-view representation learning, which is a sub-topic of multi-view learning. Multi-view representation learning methods focus on building embedding models for each particular view and on the way that these independent models can be jointly optimized in order to build a complex model that leverages information from multiple views. The reader who is interested in this topic could refer to the survey of [61] that reviews many approaches on multi-view

representation learning.

2.3.2 Structured output prediction

Structured output prediction refers to methods that exploit semantic relations that may exist between the concept labels, and has received a lot of attention in the deep learning and the broader machine learning literature. In contrast to MTL that exploits the common structure that task parameters or low-level features may have across the different tasks, structured output prediction focuses on the semantic relations that exist at the outputs, e.g., concept correlations. Video/image concept annotation is a multi-label learning problem, where given a set of concept labels, each keyframe/image is often associated with more than one labels. In most concept annotation datasets, ground-truth annotation is provided without any accompanying structure information concerning the concept labels; however, in many cases the concept labels are statistically related. For example, in the TRECVID-SIN video annotation dataset [82], which is one of the datasets used in this thesis, there are several groups of mutually exclusive labels, such as *indoor-outdoor* or *nighttime-sun*. The dataset also includes several positive correlations, such as *car-vehicle* and *dog-animal*. The automated learning of such relationships can incorporate useful knowledge into the model, improving the accuracy of the DCNN. In order to do so, many structured output prediction methods impose some label structural constraints either explicitly, i.e., using predefined rules that are known for the training dataset, or implicitly, i.e., the model is forced to discover existing label relations and considers them as label constraints. Existing methods can again be divided in those that take as input any pre-computed features and those that are tightly integrated with deep learning architectures.

With respect to the the first category, i.e., methods that take as input pre-computed features, two main sub-categories have appeared in the literature: a) Stacking-based approaches that collect the concept annotation scores produced either by a baseline set of concept classifiers (e.g., SVMs) or by a DCNN when used as a standalone classifier,

and introduce a second learning step in order to refine them, and b) Inner-learning approaches that follow a single-step learning process, which jointly considers extracted features and semantic relations. Stacking approaches aim to detect relations across concepts in the last layer of the stack. The simple process of training each concept classifier independently is known as Binary Relevance (BR) transformation and is an elementary way of solving MLC problems. One popular group is the BR-based stacking approaches. For example, in [95] concept annotation scores are obtained from individual (BR-trained) concept classifiers in the first layer, in order to create a *model vector* for each shot. These vectors form a meta-level training set, which is used to train a second layer of independently trained concept classifiers. In [122], a graph-based method is proposed that uses the ground-truth annotation to build decision trees that describe the relations across concepts, separately for each concept, and refines the initial scores by approximating these graphs. Using external knowledge of label relations, Deng et al. [24] proposed a representation, the HEX graph, to express and enforce exclusion, inclusion and overlap relations between labels. This model was further extended for "soft" label relations using the Ising model by Ding et al. [27]. Nevertheless, none of the above approaches considers the use of multi-label classification methods as part of a stacking architecture. The latter is the focus of Chapter 3 of this thesis, where we describe and evaluate the use of such methods for building models in the second-layer of the stacking architecture that learns the semantic relations across labels. All the above-mentioned approaches implicitly capture label relations from the meta-level training set of model vectors. This is an inherent flaw of all those methods that take as input existing concept annotation scores and try to refine them, as a result, they rely on starting with good concept probability estimates in the model vectors, otherwise the errors are propagated to the next layers. Some good indicators of first-layer systems that produce good concept probability estimates can be found in Section 3.3 of Chapter 3.

Inner-learning approaches, on the other hand, use the extracted features and exploit

concept relations in a single learning step in order to build concept classifiers. For example, the authors of [84] and [129] propose methods that simultaneously learn the relation between visual features and concepts and also the semantic relations between concepts. In [138] a joint learning-to-rank approach is proposed, which naturally combines the benefits of training a DCNN with a structural SVM model that is used for concept ranking. However, inner-learning approaches suffer from computational complexity. For example, [84] has complexity at least quadratic to the number of concepts, making it inapplicable to real video/image concept annotation problems, where the number of concepts is large (e.g. hundreds or thousands). The LMGE algorithm (Label correlation Mining with relaxed Graph Embedding) [129], is a faster approach with linear complexity with respect to the number of concepts; however, the complexity of the training process is about n^3, where n refers to the number of training samples. Many more methods can be found in this category for multi-label image annotation, which explore such label relations to improve the classification accuracy at the expense of increased computational complexity compared to the stacking-based ones, e.g.,[118, 117, 135, 63, 8, 65, 14, 107, 25, 104].

With respect to the second category, i.e., methods that are an integral part of DCNN architectures, structured output prediction techniques have been proposed for application mainly to the pixel-wise semantic segmentation problem. The most popular approach is to combine a DCNN with a graphical model [92], [26], [139]. For example, in [92] a Markov random field is jointly used on top of a DCNN architecture in order to incorporate the spatial relations and label correlations of the assigned labels on the pixels of an image. Similarly, in [139] the conditional random field model is formulated as a recurrent neural network (RNN) and plugged in as part of a DCNN. Structured output prediction for DCNNs has also been proposed for other visual recognition problems, such as group activity recognition [26]. All of these methods employ probabilistic inference to correct the marginal probability of each label. In contrast to the above methods that use graphical models, in [69] an auxiliary cost function

that approximates the correlations between the concept labels gets added to the total network's cost. This auxiliary cost function takes the form of a constraint over the task-specific parameters of the network and is shown to improve its accuracy.

2.4 Zero-shot learning

Zero-shot learning for image retrieval is an active field that is referred to the problem of retrieving images for a single unknown concept label. This is a simpler problem compared to the one that we investigate in this thesis, i.e., retrieving video shots given a complex textual query AKA ad-hoc video search (AVS). AVS could also be placed under the umbrella of zero-shot learning considering that no training data is available with respect to the textual query. Fully-automatic AVS is a very challenging problem, where the complete video search is performed without any user intervention. Typically, the query is broken down to a set of concepts using NLP. Each video shot from the test video collection is annotated with the same set of concepts, e.g., using DCNNs, and a distance measure is applied in order to retrieve those video shots that are closer to the concept-based query representation [133, 87, 28, 111, 79, 29, 67, 64]. Building the concept-based query representation starts by using simple NLP rules, e.g., removing stop-words, extracting nouns, verbs etc. or simply space-separating the whole query, which results in a set of terms for the query. Then, the semantic relation between each of the terms and the concepts is calculated, and the most semantically similar concepts to these terms are selected. The novelty of [111] is that they also enrich each concept with additional information captured by Google or Wikipedia, while in [29] an inverted index structure is used in order to associate the query with the concepts. A semi-automatic system is presented in [112], where the user is asked to choose keywords given a test query. All the above methods treat the query as a set of simple terms. However, detecting the most useful parts of it, e.g., subsequences that contain the main content that the user asks for retrieval, could further improve the video search accuracy. Such a method is proposed in Chapter6 of this thesis. In contrast to the

above methods, in [31], query models are trained with videos retrieved from websites, which is significantly slower compared to all the other methods discussed here.

Some recent methods for concept-based search, for example word2vec [75, 55, 56], train semantic embedding spaces of words or sentences from a large corpus using simple architectures of neural networks. After the embedding space is established, both video shots and concepts can be projected to it in order to directly measure their distance [79], [38], [99]. For example, in [79] images are mapped into a semantic embedding space by combining the class label embeddings with the concept-based annotation results. It should be noted that combining the distances of the video shots from the target query calculated with respect to both concept-based and semantic embedding representations has not been investigated before.

2.5 Benchmarking datasets and evaluation strategies

TRECVID Semantic Indexing (SIN) task [82] is a popular benchmarking activity that provides a large-scale dataset for video concept-based video annotation. 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 was assumed to be binary, i.e., it was 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 500 target concepts has been produced, 346 of which have been collaboratively annotated by the participants and by Quaero annotators. A subset of 60 of them was selected for participants' submissions. Each year a different subset of the 60 concepts is finally officially evaluated.

In this thesis our experiments were performed on the TRECVID SIN 2013 dataset

[82] 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 345 concepts;
- elements of ground-truth: some shots were collaboratively annotated. For each shot and each concept, four possibilities are available: the shot has been annotated as positive (it contains the concept), the shot has been annotated as negative (it does not contain the concept), the shot has been skipped (the annotator cannot decide), or the shot has not been annotated (no annotator has seen the shot).

TRECVID SIN 2013 task was finally evaluated for 38 semantic concepts by calculating the Mean Extended Inferred Average Precision (MXinfAP) at depth 2000 [130]. MXinfAP is an approximation of the Mean Average Precision (MAP) that has been adopted by TRECVID [82] because it is suitable for the partial ground-truth that accompanies the TRECVID dataset [82].

From 2016 the TRECVID SIN task has been replaced by the new Ad-hoc Video Search (AVS) task [5]. According to the AVS task, given a set of shot boundaries for the AVS test dataset and a set of ad-hoc queries, for which ground-truth annotation does not exist, participants are asked to return for each query, the top 1000 video shots from the test set, ranked according to the highest possibility of being described by the query. A list of 30 ad-hoc queries was given for participants' submission in AVS 2016. A query is a textual description of persons, objects, activities, locations

etc. and combination of them, e.g., "Find shots of a person playing guitar outdoors". TRECVID AVS task is also evaluated in terms of the MXinfAP measure.

TRECVID AVS 2016 datasets provide the following materials:

- a development set that refers to the TRECVID SIN dataset
- a test set that contains roughly 600 hours of videos, comprising 335944 shots;
- shot boundaries (for both sets);
- a set of 30 queries.

TRECVID SIN (1,000h, 345 concepts) and TRECVID AVS (600h, 30 ad-hoc labels) datasets are annotated and evaluated on video-fragment level, which is the focus of this thesis. Other related video datasets also exist that focus on recognising concepts or more complex event activities in video level. For example, the EventNet dataset consists of 95,321 videos and 500 events and 4,490 concepts related to each event [44], the TRECVID MED [5] provides 1000h videos and 20 event classes and the FCVID is also one of the largest video datasets with 239 event-related categories and 91K youtube videos [53]. The Sports-1m (1 million videos, 487 sports-related activities) [54] and UCF-101 (13,320 videos, 101 activities) [100] are activity detection datasets; and the YouTube-8M (450,000h, 4,716 concepts) [1] is a concept-annotated video datasets. All the above datasets are among the most large-scale and most challenging benchmarking datasets for automatic video understanding. Related can be considered also benchmarking image annotation datasets such as the ImageNet [89] (tens of millions images, 100,000 concepts), the PASCAL-VOC 2012 (11,540 images, 20 concepts) [34], the NUS-WIDE (269,648 images, 81 concepts) [21], and the Places (over 7 million images, 205 scene-related concepts) [140].

2.6 Conclusions

In this chapter we presented the most important works concerning the problems of concept-based video annotation and ad-hoc video search. Based on such state-of-the-art approaches we propose different extended and varied architectures for solving the above challenging problems. We began by presenting approaches on visual feature extraction using either handcrafted or DCNN-based features, then we presented classical learning architectures for training independent binary concept classifiers, e.g., using SVMs, or multi-label classifiers, e.g., using deep learning. We continued with more advanced approaches that exploit concept relations at two different levels. Either at feature-level relations between concepts building on ideas from multi-task learning or at semantic level relations between concepts, e.g., by exploiting concept label correlations. Having noted the limitations of the methods discussed in this section we will present improved machine learning architectures that go beyond the state-of-the-art as it is described below:

We firstly improve existing architectures that are based on binary, independently trained concept classifiers (Chapter 3). Specifically, we deal with the problem of extending and using different local descriptors, as well as exploiting concept semantic relations, towards improved concept-based video annotation. We examine if state-of-the-art binary local descriptors that have been previously used only for image annotation([88], [59]), can also facilitate concept-based video annotation. We propose color extensions of them inspired by previously proposed color extensions of SIFT [62]. Concerning the learning stage of our independently trained concept classifiers, we perform a comparative study and propose an improved way of employing stacked models, which capture concept relations, by using multi-label classification algorithms in the last layer of the stack, in contrast to existing methods that use model vectors in order to perform a second round of training binary classifiers that refine the initial prediction scores [95].

Secondly, we propose a cascade architecture which is another way of improving both the accuracy and the prediction time of independently trained concept classifiers (Chapter 4). Our proposed cascade architecture dynamically selects, orders and combines many base classifiers, trained independently with different feature-based keyframe representations, in contrast to existing cascades that are based on a fixed ordering of the cascade stages [78], as presented in Section 2.2.

Then, we aim to improve existing multi-label deep learning architectures by incorporating convolutional layers that enforce the sharing of latent concept representations and model the correlations between the concept labels (Chapter 5). Our method has some similarities with the MTL methods [58] and [69]. However, in contrast to [58] that uses pre-computed features and a shallow linear model to factorize the 2D weight matrix that encodes concept-specific features, our approach learns shared representations as an integral part of a DCNN architecture implementing the factorization in two standard CNN layers. Furthermore, in contrast to [69] that uses a two-sided CNN taking as input in the one side a keyframe/image, and in the other side a semantic descriptor that verifies whether a certain concept is present in the keyframe/image, our method requires only the raw keyframe/image and no additional information regarding the task that should be learned/predicted. Both [58] and [69] can be optimized for only binary or multi-class classification cost functions (e.g., logistic loss), thus ignoring the multi-label nature of the concept-based video/image annotation problem. In contrast, our method works for any multi-label classification cost (e.g., cross-entropy). Our method is also most closely related to [69] and [70], that jointly consider MTL and structured outputs, but differs from them as follows: In [70] pre-computed features are used, specifically, a shallow linear MTL method is proposed that is instantiated with a new cost function that exploits concept correlations and takes as input pre-computed DCNN-based features. In contrast, we propose a single DCNN architecture, trained end-to-end, that incorporates both MTL and structured-output prediction, both of which are implemented using standard convolutional layers, resulting in a more ef-

fective classifier. In [69] an auxiliary label-based cost function is proposed that forces the network's output to fit the distribution of a single row of the concept correlation matrix. In contrast, our approach adds a concept correlation cost term to the network's main cost function that forces positively-correlated concepts to receive similar scores and negatively-correlated ones to receive dissimilar scores, again leading to better results.

Finally, we present a fully-automatic method that combines concept-based video annotation and textual query analysis in order to solve the problem of ad-hoc video search (Chapter 6). We present a set of NLP steps that cleverly analyse different parts of the query in order to convert it to related semantic concepts. Specifically we can detect the most useful parts of it, e.g., subsequences that contain the main content that the user asks for retrieval, in contrast to existing approaches that simply use a parser to extract parts of speech such as verbs, adjectives etc. We also propose a new method for transforming concept-based keyframe and query representations into a common semantic embedding space, and we show that our proposed combination of concept-based representations with their corresponding semantic embeddings results in improved video search accuracy. Existing methods simply use concept-based representations [133, 87, 28, 111, 79, 29, 67, 64].

Chapter 3

Learning with Local Features and a Two-layer Stacking Architecture

Contents

In Chapter 1 we presented a typical concept-based video annotation system that learns independent concept classifiers and consists of three main modules: the video decomposition module, the feature extraction module, and finally the learning module (Fig. 1.1). In this typical system, on the one hand, the keyframe features that will be extracted for training concept classifiers should be carefully selected in order to train discriminative models, and on the other hand, concept relations should be exploited in a post-processing step, because any existing semantic relations among concepts are not taken into account (e.g., the fact that *sun* and *sky* will often appear together in the same video shot). In this chapter we focus on two directions: firstly, on feature-based video representation and secondly, on learning algorithms that exploit semantic concept relations. We present how different binary and non-binary local descriptors

can be extended and used for building effective independent concept classifiers. We also present a stacking architecture that is able to further improve the prediction scores of a given set of independently trained concept classifiers, by exploiting label relations in the last layer of the stack.

Considering feature extraction for video representation, Scale Invariant Feature Transform (SIFT) [62] and Speeded Up Robust Features (SURF) [9] are probably the two local descriptors that are most-widely used. However, they are non-binary descriptors, which makes them not so suitable for modern applications requiring the transmission of descriptor vectors. For example, when considering a mobile application where pictures are taken with a mobile device and local descriptors from these pictures need to be sent to a server for semantic analysis, then it is very important that the local descriptors are as compact as possible, to minimize transmission requirements [18]. Although mobile applications and improvement in terms of time efficiency is not the scope of this thesis, we are always interested in having features that are as compact as possible in order to provide a competitive and fast video annotation algorithm. This is very important for other researchers that do research in this field in order to realize that a successful video annotation algorithm should be a combination of accuracy and time efficiency. ORB (Oriented FAST and Rotated BRIEF) [88] and BRISK (Binary Robust Invariant Scalable Keypoints) [59] are two binary local descriptors, which were originally proposed for similarity matching between local image patches. On the machine learning front, the majority of concept-based video annotation systems learn supervised classifiers separately for each semantic concept. However, assigning concepts to video shots is by definition a multi-label classification problem, since multiple concepts may describe a single video shot. The simple process of training each concept classifier independently is known as Binary Relevance (BR) transformation and is an elementary way of solving multi-label learning problems. One way of improving this baseline BR approach, is to consider concept relations. A group of methods in this category follow a stacking architecture (e.g. [95], [51]). The predictions of multiple

BR-trained concept classifiers form model vectors that are used as a meta-learning training set for a second learning round (mainly by adopting a second round of BR models).

Our contributions are:

- A thorough examination of ORB and BRISK in the task of concept-based video annotation that shows that they constitute a viable alternative to the non-binary descriptors currently used in this task, while their compact size and low storage needs make them appealing for mobile applications.
- Color extensions for the three local descriptors considered in this chapter (SURF, ORB, BRISK). Specifically, inspired by two color extensions of SIFT [113], namely RGB-SIFT and OpponentSIFT, we define the corresponding color extensions for SURF, ORB and BRISK, and we show that this relatively straightforward way of introducing color information is in fact a generic methodology that works similarly well for different binary and non-binary local descriptors.
- A different way of performing Principal Component Analysis (PCA) [123] for feature reduction, which often improves the results of SIFT/SURF/ORB/BRISK color extensions when combined with Vector of Locally Aggregated Descriptors (VLAD) encoding [48].
- An improved stacking architecture that elaborates multi-label classification algorithms instead of BR models for the second-layer learning, in order to capture semantic concept relations.

Another distinguishing feature of this chapter is the way that concept-based video annotation is evaluated. A closer look at the literature shows that researchers focus on evaluating video annotation in a semantics-based indexing and retrieval setting, i.e. given a concept, measure how well the top retrieved video shots for this concept

truly relate to it. However, besides the retrieval problem, another important problem related to semantics-based video manipulation is the annotation problem, i.e. the problem of estimating which concepts best describe a given video shot. We report evaluation considering both criteria and compare them.

The rest of this chapter is organized as follows: Section 3.1 examines how two binary descriptors can be used for video concept detection, introduces the color extensions of SURF, ORB and BRISK and discusses a different way of employing PCA for color descriptors. Section 3.2 presents the proposed stacking architecture for exploiting concept relations and multi-label learning algorithms that are suitable for instantiating this architecture. Section 3.3 reports our experiments and results, and finally Section 3.4 summarizes our main conclusions.

3.1 Building independent concept classifiers

In this section we present how different local descriptors can be extended and used for building effective independent concept classifiers. The classifiers can be used as stand alone classifiers or alternatively as part of a stacking architecture.

3.1.1 Using a binary local descriptor for concept-based video annotation

ORB [88] and BRISK [59] are two binary local image detectors and descriptors that present similar discriminative power compared to SIFT and SURF in image matching problems, they have similar properties such as invariance in rotation, scale and illumination, but at the same time are more compact and faster to be computed. A 256-element binary ORB vector requires 256 bits to be stored (similarly a 512-element binary BRISK vector requires 512 bits); in contrast, an integer-quantized 128-element SIFT vector requires 1024 bits. In addition, according to [88] and [59], ORB and BRISK are an order of magnitude faster than SURF to compute, which in turn is

faster than SIFT.

There is not a single way for introducing binary descriptors in the video annotation pipeline. [43] did so by considering the BoW encoding, and proposed a modified K-means algorithm (the "K-majority" algorithm) for generating the codebook (vocabulary) of BoW, that would result in a binary codebook.

In this work we claim that binary descriptors (ORB, BRISK) can be used for video annotation in the same way as their non-binary counterparts. Specifically, let us assume that I is a set of images and $\boldsymbol{x_i}$ $i = 1, ..., N$ are ORB or BRISK descriptors extracted from I, where $\boldsymbol{x_i} \in \{0, 1\}^d$. N is the total number of extracted local descriptors and d is their dimension. From these binary descriptors, we generate a floating-point codebook of K visual codewords $w_k \in \mathbb{R}^d$, $k = 1, ..., K$, using a standard K-means. The distances between the binary ORB/BRISK descriptors and the codewords are calculated by the L2 norm. The update of the cluster centres is also performed as in the original K-means (calculating the mean of a set of vectors). We compare these two codebook creation strategies (that of [43] and the one described in this section) in Section 3.3.1.

3.1.2 Color extensions of binary and non-binary local descriptors

Based on the good results of two color extensions of SIFT, namely RGB-SIFT and OpponentSIFT [113], we examine the impact of using the same methodology for introducing color information to other descriptors (SURF, ORB, BRISK). Our objective is to examine if this is a methodology that can benefit different local descriptors and is therefore generally applicable.

Let d denote the dimension of the original local descriptor (typically, d will be equal to 64 or 128 for SURF, 128 or 256 for ORB and 512 for BRISK). This section summarizes the process of extracting RGB-SURF, RGB-ORB, RGB-BRISK, OpponentSURF, OpponentORB and OpponentBRISK descriptors. An RGB image has three 8-bit

channels (for red, green and blue). The original non-color local descriptors are calculated on 8-bit grayscale images, so they first transform the RGB image to grayscale. In contrast to this, our RGB-SURF/ORB/BRISK apply the corresponding original descriptor directly to each of the three R, G, B channels and for each keypoint extract three d-element feature vectors. These are finally concatenated into one $3 \cdot d$-element feature vector, which is the RGB-SURF, RGB-ORB or RGB-BRISK descriptor vector.

Similarly, our OpponentSURF/ORB/BRISK descriptors firstly transform the initial RGB image to the opponent color space [113]. We refer to the transformed channels as O_1, O_2 and O_3. O_3 is the luminance channel, i.e. the one that the original SURF/ORB/BRISK descriptors use, while the other two channels (O_1 and O_2) capture the color information. Following the transformation, a normalization step that converts the ranges of each channel within the [0,255] range is employed, as in [113]. Then, similarly to RGB-SURF/ORB/BRISK, the original SURF, ORB or BRISK descriptor is applied separately to each transformed channel and the final $3 \cdot d$-element feature vector is the concatenation of the three feature vectors extracted from the three channels.

3.1.3 Reducing the dimensionality of local color descriptors

State-of-the-art local descriptor encoding methods generate high-dimensional vectors that make training of machine learning algorithms difficult. For example, while the BoW model generates a k-element feature vector, where k equals to the number of visual words, VLAD encoding generates a $k \cdot l$-element feature vector (where l is the dimension of the local descriptor; in the case of the color extensions of descriptors discussed in the previous section, $l = 3 \cdot d$). Thus, it is common to employ dimensionality reduction before the construction of VLAD vectors, on local descriptors, mainly using PCA [123]. In this section we explain that directly applying PCA to the full vector of color descriptors, as implied from previously published works (e.g. [16]; termed "typical-PCA" in the sequel), is not the only possible solution, and we propose a simple

modification of this descriptor dimensionality reduction process that it experimentally shown to improve the concept-based video annotation results in several cases.

PCA projects linearly l-dimensional features to a lower-dimensional feature space. Given a matrix A with dimension $l \times n$, where n is the number of observations (i.e., of keyframes in a video training dataset), if we want to perform dimensionality reduction (from l to l') with PCA, the reduced matrix A' will be $A' = E^T \cdot A$, where E is the projection matrix (of dimension $l \times l'$) and T denotes the transpose of a matrix.

PCA aims to find those directions in the data space that present high variance. When PCA is applied directly to the entire vector of one of the color extensions of (binary or non-binary) local descriptors, if one or two of the three color channels of the descriptor exhibit lower diversity than the others, then these risk being under-represented in the reduced dimensionality space. To avoid this, we propose performing PCA separately for each color channel and consider an equal number of principal components from each of them, to create three projection matrices that correspond to each of the three channels (termed "channel-PCA" in the sequel), instead of one projection matrix that corresponds to the complete descriptor vector. The three reduced single-channel descriptor vectors that can be obtained for a color descriptor using the aforementioned projection matrices are finally concatenated in a reduced color-descriptor vector. In some of our preliminary experiments we chose different number of principal components from each color channel based on their variance. However, this did not present large fluctuations in the overall system's accuracy and this is the reason why we finally take an equal number of components from each color channel. However, more investigation towards this direction could be performed in the future for finding a better way of choosing the number of components per channel in accordance with their influence on the entire video frame encoding vector.

3.2 Stacking for exploiting concept relations

Having presented our methods for building concept classifiers in Section 3.1, this section deals with refining the scores of such independently learned concept classifiers by exploiting label relations. Assuming that we first train a set of SVM-based or LR-based independent concept classifiers, here we propose an improved way of subsequently employing stacked models, by using appropriate multi-label classification methods, able to capture concept relations, in the last layer of the stack.

3.2.1 Proposed stacking architecture

Let $D_1, ..., D_N$ denote a set of N trained independent concept classifiers on N different concepts. Let V denote a validation set of video shots, which will be used for training the second layer of the stacking architecture, and m denote the model vector of a new unlabeled video shot. Fig. 3.1 summarizes the full pipeline from training the second-layer classifiers to using them for classifying an unlabeled sample when using: (1) the BR stacking architecture (Fig. 3.1(b),(d)) presented in Section 2.3.2, and (2) the proposed stacking architecture (Fig. 3.1(c),(e)). Both architectures use exactly the same strategy to create the meta-level training set; the trained BR models $(D_1, ..., D_N)$ of the first layer are applied to the validation dataset T and in this way a model vector set M is created, consisting of the scores that each of $D_1, ..., D_N$ has assigned to each video shot of V for every concept (Fig. 3.1(a)). What distinguishes the two architectures is the way that this meta-learning information is used and therefore the way that the second-layer learning is performed.

During the training phase, the BR stacking architecture builds a new set of BR models $(D'_1, ..., D'_N)$. To train each model, a different subset of M that is ground-truth annotated for the corresponding concept C_n that the meta-concept classifier D'_n will be trained for, is used (Fig. 3.1(b)). In contrast, the proposed architecture uses the whole model vector set and ground-truth annotation at once in order to train a

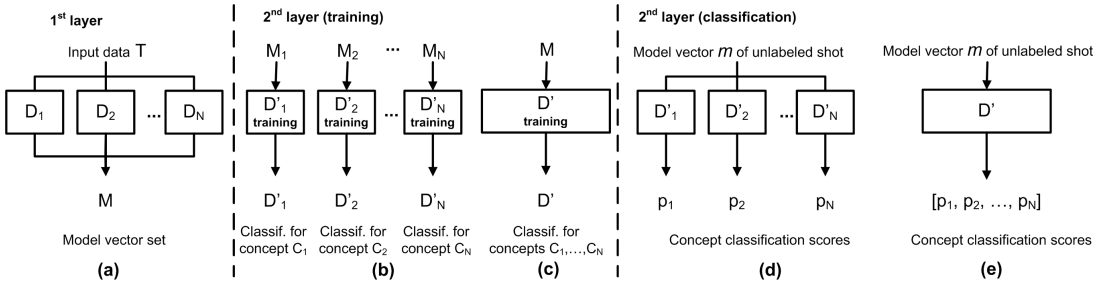

Figure 3.1: Comparing BR and the proposed stacking architecture. (a) First layer of a stacking architecture. (b) Training of the second layer of a BR-stacking architecture. (c) Training of the second layer of the proposed stacking architecture. (d) Classification phase of the BR stacking architecture. (e) Classification phase of the proposed architecture.

single multi-label classification model D', able to capture concept relations, instead of separate models $D'_1, ..., D'_N$ (Fig. 3.1(c)).

During classification, a new unlabeled video shot is given to the first layer BR models $(D_1, ..., D_N)$ and a model vector m is returned. Then on the one hand, the BR stacking architecture lets the $D'_1, ..., D'_N$ models to classify m and one score is returned separately from each (Fig. 3.1(d)). On the other hand, the proposed architecture uses the single trained model D' in order to return a final score vector (Fig. 3.1(e)).

With respect to learning concept relations, the BR-based stacking methods learn them only by using the meta-level feature space. BR algorithm is explained in details in Section 2.3.2. One classifier is trained separately from the positive and negative frames of each concept. Then the independently trained concept classifiers are used to classify a validation set, their prediction scores on this set form model vectors that are used as feature representations for training a second round of concept classifiers separately for each concept. BR is a multi-label classification algorithm that solves the problem by utilizing label transformations that ignore label relations. Subsequently, the learning of each concept is still independent of the learning of the rest of the concepts. The rationale behind us proposing the use of other multi-label learning algorithms in replacement of the BR models at the second layer of the stacking archi-

tecture is based on the assumption that if we choose algorithms that consider label relations as part of the second-layer training, improved detection can be achieved. Our stacking architecture learns concept relations in the last layer of the stack both from the outputs of first-layer concept classifiers and by modelling relations directly from the ground-truth annotation of the meta-level training set. This is achieved by instantiating our architecture in our experiments with different second-layer algorithms that model:

- Relations between pairs of concepts;
- Relations across sets of more than two concepts;
- Multiple relations in the neighbourhood of each testing instance.

3.2.2 Learning algorithms for stacking

To model the relations described above, we exploit methods from the multi-label learning field [40], [86], [108], [136]. Pairwise methods can consider pairwise relations between labels; similar to the multi-class problem, one versus one models are trained and a voting strategy is adopted in order to decide for the final classification. In this category we choose the Calibrated Label Ranking (CLR) algorithm [40] that combines pairwise and BR learning. Label powerset (LP) methods search 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. We choose the original LP tranformation [108], as well as the Pruned Problem Transformation algorithm (PPT) [86] that reduces the class imbalance problem by pruning label sets that occur less than l times. Finally, lazy style methods most often use label relations in the neighbourhood of the tested instance, to infer posterior probabilities. In this direction we choose ML-kNN algorithm [136], which models exactly this information. In selecting the above methods, we took into account the computational complexity of these and other similar

methods and tried to avoid using particularly computationally expensive ones, e.g., RAkEL and HOMER that are based on ensembles of classifiers [77].

The use of multi-label classification algorithms as the second layer of a stacking architecture has the significant advantage of allowing the representation of the videos using state-of-the-art high dimensional low-level features (for describing the video at the first layer of the stack), as opposed to simpler features used in e.g. [77], [120], while at the same time keeping relatively low the dimensionality of the input to the multi-label classifier of the second layer, thus making the overall concept-based video annotation architecture applicable even to large-scale problems.

3.3 Experimental study

Our experiments were performed on the TRECVID SIN 2013 dataset presented in Section 2.5. Firstly, in Section 3.3.1 we performed experiments to assess the performance of many different binary and non-binary local descriptors and also color variants of them that were presented in Sections 3.1.1, 3.1.2 and 3.1.3. On these experiments we also want to examine the performance of the different methods both on the video indexing and on the video annotation problem. Based on this, we adopt two evaluation strategies: i) Considering the video indexing problem, given a concept, we measure how well the top retrieved video shots for this concept truly relate to it. ii) Considering the video annotation problem, given a video shot, we measure how well the top retrieved concepts describe it. For the indexing problem we calculated the Mean Extended Inferred Average Precision (MXinfAP) at depth 2000 [130], presented in Section 2.5. For the annotation problem we calculate the Mean Average Precision at depth 3 (MAP@3). In the latter case, our evaluation was performed on shots that are annotated with at least one concept in the ground-truth.

For experimenting with different local descriptors, one keyframe was initially extracted for each video shot and was scaled to 320×240 pixels prior to feature extraction.

For some of our final experiments, we also extracted two visual tomographs [93] from each shot. Tomographs are spatio-temporal video slices that are extracted and used in the same way that keyframes are typically used in video concept annotation schemes. These spatio-temporal slices capture in a compact way motion patterns that are useful for detecting semantic concepts and are used for training the concept classifiers. Regarding feature extraction, we followed the experimental setup of [16] and we used the toolbox that its authors have published. More specifically, we used the dense SIFT descriptor, that accelerates the original SIFT descriptor, in combination with the Pyramid Histogram Of visual Words (PHOW) approach [13]. For SURF, ORB and BRISK we used their implementations included in OpenCV, and further extended these implementations with the corresponding color variants that we introduced in Section 3.1.2. The same square regions at different scale levels of the PHOW approach were used as the image patches that were described by SURF, ORB and BRISK. We calculated 128-SIFT, 128-SURF, 256-ORB and 512-BRISK grayscale descriptors; then, each color extension of a descriptor resulted in a color descriptor vector three times larger than that of the corresponding original descriptor, as explained in Section 3.1.2. All the non-binary local descriptors (SIFT, SURF and their color extensions) were compacted to 80 dimensions, using PCA, following the recommendations of [16] and [49]. Since there is no previous research on the influence of dimensionality reduction on binary descriptors when they are used for video annotation, ORB, BRISK and their color extensions were compacted both to 80 and to 256 dimensions (the latter is the original size of ORB), in order to investigate this. All the compacted local descriptors (binary and non-binary) were subsequently aggregated using VLAD encoding. Similarly with the authors of [16], we divided each image into the same 8 regions using spatial binning and we used sum pooling to combine the encodings from different regions. As a result of the above process, a VLAD vector of 163840 elements for descriptors compacted to 80 dimensions and of 524288 elements for descriptors compacted to 256 dimensions was extracted for each image (by image we mean here

either a keyframe or a visual tomograph). These VLAD vectors were compressed into 4000-element vectors by applying a modification of the random projection matrix [10]. The reduced VLAD vectors were L2 normalized according to [16] and served as input to the Logistic Regression (LR) classifiers that we used. Following the *cross validated committees* methodology of [72], we trained five LR classifiers per concept and per local descriptor (SIFT, ORB, RGB-ORB etc.), and combined the output of these five by means of late fusion (averaging). When different descriptors were combined, again late fusion was performed by averaging the classifier output scores.

Secondly, in Section 3.3.2 we assess the usefulness of the stacking architecture, presented in Section 3.2, that uses multi-label learning algorithms to capture label relations. 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 algorithms for the second layer of the stack. The first layer of the employed stacking consisted from the independent classifiers evaluated in Section 3.3.1. Their output was used to construct model vectors that were introduced in the second layer. It would be useful here to remind that, as presented also in Fig. 3.1, model vectors consist of the scores that each of first layer classifiers has assigned to each video shot of the validation set (i.e., TRECVID 2012) for every concept. We instantiated the second layer of the proposed architecture with four different multi-label learning algorithms as described in Section 3.2.2, and will refer to our framework as P-CLR , P-LP, P-PPT and P-MLkNN when instantiated with CLR [40], LP [108], PPT [86] and ML-kNN [136] respectively. The value of l for P-PPT was set to 30. We compared these instantiations of the proposed framework against BCBCF [51], DMF [95], BSBRM [110], MCF [122] and CF [45]. For BCBCF we used the concept predictions instead of the ground-truth in order to form the meta-learning dataset, as this was shown to improve its performance in our experiments; we refer to this method as CBCFpred in the sequel. Regarding the concept selection step we used these parameters: $\lambda = 0.5$, $\theta = 0.6$, $\eta = 0.2$, $\gamma =$ the mean of Mutual Information values. For MCF we only used the

Table 3.1: Performance (MXinfAP, %, and MAP@3, %) for ORB, when the binary codebook proposed in [43] and when a floating-point codebook is used. In parenthesis we show the relative improvement w.r.t. the binary codebook.

	MXinfAP (indexing)		MAP@3 (annotation)	
Descriptor	Binary codebook [43]	Floating-point codebook	Binary codebook [43]	Floating-point codebook
ORB	4.52	10.36 (+129.2%)	66.85	71.05 (+6.3%)

spatial cue, so temporal weights have been set to zero. The ϕ coefficient threshold, used by BSBRM, was set to 0.09. Finally, for CF we performed two iterations without temporal re-scoring (TRS). We avoided using TRS in order to make this method comparable to the others. For implementing the above techniques, the WEKA [123] and MULAN [109] machine learning libraries were used as the source of single-class and multi-label learning algorithms, respectively.

In all of the experiments of this section, the final step of video annotation was to refine the calculated detection scores by employing the re-ranking method of [91]. This method slightly changes the score that was given per concept based on the scores that were assigned to its neighbors using the Gaussian distribution. Using this re-ranking method improves the final results by approximately 1-2 percentage points. We evaluated all the methods on the test set of the TRECVID 2013 SIN task [82] using the subset of 38 concepts that were also evaluated as part of the task. For the stacking architecture (Section 3.3.2), similar to Section 3.3.1, we examined the performance of the different methods both on the video indexing and on the video annotation problem.

3.3.1 Binary vs. non-binary local descriptors and their combinations

In this subsection we performed experiments to assess the performance of many different binary and non-binary local descriptors and also color variants of them that were presented in Sections 3.1.1, 3.1.2 and 3.1.3. Such experiments are very useful for establishing a baseline concept-based video annotation system that consists of in-

Table 3.2: Performance (MXinfAP, %, and MAP@3, %) for the different descriptors and their combinations, when typical and channel-PCA is used for dimensionality reduction. In parenthesis we show the relative improvement w.r.t. the corresponding original grayscale local descriptor for each of the SIFT, SURF, ORB, BRISK color variants.

		MXinfAP (indexing)			MAP@3 (annotation)		
Descriptor	Descriptor size in bits	Keyframes, typical-PCA	Keyframes, channel-PCA	Boost(%) w.r.t typical-PCA	Keyframes, typical-PCA	Keyframes, channel-PCA	Boost(%) w.r.t typical-PCA
SIFT	1024	14.22	14.22	-	74.32	74.32	-
RGB-SIFT	3072	14.97 (+5.3%)	14.5 (+2.0%)	-3.1%	74.67 (+0.5%)	74.07 (-0.3%)	-0.8%
OpponentSIFT	3072	14.23 (+0.1%)	14.34 (+0.8%)	+0.8%	74.54 (+0.3%)	74.53 (+0.3%)	0.0%
All SIFT (SIFTx3)	-	**19.11 (+34.4%)**	**19.24 (+35.3%)**	**+0.7%**	**76.47 (+2.9%)**	**76.38 (+2.8%)**	**-0.1%**
SURF	1024	14.68	14.68	-	74.25	74.25	-
RGB-SURF	3072	15.71 (+7.0%)	15.99 (+8.9%)	+1.8%	74.58 (+0.4%)	74.83 (+0.8%)	+0.3%
OpponentSURF	3072	14.7 (+0.1%)	15.26 (+4.0%)	+3.8%	73.85 (-0.5%)	74.07 (-0.2%)	+0.3%
All SURF (SURFx3)	-	**19.4 (+32.2%)**	**19.48 (+32.7%)**	**+0.4%**	**75.89 (+2.2%)**	**76.12 (+2.5%)**	**0.3%**
ORB 256 (no PCA)	256	10.36	10.36	-	71.05	71.05	-
RGB-ORB 256	768	13.02 (+25.7%)	13.58 (+31.1%)	+4.3%	72.86 (+2.6%)	73.21 (+3.0%)	+0.5%
OpponentORB 256	768	12.61 (+21.7%)	12.73 (+22.9%)	+1.0%	72.66 (+2.3%)	72.46 (+2.0%)	-0.3%
All ORB 256	-	**16.58 (+60.0%)**	**16.8 (+62.2%)**	**+1.3%**	**74.32 (+4.6%)**	**74.20 (+4.4%)**	**-0.2%**
ORB 80	256	11.43	11.43	-	72.02	72.02	-
RGB-ORB 80	768	13.79 (+20.6%)	13.48 (+17.9%)	-2.2%	73.20 (+1.6%)	72.96 (+1.3%)	-0.3%
OpponentORB 80	768	12.81 (+12.1%)	12.57 (+10.0%)	-1.9%	72.56 (+0.7%)	72.01 (0.0%)	-0.8%
All ORB 80 (ORBx3)	-	**17.48 (+52.9%)**	**17.17 (+50.2%)**	**-1.8%**	**74.64 (+3.6%)**	**74.58 (+3.6%)**	**-0.1%**
BRISK 256	512	11.43	11.43	-	72.36	72.36	-
RGB-BRISK 256	1536	11.78 (+3.1%)	12 (+5.0%)	+1.9%	72.74 (+0.5%)	72.67 (+0.4%)	-0.1%
OpponentBRISK 256	1536	11.68 (+2.2%)	11.96 (+4.6%)	+2.4%	72.42 (+0.1%)	72.35 (0.0%)	-0.1%
All BRISK 256 (BRISKx3)	-	**16.4 (+43.5%)**	**16.47 (+44.1%)**	**+0.4%**	**74.56 (+3.0%)**	**74.58 (+3.1%)**	**0.0%**
BRISK 80	512	10.73	10.73	-	71.79	71.79	-
RGB-BRISK 80	1536	12.21 (+13.8%)	11.6 (+8.1%)	-5.0%	72.70 (+1.3%)	72.29 (+0.7%)	-0.6%
OpponentBRISK 80	1536	11.05 (+3.0%)	11.15 (+3.9%)	+0.9%	72.10 (+0.4%)	71.49 (-0.4%)	-0.9%
All BRISK 80	-	**16.43 (+53.1%)**	**15.95 (+48.6%)**	**-2.9%**	**74.51 (+3.8%)**	**74.39 (3.6%)**	**-0.2%**

Table 3.3: Performance (MXinfAP, % ; MAP@3, %) of pairs and triplets of the best combinations of Table 3.2 descriptors (SIFTx3 channel-PCA, SURFx3 channel-PCA, ORBx3 typical-PCA, BRISKx3 channel-PCA).

(a) Descriptor pairs	+SURFx3	+ORBx3	+BRISKx3
SIFTx3	**22.4**; 76.64	21.31; **76.81**	20.71; 76.53
SURFx3		21.6; 76.43	21.13; 76.68
ORBx3			19.08; 75.34

(b) Descriptor triplets	+ORBx3	+BRISKx3
SIFTx3+SURFx3	**22.9**; 77.29	22.52; **77.39**
SIFTx3+ORBx3		21.5; 76.61
SURFx3+ORBx3		21.76; 76.56

Table 3.4: Performance (MXinfAP, %, and MAP@3, %) for the best combinations of local descriptors (SIFTx3 channel-PCA, SURFx3 channel-PCA, ORBx3 typical-PCA, BRISKx3 channel-PCA). (a) When features are extracted only from keyframes, (b) when horizontal and vertical tomographs [93] are also examined.

	MXinfAP (indexing)			MAP@3 (annotation)		
Descriptor	(a) Keyframes	(b) Keyframes+ Tomographs	Boost (%) w.r.t (a)	(a) Keyframes	(b) Keyframes+ Tomographs	Boost (%) w.r.t (a)
SIFTx3	19.24	20.28	+5.4%	76.38	76.30	-0.1%
SURFx3	19.48	19.74	+1.3%	76.12	75.98	-0.2%
BRISKx3	16.47	19.08	+15.8%	74.58	75.26	+0.9%
ORBx3	17.48	19.24	+10.1%	74.64	75.16	+0.7%
SIFTx3+SURFx3+ORBx3	22.9	24.57	+7.3%	77.29	77.79	+0.7%

dependently trained classifiers (one per concept) and also understand the usefulness of different feature types e.g., grayscale vs. color-based alternatives and binary vs. non-binary descriptors, the influence of PCA on the local vectors, and the usefulness of combining different feature types for the same concept. Finally, we show an altern-

ative technique for annotating video shots with semantic concepts where instead of extracting a representative keyframe per shot we extract a video tomograph per shot [93] that is able to capture the motion information of it.

We start by assessing the performance of classifiers in relation to the indexing problem. In Table 3.1 we compare the performance of the original grayscale ORB descriptor in concept-based video annotation, when used in conjunction with a binary codebook (as in [43]) and a floating-point one (as in Section 3.1.1). In both cases, VLAD encoding is employed. We can see that the binary codebook proves ineffective; the floating-point one outperforms it by more than 129%. It should be noted that the MXinfAP of random classification is <0.1%, which indicates the difficulty of the problem. Based on this result, in all subsequent experiments with ORB, BRISK and their color extensions a floating-point codebook was used.

In Table 3.2 we evaluate the different local descriptors and their color extensions considered in this work, as well as combinations of them. First, comparing the original ORB and BRISK descriptors with the non-binary ones (SIFT, SURF), we can see that binary descriptors perform a bit worse than their non-binary counterparts but still reasonably well. This satisfactory performance is achieved despite ORB, BRISK and their extensions being much more compact than SIFT and SURF, as seen in the second column of Table 3.2. Second, concerning the methodology for introducing color information to local descriptors, we can see that the combination of the original SIFT descriptor and the two known color SIFT variants that we examine ("All SIFT" in Table 3.2) outperforms the original SIFT descriptor alone by 34.4% (35.3% for channel-PCA). The similar combination of the SURF color variants with the original SURF descriptor, is shown in Table 3.2 to outperform the original SURF by 32.2% (which increases to 32.7% for channel-PCA), and even more pronounced improvements are observed for ORB and BRISK. These results show that this relatively straightforward way for introducing color information is in fact generally applicable to heterogeneous

local descriptors.

We also compare the performance of each binary descriptor when it is reduced to 256 and to 80 dimensions. Reducing ORB and its color variants to 80 dimensions and combining them performs better than reducing them to 256 dimensions (both when applying typical- and channel-PCA). On the other hand, reducing BRISK and its two color variants to 256 dimensions and combining them gave the best results (in combination with channel-PCA).

To analyse the influence of PCA on the vectors of local color descriptors, in Table 3.2 we also compare the channel-PCA of Section 3.1.3 with the typical approach of applying PCA directly on the entire color descriptor vector. In both cases PCA was applied before the VLAD encoding, and in applying channel-PCA we kept the same number of principal components from each color channel (e.g. for RGB-SIFT, which is reduced to $l' = 80$ using typical-PCA, we set $p_1 = p_2 = 27$ for the first two channels and $p_3 = 26$ for the third color channel; $p_1 + p_2 + p_3 = l'$). According to the relative improvement figures reported in the fifth column of Table 3.2 (i.e., for the indexing problem), performing the proposed channel-PCA in most cases improves the concept detection results, compared to the typical-PCA alternative, without introducing any additional computational overhead.

According to Table 3.2, for each local descriptor, the combination with its color variants that presents the highest MXinfAP is the following: SIFTx3 with channel-PCA, SURFx3 with channel-PCA, ORBx3 with typical-PCA, BRISKx3 with channel-PCA. In Table 3.3 we further combine the above to examine how heterogeneous descriptors would work in concert. We can see from the results that the performance increases when pairs of local descriptors (including their color extensions) are combined (i.e., SIFTx3+SURFx3, SIFTx3+ORBx3, SIFTx3+BRISKx3 etc.), which shows a complementarity in the information that the different local descriptors capture. The performance further increases when triplets of different descriptors are employed, with

the best combination being SIFTx3+SURFx3+ORBx3. Combining all four considered local descriptors and their color variants did not show in our experiments to further improve the latter results.

In Table 3.4 we improve selected results of Tables 3.2 and 3.3 by additionally exploiting the literature technique of video tomographs [93] (for simplicity, these tomographs are described using only SIFT and its two color extensions). The results of Table 3.4 indicate that introducing temporal information (through tomographs) can give an additional 7.3% relative improvement to the best results reported in Table 3.3 (MXinfAP increased from 22.9 to 24.57).

Concerning the performance of independent classifiers with respect to the annotation problem, for which results are also presented in Tables 3.1, 3.2, 3.3 and 3.4, similar conclusions can be reached regarding the usefulness ORB and BRISK, and how color information is introduced to SURF, ORB and BRISK. Concerning channel-PCA, in this case it does not seem to affect the system's performance: the differences between classifiers that use the typical-PCA and channel-PCA are marginal. Another important observation is that in all the above tables a significant improvement of the MXinfAP (i.e., of the indexing problem results) does not lead to a correspondingly significant improvement of results on the annotation problem.

3.3.2 Stacking for exploiting label relations

In this subsection, having a complete approach of independently trained concept classifiers we evaluate the refinement of their prediction scores using the proposed stacking architecture, presented in Section 3.2 and other compared methods. The methods of this subsection try to capture information that exists between the presence or absence of pairs or groups of concepts in order to improve the prediction scores of the independently trained classifiers. This is useful for improving existing pre-trained concept-based video annotation systems. However, more advanced methods such as deep networks

Table 3.5: Performance, (MXinfAP (%), MAP@3 (%) and CPU time), for the methods compared on the TRECVID 2013 dataset. The meta-learning feature space for the second layer of the stacking architecture is constructed using detection scores for (i) 346 concepts and (ii) a reduced set of 60 concepts. CPU times refer to mean training (in minutes) for all concepts, and application of the trained second-layer classifiers on one shot of the test set (in milliseconds). Columns (a) and (c) show the results of the second layer classifiers only. Columns (b) and (d) show the results after combining the output of first and second layer classifiers, by means of arithmetic mean. "Baseline" denotes the output of the independent concept classifiers that constitute the first layer of the stacking architecture (i.e. the best classifiers reported in Table 3.4). In parenthesis we show the relative improvement w.r.t. the baseline.

Method	MXinfAP (indexing)		MAP@3 (annotation)		
	2nd layer (a)	1st and 2nd layer combination (b)	2nd layer (c)	1st and 2nd layer combination (d)	Mean Exec. Time Training/Testing (e)
Baseline	24.57	24.57	77.79	77.79	N/A
	(i) Using the output of 346 concepts' classifiers for meta-learning				
DMF [95]	23.97 (-2.4%)	25.38 (+3.3%)	78.71 (+1.2%)	79.12 (+1.7%)	27.62/0.61
BSBRM [110]	24.7 (+0.5%)	24.95 (+1.5%)	79.31 (+2.0%)	79.06 (+1.6%)	1.02/0.08
MCF [122]	24.33 (-1.0%)	24.53 (-0.2%)	76.14 (-2.1%)	77.31 (-0.6%)	1140.98/0.22
CBCFpred [51]	24.32(-1.0%)	24.56 (0%)	78.95 (1.5%)	78.39 (0.8%)	26.84/0.27
CF [45]	23.34 (-5.0%)	25.27 (+2.8%)	78.13 (+0.4%)	78.81 (+1.3%)	55.24/1.22
P-CLR	14.01 (-43.0%)	24.52 (-0.2%)	79.17 (+1.8%)	79.26 (+1.9%)	49.40/9.85
P-LP	**25.23 (+2.7%)**	**25.6 (+4.2%)**	**80.88 (+4.0%)**	79.06 (+1.6%)	549.40/24.93
P-PPT	23.8 (-3.1%)	24.94 (+1.5%)	79.39 (+2.1%)	78.3 (+0.7%)	392.49/0.03
P-MLkNN	19.38 (-21.1%)	24.56 (0.0%)	77.55 (-0.3%)	**79.64 (+2.4%)**	607.40/273.80
	(ii) Using the output of a subset of the 346 concepts' classifiers (60 concepts) for meta-learning				
DMF [95]	24.32 (-1.0%)	25.04 (+1.9%)	79.47 (+2.2%)	79.19 (+1.8%)	2.64/0.30
BSBRM [110]	24.71 (+0.6%)	24.96 (+1.6%)	79.82 (+2.6%)	79.26 (+1.9%)	0.65/0.08
MCF [122]	**24.85 (+1.1%)**	24.74 (+0.7%)	77.84 (+0.1%)	77.88 (+0.1%)	466.69/0.18
CBCFpred [51]	15.66 (-36.3%)	22.41 (-8.8%)	79.58 (+2.3%)	79.01 (+1.6%)	2.42/0.25
CF [45]	24.8 (+0.9%)	25.18 (+2.5%)	79.02 (+1.6%)	79.04 (+1.6%)	5.28/0,60
P-CLR	16.16 (-34.2%)	24.44 (-0.5%)	78.85 (+1.4%)	79.12 (+1.7%)	6.32/5.82
P-LP	23.85 (-2.9%)	**25.28 (+2.9%)**	**80.22 (+3.1%)**	79.04 (+1.6%)	208.9/41.43
P-PPT	24.12 (-1.8%)	24.96 (+1.6%)	79.6 (+2.3%)	78.45 (+0.8%)	90.13/0.31
P-MLkNN	22.21 (-9.6%)	24.94 (+1.5%)	77.68 (-0.1%)	**79.42 (+2.1%)**	167.40/72.54

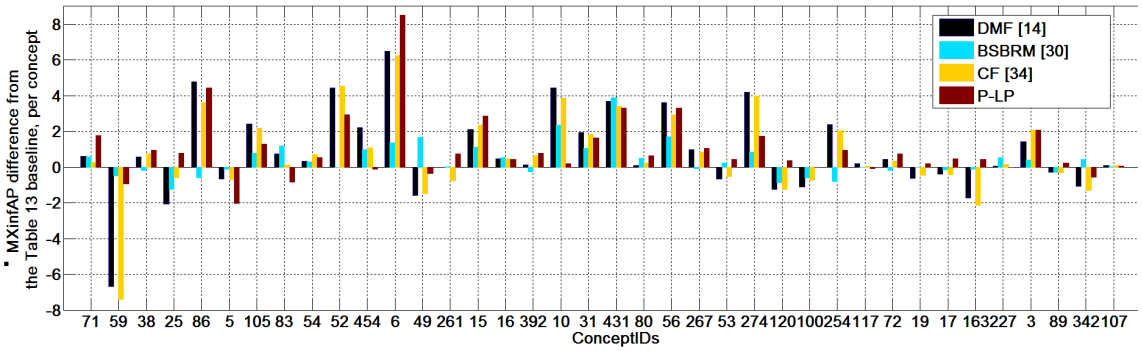

Figure 3.2: Differences of selected second layer method from the baseline per concept with respect to the indexing problem when a meta-learning set of 346 concepts is used. Concepts ordered according to their frequency in the test set (in descending order). Concepts on the far right side of the chart (most infrequent concepts) seem to be the least affected, either positively or negatively, by the second-layer learning.

that train a unique model that predicts all the concept in once and also exploit concept relations are presented in Chapter 5.

In Table 3.5 we report results of the proposed stacking architecture and compare with other methods that exploit label relations. As first layer of the stack we use the best-performing independent classifiers of Table 3.4 (i.e., the last line of Table 3.4, fusing keyframes and tomographs). We start the analysis with the upper part of Table 3.5, where we used the output of such classifiers for 346 concepts.

In relation to the indexing problem (Table 3.5:(a),(b)), we observe that the second layer concept classifiers alone do not perform so well; in many cases they are not able to outperform the independent first layer classifiers (baseline). However, when the concept classifiers of the two layers are combined (Table 3.5:(b)), i.e. the second layer concept detection scores are averaged with the initial scores of the first layer, the accuracy of all the methods is improved. More specifically, P-LP outperforms all the compared methods, 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. PPT models label relations on a similar manner, however, it prunes away label sets that occur less times than a threshold. Modelling different kinds of relations (e.g. by using ML-kNN, CLR) exhibits moderate to low performance. To investigate the statistical significance of the difference of each method from the baseline we used a two-tailed pair-wise sign test [12] and found that only differences between P-LP and the baseline are significant (at 1% significance level).

In relation to the annotation problem (Table 3.5:(c),(d)) the results show again the effectiveness of the proposed stacking architecture when combined with P-LP, reaching a MAP@3 of 80.88 and improving the baseline results by 4.0%. In this problem also P-MLkNN presents good results, reaching top performance when combined with the classifiers of the first layer. Also, for P-LP the relative boost of MXinfAP with respect

to the baseline is of the same order of magnitude as the relative boost of MAP@3 (which, as we recall from Section 3.3.1, is not the case when examining independent concept classifiers).

To assess the influence of the number of input classifiers in the second layer we also performed experiments where the predictions of a reduced set of 60 concept classifiers (the 60 concepts that NIST pre-selected for the TRECVID SIN 2013 task [82]) is used for constructing the meta-level dataset (Table: 3.5:(ii)). Results show that usually a larger input space (classifiers for 346 concepts instead of 60) is better, increasing both MXinfAP and MAP@3. This outcome was expected as more classifiers/concepts means more opportunity to find relations between concepts.

To investigate the importance of stacking-based methods separately for each concept, we closely examine the four best-performing methods of column (b) in Table 3.5:(i). Fig. 3.2 shows the difference of each method from the baseline. We observe that the majority of concepts exhibit improved results when any of the second-layer methods is used. The most concepts benefit from the use of P-LP (29 of the 38 concepts), while the number of concepts that benefit from DMF, BSBRM and CF, compared to the baseline, is 25, 21, and 25 respectively. One concept (6:animal) consistently presents a great improvement when label relations are considered, while there are 3 concepts (5:anchorperson, 59:hand and 100:running) that are negatively affected regardless of the employed stacking method.

Finally, we take a look at the execution times that each method requires (Table 3.5:(e)). One could argue that the proposed architecture that uses multi-label learning methods requires considerably more time than the typical BR-stacking one. However, we should note here that extracting one model vector from one video shot, using the first-layer classifiers for 346 concepts requires approximately 3.2 minutes in our experiments, which is about three orders of magnitude slower than the slowest of the second-layer methods. As a result of the inevitable computational complexity of the

first layer of the stack, the execution time differences between all the second-layer methods that are reported in Table 3.5 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. In some preliminary experiments we performed PCA in order to reduce the feature space and then applied such multi-label learning algorithms directly to the reduced low-level visual features. However, this performed approximately two percentage points worse compared to the proposed two-layer architecture.

Specifically, the computational complexity of BR, CLR, LP and PPT when used in a single-layer architecture depends on the complexity of the base classifier, in our case the LR, and on the parameters of the learning problem. Given that the training dataset used here consists of more than 500.000 training examples, and each training example (video shot) is represented by a 4000-element low-level feature vector for each visual descriptor, the BR algorithm, which is the simplest one, would build N models for N concepts; CLR, the next least complex algorithm, would build N BR-models and $N * (N-1)/2$ one-against-one models. LP and PPT, would build a multi-class model, with the number of classes being equal to the number of distinct label sets in the training set (after pruning, in the case of PPT); this is in order of N^2 in our dataset. Finally ML-kNN would compare each training example with all other (500.000) available examples; in all these cases, the 4000-element low-level feature vectors would be employed. Taking into consideration the dimensionality of these 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 proposed stacking architecture. In addition to this, typically, multi-label learning algorithms require the full training set to be loaded on memory at once (e.g. [109]), which would be practically unfeasible in a single-layer setting, given the dimensionality of the low-level feature vectors. We conclude that the two major

obstacles of using multi-label classification algorithms in a one-layer architecture are the high memory space and computation time requirements, and this finding further stresses the merit of our proposed multi-label stacking architecture.

3.4 Conclusion

In this chapter we first dealt with video frame description and representation for concept-based video annotation. We showed that two binary local descriptor (ORB, BRISK) can perform reasonably well compared to their state-of-the-art non-binary counterparts in the video semantic concept detection task. We subsequently showed that a methodology previously used for defining two color variants of SIFT is a generic one that is also applicable to other binary and non-binary local descriptors. We also proposed a different way of employing PCA for dimensionality reduction of color descriptors that are used in combination with VLAD (channel-PCA). A second major focus of this chapter was to take advantage of concept relations information for building better classifiers. For this we proposed an alternative way of employing the stacking architecture, using multi-label learning algorithms in the last level of the stack. We showed that using the proposed architecture in combination with the Label Powerset (LP) method represents an attractive solution. Furthermore, in this chapter we compared video concept annotation approaches on two different experimental settings: video indexing and annotation. In relation to this comparison, the message that this chapter aims to pass is that the usual evaluation of video annotation results in a retrieval-based problem setting is not sufficient for assessing the performance of concept classifiers in the context of the annotation problem, and we experimentally underline the importance of reporting evaluation results in both these directions. In addition, for building a competitive system the most important is to use a combination of many different types of visual features for the same concept i.e., binary, non-binary, grayscale, color-based, and then further refine the output of such independently trained classifiers with a stacking-based approach that exploits concept relations in the second

layer. The dimension of local descriptors and the way that PCA will be applied to color-extensions did not show important properties that someone should consider in order to build a competitive system, so when building a new system one could simply use what is already recommended in the bibliography.

Our video annotation architectures has been evaluated only on hand-crafted features (i.e., local descriptors). However, introducing also DCNN-based features could further improve the overall results of our system. With the goal of further increasing concept annotation accuracy of our system, we are going to investigate the use of DCNN-based features in combination with hand-crafted ones by developing more advanced classifier combination techniques on the following chapter.

Chapter 4

Cascades of Pre-trained Classifiers

Contents

In this chapter we present a cascade architecture that combines many pre-trained concept classifiers on different visual features for the same concept. 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 video concept annotation accuracy. Motivated by this, we present a cascade architecture (Fig. 4.1) suitable for combining many base classifiers that have been trained for the same concept.

Our contributions are:

- A cascade architecture that combines many base classifiers in order to perform fast and accurate video concept annotation.

- An algorithm that uses a fixed ordering of cascade stages and a simple thresholding strategy in order to instantiate the proposed cascade.
- A more advanced algorithm that sets the ordering of cascade stages (i.e. the ordering of stage classifiers) and dynamically assigns thresholds to each stage in order to instantiate the proposed cascade.

The chapter is organized as follows: Section 4.1 presents the proposed cascade architecture, Section 4.2 introduces a simple algorithm that uses fixed stage ordering and a simple thresholding strategy. Section 4.3 presents a more advanced algorithm that sets the ordering of cascade stages and dynamically assigns thresholds to each stage. Section 4.4 presents the experimental results and, finally, Section 4.5 presents conclusions.

4.1 Cascade architecture overview

Figure 4.1 shows the proposed cascade architecture suitable for combining many base classifiers that have been trained for the same concept [68]. 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 two different algorithm in order to instantiate

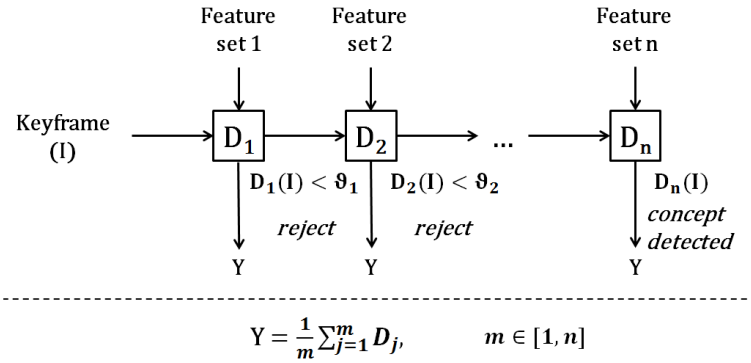

Figure 4.1: Block diagram of a cascade architecture for one concept.

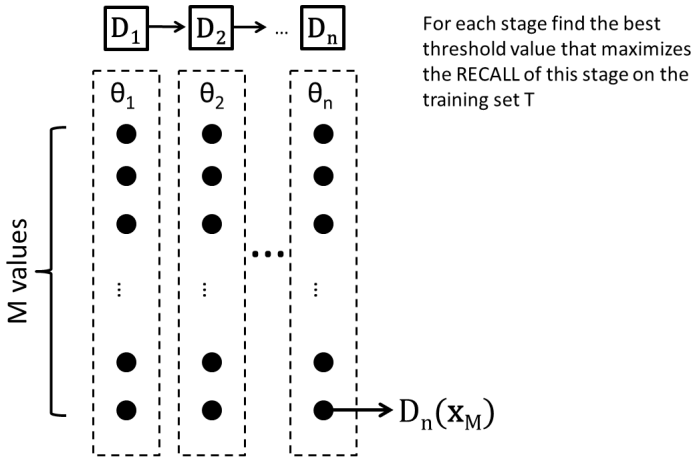

Figure 4.2: Threshold assignment of the proposed cascade architecture (Fig. 4.1) with fixed ordering of stages.

the above cascade (i.e., set the ordering of cascade stages and assign thresholds to each stage).

4.2 Simple cascades with fixed stage ordering

In this section we present an algorithm that uses a fixed ordering of cascade stages in terms of classifier accuracy and a simple thresholding strategy that optimizes recall in order to instantiate the proposed cascade of Fig. 4.1. Specifically, less accurate classifiers are introduced on early stages (e.g. binary descriptors), while more accurate classifiers are used on later stages (e.g. non-binary, DCNN-based). Subsequently, the

threshold for each stage is chosen independent of the other stages.

Algorithm 1 presents the training and the classification process of this cascade instantiation and Fig. 4.2 shows an illustrative example of the thresholding strategy that is used to instantiate the thresholds for each stage. During the training phase, the learning goal of the n-stage cascade is the calculation of n thresholds θ_j, one for each stage ($j = 1 : n$). During the classification phase every next stage is triggered only if the stage output score is higher than the rejection threshold; 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

Algorithm 1 Algorithm for the training and classification process of a cascade with fixed stages

Training
Input: Training set T= $\{x_i, y_i\}_{i=1}^{M}$, $y_i \in \{\pm 1\}$; n trained classifiers $D = \{D_1, D_2, ..., D_n\}$ ordered in terms of classifier accuracy
Output: A vector of thresholds $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_n]^\top$
for $j = 1$ to n **do**
    1. Find the threshold θ_j, that optimizes the recall of D_j on T
end for
Classification
Input: Unlabeled sample = x', n-stage trained cascade of $D_1, ..., D_n$ classifiers, and $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_n]^\top$ 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. $Y = \frac{Y}{j-1}$

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 performance 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.

4.3 Ordering of visual descriptors in a classifier cascade

In this section we present a more 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. 4.1. In this case 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 as presented in Section 4.2.

4.3.1 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

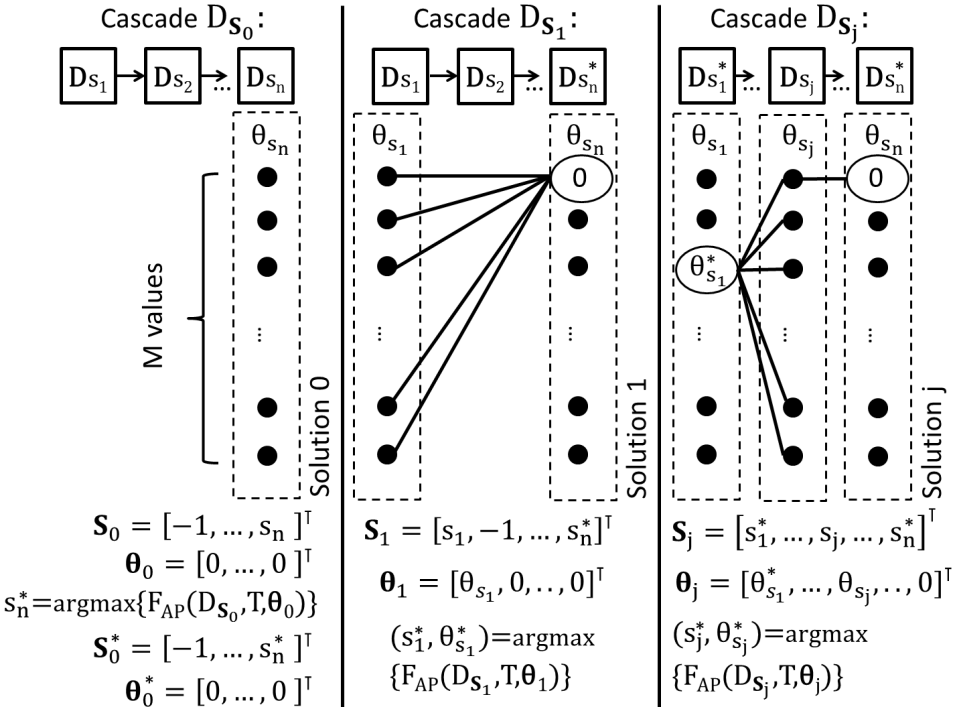

Figure 4.3: Threshold assignment and stage ordering of the proposed cascade architecture (Fig. 4.1).

for the given concept ($\boldsymbol{x_i}$ being the feature vectors, e.g., SIFT descriptors aggregated using the VLAD encoding, 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:

$$(\mathbf{S}^\star, \boldsymbol{\theta}^\star) = \underset{(\mathbf{S}, \boldsymbol{\theta})}{\text{argmax}}\{F(D_{\mathbf{S}}, T, \boldsymbol{\theta})\}, \tag{4.1}$$

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 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.

4.3.2 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 2 presents the proposed greedy search algorithm that instantiates the proposed cascade (Fig. 4.1), i.e., sets the ordering of cascade stages and assign thresholds to each stage. Fig. 4.3 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. Let as remind that each number represents the index of a classifier from D and appears at most once. 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. 2: 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 (4.1),

$$s_n^\star = \underset{s_n}{\text{argmax}}\{F_{AP}(D_{\mathbf{S}_0}, T, \boldsymbol{\theta}_0)\}. \tag{4.2}$$

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 (4.1). In simpler words, this is the optimal solution if we had a single-stage cascade $l = 1$, so what the algorithm chooses her is the best performing classifier from D. The selected classifier in this first step of our algorithm occupies the last position of the cascade. Then we continue finding the best pair of stage-classifier and threshold for the first to the last-1 position of the cascade. In iteration j, our algorithm (Alg. 2: 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 (4.1):

$$(s_j^\star, \theta_{s_j}^\star) = \underset{(s_j, \theta_{s_j})}{\text{argmax}}\{F_{AP}(D_{\mathbf{S}_j}, T, \boldsymbol{\theta}_j)\}. \tag{4.3}$$

The algorithm finds the pair of (s_j, θ_{s_j}) that optimizes (4.1). 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. So, in simpler words, in any intermediate iteration of our algorithm we add a new stage classifier and also its

Algorithm 2 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. $s^\star_n = \text{argmax}_{s_n}\{F_{AP}(D_{\mathbf{S}_0}, T, \boldsymbol{\theta}_0)\}$ (4.1),
$\mathbf{S}_0 = [-1, -1, ..., s_n]^\top$, $\boldsymbol{\theta}_0 = [0, 0, .., 0]^\top$
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. $(s^\star_j, \theta^\star_{s_j}) = \text{argmax}_{(s_j, \theta_{s_j})}\{F_{AP}(D_{\mathbf{S}_j}, T, \boldsymbol{\theta}_j)\}$ (4.1),
  $\mathbf{S}_j = [..., s_j, -1, ..., s^\star_n]^\top$, $\boldsymbol{\theta}_j = [..., \theta_{s_j}, 0, ..., \theta^\star_{s_n}]^\top$
  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

associated stage threshold by calculating which of the remaining stage classifiers, those that have not been considered as part of the current cascade, optimizes the cascade up to this iteration Finally, the optimal sequence $\mathbf{S}^\star$ equals to

$$\mathbf{S}^\star = \underset{\mathbf{S} \in \{\mathbf{S}^\star_0, \mathbf{S}^\star_1, ..., \mathbf{S}^\star_{n-1}\}}{\text{argmax}} \{F_{AP}(D_{\mathbf{S}}, T, \boldsymbol{\theta})\}, \tag{4.4}$$

which is the sequence that optimizes (4.1) within all the iterations of the algorithm (Alg. 2: States 6-7). The optimal threshold vector $\boldsymbol{\theta}^\star$ is the vector connected to the optimal sequence $\mathbf{S}^\star$. The algorithm terminates when the optimal stage classifier for each position in $\mathbf{S}$ has been selected and consequently the optimal threshold per stage has also been selected. In other words, when all of the stage classifiers have been either used or omitted in each position of $\mathbf{S}$ and not any stage classifier has been remained to be evaluated. 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.

4.4 Experimental study

Our experiments were performed on the TRECVID 2013 Semantic Indexing (SIN) dataset [82] (for more details refer to Section 2.5). We evaluated our system on the test set using the 38 concepts that were evaluated as part of the TRECVID 2013 SIN Task [82]. 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 [68]. More specifically, we extracted nine local descriptors, presented in Table 4.1. These are the best performing local descriptors of Section 3.3.1. 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 [94], ii) the 22-layer GoogLeNet network provided by [106], and iii) the 8-layer CaffeNet network described in [57]. 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. These base classifiers are denoted as $B_1, B_2, ..., B_{f_j}$ in our cascade presented in Section 4.1 and a stage classifier D_j can either combine 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 all cases, the final step of concept-based video annotation was to refine the calculated detection scores by employing the re-ranking method proposed in [91].

We compared the proposed cascade (Fig. 4.1) instantiated with the dynamic algorithm of Section 4.3 with five different ensemble combination approaches: i) Late-fusion with arithmetic mean [102]. ii) The ensemble pruning method proposed by [93]. iii) The instantiation of the proposed cascade with the simple algorithm that assumes fixed ordering of the stages in terms of classifier accuracy, presented in Section 4.2. 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 [17]. In contrast to [17] 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 [7]). 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 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.

Table 4.1: 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 4.2: Performance (MXinfAP, %) for different classifier combination approaches.

		M1	M2	M3	M4	M5	M6
RunID	Stage classifiers	Late-fusion [102]	Ensemble pruning [93]	Cascade-thresholding (Section 4.2)	Cascade-dynamic [17]	AdaBoost [7]	Cascade-proposed (Section 4.3)
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**

4.4.1 Cascade architecture based on local and DCNN-based descriptors

Tables 4.1 and 4.2 present the results of our experiments in terms of MXinfAP [130]. Table 4.1 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 4.2: 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. Let us remind here that in our proposed cascade presented in Section 4.1 base classifiers are denoted as $B_1, B_2, ..., B_{f_j}$ and stage classifiers are denoted as $D_1, D_2, ..., D_n$. For the late fusion

methods (Table 4.2: M1, M2) and the boosting-based method (Table 4.2: M5), the corresponding base classifiers per line of Table 4.1 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 4.2 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 annotation. 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.

Computational Complexity

We continue the analysis of our results with respect to the computational complexity of the different methods compared in Table 4.2 during the training and classification

Table 4.3: 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 [102]	-	-
M2	Ensemble pruning [93]	$(2^n - 1)M$	-
M3	Cascade-thresholding (Section 4.2)	$\sum_{j=0}^{n} M_j,\ M_j \subseteq Mj - 1$	-
M4	Cascade-dynamic [17]	$(n-2)Q^2$	-
M5	AdaBoost [7]	$M(n(n+1)/2)$	$n(n+1)/2$
M6	Cascade-proposed (Section 4.3)	$Q(n(n+1)/2)$	-

phase. Table 4.3 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 [102], which builds n models (one for each set of features), is the simplest one. Cascade-thresholding (Section 4.2) follows, which evaluates n cascade stages in order to calculate the appropriate thresholds per stage. Cascade-dynamic [17] 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 [93] 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 4.4 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 4.4: R4-M1) and the cascade-dynamic alternative (Table 4.4: R4-M4), respectively, which

Table 4.4: 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 [102]	Ensemble pruning [93]	Cascade-thresholding (Section 4.2)	Cascade-dynamic [17]	AdaBoost [7]	Cascade-proposed (Section 4.3)
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**

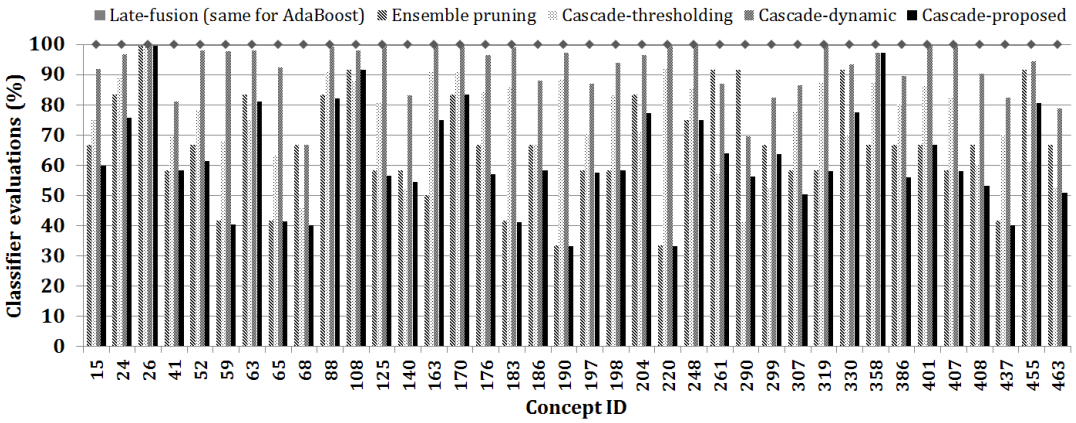

Figure 4.4: Relative amount of classifier evaluations (%) per concept for R4 of Table 4.4.

are the two most accurate methods after the proposed-cascade. Figure 4.4 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.

To sum up, according to Tables 4.2 and 4.4, the three best-performing methods are the proposed-cascade, the late fusion [102] and the cascade-dynamic [17]. With re-

spect 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 4.4 for the proposed cascade very important. For example, the last configuration of Table 4.4 (R4), would take approximately 10 hours for classifying the TRECVID SIN 2013 test dataset when using late fusion, and approximately 6.2 hours when the proposed cascade is used. This dataset consists of 200 hours of video, 2420 videos that have been decoded into 112677 keyframes, and also 38 concepts. So, this means that the proposed cascade requires approximately 0.20 seconds/keyframe and 9.4 seconds/video. This is many times faster than real time.

4.5 Conclusions

In this chapter we presented a cascade architecture (Fig. 4.1) that can be served with many different keyframe representations and we proposed two different algorithm in order to instantiate it (i.e., set the ordering of cascade stages and assign thresholds to each stage). Our first algorithm is a simple approach that uses fixed stage ordering and static threshold assignment (Section 4.2). The stage classifiers are ranked from the less accurate to the most accurate based on their retrieval accuracy in a validation

set and each stage threshold is assigned to that value that lets all the keyframes that depict the target concept to pass to the next stage (i.e., optimizing recall per stage). In contrast, our second algorithm (Section 4.3) can optimally prune, order and combine the cascade stages in order to perform fast and accurate video concept annotation. This more advanced algorithm sets the ordering of cascade stages (i.e. the ordering of stage classifiers) and assigns thresholds to each stage towards the optimization of the complete cascade and not the optimization of each stage separately from the other stages as our first simple approach does. For all of the compared methods we observe that combining many classifiers trained on different features for the same concept is able to improve concept annotation accuracy. The proposed more advanced cascade-based approach, presented small improvements in terms of MXinfAP that are accompanied by considerable improvements in computational complexity. As a result, the proposed cascade could be useful for applications that require small detection times, however, it does not really provide any significant improvement compared to the simple late fusion scheme in terms of accuracy.

Another observation is that DCNN-based features significantly outperform the hand-crafted local descriptors and this is clearly in accordance with the fact that local descriptors have been completely replaced by DCNN-based features the last few years. According to Table4.1, classifiers trained on DCNN-based features perform better than the combinations of SIFT, SURF and ORB with their color variants. (With only one exception for SIFTx3 that slightly outperforms CAFFE). It is also observed that each of the base classifiers of these groups (e.g. RGB-SIFT) is rather weak, achieving MXinfAP that ranges from 12.18 to 16.75 (depending on which descriptor is used). In the next chapter we will focus on ways that existing deep architectures can be improved, in order to achieve higher concept-based video annotation accuracy.

Chapter 5

Multi-task Learning and Structured Outputs for DCNNs

Contents

In this chapter we propose a DCNN (Deep Convolutional Neural Network) architecture that addresses the problem of video/image concept annotation by exploiting concept relations at two different levels. At the first level, we build on ideas from multi-task learning, and propose an approach to learn concept-specific representations that are sparse, linear combinations of representations of latent concepts. By enforcing the sharing of the latent concept representations, we exploit the implicit relations between the target concepts. At a second level, we build on ideas from structured output learning, and propose the introduction, at training time, of a new cost term that explicitly models the correlations between the concepts. By doing so, we explicitly model the structure in the output space (i.e., the concept labels). Both of the above are implemented using standard convolutional layers and are incorporated in a single DCNN

architecture that can then be trained end-to-end with standard back-propagation [1].

As discussed in chapters 2 and 4, the typical pipeline of concept-based video annotation systems that consists of hand-crafted feature extraction and training of concept classifiers separately for each concept, has been replaced by deep learning architectures. The dominant approach for doing this is training DCNNs in whose architectures the concepts share features up to the very last layer, and then branch off to T different classification branches (using typically one layer), where T is the number of concepts [83]. However, in this way, the implicit feature-level relations between concepts, e.g. the way in which concepts such as a *car* and *motorcycle* share lower-level features modeling things like their wheels, are not directly considered. Also, in such architectures, the relations or interdependencies of the concepts at a semantic level, i.e. the fact that two specific concepts may often appear together or, inversely, the presence of the one may exclude the other, are also not directly taken into consideration. While some methods have been proposed for exploiting in a more elaborate way one of these two different types of concept relations, there is no single method that jointly exploits visual- and semantic-level concept relations in a unified DCNN architecture.

In this chapter we propose a DCNN architecture that captures therefore both implicit and explicit concept relations, i.e., both visual-level and semantic-level concept relations. First, implicit concept relations are modeled in a DCNN architecture that learns T concept-specific feature vectors that are themselves linear combinations of $k < T$ latent concept feature vectors. In this way, in the shared representations (i.e., the latent concepts feature vectors), higher-level concepts may share visual features - for example, concepts such as *car*, *motorcycle*, and *airplane* may share features encoding the *wheels* in their depiction [47]. This bears similarities to multi-task learning (MTL) schemes, like GO-MTL [58] and the two-sided network proposed in [69] that factorize the 2D weight matrix that encodes concept specific features. However, in

[1] Source code available at: https://github.com/markatopoulou/fvmtl-ccelc

contrast to GO-MTL [58], in our work the factorization is achieved in two standard convolutional network layers, and in contrast to [69], our network does not only verify whether a certain concept that is given as input to the one side of the network is present in the video/image that is given as input to the other side. Instead, it provides scores for all concepts in the output, similar to classical multi-label DCNNs. Second, explicit concept relations are introduced by a new cost term, implemented using a set of standard CNN layers that penalize differences between the matrix encoding the correlations among the ground-truth labels of the concepts, and the correlations between the concept label predictions of our network. In this way, we introduce constraints on the structure of the output space by utilizing the label correlation matrix - this will explicitly capture, for example, the fact that *daytime* and *nighttime* are negatively correlated concepts.

The main contributions of this chapter are:

- We propose a new approach, named FV-MTL (Shared Latent Feature Vectors using Multi-task Learning), that learns shared feature vectors corresponding to latent concepts, and expresses the concept-specific feature vectors as their sparse linear combination. This corresponds to a factorization of the weight matrix and is implemented using a set of standard CNN layers.

- We propose a new cost function, named CCE-LC (Cost Sigmoid Cross-entropy with Label Constraint), which exploits a form of semantic relations, namely correlations between pairs of concepts, in order to predict structured outputs. This is again implemented using a set of standard CNN layers.

- We incorporate both of the above in a single DCNN architecture that is trained end-to-end to solve the video/image concept annotation problem. We evaluate the trained DCNN both as a standalone classifier, where the direct output of the complete network is evaluated, and as feature generator, where SVM classifiers

are trained on DCNN-generated features. In both cases we obtain state of the art results in publicly available datasets and show the benefits of each of the proposed contributions both alone and in tandem.

The remainder of the chapter is organized as follows: Section 5.1 presents the proposed FV-MTL approach, the proposed CCE-LC cost function, and also the way that FV-MTL can be used jointly with the CCE-LC cost in a unified DCNN architecture. Section 5.2 reports our experiments, results, and comparisons, and finally, Section 5.3 summarizes our main conclusions.

5.1 Multi-task learning and structured output predictions in deep networks

5.1.1 Problem formulation and method overview

We consider a set of concepts $C = \{c_1, c_2, ..., c_T\}$ and a multi-label training set $\mathcal{P} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) : \boldsymbol{x}_i \in \mathcal{X}, \boldsymbol{y}_i \in \{0,1\}^{T\times 1}, i = 1...N\}$, where $\boldsymbol{x}_i$ is a 3 channel keyframe/image, and $\boldsymbol{y}_i$ is its ground-truth annotation. A video/image concept annotation system learns T supervised learning tasks, one for each target concept c_j. More specifically, it learns a real-valued function $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{Y} = [0,1]^{T\times N}$ could then be binarized (e.g., thresholded) in order to provide a hard classification result, if needed.

We propose a DCNN architecture that exploits both implicit visual-level and explicit semantic-level concept relations for video/image concept annotation by building on ideas from MTL and structured output prediction, respectively. In Fig. 5.1 (i) we illustrate a typical $(\Pi + 1)$-layer DCNN architecture, e.g., ResNet, that shares all the layers but the last one (steps (a),(b))) [94, 46]; in Fig. 5.1 (ii) we illustrate how the typical DCNN architecture of Fig. 5.1 (i) is extended by one FC extension layer, which was shown to outperform the typical DCNN architecture when used in transfer learning problems [83] (steps (c)-(e)); and finally, in Fig. 5.1 (iii) we present the proposed

Table 5.1: Definition of the symbols

Symbols	Definitions
$\boldsymbol{x}$	A keyframe/image
$\boldsymbol{y}$	A vector containing the ground-truth concept annotations for a keyframe/image
N	The number of training keyframes/images
c	A concept
T	The number of concepts, i.e., number of tasks
i	Keyframe/image index, i.e., $i = 1...N$
j	Concept/task index, i.e., $j = 1...T$
$\hat{\boldsymbol{y}}$	A vector containing the concept prediction scores for a keyframe/image
$\boldsymbol{L_x}$	Latent concept feature vectors of a keyframe/image
$\boldsymbol{S}$	Concept-specific weight matrix, each column corresponds to a task containing the coefficients of the linear combination with $\boldsymbol{L_x}$
$\boldsymbol{L_xS}$	Concept-specific feature vectors incorporating information from k latent concept representations
$\boldsymbol{U}$	Concept-specific parameter matrix for the final classification
k	The number of latent tasks
$\sigma(.)$	The sigmoid function
$\boldsymbol{\Phi}$	The concept correlation matrix calculated from the ground-truth annotated training set
$\boldsymbol{m}$	A cost vector utilized for data balancing
β	Regularization parameter
$\boldsymbol{z}$	Normalization factor vector

DCNN architecture (steps (f)-(k)). In the next subsections we first introduce the new FV-MTL approach for learning implicit visual-level concept relations; this is done using the network layers as shown in Fig. 5.1 in steps (f) to (i). Second, we introduce the new CCE-LC cost function that learns explicit semantic-level concept relations, which is done in step (k). CCE-LC predicts structured outputs by exploiting concept correlations that we can acquire from a training dataset's ground-truth annotations. The source code of our method can be found at https://github.com/markatopoulou/fvmtl-ccelc.

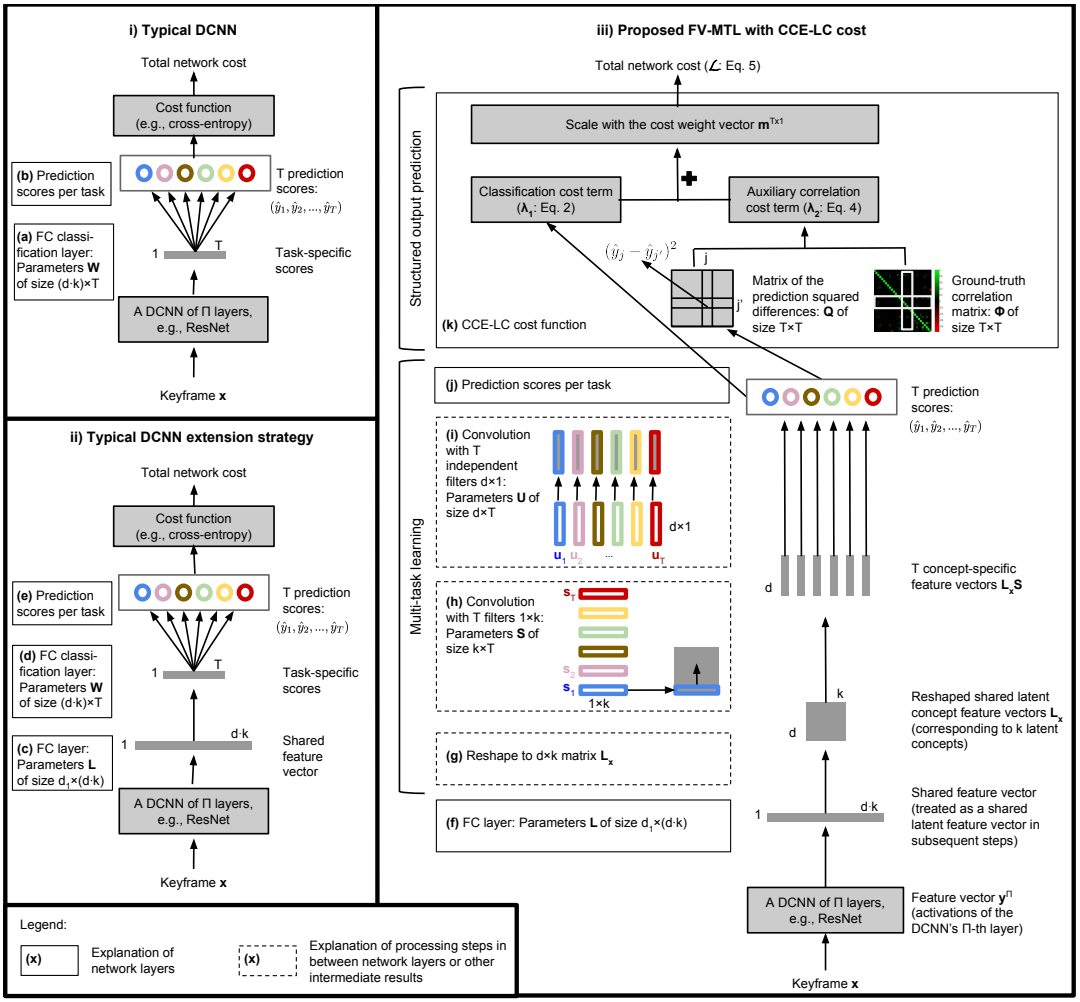

Figure 5.1: Sub-figure (i) presents the typical DCNN architecture (e.g., ResNet [46]). Sharing all layers but the last one. Sub-figure (ii) presents the typical DCNN extension strategy proposed in [83]. A shared fully-connected layer, a.k.a the extension layer, and a concept-specific classification layer (a second fully-connected classification layer that maps a common feature representation to concept categories, independently for each concept), are placed on the top of a typical DCNN architecture (e.g., ResNet [46]). Sub-figure (iii) presents the proposed FV-MTL with CCE-LC cost function: FV-MTL is modeled as a stack of standard CNN layers, on the top of which the CCE-LC cost function is placed, which consist of two terms i) the cross-entropy cost term and ii) the auxiliary correlation cost term that integrates structural information. CCE-LC is also modeled as a stack of standard CNN layers.

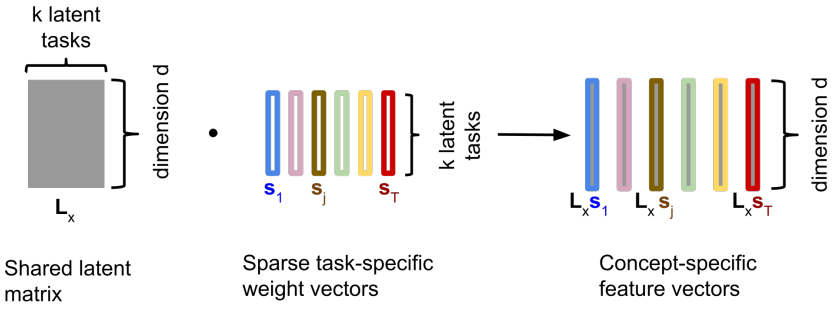

Figure 5.2: Shared latent feature vectors using multi-task learning (FV-MTL).

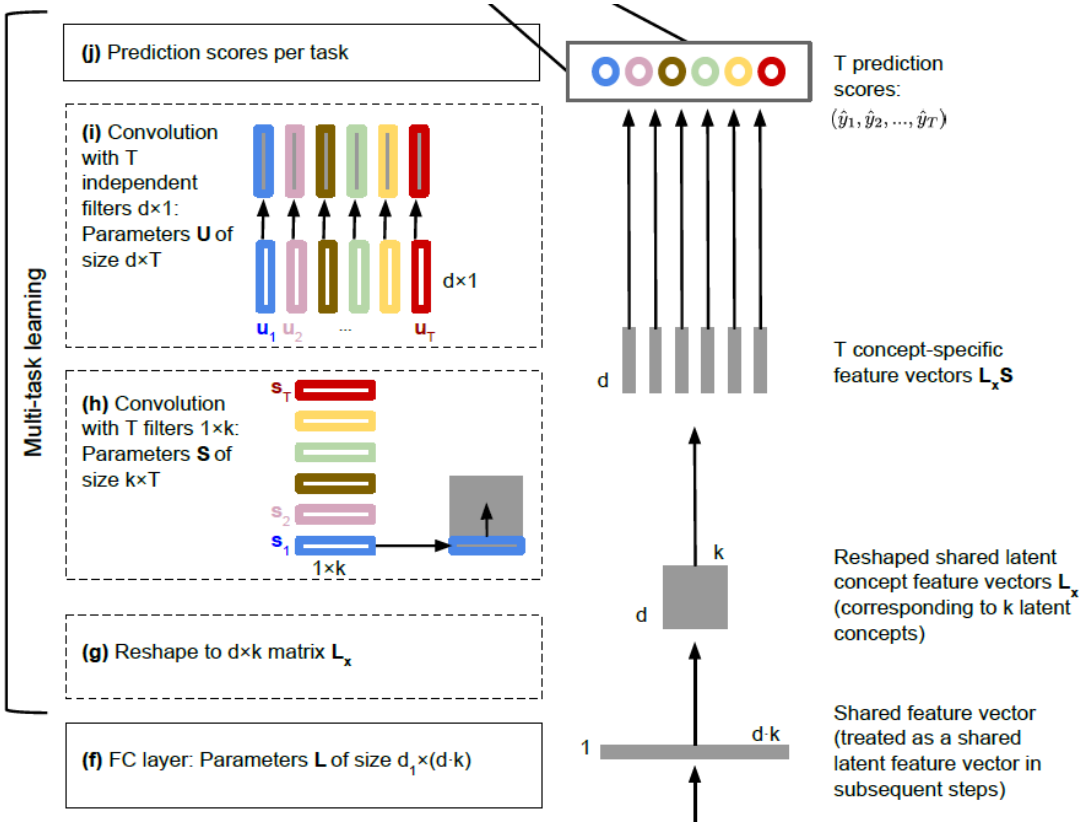

Figure 5.3: MTL part of the proposed FV-MTL with CCE-LC cost function. (Part of Fig. 5.1 (iii))

5.1.2 Shared latent feature vectors using multi-task learning (FV-MTL)

In our approach, similarly to GO-MTL [58], we assume that the parameter vectors of the tasks that present visual-level concept relations (i.e., defined in GO-MTL as be-

longing to the same group) lie in a low-dimensional subspace, thus sharing information; and, at the same time, dissimilar tasks (i.e., belonging to different groups) may also partially overlap by having one or more bases in common. Allowing the sharing also between dissimilar tasks is more natural than creating disjoint groups of task models. In order to do so, see Fig. 5.2, we learn T concept-specific feature vectors that are linear combinations of a small number of latent concept feature vectors that are themselves learned as well. Specifically, our approach uses a shared latent feature vector $\boldsymbol{L_x} \in \mathbb{R}^{d \times k}$ for all task models, where the columns of $\boldsymbol{L_x}$ correspond to d-dimensional feature representations of k latent tasks; and produces T different concept-specific feature vectors $\boldsymbol{L_x s}_j$, for $j = 1...T$, where each of them incorporates information from relevant latent tasks, with $\boldsymbol{s}_j \in \mathbb{R}^{k \times 1}$ being a task-specific weight vector that contains the coefficients of the linear combination. Each linear combination is assumed to be sparse in $\boldsymbol{L_x}$, i.e, $\boldsymbol{s}_j$'s are sparse vectors. In this way we assume that there exist a small number of latent basis tasks and each concept-specific feature vector is a linear combination of them. The overlap in the sparsity patterns of any two tasks, (i.e., how much overlap is observed between two different task-specific weight vectors $\boldsymbol{s}_j$ and $\boldsymbol{s}_{j'}$, where $j \neq j'$) controls the amount of sharing between them.

The above can be implemented in a DCNN architecture by using the network layers depicted in Fig. 5.1 in steps (f) to (i). This part of our architecture is also presented in Fig. 5.3 in order to make it easier to the reader to follow this MTL part of our approach. Specifically, an input training-set keyframe is processed by a typical DCNN architecture (e.g., ResNet) and a fully-connected layer, to produce a shared representation of the keyframe across all of the tasks (Fig. 5.1, 5.3: step (f); this is the same as step (c) in the typical DCNN extension architecture). Subsequently, the output of the fully-connected layer is reshaped to the matrix $\boldsymbol{L_x}$ (Fig. 5.1, 5.3: step (g)). Consequently, the reshaped layer outputs k feature vectors that correspond to k latent concepts. Those representations are shared between the T concepts. The subsequent layers calculate T concept-specific feature vectors, where T is the number of the con-

cepts we are interested in detecting. Each of those feature vectors is a combination of k latent concept feature vectors, with coefficients that are specific to the concept in question. This is implemented as a 1D convolutional layer on the k feature masks - in the case that the 1D convolutional layer implements a linear transform, i.e., we do not use a non-linear activation function, then these two layers implement a feature extraction scheme with a bilinear factorization of the weight matrix (Fig. 5.1, 5.3 step (h)). Once T feature vectors are extracted, then an additional layer (Fig. 5.1, 5.3: step (i)) transforms each of the T feature vectors into T concept annotation scores, one for each of the concepts that we are set to recognize (Fig. 5.1, 5.3: step (j)). The above process leads to a *soft* feature sharing, because the latent concept feature vectors adjust how much information and across which tasks should be shared. By contrast, both the typical DCNN and the DCNN extension architecture of [83] output a single feature vector (Fig. 5.1: step (a) and (d), respectively) that is shared across all of the target concepts and it is subsequently *hard* translated into concept annotation scores independently for each concept (Fig. 5.1: step (b) and (e), respectively), as was also discussed in Section 2.3.

Formally, the predicted score for the j-th task (concept) and the i-th datapoint (keyframe/image) is given by:

$$\hat{y}_{i,j} = \text{diag}(\boldsymbol{u}_j^\top (\boldsymbol{L}_{\boldsymbol{x}_i} \boldsymbol{s}_j)), \tag{5.1}$$

where $\boldsymbol{L}_{\boldsymbol{x}_i}$ is the output of the last fully-connected layer of the right part of Fig. 5.1, 5.3 (see step (f)), after reshaping the calculated vector of dimension $1 \times (d \cdot k)$, in order to have a matrix of d rows and k columns (Fig. 5.1, 5.3: step (g)). Specifically, $\boldsymbol{L}_{\boldsymbol{x}_i} = \text{reshape}(\alpha(\boldsymbol{L}' \boldsymbol{y}_i^{(\Pi)} + \boldsymbol{b}))$, where $\boldsymbol{L} \in \mathbb{R}^{d_1 \times d \cdot k}$ the parameters of the last fully-connected layer, $\boldsymbol{y}_i^{(\Pi)} \in \mathbb{R}^{d_1 \times 1}$ the output of the previous layer, and α the layer's activation functions, e.g. the ReLU. $G = \{g^{(\pi)}\}_{\pi=1}^{\Pi}$ is the set of network parameters for the first Π layers. $\boldsymbol{s}_j, \boldsymbol{u}_j$ are the j-th columns of the parameter matrices $\boldsymbol{S} \in \mathbb{R}^{k \times T}$ and $\boldsymbol{U} \in \mathbb{R}^{d \times T}$, respectively. Each $\boldsymbol{s}_j$ contains a task-specific weight vector of the

coefficients of the linear combination with the shared latent feature vector $\boldsymbol{L}_{\boldsymbol{x}_i}$. This linear combination indicates for each concept which latent tasks describe it. Each $\boldsymbol{u}_j$ contains a concept-specific weight vector that transforms the concept-specific feature vectors $\boldsymbol{L}_{\boldsymbol{x}_i}\boldsymbol{S}$ to concept scores.

Similarly to other DCNN works, we optimize the sigmoid cross entropy between the predicted and the ground truth labels that is formed as:

$$\lambda_{1_{i,j}} = y_{i,j} log\sigma(\hat{y}_{i,j}) + (1 - y_{i,j}) log(1 - \sigma(\hat{y}_{i,j})), \tag{5.2}$$

where $\sigma(\cdot)$ refers to the sigmoid function $\sigma(x) = 1/(1+exp(-x))$. That is, we optimize Eq. 5.2 with respect to the parameters of the network. This is the cost of the *classification cost term* branch in Fig. 5.1 (iii) and differs from the GO-MTL cost function [58] in the following ways:

First, while GO-MTL aims to approximate the parameter vector of the j-th observed task $\boldsymbol{w_j}$ by a linear combination of a subset of latent tasks $\boldsymbol{w_j} = \boldsymbol{V}\boldsymbol{s_j}$, where $\boldsymbol{V} \in \mathbb{R}^{d \times T}$ is a shared knowledge basis, our goal is given a keyframe/image i, to learn a new set of concept-specific feature vectors $\boldsymbol{L}_{\boldsymbol{x}_i}\boldsymbol{s}_j$, one per task, that leverage shared properties with all the other tasks. Our assumptions are similar, and we also use a predictor matrix factorization approach $\boldsymbol{L}_{\boldsymbol{x}_i}\boldsymbol{S}$, however, in a different way: In the proposed approach, given an input keyframe/image our method transforms it into T different concept-specific feature vectors that leverage information from a set of latent concept feature vectors using a bilinear factorization of the weight matrix, as described above. Subsequently, parameter matrix $\boldsymbol{U}$ is used in order to transform these concept-specific representations to concept scores, i.e., $\boldsymbol{U}^\top(\boldsymbol{L}_{\boldsymbol{x}_i}\boldsymbol{S})$. Differently, GO-MTL factorizes the 2D weight matrix that encodes concept-specific features and directly transforms the image/keyframe into concept scores.

Second, GO-MTL [58] uses iterative optimization and shallow linear models to learn the parameters. For example, in each iteration of the GO-MTL [58] method all para-

meters except for one are kept fixed and the function is optimized towards the non-fixed parameter. In our case a complete DCNN architecture is used, which makes it easy to calculate error differentials per layer w.r.t. its inputs, in order to back-propagate them to previous layers.

Third, the GO-MTL cost function can be optimized with respect either to regression loss (e.g., squared loss) or binary/multi-class classification loss (e.g., logistic loss), thus ignoring the multi-label nature of the problem. In contrast, our method works for any multi-label classification cost (e.g., the sigmoid cross entropy loss, presented above). It should be noted here that we use the sigmoid function on each activation separately and as a result the different outputs do not compete with each other (i.e., their sum does not equal to 1).

To make clear the difference of the proposed architecture from the typical and extension DCNN architectures (Fig. 5.1 (i) and (ii), respectively) we set $\alpha(\boldsymbol{L}\boldsymbol{y}_i^{(\Pi)} + \boldsymbol{b}) = \Delta$ and rewrite Eq. 5.1 as: $\hat{y}_{i,j} = \text{diag}(\boldsymbol{u}_j^\top(\text{reshape}(\boldsymbol{\Delta})\boldsymbol{s}_j))$.

Similarly, the predicted score for the j-th task and i-th datapoint with respect to the typical and extension DCNN architecture is given by: $\hat{y}_{i,j}^T = \boldsymbol{w}_j^{\top T}(\alpha(\boldsymbol{y}_i^{(\Pi)} + \boldsymbol{b}))$, and $\hat{y}_{i,j}^E = \boldsymbol{w}_j^{\top E}\boldsymbol{\Delta}$, respectively. The task-specific weight vector $\boldsymbol{s}_j$ used in our method contains the coefficients of the latent task feature vectors that will be combined with respect to concept j. This is exactly the way that our method achieves a *soft* feature sharing separately for each concept, i.e., by letting similar concepts to be described by the same latent task feature vectors according to $\boldsymbol{s}_j$. In contrast, the other two architectures of Fig. 5.1 do not use this linear combination of latent concept feature vectors but let the second-last layer, a single feature vector, to be shared across all of the concepts, thus, a *hard* translation into concept scores is performed independently for each concept.

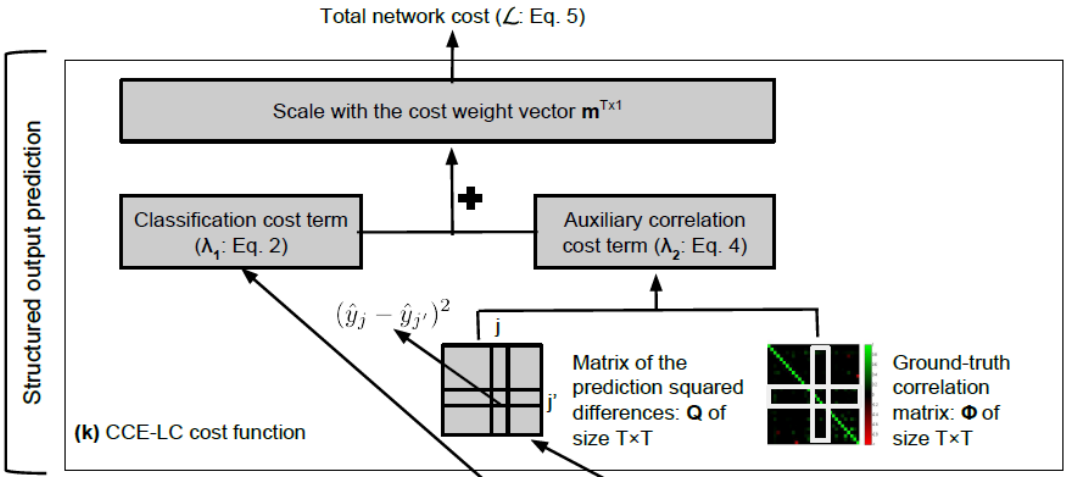

Figure 5.4: Structured output prediction part of the proposed FV-MTL with CCE-LC cost function. (Part of Fig. 5.1 (iii))

5.1.3 Label constraints for structured output prediction

Cross-entropy cost is not adequate for capturing semantic concept relations. In this section we present an additional cost term that constitutes an effective way to integrate structural information. By structural information we refer to the inherently available concept correlations in a given ground-truth annotated collection of training videos/images. It should be noted that information from other external sources, such as WordNet [37] or other ontologies, could also be used but we have not tried it in our experiments. In order to consider this information we firstly calculate the correlation matrix $\boldsymbol{\Phi} \in [-1,1]^{T\times T}$ from the ground-truth annotated data of the training set. Each position of this matrix corresponds to the ϕ-correlation coefficient between two concepts c_j, $c_{j'}$ calculated as:

$$\phi_{j',j} = \frac{AD - BC}{(A+B)(C+D)(A+C)(B+D)}, \tag{5.3}$$

where $\phi_{j',j}$ refers to j'-element of the j-th row of the correlation matrix $\boldsymbol{\Phi}$ that contains the correlation between concepts $c_{j'}$ and c_j. $A = p(c_j \wedge c_{j'}|\boldsymbol{y}_i, i = 1...N)$, $B = p(c_j \wedge \neg c_{j'}|\boldsymbol{y}_i, i = 1...N)$, $C = p(\neg c_j \wedge c_{j'}|\boldsymbol{y}_i, i = 1...N)$, $D = p(\neg c_j \wedge \neg c_{j'}|\boldsymbol{y}_i, i = 1...N)$, where $p(a|b)$ refers to the probability of a given b. The logical operator $\wedge$ expresses conjunction, e.g., $c_j \wedge c_{j'}$, means that both c_j and c_j appear on the image/keyframe,

according to its ground-truth annotations; and $\neg$ expresses negation, e.g., $c_j \wedge \neg c_{j'}$, means that $c_{j'}$ does not appear on the image/keyframe.

The proposed auxiliary concept correlation cost term that uses the correlation matrix $\boldsymbol{\Phi}$ is formed as follows:

$$\lambda_{2_{i,j}} = \frac{1}{T\text{-}1} \sum_{\substack{j'=1,\\ j' \neq j}}^{T} \begin{cases} \phi_{j',j} \left\| \sigma(\hat{y}_{i,j}) - \sigma(\hat{y}_{i,j'}) \right\|^2, \text{if } \phi_{j',j} \geq 0 \\ (-\phi_{j',j}) \left\| \sigma(\hat{y}_{i,j}) + \sigma(\hat{y}_{i,j'}) \right\|^2, \text{otherwise} \end{cases} \tag{5.4}$$

This term works as a label-based constraint and its role is to add a penalty to concepts that are positively correlated but were assigned with different concept annotation scores. Similarly, it adds a penalty to concepts that are negative-correlated but were not assigned with opposite annotation scores. Contrarily, it does not add a penalty to non-correlated concepts.

We can implement the $\lambda_{2_{i,j}}$ correlation term (Eq. 5.4) using a set of standard CNN layers, as presented on the top of the right part of Fig. 5.1. This part of our architecture is also presented in Fig. 5.4 in order to make it easier to the reader to follow this structure output prediction part of our approach. One matrix layer encodes the correlations between the ground-truth labels of the concepts (denoted as $\boldsymbol{\Phi}$), and the other matrix layer contains the correlations between the concept label predictions of our network in the form of squared differences (denoted as $\boldsymbol{Q} \in \mathbb{R}^{T \times T}$, i.e., the matrix $\mathbf{Q}$ contains the differences of activations from the previous layer). Specifically, the matrix $\mathbf{Q}$ gets multiplied, by element-wise multiplication, with the correlation matrix $\boldsymbol{\Phi}$, i.e., $\mathbf{Q} \circ \boldsymbol{\Phi}$. All the rows in the resulting $T \times T$ matrix are added, which leads to a single row vector.

5.1.4 FV-MTL with cost sigmoid cross-entropy with label constraint (FV-MTL with CCE-LC)

The two cost terms presented in Sections 5.1.2 and 5.1.3, i.e., Eq. 5.2 and Eq. 5.4, respectively, can be added in a single cost function that forms our total FV-MTL with CCE-LC network's cost as follows:

$$\mathcal{L} = \sum_{i=1}^{N} \frac{1}{T} \sum_{j=1}^{T} \frac{m_{i,j}}{z_j} \left(\lambda_{1_{i,j}} + \beta \lambda_{2_{i,j}} \right) \tag{5.5}$$

where parameter β controls the importance of concept correlation term.

In the above cost function we introduce the vector $\boldsymbol{m}_i \in \mathbb{R}^{T \times 1}$ that was originally proposed by [11] to address the problem of class imbalance. Class imbalance is a common problem in concept annotation, where for most datasets the distribution between negative to positive examples per concept is highly imbalanced, with the former outnumbering the latter in most cases. This results in bias of the classifier towards the class (positive or negative) that contains the largest number of samples. Consequently, we introduce the cost vector $\boldsymbol{m}_i$ in our cost function in order to balance the number of positive to negative examples per concept. Let us denote by p_j the number of the positive examples and n_j the number of negative examples for the concept c_j. Then, the ratio r_j of the negative to positive examples is computed as:

$$r_j = \begin{cases} \dfrac{n_j}{p_j}, & \text{if } n_j \textit{ and } p_j \neq 0 \\ 1, & \text{otherwise} \end{cases} \tag{5.6}$$

We create a weight vector $\boldsymbol{m}_i = [m_{i,1}, ..., m_{i,T}]$, for each training example i.e., for $i = 1...N$. Where $m_{i,j} = 1$ if $y_{i,j} = 0$, $m_{i,j} = 0$ if $y_{i,j}$ is unlabeled and $m_{i,j} = r_j$ if $y_{i,j} = 1$, where r_j is given by Eq. 5.6, and is different for each concept j. This weight vector is multiplied element-wise with the cost function. By doing so we adjust the misclassification cost of positive examples so as to prevent the biasing of the network towards the negative class when only a few positive examples are available. Furthermore, the normalization factor $\boldsymbol{z} \in \mathbb{R}^{T \times 1}$ that is introduced in Eq. 5.5 is

Table 5.2: Datasets and their statistics. Label cardinality (i.e., the average number of concepts presented per image/video shot), concept cardinality (i.e., the average number of positive images/video shots per concept), and missing labels (i.e., the average number of non-annotated labels per image/video shot) have been calculated on the training set for each dataset.

Dataset	Training Instances	Testing Instances	Training set Concepts	Test set Concepts	Concept Cardinality	Label Cardinality	Missing Labels
TRECVID-SIN	239495	112677	346	38	3206.3	2.2	294.6
PASCAL-VOC2012	5717	5823	20	20	416.6	1.5	0.0
PASCAL-VOC2007	5011	4952	20	20	379.2	1.4	0.0
NUS-WIDE	161789	107859	81	81	3066.1	1.9	0.0

calculated as: $\boldsymbol{z} = \sum_{i=1}^{N} \boldsymbol{m}_i$, where each position of this vector, i.e., z_j, denotes the sum of the weights for concept c_j.

In our overall network architecture, an additional layer is used in order to implement the complete FV-MTL with CCE-LC cost function, adding the two cost terms (λ_1, λ_2) and scaling their sum by the $\boldsymbol{m}$ (Eq., 5.5). In this way, the complete DCNN architecture learns by considering both the actual ground-truth annotations and also the concept correlations that can be inferred from it (Fig. 5.1: step (k)). In contrast, a typical DCNN architecture simply incorporates knowledge learned from each individual ground-truth-annotated sample.

5.2 Experimental study

5.2.1 Datasets and experimental setup

Our experiments were performed on four large multi-label video/image classification datasets, namely the TRECVID-SIN 2013 [82], the PASCAL-VOC 2007 [33], the PASCAL-VOC 2012 [34], and the NUS-WIDE [21], presented in Table 5.2. For assessing concept annotation performance, the indexing problem as defined in [82] was evaluated, i.e., given a concept, the goal was to retrieve the 2000 video shots (or images, depending on the dataset) that are mostly related with it. It should be noted here that our scope was to evaluate our algorithm in as many datasets as possible. The problem is that across existing large-scale video datasets (e.g., TRECVID MED, YouTube8m)

only TRECVID SIN dataset provides multi-label ground-truth annotations in video-shot level. The fact that we treat the problem as a still image annotation problem (analysing each keyframe separately from the others), makes it easy to also evaluate our method and all the compared methods on image annotation datasets. However, once again we need datasets that provide multi-label ground-truth annotations, so we couldn't use ImageNet but NUS-WIDE and PASCAL-VOC are two benchmarking datasets that cover the required properties and this is reason why we introduce them in this Section.

The TRECVID-SIN 2013 [82] dataset consists of approximately 600 and 200 hours of Internet archive videos for training and testing, respectively. The training set is partially annotated with 346 semantic concepts. The test set is evaluated on 38 concepts for which ground-truth annotations exist, i.e., a subset of the 346 concepts. The PASCAL-VOC 2007 [33] dataset consists of 5011 training and validation images and 4952 test images. The PASCAL-VOC 2012 [34] dataset consists of 22531 images divided into training, validation and test sets (5717, 5823 and 10991 images, respectively). We used the training set to train the various methods of our study, and evaluated them on the validation set. We did not use the original test set because ground-truth annotations are not publicly available for it (the evaluation of a method on the test set is possible only through the evaluation server provided by the PASCAL-VOC competition, submissions to which are restricted to two per week). Both for the PASCAL-VOC 2007 and 2012 the images are annotated with 20 object classes. The NUS-WIDE [21] dataset consists of 269648 Flickr images that have been annotated with 81 semantic concepts. We used a subset of 161789 images for training and the rest of them for testing. Since the available ground-truth annotations for each of the four datasets are not adequate in number in order to train a deep network from scratch without over-fitting its parameters, similarly to other studies [83], we used transfer learning. I.e., we used as a starting point the ResNet-50 network [46], which was originally trained on 1000 ImageNet categories [89], and fine-tuned its parameters

towards each of these four datasets.

In order to evaluate the methods' performance in the PASCAL-VOC 2007, 2012 and NUS-WIDE datasets we used the mean average precision (MAP) measure, while, the mean extended inferred average precision (MXinfAP) [130], which is an approximation of MAP, was used for the TRECVID-SIN dataset. MXinfAP is suitable for the partial ground-truth that accompanies the latter dataset.

5.2.2 Implementation details

For the rest of this section, when DCNN training takes place we did it by using the pre-trained ResNet-50 ImageNet network [46] (removing the last classification layer) and fine-tuning it on the target concept annotations. The network's learning rate and momentum was set to 10^{-5} and 0.9, respectively, whereas the mini-batch size was restricted by our hardware resources and set to 32. Multi-label stratification was used in order to ensure similar distribution of positive examples per class on each batch. Stochastic gradient descent (SGD) was used as the network's optimization function. All networks were trained and implemented in Caffe [50]. Regarding the proposed method, the new layers' learning rate and momentum were set to 0.1 and $5 \cdot 10^{-4}$, respectively, and β was set to 10. This value for β was chosen based on preliminary experiments on the TRECVID SIN dataset (Fig., 5.5) that show that this is an appropriate value, and also that the proposed approach is not sensitive to the value of β. The diagonal of the $\boldsymbol{\Phi}$ correlation matrix was set to zero. The model parameter values with respect to the compared methods were either selected experimentally or following the typical heuristics and strategies proposed in the corresponding works. We conducted our experiments on two NVIDIA TitanX GPUs.

Each trained DCNN was used in two different ways to annotate new images/keyframes with semantic concepts: a) As a standalone classifier, where each test image/keyframe was forward-propagated by the network and the network's output was used as the

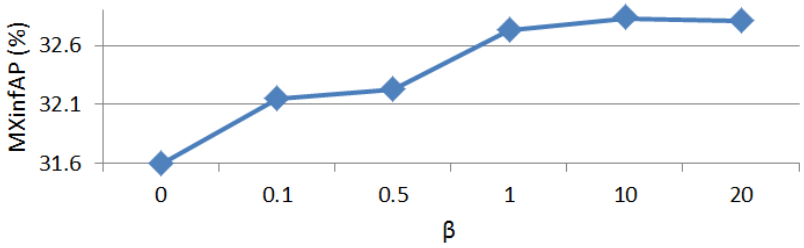

Figure 5.5: MXinfAP (%) for different values of β (Eq 5.5) for the proposed FV-MTL with CCE-LC cost.

final class distribution that was assigned to the image/keyframe. b) As a feature generator, where the training set was once again forward-propagated by the network, and the values calculated in the last layer of the network were used as feature vectors to subsequently train one Support Vector Machine (SVM) per concept. Then, each test image was firstly forward-propagated by the DCNN to extract the features and subsequently was served as input to the trained SVM classifiers.

5.2.3 Preliminary experiments - design choices

In Table 5.3 we examine the best way of using the proposed FV-MTL with CCE-LC cost by comparing different parameters and intermediate versions of them. We performed this set of experiments on the TRECVID-SIN dataset using as a starting point the ResNet-50 network.

- As a baseline we used the extension strategy proposed in [83], i.e., the DCNN architecture illustrated in Fig. 5.1 (ii). The results are presented in Table 5.3: (d)). The dimensionality of the extension layer (Fig. 5.1: step (c)) is indicated in Table 5.3: (a). Sigmoid cross-entropy was used as the network's cost function.

- We compared the baseline approach with: i) The proposed CCE-LC cost when used on the top of the baseline DCNN architecture, replacing the sigmoid cross-entropy cost (Table 5.3: (e)), i.e., the FV-MTL method was ignored. ii) The proposed FV-MTL with CCE-LC, where for the latter parameter β was set to

0, i.e., the concept correlation term λ_2 in Eq. 5.5 was ignored (Table 5.3: (f)). iii) The complete proposed FV-MTL with CCE-LC cost for $\beta = 10$, i.e., both cost terms, λ_1 and λ_2, were considered (Table 5.3: (g)). Each row of Table 5.3 corresponds to a different dimension of our FV-MTL first FC layer (shown in Fig. 5.1: step (f)).

Each of the above DCNN architectures was fine-tuned on the 346 TRECVID-SIN concepts using the TRECVID development dataset [82]. Using these results, we assess i) how the number of the latent tasks k and feature dimensionality d affect FV-MTL (Table 5.3: (a)-(c)), ii) the usefulness of exploiting semantic-level (explicit) concept relations using the CCE-LC cost instead of the typical sigmoid cross-entropy cost, iii) the usefulness of exploiting visual-level (implicit) concept relations using the proposed FV-MTL with CCE-LC when ignoring the concept correlation term λ_2 in Eq. 5.5 (Table 5.3: (f)), and iv) the usefulness of jointly exploiting visual-level and semantic-level concept relations by adopting MTL and structured output prediction using the proposed FV-MTL with CCE-LC cost when both cost terms (λ_1, λ_2 in Eq. 5.5) are considered (Table 5.3: (g)). It should be noted that our proposed FV-MTL with CCE-LC cost is most beneficial when used on datasets with non-exclusive labels (e.g., TRECVID SIN, PASCAL-VOC, NUS-WIDE) where CCE-LC can exploit and capture concept correlations across the labels. Such concept correlations are missing in single-label classification datasets such as ImageNet.

The choice of parameter k, which determines the number of latent tasks, is important because it determines the amount of sharing between the tasks. If k is very high, the tasks are not forced to share information with each other. On the other hand, if k is very low, the latent space may shrink too much. In Table 5.3 we compare different values for this parameter in order to see how it affects the proposed FV-MTL method (Table 5.3: (f),(g)). We observe that the larger the value of k the better the accuracy of the FV-MTL approach. According to the rest of the results, we observe that structured

Table 5.3: Performance (MXinfAP, %) for different dimensions of the columns of the $\boldsymbol{L_x}$ matrix (Fig. 5.1 step (e)) that we used in the experiments. For the methods (d), (e) that do not use MTL, i.e., simply use one extension layer and one classification layer on the top of it, col. c indicates the dimensionality of this extension layer. Extension strategy [83] with sigmoid cross-entropy cost serves as our baseline.

$\boldsymbol{L_x}$ **#columns** $d \times k$	**Latent tasks** k	**Feature dimension** d	**DCNN (extension strategy [83]) with STL sigmoid cross-entropy**	**Proposed CCE-LC cost**	**Proposed FV-MTL with CCE-LC** ($\beta = 0$)	**Proposed FV-MTL with CCE-LC** ($\beta = 10$)
(a)	(b)	(c)	(d)	(e)	(f)	(g)
128	4	32	23.18	28.76	30.01	29.43
256	4	64	26.91	30.84	30.50	31.38
512	8	64	28.44	30.95	30.37	31.92
1024	16	64	29.76	31.21	30.25	32.1
2048	32	64	30.95	32.44	31.60	**32.83**
4096	32	128	31.06	31.94	31.65	32.02
4096	64	64	31.06	31.94	31.71	32.07

output prediction using the proposed CCE-LC cost (Table 5.3:(e)), and MTL using the proposed FV-MTL approach (Table 5.3: (f)) are two different ways to improve concept annotation accuracy, as according to Table 5.3 the two methods always outperform the baseline (Table 5.3:(d)). Jointly using MTL and structured output prediction, in a DCNN architecture (Table 5.3: (g)) almost always outperforms all the other methods, reaching the best result of 32.83% when parameter k equals to 32 and parameter d equals to 64, i.e., the columns of $\boldsymbol{L_x}$ equal to 2048. One exception is seen in the first row of Table 5.3, where we observe a small decrease in performance of $\beta = 10$ compared to $\beta = 0$. This is due to the low number of feature dimensions and latent tasks, which are not sufficient for the CCE-LC term to capture well the correlation information.

5.2.4 Visual-level and semantic-level concept relations of the proposed method

According to Table 5.3, FV-MTL with CCE-LC for $\beta = 10$ with k equal to 32 and d equal to 64 was the pair that reached the best overall MXinfAP. In this subsection we will try to visualise what this model has learned with respect to visual-level and semantic-level concept relations. As explained in 5.1.2, the overlap in the sparsity patterns of any two tasks, (i.e., how much overlap is observed between two different task-specific weight vectors $\boldsymbol{s}_j$ and $\boldsymbol{s}_{j'}$, where $j \neq j'$) controls the amount of sharing

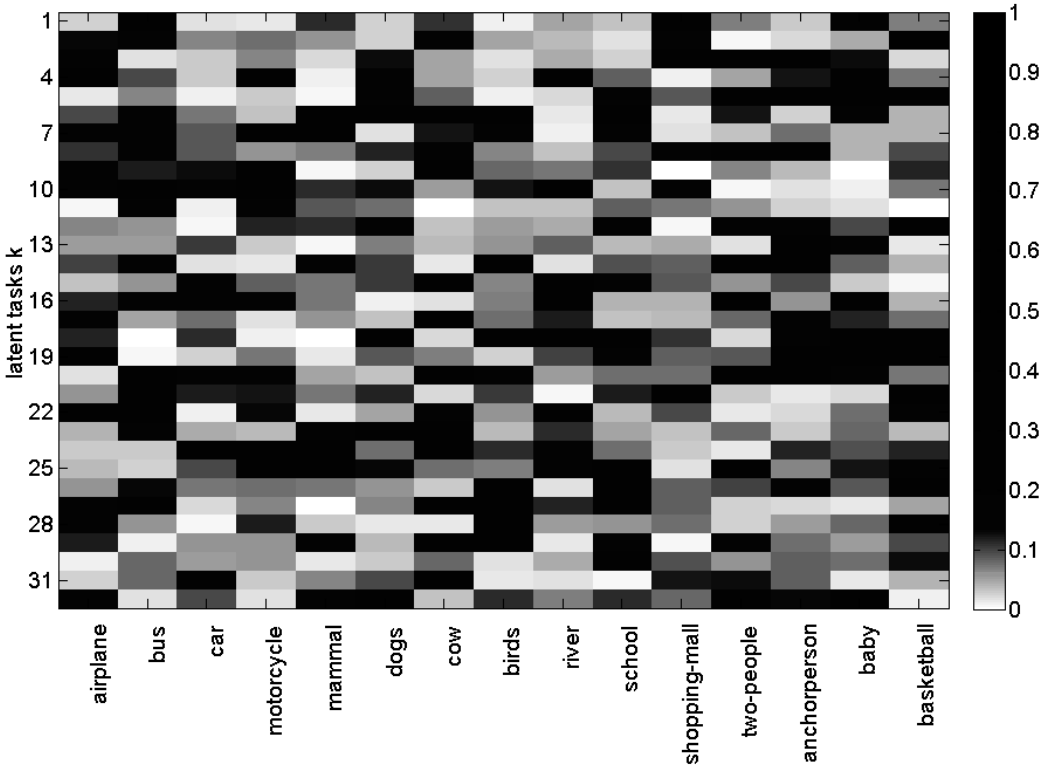

Figure 5.6: Recovered sparsity patterns (the matrix $\boldsymbol{S}$) with FV-MTL with CCE-LC for $\beta = 10$ with k equal to 32 and d equal to 64, for 15 selected concepts of the TRECVID SIN dataset. Darker color indicates higher absolute value of the coefficient. The horizontal axis depicts the 15 observed concepts and the vertical axis the 32 latent tasks.

between them. Based on this in Fig. 5.6, we recovered sparsity patterns (the matrix $\boldsymbol{S}$) with FV-MTL with CCE-LC for 15 selected concepts of the TRECVID SIN dataset (darker color indicates higher absolute value of the coefficient). The horizontal axis depicts the 15 observed concepts and the vertical axis the latent tasks (k=32) in this case. It is difficult to recover the grouping and overlap structure for the observed concepts based on this figure but some interesting observations could be found. For example, concepts with the same sparsity pattern can be considered as belonging to the same group, while concepts with orthogonal sparsity patterns can be considered as belonging to different groups. The 9th and 10th latent tasks are always active for the transport-related concepts (e.g., airplane, car, bus, motorcycle) but they are inactive,

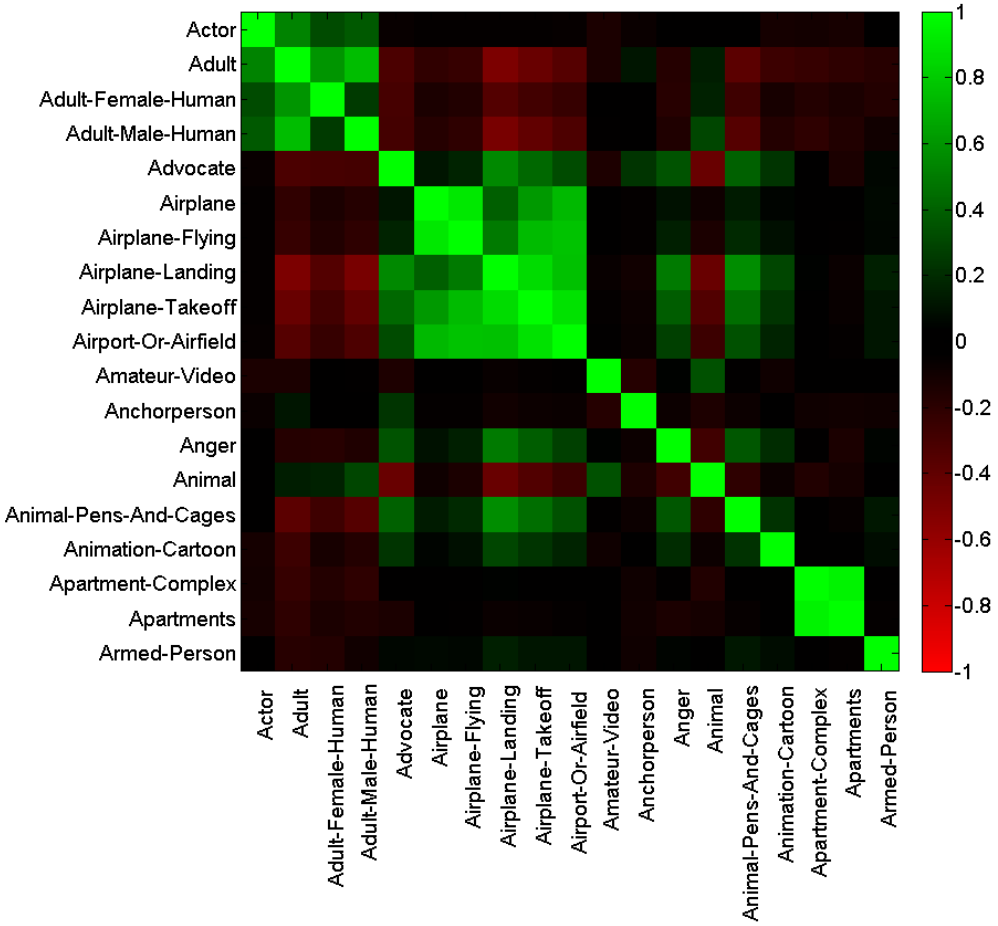

Figure 5.7: Colormap of the phi-correlation coefficient calculated on the final prediction scores of the proposed FV-MTL with CCE-LC for $\beta = 10$ with k equal to 32 and d equal to 64, when applied on the TRECVID SIN 2013 test dataset for 20 selected concepts.

at least one of the two, for any of the other concepts. Transport-related concepts can be considered as belonging to the same group. In addition, those latent tasks that are active for the concept "river" are always inactive for the concept "shopping-mall" (except for the 11th latent task), which indicates that these are two disjoint groups.

Regarding the semantic-level concept relations, Fig. reffig:correlation presents the color map of the phi-correlation coefficients, when calculated on the final prediction scores of the model when applied on the TRECVID SIN 2013 test dataset for 20 selected concepts. We can see that the model has captured many pairs of positive cor-

Table 5.4: MXinfAP (%) for 38 TRECVID-SIN and MAP (%) for 20 PASCAL-VOC2007, 20 PASCAL-VOC2012 and 81 NUS-WIDE concepts, respectively, for different STL, MTL, structured output and joint MTL and structured prediction methods using the ImageNet ResNet-50 as the base network.

Category	**Method**	**TRECVID-SIN**		**PASCAL-VOC2007**		**PASCAL-VOC2012**		**NUS-WIDE**	
		(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
		direct	last layer	direct	last layer	direct	last layer	direct	last layer
i) Baseline (without fine-tuning)	ResNet-50 [46] as feature generator	29.21	29.78	83.90	83.76	82.98	83.04	51.30	56.20
ii) Typical DCNN fine-tuning	ResNet-50 [46]	27.35	28.66	76.38	83.06	81.20	82.15	51.17	56.32
iii) DCNNs (extension strategy [83]) with STL cost functions	Hinge-loss	29.08	30.06	78.32	79.23	86.6	87.23	52.80	57.49
	Sigmoid cross-entropy	31.06	32.2	80.74	84.97	86.94	86.80	53.94	57.20
	CCE [11]	31.93	32.52	84.07	84.92	85.52	85.39	54.58	55.0
	DWE [119]	28.03	29.17	77.25	78.12	85.14	86.00	51.10	56.08
iv) MTL for DCNNs or shallow linear models	AMTL [105]	29.36	30.15	83.15	84.37	83.17	84.05	53.40	54.22
	CMTL [141]	29.89	30.45	83.44	84.42	83.55	84.60	51.80	52.40
	2-sidedNN [127]	29.91	30.01	83.50	84.53	83.70	84.45	51.97	52.67
v) Structured outputs	Stacking-LP [71]	30.01	31.05	84.68	85.12	84.25	85.30	51.96	52.98
	LMGE [129]	30.17	31.24	84.32	85.02	84.52	85.64	53.07	54.62
vi) Joint MTL + Structured outputs	ELLA_LC [70]	28.15	29.09	81.98	82.84	82.15	83.17	52.40	54.68
	DMTL_LC [69]	28.23	31.71	82.01	84.07	82.23	84.30	52.35	54.70
vii) Proposed	CCE-LC cost	32.44	33.55	85.40	86.73	86.32	86.39	**56.40**	**60.73**
	FV-MTL with CCE-LC ($\beta = 0$)	31.60	32.15	82.21	86.96	87.10	88.51	55.45	54.69
	FV-MTL with CCE-LC ($\beta = 10$)	**32.83**	**33.77**	**85.70**	**87.00**	**87.54**	**88.69**	55.54	60.22

related concepts such as "adult"-"actor", "adult"-"female human person" (green areas of the figure), pairs of negative correlated concepts such as "animal"-"airplane landing" (red areas of the figure), and non-correlated concepts such as "animal"-"actor", "anger"-"actor" (black areas of the figure). According to the observations recovered from figures 5.6 and reffig:correlation we can see that our proposed method is able to capture both visual-level and semantic-level concept relations.

5.2.5 Main findings - comparisons with related methods

Table 5.4 compares the proposed complete FV-MTL with CCE-LC (for $\beta = 10$) with other related methods on the three datasets. In addition, we evaluate the two intermediate versions of our complete DCNN architecture that were also evaluated in Table 5.3. I.e., a) Extension strategy [83] for DCNNs with the proposed CCE-LC cost, i.e., the typical complete DCNN architecture illustrated in Fig. 5.1 replacing the sigmoid cross-entropy cost with the proposed CCE-LC cost, and b) FV-MTL with CCE-LC for $\beta = 0$. We set k equal to 32 and d equal to 64, which was the pair that reached the best overall MXinfAP according to Table 5.3; similarly, in the case

that CCE-LC is used alone the dimension of the extension layer was set to 2048. We performed comparisons with the following methods:

- i) A baseline where we use the ResNet-50 pre-trained network as feature generator; one SVM classifier per concept was trained using as features either the ResNet's output or its last FC layer.

- ii) The typical DCNN architecture with sigmoid cross-entropy cost, i.e., the ResNet-50 pre-trained network fine-tuned on each of the four datasets by simply replacing the classification layer with a new layer with dimension that equals to the number of concepts in the target domain as illustrated in Fig. 5.1 (i).

- iii) Extension strategy [83] for DCNNs, i.e., the DCNN architecture illustrated in Fig. 5.1 (ii), and four different STL cost functions: a) hinge-loss, b) sigmoid cross-entropy, c) cost sigmoid cross-entropy (CCE) [11], an extended version of (b) that also addresses the class-imbalance problem, and d) dynamic weighted euclidean loss (DEW) [119], an extension of the euclidean loss suitable for multi-label classification giving a greater penalty to concept prediction scores that have been ranked higher than the negative ground-truth annotated concepts. The size of the extension layer was set to 4096, according to the findings of Table 5.3. This category of methods uses exactly the same architecture with the first intermediate version of our complete architecture (denoted as a) above), with the difference that each of the above three cost functions is used instead of the CCE-LC cost.

- iv) MTL, either as an integral part of DCNNs or for shallow linear models: a) AMTL [105], b) CMTL [141] and c) the 2-sided NN that was proposed in [127] for solving the GO-MTL method objective function [58].

- v) Structured output prediction: a) Stacking-LP [71], a two-layer stacking architecture combined with the label power-set algorithm [71]. b) LMGE [129],

an inner learning approach that uses the extracted features and exploits concept correlations in a single step.

- vi) Methods jointly using MTL and structured output prediction: a) DMTL_LC [69], and b) ELLA_LC [70].

We selected all the parameter values for these methods based on the training data, and in accordance with the recommendations provided in the corresponding papers.

We apply and evaluate all the above methods in two different ways (in a direct analogy to what is discussed in the last paragraph of Section 5.2.2); the specifics of these depended on whether they are complete DCNN architectures or shallow models that use pre-computed DCNN features. To the first category belong the following methods: Typical DCNN fine-tuning (group (ii)), all methods of group (iii) above, the 2-sided NN of [127], DMTL_LC [69], the proposed FV-MTL with CCE-LC and the latter's two intermediate versions. These methods are used a) as standalone classifiers, where the direct output of the complete network is evaluated (denoted as "direct" in Table 5.4), b) as feature generators, where SVM classifiers are trained on DCNN-based features. In the latter case, the output of the last layer of the complete trained network for each method was used as a feature vector to train one SVM per concept (denoted as "last layer" in Table 5.4). The remaining methods (that belong to the second category), i.e., the baseline of group (i) above, AMTL [105], CMTL [141], ELLA_LC [70], Stacking-LP [71] and LMGE [129], use the pre-trained ResNet-50 network as feature generator and the extracted features were used to train each of these methods. The methods specifically used in our experiments a) the ResNet-50 output layer (denoted as "direct" in Table 5.4), b) the ResNet-50 last FC layer (denoted as "last layer" in Table 5.4).

Table 5.4 presents the results in terms of MXinfAP for the TRECVID-SIN dataset and in terms of MAP for the PASCAL-VOC and NUS-WIDE datasets. With respect

to the direct output (Table 5.4: (a),(c),(e),(g)) we observe that the two intermediate versions of our proposed method perform quite well, outperforming the compared methods in the majority of cases. One exception is observed between the compared extension strategy [83] with sigmoid cross-entropy cost and the proposed FV-MTL with CCE-LC for $\beta = 0$, where their difference is that the latter also incorporates MTL. The results present fluctuations concerning which of the two methods performs better, depending on the dataset. However, jointly combining MTL and structured output prediction, using the proposed FV-MTL with CCE-LC for $\beta = 10$, further improves the concept annotation accuracy and outperforms all the other previously-published methods across all of the evaluated datasets, reaching the best overall concept annotation accuracy of 32.83%, 85.70%, 87.54% and 55.54% for TRECVID-SIN, PASCAL-VOC2007, PASCAL-VOC2012 and NUS-WIDE, respectively. The only exception is the NUS-WIDE dataset, where our intermediate version of the typical extension strategy with CCE-LC cost presents the best accuracy, and our complete architecture reaches the second-best performance. It should be noted that we compare our method with very recent methods; even our baseline is the ResNet-50 network that was ranked first in the ImageNet 2016 competition and our method outperforms it by approximately 3 to 4 percentage points. Similarly clear differences can be observed with respect to all the other compared methods. Even compared to the most recent DCNN with CCE cost [11], although the differences are smaller, we consistently outperform it by approximately 1 to 1.5 percentage points in all three datasets. Similar conclusions can be reached regarding the results presented in columns (b), (d), (f) and (h) of Table 5.4 that refers to the second way of applying the compared methods, as described in the beginning of this section. We also present in Fig. 5.8 the XinfAP per task regarding the proposed FV-MTL with CCE-LC and the other two best performing methods (i.e., DCNN with sigmoid cross-entropy cost and DCNN with CCE cost [11]) in the TRECVID-SIN dataset. Besides our overall best result (33.77% - Table 5.4), our method performs better than these other two well-performing methods for 25 out

of the 38 evaluated concepts.

To investigate the statistical significance of the difference of the results of each method from the best performing method, i.e., the proposed FV-MTL, we used a paired t-test as suggested by [12]. We found that differences between the proposed FV-MTL with CCE-LC ($\beta = 10$) and all other previously-published methods that we compare with, per column of Table 5.4, are significant at 5% significance level.

Finally, we assess the robustness of the the proposed and the other two best performing methods (i.e., Sigmoid cross-entropy, and CCE costs [83], [11]) with respect to the TRECVID SIN dataset according to Table 5.4 , when they are trained on smaller datasets for the same number of concepts. Specifically, Fig. 5.9 presents the reduction of MXinfAP when each of the compared methods is trained a) on only half of the keyframes of TRECVID SIN training set and b) on only a quarter of the keyframes for the same dataset, compared to the complete training set. We observe that the DCNN with sigmoid cross-entropy cost is affected by the smaller training datasets, as according to Fig. 5.9 its concept annotation accuracy is reduced by approximately 6 and 3 percentage points when the half and quarter training sets are used instead of the complete training set, respectively. In contrast, the proposed FV-MTL with CCE-LC for $\beta = 10$ and its intermediate versions, i.e., CCE-LC cost and FV-MTL with CCE-LC for $\beta = 0$, are robust to smaller training sets, exhibiting only a small reduction of MXinfAP compared to the case of using the complete training set.

5.2.6 Execution times

We continue the analysis of our results by assessing the execution times during the training and classification phase of the different methods compared in this study. Table 5.5 summarizes the required execution time in hours for the proposed FV-MTL with CCE-LC for $\beta = 10$ and its two intermediate versions, defined in earlier sections, and also compares it with the rest of the methods. We observe that the proposed

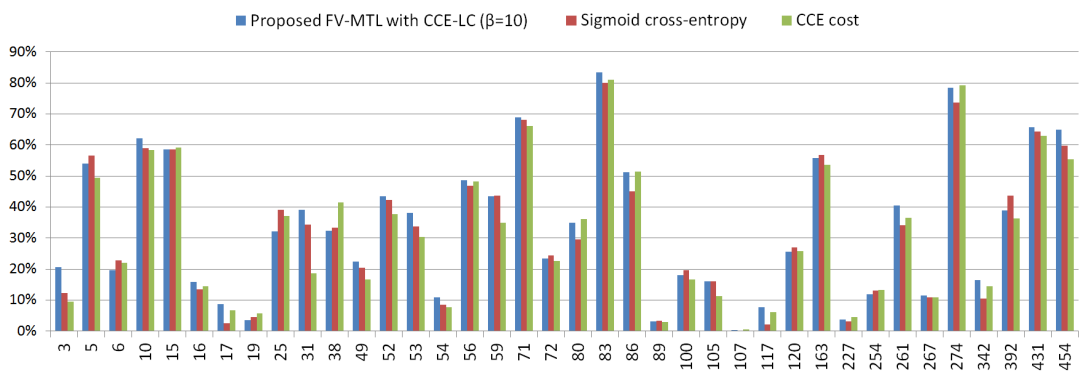

Figure 5.8: XinfAP (vertical axis) per concept (the horisontal axis shows the concept ID, according to TRECVID SIN [82]), for the top-3 best performing methods with respect to the TRECVID SIN dataset according to Table 5.4.

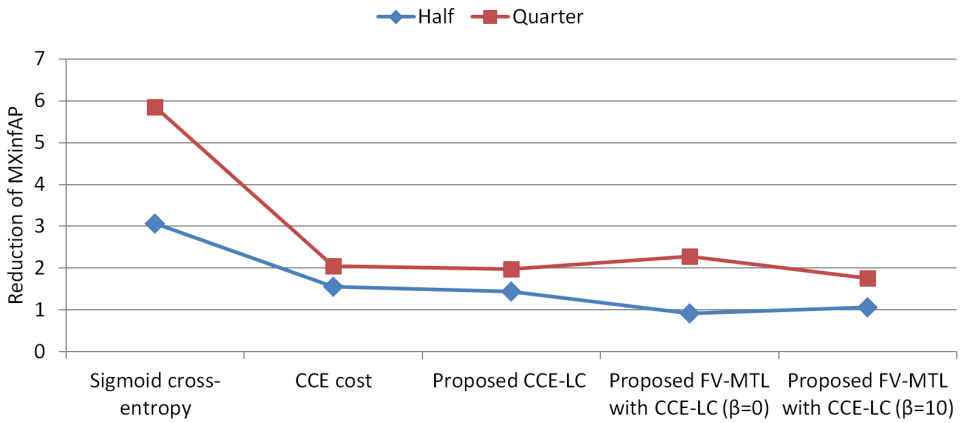

Figure 5.9: Reduction of MXinfAP when only a half and a quarter of the training samples respectively are used instead of the complete training set, for the top-5 best performing methods with respect to the TRECVID SIN dataset according to Table 5.4. Lower values are better.

method is not considerably more computationally expensive than DCNN methods that use STL cost functions. Training of the baseline, AMTL and CMTL methods that use pre-computed features is a bit faster than the proposed method and its intermediate versions; however, all these previous methods achieved lower accuracy than the proposed one, according to Table 5.4. During classification all the compared methods are executed on very similar time, except for the 2-sidedNN, Stacking-LP and DMTL_LC that are significantly slower. We conclude that our proposed FV-MTL with CCE-LC is faster than other MTL methods for DCNNs (2-sidedNN, DMTL_LC) both during

Table 5.5: Mean execution training/testing times in hours.

Category	**Method**	**TRECVID-SIN**	
		training	testing
i) Baseline (without fine-tuning)	ResNet-50 [46] as feature generator	14.25	2.45
ii) Typical DCNN fine-tuning	ResNet-50 [46]	17.33	2.47
iii) DCNNs (extension strategy [83]) with STL cost functions	Hinge-loss	17.35	2.48
	Sigmoid cross-entropy	17.45	2.47
	CCE [11]	17.75	2.50
	DWE [119]	17.85	2.50
iv) MTL for DCNNs or shallow linear models	AMTL [105]	14.75	2.50
	CMTL [141]	14.85	2.58
	2-sidedNN [127]	48.12	6.80
v) Structured outputs	Stacking-LP [71]	23.15	4.51
	LMGE [129]	15.17	2.68
vi) Joint MTL + Structured outputs	ELLA_LC [70]	20.97	2.53
	DMTL_LC [69]	49.27	6.84
vii) Proposed	CCE-LC cost	17.75	2.67
	FV-MTL with CCE-LC ($\beta = 0$)	17.53	3.17
	FV-MTL with CCE-LC ($\beta = 10$)	18.15	3.10

training and classification, and also comparable in execution time with the second-best performing method of Table 5.4, i.e., DCNN with CCE cost [11]. In addition, the proposed FV-MTL with CCE-LC, although a little bit slower, is comparable in execution time with the other most commonly used baselines (without fine-tuning, with fine-tuning and addition of one or more layers), and at the same time presents significant improvement in concept-based annotation accuracy.

5.2.7 Data augmentation and comparisons

Recently, improved accuracy has been achieved by image augmentations, i.e., feeding the DCNN with more than one image crops of the same image. For example, in the PASCAL-VOC2007 dataset this was shown to improve the MAP by 6 percentage points [121]. In Table 5.6 we compare our proposed method with these approaches,

Table 5.6: MAP (%) for 20 PASCAL-VOC2007 concepts for methods that use image augmentations.

Method	**PASCAL-VOC2007**
Simonyan et al. [94]	89.3
Wei et al. [121]	90.9
Wang et al. [119]	92.5
FV-MTL with CCE-LC ($\beta = 10$) + augmentations	**93.9**

however, due to the fact that this is a very computational intensive and time consuming process we present results only on the PASCAL-VOC2007 dataset. The following three SoA PASCAL-VOC2007 methods were selected: (i) Simonyan et al. [94]: A pre-trained ImageNet DCNN is applied on multiple image representations that are extracted and aggregated across multiple locations and scales. The resulting aggregated image descriptor (using the second-last layer as image feature representation) is used to train a linear SVM per concept.(ii) Wei et al. [121]: Many object segment hypotheses are given as input to a shared DCNN that has been pre-trained in the ImageNet dataset. The shared network's output is aggregated with max pooling in order to return a single multi-label prediction. The shared network is fine-tuned on the PASCAL-VOC dataset. (iii) Wang et al. [119]: Similar to Wei et al. [121], a pre-trained ImageNet DCNN is fine-tuned using many object segment hypotheses. Stochastic scaling and cropping over images is performed in this case in order to choose the most useful image crops. Furthermore, the DWE loss function, also presented in Table 5.4, is used on the top of the network. Finally, the proposed architecture (FV-MTL with CCE-LC ($\beta = 10$)) is fine-tuned on 15 random image object segment proposals per image extracted using the selective search method [115]. Similarly to [121] and [119] a shared DCNN is used to aggregate the probability scores w.r.t. each proposal using max-pooling. We observe that the proposed method once again outperforms all the other compared methods and also that image augmentation is a robust way of increasing the accuracy of our network by approximately 7 percentage points.

5.3 Conclusion

In this chapter we proposed a DCNN architecture that jointly exploits implicit visual-level and explicit semantic-level concept relations. We built on ideas from MTL and structured output prediction in order to develop the FV-MTL approach for learning shared latent representations across the different tasks, and the new CCE-LC cost function that exploits the correlations between the concepts, respectively. The integrated DCNN architecture that emerges from combining these approaches was shown to improve concept annotation accuracy and outperformed the related state-of-the-art methods, without introducing any significant computational overhead. Specifically, it outperforms methods that do not impose any concept relations from 1.5% to 5%, methods that solely introduce either MTL or structured outputs by ∼2%, and finally methods that jointly consider MTL and structured outputs by ∼4%, in all three evaluated datasets. Finally, introducing image augmentations during the network's training was successfully applied to our method, further increasing its accuracy by ∼7%.

Chapter 6

Zero-shot Learning for Ad-hoc Video Search

Contents

Ad-hoc video search (AVS) [5] is the problem of retrieving, from a large video collection, video fragments (e.g., video shots) that are related to a given query. A query refers to an ad-hoc textual description, e.g. "Find shots of a woman wearing glasses". This problem is closely related to the simpler problem of concept-based video search, examined in the previous chapters, where a set of video shots is retrieved given a specific keyword (a.k.a. concept). In the latter case supervised learning (e.g., DCNNs) can be used to annotate the video shots with concepts. However, AVS is more complicated because an input query could be any complex or also abstract textual description for which annotated data does not exist; as a result, unsupervised learning and natural language processing (NLP) need to be employed for generating a common representation of queries and videos.

In this chapter we present a fully-automatic AVS method that uses solely a natural-language textual query to retrieve related video shots from a video collection. The novelty of our method is:

- An efficient algorithm that performs a number of NLP and semantic analysis steps to translate a query into a set of predefined concepts.
- A new approach that projects the concept-based video shot and query representations into a common semantic embedding space.
- The combination of two different measures for the distance between the video shots and the target query, calculated on the concept-based and the semantic embedding representations respectively.

Our AVS method was evaluated on the TRECVID AVS 2016 [5] and Video Search 2008 [30] datasets. The results show that it outperforms other state-of-the-art approaches.

6.1 Concept-based query and keyframe representations

Assume that a text query Q and a set of keyframes $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ are given, where one keyframe $\mathbf{x}_i \in \mathbb{R}^d$ has been extracted from each shot of the videos in a collection. Our goal is to retrieve for query Q the k keyframes from $\mathbf{X}$ that are most closely related to it. The overview of our method is presented in Fig. 6.1. Given a predefined concept pool $C = \{c_j\}_{j=1}^{T}$, our method represents both the keyframes (Fig. 6.1 (a)) and the query (Fig. 6.1 (c)) as vectors of related concepts. Then, these concept-based representations are projected into a common semantic embedding space (Fig. 6.1 (b)). Finally, the k-nearest keyframe representations to the query representation are retrieved using a distance measure.

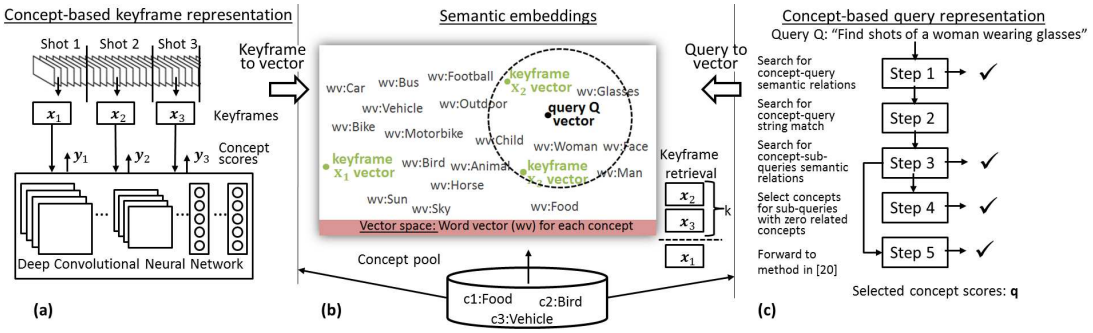

Figure 6.1: Overview of the proposed ad-hoc video search method.

6.1.1 Concept-based Keyframe Representation

Our method initially applies a DCNN $D : \mathbb{R}^d \Rightarrow [0,1]^T$, that has been trained on the T concepts, in every keyframe $\mathbf{x}_i$ in order to calculate concept-based representations $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N$. The DCNN's output for an input keyframe $\mathbf{x}$, $D(\mathbf{x}) = \mathbf{y}$, is a vector $\mathbf{y} \in [0,1]^T$ that indicates the model's belief that each of the concepts in C appears in the input keyframe.

6.1.2 Concept-based Query Representation

A set of NLP steps is applied in order to translate the query in a set of related concepts chosen from the concept set C. Let $C^Q = \{c_1^Q, c_2^Q, ..., c_{T'}^Q\} \subseteq C$ be the set of concepts selected for the query Q, where $T' \leq T$, and $\mathbf{q} = [q_1, q_2, ..., q_{T'}] \in [0,1]^{T'}$ a vector that indicates the degree to which each of the selected concepts in C^Q is related to Q. The following steps focus on analysing different parts of the query, instead of treating it as a set of single terms (words), which results to more distinctive retrieved concepts. Starting from the empty set $C^Q = \varnothing$ we calculate C^Q and $\mathbf{q}$ as follows:

Step one: The complete textual description of Q is compared with each concept in C for "semantic relatedness" in terms of the Explicit Semantic Analysis (ESA) measure (which returns a real number in the [0,1] interval) [41]. Those concepts that are semantically close to the query, i.e., the concepts that have ESA value higher than a threshold θ, are added in the set C^Q. If at least one concept is selected in this way,

we assume that the entire query is very well described by these concepts and the query processing stops; otherwise, we continue with step two.

Step two: This step searches if any of the concepts in our concept pool appears in any part of the test query by string matching. Some (complex) concepts may describe part of the query quite well, but appear in a way that is difficult to detect them due to the subsequent linguistic analysis, e.g., breaking down the query to sub-queries. Any concept that appears in the query is added in the set C^Q and the query processing continues in step three.

Step three: Queries are complex sentences, but this step automatically transforms them into elementary *sub-queries*; i.e., meaningful smaller phrases or terms that are included in the original query. For example, the query "Find shots of one or more people at train station platform" is split into the following four sub-queries: "people", "train station platform", "persons" and "train station". Then, each of the *sub-queries* is translated to a concept vector. To identify the sub-queries, part-of-speech tagging and stop-word removal are used together with a task-specific set of NLP rules. For example, extracting "Noun - Verb - Noun" sequences and considering them as sub-queries. The motivation is that such a triad is much more characteristic of the original query than any of the single terms alone. Then, the ESA measure is calculated between each sub-query and each of the concepts in the pool. Concepts that exceed the threshold θ are added in the set C^Q. In the case that for all of the sub-queries at least one concept has been selected, the query processing stops. If for a subset of the sub-queries no concepts have been selected then these sub-queries are propagated to step four. Finally, if for all the sub-queries no concepts have been selected then the test query and all of the sub-queries are propagated to step five.

Step four: For a subset of the sub-queries no concepts were selected. For each of these sub-queries, the concept with the highest value of ESA measure is selected in this step (i.e., threshold θ is ignored), and then the query processing stops.

Step five: For some queries, the processing step up to step three did not select any concepts. In this case, the query and the sub-queries are served as input to the zero-example event detection system of [111], which returns a ranked list of the most relevant concepts in accordance with relatedness scores, based on the ESA measure. In [111] the concepts are enriched with additional information captured by Google or Wikipedia, which virtually augments the concept pool. Then, the query processing has been completed.

Finally, the query's concept vector $\mathbf{q} \in [0,1]^{T'}$ is formed by the corresponding scores of the selected concepts. If a concept has been selected in steps 1, 3, 4 or 5 then the corresponding vector's element is assigned with the relatedness score (calculated using the ESA measure); if it has been selected in step 2, it is set equal to 1. A complete example of applying the above steps in a query is presented in Table 6.1.

6.2 Semantic embeddings for query and keyframe representations

Given a semantic embedding space $S \subseteq \mathbb{R}^m$, we project both the concept-based keyframe (Section 6.1.1) and the query (Section 6.1.2) representations into S, in order to directly measure their distance. Initially, we calculate the set $S^C = \{s(c_1), s(c_2), ..., s(c_T)\}$ of the semantic embedding vectors $s(c_j) \in S$ associated with each concept in C, by applying a pre-trained word2vec model [75].

Then, similarly to [79], our method calculates a keyframe semantic embedding vector $f(\mathbf{x}) \in \mathbb{R}^m$, as the combination of the semantic embeddings of the R-top retrieved concepts for $\mathbf{x}$, according to the concept-based keyframe representation $\mathbf{y} \in \mathbf{Y}$, weighted by their corresponding concept detection scores:

$$f(\mathbf{x}; \mathbf{y}, S^C) = \frac{1}{Z} \sum_{r=1}^{R} y_{g(\mathbf{x},r)} \cdot s(c_{g(\mathbf{x},r)}), \tag{6.1}$$

where $g(\mathbf{x}, r)$ denotes the r-th most likely concept label for the input keyframe $\mathbf{x}$

according to $\mathbf{y}$, $Z = \sum_{r=1}^{R} y_{g(\mathbf{x},r)}$ the normalization term and R a parameter that considers the maximum number of embeddings that will be combined.

Subsequently, we calculate the semantic embedding vector associated with the concept-based query representation by extending the above process as follows. Given the set of concepts C^Q assigned to this query and the corresponding ESA scores $\mathbf{q}$, described in Section 6.1.2, our method calculates the semantic embedding vector h(Q) for query Q, as the combination of the semantic embeddings of the concepts assigned to this query weighted by their corresponding ESA score:

$$h(Q; \mathbf{q}, S^C) = \frac{1}{Z'} \sum_{l=1}^{T'} q_l \cdot s(c_l^Q), \tag{6.2}$$

where $Z' = \sum_{l=1}^{T'} q_l$ the normalization term.

After the concept-based keyframe representations have been calculated (Section 6.1.1), our system measures their distance from the concept-based query representation (Section 6.1.2), e.g. by calculating the euclidean distance. Similarly, the distance between the semantic embedding keyframe representations and the semantic embedding query representation is calculated and the two distance vectors are combined in terms of arithmetic mean. The k keyframes with the smallest distance are then retrieved.

6.3 Experimental study

6.3.1 Dataset and Experimental Setup

Our experiments were performed on the TRECVID AVS 2016 (AVS16) [5] and Video Search 2008 (VS08) [30] datasets that consist of approx. 600 and 100 hours of internet archive videos and are evaluated on 30 and 48 queries, respectively. Ground-truth annotated training data does not exist for these queries. The AVS problem as defined in TRECVID [5] was examined, i.e., given a query, the goal was to retrieve the 1000 video shots that are mostly related with it. We analyze our results in terms of mean extended inferred average precision (MXinfAP), which is an approximation of the

Table 6.1: Concept-based query representation example.

Query: ***Find shots of three people or more walking or bicycling on a bridge during daytime***			
	Sub-queries	C^Q ($\theta = 0.8$)	**q**
Step 1	*Find shots of...daytime*	$\emptyset$	-
Step 2	***three people or more*** *walking or bicycling on a bridge during daytime*	three or more people	1.0
Step 3	people walking	walking	1.0
	bicycling	bicycle-built-for-two	1.0
		bicycles	0.85
		bicycling	0.84
	bridge	suspension bridge	1.0
		bridges	0.84
	Sub-query *daytime* also found but without corresponding concepts with ESA distance $> \theta$		
Step 4	daytime	daytime outdoor	0.74

mean average precision suitable for the partial ground-truth that accompanies the TRECVID dataset [130].

In order to create the concept-based keyframe representations, each keyframe was automatically annotated with 1000 ImageNet [89] and 346 TRECVID SIN [6] concepts. Regarding the 1000 ImageNet concepts, we applied five pre-trained ImageNet DCNNs on the keyframes and fused their outputs in terms of arithmetic mean to obtain a single score for each of the 1000 concepts. Regarding the 346 SIN concepts, we fine-tuned (FT) two pre-trained ImageNet DCNNs on the 346 concepts using the TRECVID AVS development dataset [5] and the extension strategy proposed in [83], where one extension layer with 4096 neurons was used. We used the last layer of each of these networks to train support vector machine classifiers (SVMs) for each concept. The keyframe score per concept was the average of the probabilities that the two SVM models returned. Each keyframe was finally represented by a 1345-element vector by simply concatenating the score vectors for the ImageNet and the TRECVID SIN concepts. The threshold θ for deciding whether to select a concept or not in our method

(Section 6.1.2) was set to 0.8. The pre-trained Google News Corpus word2vec model [1] was used for calculating the semantic embeddings of the concept-based representations (Section 6.2). In our preliminary experiments, small fluctuations of the overall accuracy were observed for different values of parameter R (Eq. 6.1), consequently and based on these experiments we set it to 70. The Euclidean distance was used for measuring the distance between the keyframe and query representations.

6.3.2 Experimental Results

Table 6.2: Experiments (MXInfAP (%)) on the AVS16 dataset to investigate the parameters of the proposed method.

	All	**Excluding one step:**				
Steps		step 1	step 2	step 3	step 4	step 5
(a) Concept-based representation (Sections 6.1.1 + 6.1.2)	5.94	5.92	5.74	3.96	5.95	4.53
(b) Semantic embeddings (Section 6.2)	3.77	3.86	2.98	3.22	3.75	2.80
(c) Combination	6.35	**6.51**	5.77	4.37	6.27	4.99

Table 6.3: MXInfAP (%) for different compared AVS methods.

Methods	**AVS16**		**VS08**	
(a) Literature methods				
Tzelepis et al. [111]	4.16		8.27	
Ueki et al. [112]	5.65		7.24	
Norouzi et al. [79]	3.14		7.30	
(b) Top-4 TRECVID finalists				
Top-1	Le et al. [29]	5.4	Tang et al.	6.7
Top-2	Markat. et al. [67]	5.1	Snoek al.	5.4
Top-3	Liang et al. [31]	4.0	Ngo et al.	4.2
Top-4	Zhangy et al. [133]	3.8	Mei et al.	4.1
Proposed	**6.35**		**9.11**	

Table 6.2 presents the results of some intermediate experiments that we performed in order to investigate the performance when: i) the transformation to the semantic embedding space is ignored (Table 6.2 (a)), ii) the final distance from the query is

[1] https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz

calculated solely in the semantic embedding space (Table 6.2 (b)), iii) the complete process is used, i.e., the final distance is the combination of the distances calculated in i) and ii) (Table 6.2 (c)). For each of the above cases, we also examine the usefulness of each of the steps presented in Section 6.1.2, i.e., each column shows the corresponding results when one of the steps is excluded. According to Table 6.2 we conclude as follows: Concept-based representations perform very well on the AVS problem, outperforming the semantic embedding representations. However, combining the two types of representations further improves our method, reaching a MXInfAP of 6.35 %. It seems that these types of representations are complementary. This makes sense because on the one hand, we have a representation that is based on the presence of each unique concept both in the query and in the keyframe and on the other hand, we have a representation that is based on semantic embeddings of each presented concept, which is a more rich information that has been calculated using an external trained model (word2vec).

Almost all of the steps one to five of Section 6.1.2 contribute to the improved translation of the query into related concepts; excluding one step reduces the performance in most cases. One exception is observed w.r.t. step 1. However, excluding step 1 and evaluating w.r.t. the VS08 dataset slightly reduced the accuracy, which lead us to propose the use of all five steps. Furthermore, step 4 only marginally affects the performance; thus, sub-queries that do not present high semantic relatedness with any of the concepts could be ignored when for at least one sub-query one or more concepts have been selected. Similar conclusions were reached on the VS08 dataset.

Table 6.3 compares the proposed method with 11 different literature ones. The top part of the table refers to those methods that were re-implemented in order to be adapted for this problem and datasets, whereas in the lower part we introduce the results of the top-four finalists in the AVS16 and the VS08 tasks. Especially for comparing with [112] we used a modified version that does not require the user's

involvement. In this modified version, we automatically split the query into several keywords after removing the stop-words, and for each keyword the concept with the highest ESA value is selected. Overall, as we can see in Table 3, for both datasets our proposed method performs very well compared to the other methods. Specifically, it outperforms all the compared methods, achieving an MXinfAP of 6.35% and 9.11% for AVS16 and VS08, respectively.

6.4 Conclusion

In this chapter we presented a fully-automatic method that combines video concept detection and query analysis for ad-hoc video search. Extensive experiments reveal the usefulness of the proposed NLP steps for translating a textual query to related predefined concepts and the usefulness of combining different types of keyframe-query representations (e.g., concept-based representations, semantic embeddings). Our proposed method was compared with many state-of-the-art AVS systems and was shown to outperform all fully-automatic entries to the TRECVID AVS 2016 benchmarking activity.

Chapter 7

Conclusions and Future Work

Contents

7.1 Discussion and conclusions

In this thesis we studied the problem of video annotation and retrieval; our overall aim was to develop and evaluate machine learning architectures that optimally solve this problem. For the most of this thesis we focused on annotating video content with a set of pre-defined concept labels. Then, we further used this information in order to deal with the related and more challenging problem of annotating video content with ad-hoc queries. While significant progress has been made during the last years in the task of video annotation and retrieval, we found that it remains 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. The system needs to learn hundreds or thousands of concepts that belong to different categories (e.g. landscapes, faces, actions). As a result, generality is an important property that a concept-based video annotation system should present in order to generalize its performance across many different heterogeneous concepts.

Computational requirements is another major challenge. The large number of concepts that a video annotation system should learn is computationally expensive requiring lightweight and fast methods. Finally, the most of the existing architectures focus on the above two aspects, discriminative feature extraction and computational complexity, and ignore other important aspects that can lead to robust systems such as the multi-label nature of the problem and the implicit or explicit concept relations in feature- or semantic-level, respectively. In this thesis we showed that various crucial factors in video annotation should jointly been consider in order to develop a unique accurate architecture. We presented different machine learning architectures that solve this problem focusing on the most emerging directions of the video annotation and retrieval problem: feature extraction, classifier combination and concept relation modeling.

One of the best strategies for building accurate and discriminative concept classifiers is to extract a big pool of visual features, train concept classifiers separately for each of the extracted features and combine the classifier's outputs for the same concept. After studying the related work we could see that there is a lot of research towards feature extraction, with many authors proposing local descriptors and variations of them. However, the use of binary descriptors in the specific problem had not been examined before. Since binary descriptors are more compact and faster to be computed, which could be useful to applications with space and time constraints (e.g., mobile applications), we examined the use of them in the video annotation problem. We proposed color extensions of them and we also presented general strategies of the way that dimensionality reduction can be applied to binary and non-binary local descriptors. Our experimental results revealed that the propose binary local descriptors can perform reasonably well compared to their non-binary counterparts and also that the binary and non-binary local descriptors are complementary to each other because when they are combined for the same concept, then concept annotation accuracy can be further improved. In our first attempt of building a machine learning architecture for solving the concept-based annotation problem we trained concept classifiers independently

from each other (i.e., for each concept a set of classifiers for this concept are trained on different visual features ignoring the way that the training of the other concepts takes place). However, concepts in a video keyframe do not appear in isolation from each other. For example, the concepts *sun* and *sky* are expected to appear on the same keyframe for the most of the cases. Having this in mind we extended our video annotation architecture with a second layer that refines the scores returned by the first-layer independently trained concept classifiers, using a multi-label learning algorithm. This was referred as a two-layer stacking architecture and presented an 1.5% improvement in terms of MXinfAP compared to the first-layer independently trained concept classifiers. Related stacking architectures perform a second round of binary classifications, independently for each concept, this time taking as input all the concept scores of the first layer, thus implicitly consider concept relations. Our novelty is that we use a multi-label learning algorithm that exploits sets of concept labels that occur together in the spatial domain, thus explicitly considers such concept relations.

As discussed above one of our outcomes after evaluating our first video annotation architectures was that combining many different keyframe representations (e.g. SIFT, RGB-SIFT, ORB, BRISK) for the same concept, instead of using a single feature (e.g. only SIFT), improves the concept-based video annotation accuracy. Motivated by this, we tried to find a more accurate and fast way to combine the different base classifiers for each concept. Based on the general field of ensemble learning that has been applied to other machine learning problems, we know that classifier fusion techniques such as cascade architectures can impose discriminative power to the ensemble and improve the overall classification accuracy. For this reason we proposed a cascade architecture that can be served with many different base classifiers and optimally prune, order and combine them in order to perform fast and accurate concept-based video annotation. We assumed that different concepts require a different subset of the extracted features and also an optimal ordering of them could avoid querying classifiers that are not able to further improve the annotation accuracy. Although cascade architectures have been

proposed for other computer vision learning problems, e.g., object detection, we were the first to present a suitable variation for the specific problem of concept-based video annotation. The proposed cascade is computationally more efficient during classification without presenting significant improvements in terms of MXinfAP compared to other classifier combination approaches. As a result, the proposed cascade could be useful for applications that require small detection time, however it does not really provide any significant improvement compared to the simple late fusion scheme in terms of accuracy.

All of our attempts up to this point were towards improving machine learning architectures that are based on the training of independent concept classifiers (i.e., trained either on hand-crafted features or DCNN-based features). However, the great success of deep learning techniques also in the field of video annotation resulted to the gradual replacement of such independently trained classifiers with end-to-end deep learning architectures. Towards this direction we reviewed the related work and the limitations of the proposed methods and we developed our next machine learning architecture that is based on such deep learning techniques, referring to it as FV-MTL with CCE-LC. We have noted that in a typical DCNN architecture that is used for concept-based video annotation, the concepts share features up to the very last layer, and then branch off to T different classification branches (using typically one layer), where T is the number of concepts. However, in this way, the implicit feature-level relations between concepts, e.g. the way in which concepts such as a car and motorcycle share lower-level features modeling things like their wheels, are not directly considered. Also, in such architectures, the relations or interdependencies of the concepts at a semantic level, i.e. the fact that two specific concepts may often appear together or, inversely, the presence of the one may exclude the other, are also not directly taken into consideration. While some methods have been proposed for exploiting in a more elaborate way one of these two different types of concept relations, there is no single method that jointly exploits visual- and semantic-level concept relations in a unified DCNN architecture. Thus,

we propose a DCNN architecture that captures therefore both implicit and explicit concept relations, i.e., both visual-level and semantic-level concept relations. So in our proposed method, first, implicit concept relations are modeled in a DCNN architecture that learns T concept-specific feature vectors that are themselves linear combinations of $k < T$ latent concept feature vectors. In this way, in the shared representations (i.e., the latent concepts feature vectors), higher-level concepts may share visual features - for example, concepts such as car, motorcycle, and airplane may share features encoding the wheels in their depiction. Second, explicit concept relations are introduced by a new cost term, implemented using a set of standard CNN layers that penalize differences between the matrix encoding the correlations among the ground-truth labels of the concepts, and the correlations between the concept label predictions of our network. In this way, we introduced constraints on the structure of the output space by utilizing the label correlation matrix - this would explicitly capture, for example, the fact that *daytime* and *nighttime* are negatively correlated concepts. The experimental results were very promising revealing the usefulness of jointly exploiting visual-level and semantic-level concept relations. Our method reached a MXinfAP of 33.77% in the TRECVID SIN 2013 dataset, which is a state-of-the-art outcome for this dataset.

Finally, in the last main chapter of the thesis we dealed with the problem of ad-hoc video search (AVS). AVS is the problem of retrieving, from a large video collection, video fragments (e.g., video shots) that are related to a given query. A query refers to an ad-hoc textual description, e.g. "Find shots of a woman wearing glasses". This is a very new task that was proposed as a TRECVID benchmarking activity in 2016 replacing the TRECVID SIN task. Our first attempt of developing a fully automatic ad-hoc video search system seems very promising reaching the best position in the TRECVID AVS 2016 competition, however the overall results are still very low from considering the solution as satisfactory. Our method similar to related approaches uses a pre-defined set of concepts and converts both the query and each video keyframe to concept-based vectors using these concepts. Our novelties are the specific NLP steps

that are used to extract useful keywords and sub-sequences from the target query and a new approach that projects the video shots and the query representations into a common semantic embedding space using their word2vec representations.

Overall, the lessons we learned in this thesis is that a good video annotation and retrieval architecture can be developed by carefully taking into account many different directions such as feature extraction, classifier combination, feature-level and semantic-level concept relations. Deep learning architectures are the best way of jointly considering all these, with our proposed FV-MTL with CCE-LC deep architecture consistently outperforming other related state-of-the-art ones. Other attempts that we made towards improving older video annotation architectures that are based on independently trained concept classifiers although presented promising results, their accuracy could not reach the overall accuracy of deep learning approaches. Improving computational complexity was not our focus, however, all of the proposed architectures were designed in such a way that does not introduce any significant computational overhead. Also, our cascade architecture was computationally more efficient during classification compared to other methods making it useful for applications that require small concept annotation time.

7.2 Plans for future extensions

Although concept-based video annotation accuracy has reached satisfactory performance for facilitating specific applications such as concept-based video retrieval, the problem cannot be considered as solved. A possible direction of our future work could be the extension of our FV-MTL approach in order to support incremental learning. Incremental learning in DCNN training is indeed a very active field. Traditional DCNN training/finetuning is computationally expensive; once a new concept category arrives the full network should be fine-tuned/trained from scratch. As a result, a possible future direction could be to extend the proposed FV-MTL with CCE-LC architec-

ture, with an incremental learning methodology that will allow a trained network to be extended with new classes, without losing the ability of recognising the previously learned classes. Other possible future directions could focus on the improvement of the way that implicit concept relations are captured by the proposed FV-MTL method, for example, by introducing a non-linear function, and also the exploitation of different constraints in order to capture different kinds of semantic relations (i.e., not only correlations) between concepts.

Another direction that consists an active field of our future research is the improvement of our ad-hoc video search architecture. As far as, this problem is very new and has been investigated very little there are a lot of future plans towards this direction. For example, to investigate the use of different types of semantic embeddings, and also the influence of the different keyframe-query representations on different types of queries, i.e., in which cases concept-based representations outperform semantic embeddings or the combination of both. To increase the predefined pool of concepts and search for better word-to-vector representations but also sentence-to-vector representations. Existing word2vec neural network models could be extended with MTL and structured output learning techniques, similar to what we have used in our FV-MTL with CCE-LC method, in order to improve their discriminative power. Finally, keyword extraction methods could be examined and improved in order to better analyse the textual query into meaningful keywords.

Bibliography

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *CoRR*, abs/1609.08675, 2016. 27

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *IEEE Int. Conf. CVPR 2012*, pages 510–517, 2012. 13

[3] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems (NIPS 2007)*. MIT Press, 2007. 17, 18

[4] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008. 17, 18

[5] G. Awad, J. Fiscus, and M. M. et al. Trecvid 2016: Evaluating video search, video event detection, localization, and hyperlinking. In *TRECVID 2016 Workshop*. NIST, USA, 2016. 26, 27, 103, 104, 108, 109

[6] G. Awad, C. G. M. Snoek, A. F. Smeaton, and G. Quénot. Trecvid semantic indexing of video: A 6-year retrospective. *ITE Transactions on Media Technology and Applications*, 4(3):187–208, 2016. Invited paper. 109

[7] L. Bao et al. Informedia@TRECVID 2011. In *TRECVID 2011 Workshop*, Gaithersburg, MD, USA, 2011. 16, 65, 66, 68, 69

[8] M. Baumgartner. Uncovering deterministic causal structures: a boolean approach. *Synthese*, 170(1):71–96, 2009. 17, 23

[9] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, volume 3951 of *LNCS*, pages 404–417. Springer, 2006. 12, 13, 32

[10] E. Bingham and H. Mannila. Random projection in dimensionality reduction: Applications to image and text data. In *7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 245–250, NY, 2001. ACM. 43

[11] M. Bishay and I. Patras. Fusing multilabel deep networks for facial action unit detection. In *Proc. of the 12th IEEE Int. Conf. on Automatic Face and Gesture Recognition (FG)*, 2017. 17, 85, 94, 95, 97, 98, 100

[12] H. M. Blanken, A. P. de Vries, H. E. Blok, and L. Feng. *Multimedia Retrieval*. Springer Berlin Heidelberg, NY, 2005. 50, 98

[13] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *IEEE Int. Conf. ICCV 2007*, pages 1–8, Rio de Janeiro, 2007. 42

[14] X. Cai, F. Nie, W. Cai, and H. Huang. New graph structured sparsity model for multi-label image annotations. In *IEEE International Conference on Computer Vision (ICCV), 2013*, pages 801–808, 2013. 17, 23

[15] A. Canclini et al. Evaluation of low-complexity visual feature detectors and descriptors. In *18th Int. Conf. on Digital Signal Processing (DSP), 2013*, pages 1–7, 2013. 13

[16] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *British Machine Vision Conference*, pages 76.1–76.12. British Machine Vision Association, 2011. 14, 36, 42, 43

[17] K. Chellapilla, M. Shilman, and P. Simard. Combining multiple classifiers for faster optical character recognition. In *7th International Conference on Document Analysis Systems*, DAS'06, pages 358–367, Berlin, 2006. Springer. 16, 65, 66, 68, 69

[18] D. M. Chen, M. Makar, A. F. de Araújo, and B. Girod. Interframe coding of global image signatures for mobile augmented reality. In *DCC*, pages 33–42, 2014. 32

[19] W.-C. Cheng and D.-M. Jhan. A cascade classifier using adaboost algorithm and support vector machine for pedestrian detection. In *IEEE Int. Conf. on SMC*, pages 1430–1435, 2011. 16

[20] D. Chu and A. Smeulders. Color invariant surf in discriminative object tracking. In *Trends and Topics in Computer Vision*, volume 6554 of *LNCS*, pages 62–75. Springer, 2012. 13

[21] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR 2009)*, Santorini, Greece, July 8-10, 2009. 27, 86, 87

[22] G. Csurka and F. Perronnin. Fisher vectors: Beyond bag-of-visual-words image representations. In *Computer Vision, Imaging and Computer Graphics. Theory and Applications*, volume 229 of *Communications in Computer and Information Science*, pages 28–42. Springer Berlin, 2011. 12, 14

[23] H. Daumé, III. Bayesian multitask learning with latent hierarchies. In *Proc. of the 25th Conf. on Uncertainty in Artificial Intelligence (UAI '09)*, pages 135–142, Arlington, Virginia, US, 2009. AUAI Press. 17, 18

[24] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. *Large-Scale Object Classification Using Label Relation Graphs*, pages 48–64. Springer, Zürich, Switzerland, 2014. 17, 22

[25] J. Deng, S. Satheesh, A. C. Berg, and F. Li. Fast and balanced: Efficient label tree learning for large scale object recognition. In *Advances in Neural*

Information Processing Systems, pages 567–575. Curran Associates, Inc., 2011. 17, 23

[26] Z. Deng, A. Vahdat, H. Hu, and G. Mori. Structure inference machines: Recurrent neural networks for analyzing relations in group activity recognition. *CoRR*, abs/1511.04196, 2015. 17, 23

[27] N. Ding, J. Deng, K. P. Murphy, and H. Neven. Probabilistic label relation graphs with ising models. In *Proc. of the 2015 IEEE Int. Conf. on Computer Vision (ICCV 2015)*, pages 1161–1169, Washington, DC, USA, 2015. IEEE. 17, 22

[28] M. Elhoseiny, B. Saleh, and A. Elgammal. Write a classifier: Zero shot learning using purely textual descriptions. In *Int. Conf. on Computer Vision*, 2013. 24, 30

[29] D.-D. L. et al. NII-HITACHI-UIT at TRECVID 2016. In *TRECVID 2016 Workshop*, Gaithersburg, MD, USA, 2016. 24, 30, 110

[30] G. A. et al. TRECVID 2008–Goals, Tasks, Data, Evaluation Mechanisms and Metrics. In *TRECVID 2008 Workshop*. NIST, USA, 2008. 104, 108

[31] J. L. et al. Informedia @ Trecvid 2016. In *TRECVID 2016 Workshop*, Gaithersburg, MD, USA, 2016. 25, 110

[32] M. A. et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 12

[33] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. 86, 87

[34] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes

Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 27, 86, 87

[35] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proc. of the 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, KDD '04, pages 109–117, NY, USA, 2004. ACM. 17, 18

[36] P. Fan, A. Men, M. Chen, and B. Yang. Color-SURF: A surf descriptor with local kernel color histograms. In *IEEE Int. Conf. on Network Infrastructure and Digital Content*, pages 726–730, 2009. 13

[37] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998. 83

[38] Z. Fu, T. Xiang, and E. Kodirov et al. Zero-shot object recognition by semantic manifold distance. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 5, 25

[39] J. Fu et al. C-surf: Colored speeded up robust features. In Y. Yuan, X. Wu, and Y. Lu, editors, *Trustworthy Computing and Services*, volume 320 of *Communications in Computer and Information Science*, pages 203–210. Springer, 2013. 13

[40] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, 2008. 40, 43

[41] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611, 2007. 105

[42] N. Gkalelis, V. Mezaris, and I. Kompatsiaris. A joint content-event model for event-centric multimedia indexing. In *IEEE Int. Conf. on Semantic Computing (ICSC)*, pages 79–84, 2010. 1

[43] C. Grana, D. Borghesani, M. Manfredi, and R. Cucchiara. A fast approach for integrating orb descriptors in the bag of words model. In *SPIE*, volume 8667, pages 866709–866709–8, 2013. xv, 13, 35, 44, 46

[44] Y. Guangnan, Y. L., and H. X. et al. Eventnet: A large scale structured concept library for complex event detection in video. In *ACM MM*, 2015. 27

[45] A. Hamadi, P. Mulhem, and G. Quenot. Conceptual feedback for semantic multimedia indexing. In *11th Int. Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 53–58, 2013. 43, 49

[46] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. xiii, 12, 15, 17, 19, 75, 77, 87, 88, 94, 100

[47] A. Jalali, S. Sanghavi, C. Ruan, and P. K. Ravikumar. A dirty model for multi-task learning. In *Advances in Neural Information Processing Systems*, pages 964–972. Curran Associates, 2010. 73

[48] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *IEEE Int. Conf. on CVRP 2010*, pages 3304–3311, SF, CA, 2010. 12, 14, 33

[49] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1704–1716, 2012. 14, 42

[50] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 12, 88

[51] W. Jiang, S.-F. Chang, and A. C. Loui. Active context-based concept fusion with partial user labels. In *IEEE Int. Conf. on Image Processing*, NY, 2006. IEEE. 32, 43, 49

[52] Y.-G. Jiang, C.-W. Ngo, and J. Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *Proceedings of the 6th ACM Int. Conf. on Image and Video Retrieval*, CIVR '07, pages 494–501, NY, USA, 2007. ACM. 15

[53] Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):352–364, 2018. 27

[54] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 27

[55] T. Kenter, A. Borisov, and M. de Rijke. Siamese CBOW: optimizing word embeddings for sentence representations. *CoRR*, abs/1606.04640, 2016. 25

[56] R. Kiros, Y. Zhu, and R. Salakhutdinov et al. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302. Curran Associates, 2015. 25

[57] A. Krizhevsky, S. Ilya, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS 2012)*, pages 1097–1105. Curran Associates, Inc., 2012. 3, 12, 15, 64

[58] A. Kumar and H. Daume. Learning task grouping and overlap in multi-task learning. In J. Langford and J. Pineau, editors, *Proc. of the 29th Int. Conf. on Machine Learning (ICML-12)*, pages 1383–1390, NY, USA, 2012. ACM. 17, 19, 29, 73, 74, 78, 81, 95

[59] S. Leutenegger, M. Chli, and R. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *IEEE Int. Conf. ICCV 2011*, pages 2548–2555, 2011. 13, 28, 32, 34

[60] J. Li, Y. Wu, and K. Lu. Structured domain adaptation. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(8):1700–1713, Aug 2017. 20

[61] Y. Li, M. Yang, and Z. Zhang. Multi-view representation learning: A survey from shallow methods to deep methods. *CoRR*, abs/1610.01206, 2016. 20

[62] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004. 7, 12, 28, 32

[63] Y. Lu, W. Zhang, K. Zhang, and X. Xue. Semantic context learning with large-scale weakly-labeled image set. In *Proceedings of the 21st ACM Int. Conf. on Information and Knowledge Management*, CIKM '12, pages 1859–1863, NY, USA, 2012. ACM. 17, 23

[64] Y.-J. Lu, H. Zhang, and M. de Boer et al. Event detection with zero example: Select the right and suppress the wrong concepts. In *ACM Int. Conf. on Multimedia Retrieval (ICMR)*, ICMR '16, pages 127–134, NY, USA, 2016. ACM. 24, 30

[65] Q. Luo, S. Zhang, T. Huang, W. Gao, and Q. Tian. Superimage: Packing semantic-relevant images for indexing and retrieval. In *Proceedings of International Conference on Multimedia Retrieval*, ICMR '14, pages 41–48, NY, USA, 2014. ACM. 17, 23

[66] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008.*, pages 1–8, 2008. 15

[67] F. Markatopoulou and A. M. et al. Iti-certh participation to trecvid 2016. In *TRECVID 2016 Workshop*, Gaithersburg, MD, USA, 2016. 24, 30, 110

[68] F. Markatopoulou, V. Mezaris, and I. Patras. Cascade of classifiers based on binary, non-binary and deep convolutional network descriptors for video concept

detection. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP 2015)*, pages 1786–1790, 2015. 56, 64

[69] F. Markatopoulou, V. Mezaris, and I. Patras. Deep multi-task learning with label correlation constraint for video concept detection. In *Proc. of the Int. Conf. ACM Multimedia (ACMMM 2016)*, pages 501–505, Amsterdam, The Netherlands, 2016. ACM. 17, 20, 23, 29, 30, 73, 74, 94, 96, 100

[70] F. Markatopoulou, V. Mezaris, and I. Patras. Online multi-task learning for semantic concept detection in video. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP 2016)*, pages 186–190, Sept 2016. 17, 19, 29, 94, 96, 100

[71] F. Markatopoulou, V. Mezaris, N. Pittaras, and I. Patras. Local features and a two-layer stacking architecture for semantic concept detection in video. *IEEE Transactions on Emerging Topics for Computing*, 3:193–204, 2015. 17, 94, 95, 96, 100

[72] F. Markatopoulou et al. ITI-CERTH participation to TRECVID 2013. In *TRECVID 2013 Workshop*, Gaithersburg, MD, USA, 2013. 1, 43

[73] V. Mezaris, P. Sidiropoulos, A. Dimou, and I. Kompatsiaris. On the use of visual soft semantics for video temporal decomposition to scenes. In *IEEE Int. Conf. on Semantic Computing (ICSC)*, pages 141–148, 2010. 1

[74] 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)*, pages 224–227, 2011. 1

[75] T. Mikolov, I. Sutskever, and K. Chen et al. Distributed representations of words and phrases and their compositionality. In *26th Int. Conf. on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates. 25, 107

[76] H. Mousavi, U. Srinivas, V. Monga, Y. Suo, M. Dao, and T. Tran. Multi-task image classification via collaborative, hierarchical spike-and-slab priors. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP 2014)*, pages 4236–4240, 2014. 17, 18

[77] G. Nasierding and A. Z. Kouzani. Empirical Study of Multi-label Classification Methods for Image Annotation and Retrieval. In *2010 Int. Conf. on Digital Image Computing: Techniques and Applications*, pages 617–622, China, 2010. IEEE. 41

[78] C. Nguyen, H. Vu Le, and T. Tokuyama. Cascade of multi-level multi-instance classifiers for image annotation. In *KDIR'11*, pages 14–23, 2011. 16, 29

[79] M. Norouzi, T. Mikolov, and S. B. et al. Zero-shot learning by convex combination of semantic embeddings. *CoRR*, abs/1312.5650, 2013. 5, 24, 25, 30, 107, 110

[80] G. Obozinski and B. Taskar. Multi-task feature selection. In *Proc. of the 23rd Int. Conf. on Machine Learning (ICML 2006). Workshop of Structural Knowledge Transfer for Machine Learning*, Pittsburgh, Pennsylvania, 2006. 17, 18

[81] W. Ouyang, X. Chu, and X. Wang. Multi-source deep learning for human pose estimation. *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 0:2337–2344, 2014. 19, 20

[82] 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. xiv, 21, 25, 26, 44, 51, 64, 86, 87, 90, 99

[83] N. Pittaras, F. Markatopoulou, V. Mezaris, and I. Patras. Comparison of Fine-Tuning and Extension Strategies for Deep Convolutional Neural Networks. In *Proc. of the 23rd Int. Conf. on MultiMedia Modeling (MMM 2017)*, pages 102–

114, Reykjavik, Iceland, 2017. Springer. xiii, xvii, 5, 15, 73, 75, 77, 80, 87, 89, 91, 94, 95, 97, 98, 100, 109

[84] G.-J. Qi et al. Correlative multi-label video annotation. In *15th Int. Conf. on Multimedia*, pages 17–26, NY, 2007. ACM. 17, 23

[85] G. Qiu. Indexing chromatic and achromatic patterns for content-based colour image retrieval. *Pattern Recognition*, 35:1675–1686, 2002. 14

[86] J. Read. A pruned problem transformation method for multi-label classification. In *2008 New Zealand Computer Science Research Student Conference (NZCSRS)*, New Zealand, 2008. 40, 43

[87] B. Romera-Paredes and P. H. Torr. An embarrassingly simple approach to zero-shot learning. *32nd Int. Conf. on Machine Learning (ICML)*, 2015. 24, 30

[88] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *IEEE Int. Conf. on Computer Vision*, pages 2564–2571, 2011. 12, 13, 28, 32, 34

[89] O. Russakovsky, J. Deng, and H. S. et al. ImageNet Large Scale Visual Recognition Challenge. *Int. Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 27, 87, 109

[90] B. Safadi, N. Derbas, A. Hamadi, M. Budnik, P. Mulhem, and G. Qu. LIG at TRECVid 2014 : Semantic Indexing tion of the semantic indexing. In *TRECVID 2014 Workshop*, Gaithersburg, MD, USA, 2014. 12

[91] 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*, pages 2081–2084, NY, 2011. ACM. 44, 65

[92] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *CoRR*, abs/1503.02351, 2015. 17, 23

[93] P. Sidiropoulos, V. Mezaris, and I. Kompatsiaris. 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. xv, 12, 16, 42, 45, 46, 48, 65, 66, 68, 69

[94] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR, abs/1409.1556*, 2014. 3, 12, 15, 17, 19, 64, 75, 101

[95] J. Smith, M. Naphade, and A. Natsev. Multimedia semantic indexing using model vectors. In *2003 Int. Conf. on Multimedia and Expo. (ICME)*, pages 445–448, NY, 2003. IEEE. 17, 22, 28, 32, 43, 49

[96] C. Snoek, D. Fontijne, K. E. van de Sande, and H. e. a. Stokman. Qualcomm research and university of amsterdam at trecvid 2015: Recognizing concepts, objects, and events in video. In *Proc. of TRECVID 2015*. NIST, USA, 2015. 15

[97] C. G. M. Snoek, K. E. A. V. D. Sande, D. Fontijne, S. Cappallo, J. V. Gemert, and A. Habibian. MediaMill at TRECVID 2014 : Searching Concepts , Objects , Instances and Events in Video. In *TRECVID 2014 Workshop*, Gaithersburg, MD, USA, 2014. 12

[98] C. G. M. Snoek and M. Worring. Concept-Based Video Retrieval. *Foundations and Trends in Information Retrieval*, 2(4):215–322, 2009. 1, 2, 12

[99] R. Socher, M. Ganjoo, and C. D. Manning et al. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems*, pages 935–943. Curran Associates, 2013. 5, 25

[100] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. 27

[101] S. Strat, A. Benoit, P. Lambert, and A. Caplier. Retina enhanced surf descriptors for spatio-temporal concept detection. *Multimedia Tools and Applications*, 69(2):443–469, 2014. 13

[102] 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*, volume 7585 of *Lecture Notes in Computer Science*, pages 335–344. Springer, 2012. 16, 65, 66, 68, 69

[103] S. T. Strat, A. Benoit, and P. Lambert. Retina enhanced bag of words descriptors for video classification. In *22nd Europ. Signal Processing Conf. (EUSIPCO)*, pages 1307–1311, 2014. 13

[104] L. E. Sucar, C. Bielza, E. F. Morales, P. Hernandez-Leal, J. H. Zaragoza, and P. Larrañaga. Multi-label classification with bayesiannetwork-based chain classifiers. *Pattern Recognition Letters*, 41:14 – 22, 2014. 17, 23

[105] G. Sun, Y. Chen, X. Liu, and E. Wu. Adaptive multi-task learning for fine-grained categorization. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP 2015)*, pages 996–1000, Sept 2015. 17, 19, 94, 95, 96, 100

[106] C. Szegedy et al. Going deeper with convolutions. In *CVPR 2015*, 2015. 12, 64

[107] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Proc. of the 16th Int. Conf. on Neural Information Processing Systems (NIPS 2003)*. MIT Press, 2003. 17, 23

[108] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–686. Springer, Berlin, 2010. 40, 43

[109] G. Tsoumakas, E. Spyromitros-xioufis, J. Vilcek, and I. Vlahavas. MULAN : A Java Library for Multi-Label Learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011. 44, 52

[110] G. Tsoumakas et al. Correlation-Based Pruning of Stacked Binary Relevance Models for Multi-Label learning. In *ECML/PKDD 2009 Workshop on Learning*

from Multi-Label Data (MLD'09), pages 101–116, Berlin, 2009. Springer-Verlag. 43, 49

[111] C. Tzelepis, D. Galanopoulos, and V. Mezaris et al. Learning to detect video events from zero or very few video examples. *Image Vision Computing*, 53:35–44, Sept. 2016. 24, 30, 107, 110

[112] K. Ueki, K. Kikuchi, and T. Kobayashi. Waseda at TRECVID 2016: Ad-hoc Video Search. In *TRECVID 2016 Workshop*, Gaithersburg, MD, USA, 2016. 24, 110, 111

[113] 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*, 32(9):1582–1596, 2010. 12, 13, 33, 35, 36

[114] K. E. A. Van de Sande, C. G. M. Snoek, and A. W. M. Smeulders. Fisher and vlad with flair. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 14

[115] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. In *Int. Conf. on Computer Vision*, pages 1879–1886, Nov 2011. 101

[116] 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)*, volume 1, pages 511–518, 2001. 16

[117] H. Wang, H. Huang, and C. Ding. Image annotation using multi-label correlated green's function. In *IEEE 12th Int. Conf. on Computer Vision*, pages 2029–2034, Sept 2009. 17, 23

[118] H. Wang, H. Huang, and C. Ding. Image annotation using bi-relational graph of images and semantic labels. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 793–800, June 2011. 17, 23

[119] M. Wang, C. Luo, R. Hong, J. Tang, and J. Feng. Beyond Object Proposals: Random Crop Pooling for Multi-Label Image Recognition. *IEEE Transactions on Image Processing*, 25(12):5678–5688, Dec 2016. 17, 20, 94, 95, 100, 101

[120] M. Wang, X. Zhou, and T.-S. Chua. Automatic image annotation via local multi-label classification. In *Int. Conf. on Content-based image and video retrieval - CIVR '08*, pages 17–26, NY, 2008. ACM. 41

[121] Y. Wei, W. Xia, M. Lin, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. Hcp: A flexible cnn framework for multi-label image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1901–1907, Sept 2016. 17, 100, 101

[122] M.-F. Weng and Y.-Y. Chuang. Cross-Domain Multicue Fusion for Concept-Based Video Indexing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34(10):1927–1941, 2012. 17, 22, 43, 49

[123] I. Witten and E. Frank. *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, second edition, 2005. 14, 33, 36, 44

[124] G. S. Xie, X. Y. Zhang, S. Yan, and C. L. Liu. Hybrid cnn and dictionary-based models for scene recognition and domain adaptation. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(6):1263–1274, June 2017. 20

[125] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016. 12

[126] X. Xu, T. M. Hospedales, and S. Gong. Discovery of shared semantic spaces for multiscene video query and summarization. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(6):1353–1367, June 2017. 20

[127] Y. Yang and T. M. Hospedales. A unified perspective on multi-domain and multi-task learning. In *ICLR 2015*, 2015. 17, 19, 20, 94, 95, 96, 100

[128] Y. Yang and T. M. Hospedales. Deep multi-task representation learning: A tensor factorisation approach. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017. 19

[129] Y. Yang, F. Wu, F. Nie, H. T. Shen, Y. Zhuang, and A. G. Hauptmann. Web and personal image annotation by mining label correlation with relaxed visual graph embedding. *IEEE Transactions on Image Processing*, 21(3):1339–1351, March 2012. 17, 23, 94, 95, 96, 100

[130] 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*, pages 603–610, USA, 2008. ACM. 26, 41, 66, 88, 109

[131] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014. 19

[132] X. Zhai, Y. Peng, and J. Xiao. Learning cross-media joint representation with sparse and semisupervised regularization. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(6):965–978, June 2014. 20

[133] H. Zhang, L. Pang, Y. Lu, and C. Ngo. VIREO @ TRECVID 2016: Multimedia Event Detection, Ad-hoc Video Search, Video to Text Description. In *TRECVID 2016 Workshop*, Gaithersburg, MD, USA, 2016. 24, 30, 110

[134] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *Int. Journal on Computer Vision*, 73(2):213–238, 2007. 15

[135] M.-L. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, KDD '10, pages 999–1008, NY, USA, 2010. ACM. 17, 23

[136] M.-L. Zhang and Z.-H. Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007. 40, 43

[137] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *Proc. of the 13th Europ. Conf. on Computer Vision (ECCV 2014)*, pages 94–108, Zurich, Switzerland, 2014. Springer. 17, 19

[138] X. Zhao, X. Li, and Z. Zhang. Joint structural learning to rank with deep linear feature learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(10):2756–2769, Oct 2015. 17, 23

[139] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *Proc. of the Int. Conf. on Computer Vision (ICCV 2015)*, 2015. 17, 23

[140] B. Zhou, A. Lapedriza, and J. e. a. Xiao. Learning deep features for scene recognition using places database. In *NIPS*, pages 487–495, 2014. 27

[141] J. Zhou, J. Chen, and J. Ye. Clustered multi-task learning via alternating structure optimization. *Advances in Neural Information Processing Systems (NIPS 2011)*, 2011. 17, 19, 94, 95, 96, 100