Title
Video and Image Analysis Using Local Information

Author
Lian, Xiaochen

Publication Date
2017

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Video and Image Analysis Using Local Information

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Statistics

by

Xiaochen Lian

2017

ABSTRACT OF THE DISSERTATION

Video and Image Analysis Using Local Information

by

Xiaochen Lian
Doctor of Philosophy in Statistics
University of California, Los Angeles, 2017
Professor Alan Loddon Yuille, Chair

Local information is very crucial in many image and video analysis tasks. In this thesis, we introduce four representative works in exploiting local information. We first introduce a set of per-pixel labeling datasets, which provide a good platform for studies of using local information in image analysis. Based on this dataset, we propose a novel segmentation method which utilizes local appearance consistency for car semantic part parsing task. We then address the attention issue in video action recognition tasks, by designing a latent attention module, which is jointly learned with video recognition components. Last, we improve the attention mechanism to explicitly detect spatial and spatio-temporal regions that are related to actions (ROIs).

The dissertation of Xiaochen Lian is approved.

Yingnian Wu

Hongjing Lu

Adnan Youssef Darwiche

Alan Loddon Yuille, Committee Chair

University of California, Los Angeles

2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENT

I am sincerely thanksful to many people for their help and support along the way.

First and foremost, to my advisor, Alan L. Yuille. He is a smart person. He has deep understanding about the fundamental problems in this field, and therefore can quickly find out what is the problem is and give me suggestion. I'm often surprised by his ability to connect different problems. He is a considerate advisor. He always gives me the freedom to try my ideas and help me regardless of his interests. I'm also impressed by his passion and inspire by it. We can talk over hours without feeling tired. Most importantly, I feel like my personality has been influenced by him in a good way.

To my thesis committee, Professor Adnan Y. Darwiche, Professor Ying Nian Wu, and Professor Hongjing Lu, for their feedback and advice of this work.

To many great researchers and engineers whom I was lucky to collaborate with during the pursuit of Ph.D. degree. Thank Zhuoyuan Chen, Jiang Wang, Wei Xu and Yi Yang, whose had helped me during my internship at Baidu. I will never forget their kindness and patience. Thank Zhiwei Li, Changhu Wang, Lei Zhang, Xuezheng Liu, Xi Wang for their help and advice during my internship at Microsoft Research Asia. Thank Professor Bao-Liang Lu and all professors and teachers that I had class with during my study at Shanghai Jiao Tong University. They help me build up a solid foundation of math and computer science, which I have benefited, am benefiting and will benefit in the future.

Last but not least, to my family for their support. Thank my father, Huanxing Lian and mother Hua Chen, for their unconditional love, understanding, and support. Thank my girlfriend, Xin Lu, for accompanying me throughout the journey. I could not write any sentence to express my love sufficiently to them.

VITA

2004-2008	B.E. in Computer Science and Engineering, Shanghai Jiao Tong University.
2008-2011	M.S. in Computer Science and Engineering, Shanghai Jiao Tong University.
2011-2017	Ph.D Candidate in Department of Statistics, University Of California, Los Angeles.

PUBLICATIONS

Wenhao Lu, Xiaochen Lian and Alan Yuille, Parsing Semantic Parts of Cars Using Graphical Models and Segment Appearance Consistency, *B*MVC 2014

Bing Li, Xiaochen Lian and Bao-Liang Lu, Gender classification by combining clothing, hair and facial component classifiers, Neurocomputing 76(1), 18–27, 2012

Tianxiang Wu, Xiaochen Lian and Bao-Liang Lu, Multi-view gender classification using symmetry of facial images, Neural Computing and Applications 21(4), 661–669, 2012

Mu Li, Xiaochen Lian, James Kwok and Bao-Liang Lu, Time and Space Efficient Spectral Clustering via Column Sampling, *C*VPR 2011

Xiao-Chen Lian, Zhiwei Li, Bao-Liang Lu and Lei Zhang, Max-Margin Dictionary Learning for Multiclass Image Categorization, *E*CCV 2010

Xiao-Chen Lian, Zhiwei Li, Changhu Wang, Bao-Liang Lu and Lei Zhang, Probabilistic Models for Supervised Dictionary Learning, *C*VPR 2010

Xiao-Chen Lian, and Bao-Liang Lu, Gender Classification Combining Facial and Hair Information, *I*CONIP 2008

Xuezheng Liu, Zhenyu Guo, Xi Wang, Feibo Chen, Xiao-Chen Lian, Jian Tang, Ming Wu, M. Frans Kaashoek, and Zheng Zhang, D3S: Debugging Deployed Distributed Systems, *N*SDI 2008

CHAPTER 1

Introduction

When processing visual inputs coming from the environment, human does not have a detailed and coherent representation of the entire scene. Instead, human can focus on local regions, *e.g.*, locations where are salient, where task-related objects are at, or where motion happens.

In the field of computer vision, human visual perception mechanism has been considered as the oracle and the target model of computer vision algorithms. The idea of using local information has been practiced by computer vision researchers for a long time. First, various local descriptors have been proposed to capture local low-level appearance or semantic information. In image analysis, we have Scale Invariant Feature Transform (SIFT) [Low04], Histogram of Oriented Gradients (HOG) [DT05] and so on. In video applications, we have Histograms of Optical Flow (HOF) [LMS08] and so on. These descriptors are aiming to capture the local appearance. Also, a lot of studies have been done to make local representation more semantic meaningful, such as mid-level patches (*e.g.*, poselet [BM09a]), supervised dictionary [LLL10] and spatio-temporal sub-volumes (*e.g.*, actons [ZWY13]). Then to aggregate local descriptors, Hierarchical Graphic Models [ZM07, JG06, ZCY10] or Tree-Structured Models [FH05] have been proposed.

Recently, deep learning techniques, especially deep convolutional neural network (DCNN), have achieved great success in various image and video analysis tasks. DCNNs are usually operated on a very large field of view (a typical spatial range is hundred pixels by hundred pixels), yet they outperform existing shallow models even on tasks that requires very local and detailed cues (*e.g.*, image segmentation [CPK15] and fine-grained classification [LRM15]). However, this does not mean local information becomes less important. In fact, DCNN can be seen, implicitly, as a unified model of local descriptor and global aggregation, both of

which are learned jointly. There also has been a trend of explicitly integrate local information with deep learning techniques.

My PhD research has always been driven by ideas about using local information to help improve computer vision tasks. This thesis will present my research effort in this direction. It starts with the area of image analysis. In order to better study and benchmark algorithms that use local information in image tasks, we need a dataset with detailed annotations. Per-pixel labeling image datasets are very good data sources for such purpose. My first and also long-term project is to provide detailed per-pixel labeling on PASCAL VOC 2010 images [EVW10a]. This includes a range of different, but related, annotations which are suitable for a variety of visual tasks required for image understanding which involving high precision tasks requiring per-pixel accuracy. More specifically, we conduct three related annotations. Firstly, we provide a PASCAL Semantic Segmentation dataset, known as the PASCAL Context dataset, which gives per-pixel annotations for up to 459 categories, of which 60 categories are most frequently represented. Secondly, we derive a Boundary Segmentation Dataset which labels the boundaries between different objects and background regions (those labeled in the PASCAL Semantic Segmentation dataset). Thirdly, we present the PASCAL Semantic Part dataset, which provides object semantic part segmentation for 16 of the PASCAL VOC objects, where each object is further partitioned into its semantic parts. We believe these datasets can help the vision and machine learning communities for fine grained and detailed object understanding.

With the help of the per-pixel labeling image dataset, I start to study object semantic part segmentation and benchmark my algorithms on the dataset. I conducted the project to parse car parts (*i.e.*,wheels, body, windows, lights and license plates). The problem is formulated as landmark identification. We first select representative locations on the boundaries of the parts to serve as landmarks. They are selected so that locating them yields the silhouette of the parts, and hence enables us to do object part segmentation. We use a mixture of graphical models to deal with different viewpoints so that we can take into account how the visibility and appearance of parts alter with viewpoint. The novel aspect of our graphical model is that we couple the landmarks with the segmentation of the image to exploit the

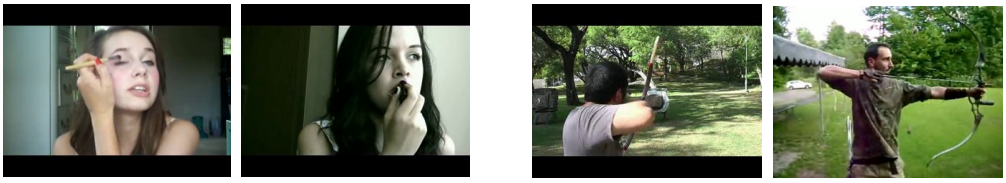


Figure 1.1: The challenges in video action recognition.

image contents when modeling the pairwise relation between neighboring landmarks. In the ideal case where part boundaries of the cars are all preserved by the segmentation, we can assume that the landmarks lie near the boundaries between different segments. Each landmark is then associated to the appearance of its two closest segments. This enables us to associate appearance information to the landmarks and to introduce pairwise coupling terms which enforce that the appearance is similar within parts and different between parts. We call this segmentation appearance consistency (SAC) between segments of neighboring landmarks.

Compared with images, videos are closer to what our visual system is processing, as most of the time, we are perceiving the relative movement between us and the environment. There are two major challenges in video recognition. First, difference between two actions are sometimes subtle. For example, as shown on the left of Figure 1.1, applying eye makeup and applying lipstick looks very similar, with the difference of where the hand is moving around (eyes vs. mouth). Second, within the same action category, due to the variation of lighting, view point and actor's pose, videos can look quite differently, as shown on the right of Figure 1.1. Using local regions is a very natural solution to this issues. There have been lots of studies of the framework that incorporates location information. One typical framework consists of three components: location sampling, hand-crafted representation and feature encoding. Location sampling (*e.g.*, spatial-temporal interesting points and dense trajectories) detects where the actions are likely happening or where contain action-related cues. Then hand-crafted features are extracted at these locations. Last, feature encoding methods (*e.g.*, Fisher Vector and VLAD) are applied to aggregate local features.

Motivated by the success of deep learning techniques, researchers start to exploit the combination of local information and DCNN. My first attempt in this direction is a video

action recognition framework that is able to estimate the action critical places of a video and infer the action happening in the video by attending only to relevant places in each frame. The framework consists of two parts: a classification module and a soft attention module. The classification module applies a fully convolutional neural network (FCN) [LSD15], which produces a dense classification score map for the input video. On the other hand, the attention module, computes an dense attention map with the same size as the classification score map, based on the same input video. The final output of our model is produced by the weighted sum of classification scores across all locations. Intuitively, each value at a location of the attention map indicates how much "attention" we should pay to the classification result at that location.

The attention learned from the above project is only latent, due to the lack of related supervision information. To improve it, we need to have a more accurate attention detection mechanism. Oftentimes, action-related information exists in certain spatial and spatio-temporal regions of interests (ROIs). For example, regions of single video frames (*i.e.*,static ROIs) include not only the people performing the action but also the objects that people interact with (*e.g.*, bicycles in *Biking*) or which often co-occur with the actions (*e.g.*, basket board in *Basketball*). Similarly, the spatio-temporal ROIs can track motion of the entire body, the motion of body parts, the movements of objects (*e.g.*, barbell in *Clean and Jerk*), and background motion (*e.g.*, sea waves in *Surfing*). These considerations motivate us to propose a video action recognition method that attends to regions of the videos. I design an algorithm which can proposes candidates of action-related regions (ROIs). Information in ROIs could be noisy, as some of ROIs are irrelevant to the actions or even causing confusions. This issue becomes worse when only video-level labeling is available. To solve this, we use multiple instance learning (MIL), where a video frame or a video clip is a "bag" and the ROIs are its "instances". In the mining component, we propose a novel aggregation module that learns to robustly combine instance features. We combine MIL with deep convolutional neural networks (CNNs) to enable joint learning of ROI mining and deep features.

In sum, I have investigated the following tasks in the direction of exploiting local information for vision tasks during my PhD:

1. Per-pixel image annotation dataset.

2. Car semantic part parsing using local appearance consistency.

3. Learning latent attention for video recognition.

4. Mining spatial and spatio-temporal ROIs for video recognition.

The thesis is organized as follows. In Chapter 2, we introduce the per-pixel labeling datasets, which provides a good benchmark for studies of using local information. Then in Chapter 3, we present a novel segmentation algorithm which incorporates local appearance consistency into consideration. After that, we switch to video recognition tasks. In Chapter 4, we presents our model with latent attention module. Then in Chapter 5 we describe a improved model that explicitly addresses the candidate ROIs.

CHAPTER 2

PASCAL Per-Pixel Labeling Dataset

Per-pixel labeling image datasets are very good data sources for training and testing algorithms that use local information. In this chapter, we describe a long-term image annotation project which provides detailed per-pixel labeling on PASCAL VOC 2010 images [EVW10a]. This includes a range of different, but related, annotations which are suitable for a variety of visual tasks required for image understanding which involving high precision tasks requiring per-pixel accuracy. More specifically, we describe three related annotations. Firstly, we provide a PASCAL Semantic Segmentation dataset, known as the PASCAL Context dataset, which gives per-pixel annotations for up to 459 categories, of which 60 categories are most frequently represented. Secondly, we derive a Boundary Segmentation Dataset which labels the boundaries between different objects and background regions (those labeled in the PASCAL Semantic Segmentation dataset). Thirdly, we present the PASCAL Semantic Part dataset, which provides object semantic part segmentation for 16 of the PASCAL VOC objects, where each object is further partitioned into its semantic parts. We believe these datasets can help the vision and machine learning communities for fine grained and detailed object understanding.

2.1 Introduction

Humans can extract an enormous amount of information from natural images. We can detect the objects, infer their positions, and even reconstruct the 3D structure of the scene. Extracting this information requires the ability to solve a variety of detailed visual tasks including *s*emantic segmentation, *b*oundary detection, *s*emantic part segmentation, and *i*nstance seg-

mentation. In order for vision researchers to address these visual tasks it is desirable to have large annotated datasets on challenging images which can be used to train and test vision algorithms.

This chapter provides benchmarked datasets for all these four visual tasks. We argue that datasets of these types are of particular importance as researchers build on the successes of object detection (e.g., the "cat in the box" problem) to more detailed and challenging tasks such as image parsing and image understanding. With the increasing interests in this area, it is necessity to have a platform for researchers to evaluate their methods. The platform should have a publicly available dataset of challenging images and annotations, and a standard evaluation methodology so that performance of algorithms can be compared.

The purpose of this chapter is to describe a set of four datasets which provide per-pixel annotations. They are designed to serve the tasks of semantic segmentation, boundary detection, and semantic segmentation of object parts. Studies on these datasets can provide important cues and information about the objects in the images and the scenes structure, which will be very helpful for solving many other vision tasks, such as image parsing and scene understanding.

The datasets were constructed using PASCAL VOC 2010 images since these were carefully chosen to be representative of natural images. They include: (i) a large range of viewing conditions (pose, lighting, etc.); (ii) objects of various sizes and at arbitrary locations; (iii) images of objects with cluttered background and occlusions. These datasets are large which makes them well suited for recent machine learning techniques, such as deep networks, which rely on large amounts of training data. We focus on PASCAL VOC 2010 for all this annotations because they complement each other.

PASCAL Semantic Segmentation Dataset (PASCAL Context). Semantic segmentation is a classic visual task dating from the last century, e.g., see [KY00]. Previous datasets have either restricted themselves to only labeling foreground objects (*e.g.*, PASCAL VOC [EVW10a] and Microsoft COCO [LMB14a]), or contain far fewer images (*e.g.*, MSRC [SJC08], Sowerby [HZR06] and San Francisco [HZR06]). In this dataset, we label ev-

ery pixel of the training and validation sets of the PASCAL VOC 2010 detection challenge. This involves labeling multiple background classes (e.g., sky, water, and grass) as well as objects. This new dataset has higher class entropy, and most pixels belong to a wide variety of object categories beyond the 20 PASCAL object classes.

PASCAL Semantic Boundary Dataset (PASCAL Boundary). Performance on current edge detection datasets, such as the BSDS dataset [MFT01], is becoming saturated and there is need for a much larger dataset to suit the new generation of machine learning methods. Moreover, although datasets like BSDS contain boundaries of objects they also contain other internal edges [HYK13]. Segmentation annotations from Datasets like Microsoft COCO [LMB14a] and PASCAL Segmentation [EVW10a] only contain a fraction of boundaries (roughly one half) because those datasets only label the boundaries of foreground objects and neglect the backgrounds (e.g., sky and water) and other objects. In our semantic boundary dataset, we provide instance-level labels for objects and with the labels we derive the semantic boundary from the semantic segmentation labeling.

PASCAL Semantic Part Segmentation Dataset (PASCAL Part). Part segmentation is a visual task that has started attracting attention. It is a pre-require for tasks such as action recognition and image parsing. To the best of our knowledge, there have not been datasets for part segmentation except for humans [YKO12, DCX13, BF11a, WSS07, BM09b] and vehicles [TFL08b]. Our new dataset provides semantic part segmentation for 16 out of 20 categories from PASCAL VOC 2010 dataset. To ensure fair comparisons, we build a benchmark, together with an evaluation server. The benchmark currently uses 7 articulated categories, due to their popularity in part-based methods and significant variability in terms of their poses and the sizes and shapes of their parts. An evaluation toolkit is also provided to enable a "plug and play" training and testing harness.

This paper is organized as follows. In section (2.2) we discuss related datasets for the four visual tasks. The following four sections (2.3,2.4 and 2.5) describe the semantic segmentation, boundaries, semantic part segmentation and instance segmentation datasets respectively. In all cases we describe the annotation strategy, the evaluation criteria, give statistical analysis of how they relate to alternative datasets, discuss performance on the

datasets and transfer to other datasets. Section (2.6) gives a discussion of future work.

2.2 Background: Related Datasets

The computer vision community has constructed many challenging annotated datasets, such as PASCAL VOC [EVW10a]. These have served to greatly advance the performance of computer vision algorithms by providing benchmarks for comparison and enabling the use of machine learning methods. In this section we briefly review the datasets most similar to those in this paper.

2.2.1 Semantic Segmentation

The Sowerby dataset (British Aerospace) was one of the first datasets to enable computer vision researchers to benchmark semantic segmentation algorithms. This dataset contained roughly 100 images annotated with XXX labels, Another samll dataset of 50 San Francisco images was provided by [KY00]. These datasets concentrated on mainly on "background regions", such as sky and road, and only labeled a few objects. The larger MSRC [SJC08] raised interest in semantic segmentation and contained xxx images and yyy classes. It played an important role in the study of semantic segmentation but its limited size eventually led to performance saturation and failure to transfer to other datasets.

The PASCAL segmentation dataset [EVW10b] is more challenging and much larger than MSRC but it restricts itself to labeling the boundaries of the twenty main foreground objects in PASCAL and treats the background as a single class (i.e. it does not include labels like sky and water). The COCO dataset [LMB14b] includes more images and objects but also only has a single background class. Hence both differ from the PASCAL Context dataset [MCL14a] described in this paper which has 59 frequently occurring labels including several different types of backgrounds, see sections (2.3,2.2) for more detailed comparisons.

There are other datasets which contain per-pixel labeling and include multiple background class. The SIFT flow dataset contains 2688 images and has 33 semantic labels

including background reshions [LYT09]. SUN2012 [XHE10] is a subset of LabelMe, which contains 16873 images and 3819 object classes, though many have only a few training examples. Barcelona [TL10] is another subset of LabelMe, which includes 15150 images and 170 categories. Other datasets have per-pixel labeling of material properties. There are also datasets of indoor scenes [SKH12] which contains 1449 RGB-D images and 894 object labels.

2.2.2 Boundary Segmentation Dataset

The Sowerby database of outdoor images [BP98] also gave annotations for edges and was used for statistical edge detection [KYC99, KYC03]. Another smaller (50 images) edge detection dataset, called the South Florida dataset, was presented in [BP98]. This removed textured regions of the images and concentrated more on edge localization than edge detection. It was observed [KYC99, KYC03] that while many edge detection methods, such as traditional approaches like Canny [Can86], performed well on South Florida only the statistical methods did well on Sowerby because they alone were able to avoid false positive edges in the texture regions.

The BSDS dataset originally contained 300 images [MFT01, MFM04] but was later extended to 500 images in [AMF11a]. This was developed because of the limited size of Sowerby and the concerns that the edges were found by running a set of edge detectors and using annotators to prune out the false positives. Instead BSDS was annotated by assigning about five annotators to each image, giving them intentionally vague instructions to label what they considered to be edges. This meant that there was considerable variability between different annotators, for example some would label detailed edges which others would only label coarse one. Psychophysics studies [HYK13] tested the reliability of these annotations and concluded that edges which were labelled by three or more annotators were typically real edges those labelled by only one or two annotators where problematic, in the sense that there were many unlabelled pixels in the image which appeared equally "edge-like" to observers in the study.

Although BSDS has significantly improved the state of the art of edge detection, e.g., see [AMF11a], BEL [DTB06], there is growing evidence that performance on it is becoming saturated due to the success of recent methods such as random forests and Sketch Tokens [LZD13], SE [DZ14] and deep neural networks like HED [XT15]. Larger datasets may be needed to take advantage of the computational power of deep network methods. We also note that BSDS labels internal edges as well as boundary edges. This motivates us to propose the PASCAL boundary segmentation dataset. Endres and Hoiem [EH10] cleaned up the BSDS300 dataset by grouping multiple segments within an object into a single object. Another related work is that of Zhu et al. [ZTM15], who recently proposed an amodal segmentation dataset, where they label the complete extents of an object (even if they are occluded) on the 500 images of the BSDS500 dataset.

In addition other boundary datasets can be obtained from several of the semantic segmentation datasets mentioned above [EVW10b] [LMB14b] [SKH12]. This is only possible because these datasets are highly accurate near the boundaries. We will return to these datasets in the boundary segmentation section.

2.2.3 Semantic Part Segmentation

Semantic part segmentation is like a more detailed way of doing segmentation. For example, instead of labeling pixels as "horse" we label them as "horse head", "horse torso", and so on. The most popular semantic part segmentation datasets are for humans. These include *F*ashionista (FS) Dataset [YKO12] which consists of 685 photographs of models, with labels for clothing items (*e.g.*, t-shirt, blouse, bag) and human hair and skin. Another related dataset is *D*aily Photos (DP) Dataset [DCX13] with 2500 images. In [BF11a], Bo and Fowlkes created a pedestrian parsing dataset, which contains 937 training images from HumanEva [SBB10], and 170 testing images with 345 labeld pedestrians from Penn-Fudan database [WSS07]. The authors provide segmentations into "hair", "face", "upper clothes", "arms", "lower clothes", "legs" and "background". *C*UHK Pedestrian Parsing (PPSS) Dataset [LWT13] contains 3,673 images from 171 videos of different surveillance

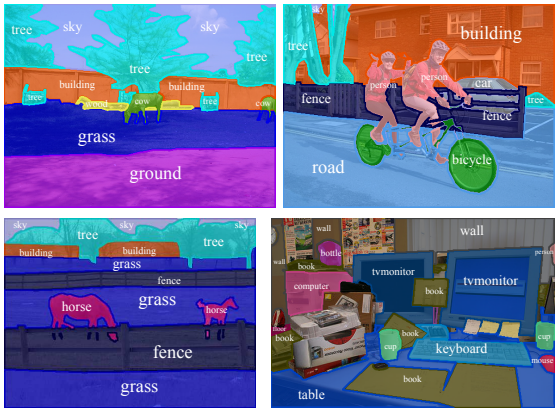


Figure 2.1: PASCAL Context: Examples of our annotations, which contain semantic segmentation of 459 categories in the PASCAL VOC 2010.

scenes, with the same part definitions as in [BF11a].

One drawback of these datasets is that the images are largely without clutter, occlusion. or large pose variations (most people are standing). By contrast, the PASCAL images are far more varied and more typical of real world situations. Also the number of humans in PASCAL is much larger (7296 instances). Bourdev and Malik [BM09b] have labeled PASCAL images with clothing items and human hair and skin. In our dataset, for human we use a more pose-oriented definition of parts, which is based on the skeletal system (*e.g.*, left lower arm, torso). Besides human, we also include other 15 categories, which makes our dataset more comprehensive. We have previously introduced this dataset in [CML14] and have also presented mofified subsets of it on cars [LLY14] and some animals (horses and cows) [WY15].

2.3 The Semantic Segmentation Dataset: PASCAL Context

PASCAL context [MCL14b] contains pixel-wise labels for the 10,103 *trainval* images of the PASCAL VOC 2010 detection challenge (Fig. 2.1 shows example labels). There are 459 categories in the dataset, divided into three types: (i) objects, (ii) stuff and (iii) hybrids. *O*bjects are classes that are defined by shape. This includes the original 20 PASCAL categories as well as classes such as fork, keyboard, and cup. *S*tuff denotes classes that do not have specific shape and appear as regions in images, e.g., sky, water. *H*ybrid classes

are classes for which shape is so variable that it cannot be easily modeled, e.g., roads have clear boundaries (unlike sky), but their shape is more complex than the shape of a cup. In practice, only 59 of these types happen frequently.

2.3.1 Annotation strategy for PASCAL Context

Our annotation effort took three months of intense labeling performed by six in-house annotators. This resulted in much more accurate segmentations than when using online systems such as MTurk. While this increased the labeling cost significantly, we wanted to assure the highest possible accuracy and consistency of the annotations. The annotators were asked to draw a region and assign it a label using an interface similar to LabelMe [RTM08]. There are about 12 regions in each image on average and the annotators spent about 3 to 5 minutes per image.

We provided the annotators with an initial set of 80 carefully chosen labels and asked them to include more classes if a region did not fit into any of these classes. Some cases were ambiguous to annotate; for example, the annotators were not sure how to label a tree visible through a window. We decided to go for a rich set of annotations, and thus allowed some pixels to have multiple labels (tree and window in this example). If the annotators were unable to recognize a region, they labeled it as unknown. We double checked each annotation and revised the ones that were not coherent in terms of category name or the region covering the object.

2.3.2 Analysis and Comparison for PASCAL Context

The occurrence of categories in PASCAL Context follow a power law distribution. If we select the 59 most frequent classes and assign to the rest the background label then 87.2% of the pixels are labeled as foreground, and the rest as background. By comparison, note that the 20 object classes of PASCAL VOC cover only 29.3% of the pixels. Fig. 2.2 shows the distribution of pixels and images among the 59 most frequent categories.

The PASCAL Context dataset was used [MCL14b] to study the frequency of contextual

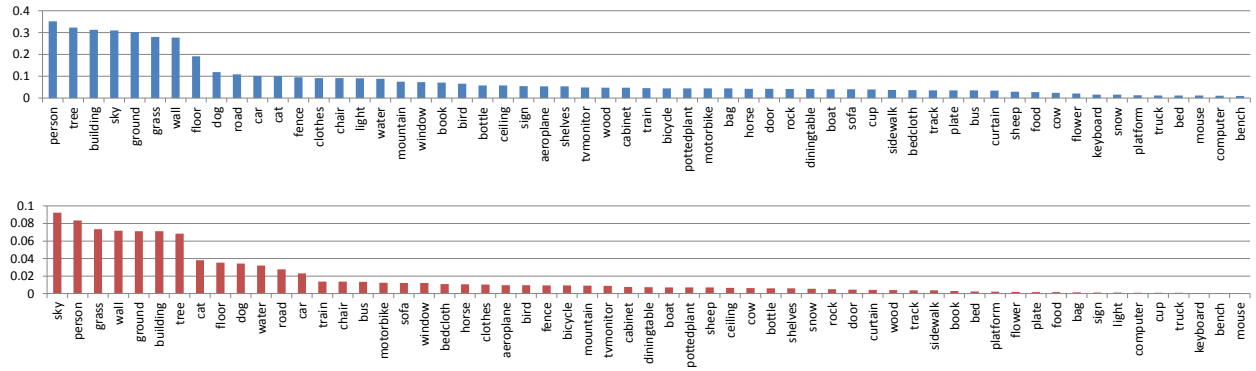


Figure 2.2: Distribution of pixels (above) and images (below) for the 59 most frequent categories. See text for the statistics.

categories around objects in terms of different sizes of objects. Several interesting trends were found. The amount of sky in the bottom region of airplanes increases as airplanes become smaller, which shows that small airplanes typically appear in the sky. Also we see more sky pixels in the top region of buses compared to cars, presumably because buses are taller than cars. This suggests that context can help predict objects and may be of value for small objects.

The statistics of PASCAL context differ from other datasets with contextual cues. These suggest that PASCAL-context is more challenging. Among the 35 most frequent categories of SUN [XHE10], 87.2% of the pixels are "stuff", 94.5% for Barcelona [TL10], while 60.1% for PASCAL. Thus, the number of "things" and "stuff" pixels is more balanced in PASCAL. The entropy[1] for the most frequent 35 classes in Barcelona is 1.78, for SUN is 2.11 and for PASCAL is 3.11, which shows that more pixels are assigned to fewer classes in SUN and Barcelona. Thus, PASCAL images are more diverse than SUN's and Barcelona's.

[1]For each category, we divide the number of pixels of that category to the total number of pixels in the dataset. This probability is used to compute entropy.

2.4 PASCAL Boundaries

The PASCAL Boundaries dataset was derived from PASCAL Context. Hence, unlike BSDS, it only labels the boundaries of objects and backgrounds. If differs from "boundary datasets" which can be derived from PASCAL segmentation and Microsoft COCO because its contains boundaries between background classes such as "sky" and "ground".

2.4.1 PASCAL Boundaries: annotation strategy

Converting PASCAL Context [MCL14b] to PASCAL Boundaries requires two stages: (I) Fine-tuning the annotations at the boundaries, which are typically imprecise for segmentation datasets (e.g., PASCAL Context and MRSC), by a two stage strategy with *t*riage (annotators rank boundaries from 1 to 5) and *c*orrection of the badly ranked boundaries. (II) Providing instance level labels for objects.

Converting any semantic segmentation dataset into a boundary dataset is straightforward provided the semantic annotations are accurate near the boundaries (i.e. within two or three pixels). Considerable work was done to ensure that the Pascal Context dataset was very accurate near the boundaries (after the initial annotations in Korea, there was refinement and checking at UCLA over a period of several months). Not all semantic segmentation datasets have this high precision near the boundaries because: (i) they are often unnecessary since the proportion of pixels near the boundary is small and hence has negligible effect on evaluating semantic segmentation, and (ii) obtaining precision near boundaries is hard and it much simpler, for example, to label regions by allowing annotators to mark regions using polygons which is very fast for annotating regions but is inaccurate near boundaries.

2.4.2 Analysis and Comparison for PASCAL Boundaries

We contrast PASCAL Boundaries with PASCAL Segmentation and BSDS by computing the average number of boundaries in each image. PASCAL Boundaries has images of 360×496 pixels on average, from which an average of 1.45% of pixels are annotated as boundaries.

This is in comparison to the SBD dataset [HAB11], which has only 0.77% of pixels labeled as boundaries. This clearly shows that labeling the boundaries of only the foreground objects ignores at least 50% of all the existing boundaries in an image. The percentage of pixels labeled as boundary in the PASCAL Boundaries dataset, however, is slightly lower than the 1.81% of pixels annotated as edges in BSDS500, on images of 321×481 pixels size. This is understandable since the BSDS annotations consisted of edges from different levels of granularity (e.g. the interiors of objects). This number drops to 0.91% if we consider only those pixels of the BSDS500 annotations that were labeled by all the annotators annotating the image.

Many of the images in the PASCAL dataset are non-iconic. They contain multiple objects, and so are not biased towards posed "photography images" which contain a salient foreground object in the center with high contrast with respect to the background. It is obviously important that computer vision algorithms are trained and tested on non-iconic images. We also emphasize that the number of images in the PASCAL Boundaries dataset ($\sim$ 10k) is much larger than in existing edge detection datasets. The increased scale of this dataset provides more variation in the boundary types and is beneficial for learning deep models. Moreover, testing algorithms on thousands of images, as opposed to testing them on a couple hundred images, will provide more credibility to future boundary detection models.

2.5 PASCAL Semantic Part Dataset and Benchmark

The goal of the dataset is to provide a platform on which researchers can investigate the performance of semantic part segmentation methods on challenging PASCAL imagees. To this end, it is required that the PASCAL Part contains well-defined and semantically meaningful parts, and that the annotation of parts are consistent and accurate. This section describes the processes used for annotating the PASCAL Part.

Table 2.1: Definitions of semantic parts for 16 Categories in the PASCAL Part. For boat, chair, dining table and sofa, we do not have parts. Bold font of part names means the corresponding parts could appear multiple times in one object instance*.

aeroplane	body, engine, left wing, right wing, stern, tail, **wheel**
bicycle	back wheel, chain wheel, front wheel, handlebar, **headlight**, saddle
bird	beak, head, left eye, left foot, left leg, left wing, neck, right eye, right foot, right leg, right wing, tail, torso
bottle	body, cap
bus	back license plate, back side, **door**, front license plate, front side, **headlight**, left mirror, left side, right mirror, right side, roof side, **wheel**, **window**
car	back license plate, back side, **door**, front license plate, front side, **headlight**, left mirror, left side, right mirror, right side, roof side, **wheel**, **window**
cat	head, left back leg, left back paw, left ear, left eye, left front leg, left front paw, neck, nose, right back leg, right back paw, right ear, right eye, right front leg, right front paw, tail, torso
cow	head, left back lower leg, left back upper leg, left ear, left eye, left front lower leg, left front upper leg, left horn, muzzle, neck, right back lower leg, right back upper leg, right ear, right eye, right front lower leg, right front upper leg, right horn, tail, torso
dog	head, left back leg, left back paw, left ear, left eye, left front leg, left front paw, muzzle, neck, nose, right back leg, right back paw, right ear, right eye, right front leg, right front paw, tail, torso
horse	head, left back hoof, left back lower leg, left back upper leg, left ear, left eye, left front hoof, left front lower leg, left front upper leg, muzzle, neck, right back hoof, right back lower leg, right back upper leg, right ear, right eye, right front hoof, right front lower leg, right front upper leg, tail, torso
motorbike	back wheel, front wheel, handlebar, **headlight**, saddle
person	hair, head, left ear, left eye, left eyebrow, left foot, left hand, left lower arm, left lower leg, left upper arm, left upper leg, mouth, neck, nose, right ear, right eye, right eyebrow, right foot, right hand, right lower arm, right lower leg, right upper arm, right upper leg, torso
pottedplant	plant, pot
sheep	head, left back lower leg, left back upper leg, left ear, left eye, left front lower leg, left front upper leg, left horn, muzzle, neck, right back lower leg, right back upper leg, right ear, right eye, right front lower leg, right front upper leg, right horn, tail, torso
train	**coach back side**, **coach front side**, **coach left side**, **coach right side**, **coach roof side**, **coach**, head, head back side, head front side, head left side, head right side, head roof side, **headlight**
tvmonitor	screen

* We do not give these parts different names as most of time it is difficult to distinguish them.

Table 2.2: Definition of semantic parts for 7 articulated object categories used in Part Segmentation benchmark. In the second column are the higher-level parts used in the benchmark and their constituent parts. The last column shows the frequencies of these parts and their proportion (in terms of percentage of pixels) in the training dataset.

Category	Higher-level Parts and Constituents		Freq. / Prop.
dog	head	head, left ear, left eye, nose, muzzle, right ear, right eye	1388 / 46.1%
	body	torso, neck	1360 / 36.9%
	leg	left (right) back leg, left (right) back paw, left (right) front leg, left (right) front paw	1130 / 13.1%
	tail	tail	577 / 1.87%
cat	head	head, left ear, left eye, nose, right ear, right eye	1124 / 39.2%
	body	torso, neck	1101 / 42.4%
	leg	left (right) back leg, left (right) back paw, left (right) front leg, left (right) front paw	881 / 14.0%
	tail	tail	459 2.82%
cow	head	head, left ear, left eye, right ear, right eye, left horn, muzzle, right horn	390 / 34.5%
	body	neck, torso	410 / 54.5%
	leg	left (right) back lower leg, left (right) back upper leg, left (right) front lower leg, left (right) front upper leg	312 / 8.42%
	tail	tail	118 / 0.79%
horse	head	head, left ear, left eye, right ear, right eye, muzzle	593 / 27.2%
	body	neck, torso	616 / 53.5%
	leg	left (right) back lower leg, left (right) back upper leg, left (right) back hoof, left (right) front lower leg, left (right) front upper leg, left (right) front hoof	525 / 12.6%
	tail	tail	363 / 3.33%
sheep	head	head, left ear, left eye, right ear, right eye, left horn, muzzle, right horn	607 / 28.2%
	body	neck, torso	663 / 61.2%
	leg	left (right) back lower leg, left (right) back upper leg, left (right) front lower leg, left (right) front upper leg	163 / 3.07%
	tail	tail	201 / 0.77%
person	head	hair, head, left ear, left eye, left eyebrow, nose, right ear, right eye, right eyebrow, mouth	7252 / 25.5%
	body	neck, torso	7313 / 33.7%
	arm	left (right) hand, left (right) lower arm, left (right) upper arm	6739 / 22.0%
	leg	left (right) foot, left (right) lower leg, left (right) upper leg	4730 / 16.0%
bird	head	head, beak, left eye, right eye	897 / 18.9%
	body	neck, torso	934 / 55.6%
	wing	left wing, right wing	550 / 3.14%
	leg	left foot, left leg, right foot, right leg	364 / 15.3%
	tail	tail	654 / 5.79%

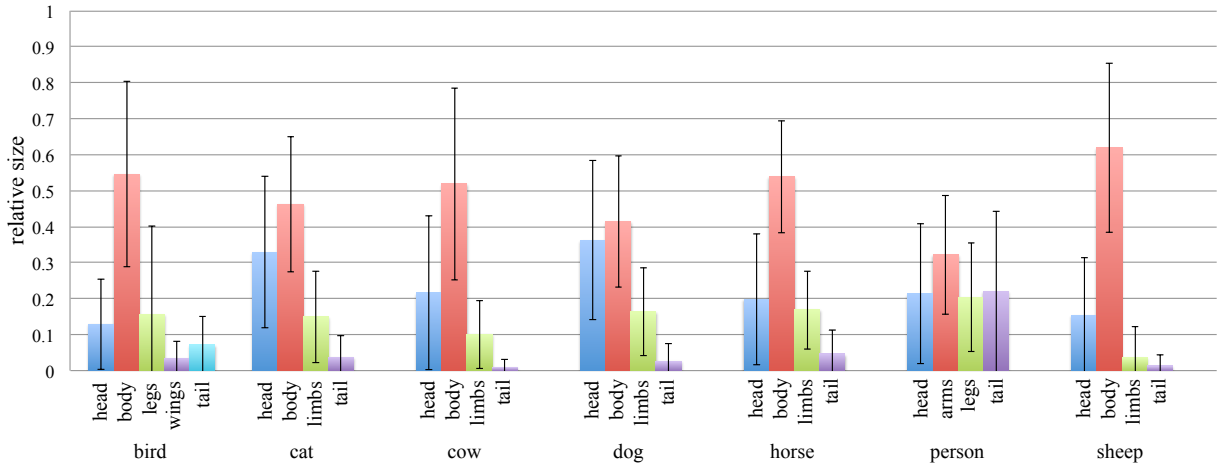


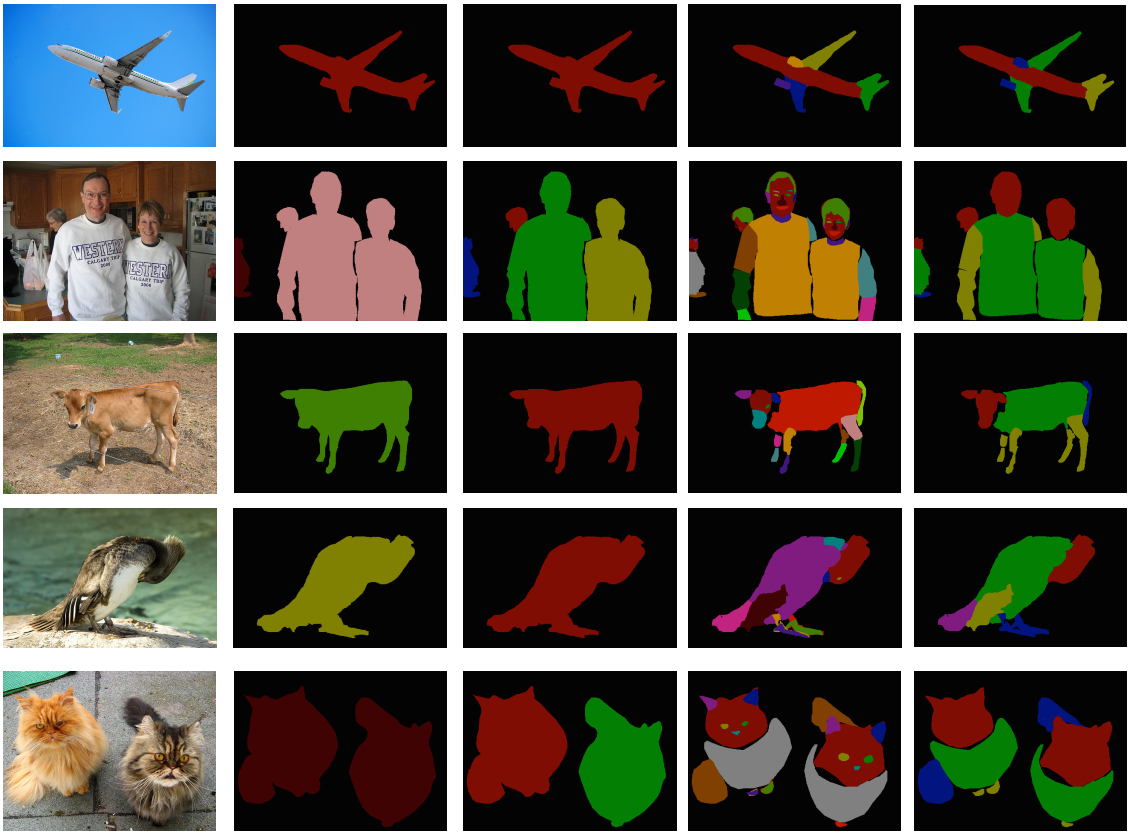
Figure 2.3: Mean and variance of relative sizes of parts for 7 articulated object categories used in Part.

2.5.1 Annotation Strategy: PASCAL PARTS

We do not consider the categories of boat, chair, dining table and sofa because nstances from these categories have extremely large variation in their constituent parts. For each of the rest 16 categories, we define a set of semantic parts. Table 2.1 lists the semantic parts. The parts of articulated categories (dog, cat, cow, horse, sheep, person and bird) are defined based on their skeletal systems. We also define parts for the components of their heads (*e.g.*, eyes, ears, muzzle) as they are important discriminative information visually, especially in fine-grained applications. For vehicles (car, bus, motorbike) and aeroplane, we define the parts based on their functions (*e.g.*, car doors and bicycle chain wheel) and visual distinctiveness (*e.g.*, car license plate). For bottle, pottedplant and tvmonitor, we choose parts that are common across all instances. For example, pottedplant always has a plant part and a pot part.

Notice that in Table 2.1, for some categories, especially the animals, we define parts at a very detailed level and organize them in a hierarchical way. This gives researchers freedom to build up a hierarchy of parts and focus on the level suitable to their projects.

The choice of parts also supports research in exploiting visual properties common to



(a) Original image (b) Category map (c) Instance map (d) Semantic part map (e) Higher-level parts

Figure 2.4: Examples of semantic part annotations. WHAT IS THE POINT OF THIS FIGURE??

categories *e.g.*, vehicle wheels, for example in the form of "part sharing"This is very essential for methods to scale up to more categories or images, and for learning intermediate representation

We developed a web-based annotation tool. For each image, the annotation contains three steps: First, we label each pixel in the image with one of the 20 categories or background, which gives us a category map of the images (Column (b) of Figure 2.4); Second, we identify different instances of the 20 categories in the image and label its pixels with a unique number, which yields an instance map (Column (c) of Figure 2.4); Finally, within each instance, we label its subregions as different semantic parts (Column (d) of Figure 2.4). We do not consider object instances annotated as "difficult" by VOC2010.

The accuracy of the annotation was ensured by one annotation phase and two polishing phases, and to achieve consistency, we made each phase take place at a single place. At the first phase, all the annotation took place at Korea University, following a set of annotation instructions. We started with 15 labelers. We requested each annotator attend an in-person training session. For each object, we selected 20-50 exemplar images and asked each annotator to label object parts following the instructions. The annotations were observed and meanwhile the instructions were discussed in details. An annotator could only proceed to work on more images if he/she could correctly annotate all these exemplar images. Depending on the training curve, this training session lasted 2-5 hours. After all the images were annotated, we started a polishing phase immediately at Korean University, by the same group of people. Then the annotations were sent to 15 graduate students of University of California, Los Angeles with expertise at machine learning and computer vision for final review.

2.5.2 Benchmark

We've constructed a Semantic Part Segmentation benchmark, which uses a subset of the 20 categories and higher-level parts.

Currently the benchmark focuses on the 7 articulated categories: dog, cat, cow, dog,

horse, sheep, person and bird. These categories are widely used to evaluate part-based methods [HAG15, AL12, DCS14, TKP15, ZDG14], and their large variation due to poses and the interaction with other objects (e.g. occlusions) bring challenges. Table 2.2 lists the 7 categories and their semantic parts used in the benchmark. As illustrated in the second column, we group the original parts defined in the PASCAL Part into higher-level parts: head, body, arm and leg for the person, head, body, leg and tail for the quadrupeds, and head, body, leg, wing and tail for the bird. See Column (e) of Figure 2.4 for examples. The reasons are twofold: first some parts are very difficult to learn individually (*e.g.*, bird's beak and horse's left back lower leg); Second semantic parts at this level are commonly adopted in many existing works [HAG15, TKP15, WY15, BF11a, LWT13].

The training images of the benchmark are those of the PASCAL Part that contain the target categories (6,732 images in total). We do not divide them into training and validation sets as suggested in VOC2010, which we leave to the users. The testing images are images used in VOC2010 segmentation task that contain the target categories (498 images in total). The ground truth annotation of part segmentation on testing images is not published. We plan to include more categories and extend testing images to the whole testing set of PASCAL Part in the future.

The last column of Table 2.2 shows the frequencies of these higher-level parts, and their proportions (computed by the ratio of total number of pixels of a part and the total number of pixels of its parent category), from which we can see that the benchmark contains both larger parts (*e.g.*, head and body) and smaller parts (*e.g.*, wing and tail). Figure 2.3 illustrates the variation of relative sizes of parts in the whole dataset, which indicates that our benchmark is unbiased in the sense that the relative size of a part varies a lot across images.

For each test image, participants need to predict the part class of each pixel, or "background" if the pixel does not belong to one of the target categories. Participants are not required to distinguish between instances from the same category.

For evaluation, we adopt the overall intersection over union (IOU) criteria used in PASCAL VOC segmentation task. For part p of category c, the IOU is computed (across all

testing images) as

$$\mathrm{IOU}(p,c) = \frac{\#\text{true positive pixels}}{\#\text{predicted pixels} + \#\text{false negative pixels}} \tag{2.1}$$

Then the IOU of category c is the average of IOU of its parts, *i.e.*,

$$\mathrm{IOU}(c) = \frac{1}{|\mathcal{P}_c|} \sum_{p \in \mathcal{P}_c} \mathrm{IOU}(p,c) \tag{2.2}$$

where $\mathcal{P}_c$ is the set of parts of category c.

2.6 Discussion: Future Works

A first extension is to include more categories in the Part Segmentation benchmark, and use the whole testing set of VOC2010. Further we would like to go beyond the PASCAL VOC dataset and increase the number of object categories, which will stimulate research in part sharing. Fixing the level of part hierarchy in current benchmark setting is not ideal. We plan to provide evaluation methodology for hierarchical part segmentation.

Table 2.3: Working hours of annotation and polishing phases on training/validation and testing sets.

Category	trainval	test	polishing
aeroplane	59.04	65.28	46.62
bicycle	49.12	51.20	37.62
bird	67.97	76.44	61.89
boat	6.87	9.82	50.07
bottle	20.28	28.70	73.47
bus	49.80	55.80	31.68
car	177.40	130.96	102.33
cat	101.88	102.96	68.28
chair	18.90	35.31	162.63
cow	46.40	59.70	31.83
diningtable	4.68	10.18	44.58
dog	127.44	129.96	85.80
horse	74.52	85.80	40.08
motorbike	30.55	13.50	26.43
person	656.64	383.13	346.59
pottedplant	24.63	33.60	58.23
sheep	70.10	87.20	47.19
sofa	4.51	31.62	45.15
train	41.92	45.04	32.61
tvmonitor	13.66	16.10	44.64
Total	1,646.31	1,452.30	1,437.72
Average	82.32	72.62	71.89

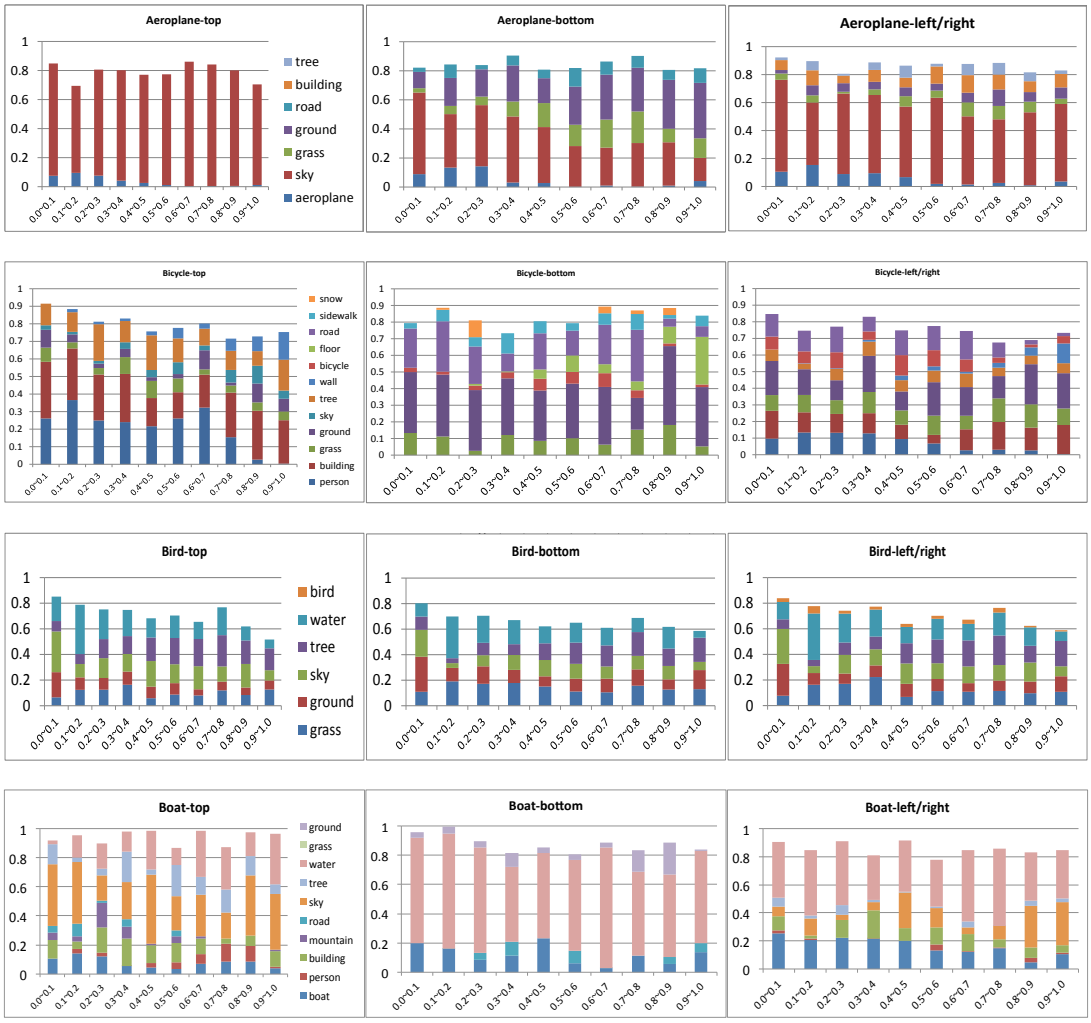


Figure 2.5: Pixel-wise frequency of context classes in top, bottom, and left/right contextual parts. The x-axis corresponds to size percentile and the y-axis represents the frequency of appearance. Only the most correlated classes are shown.

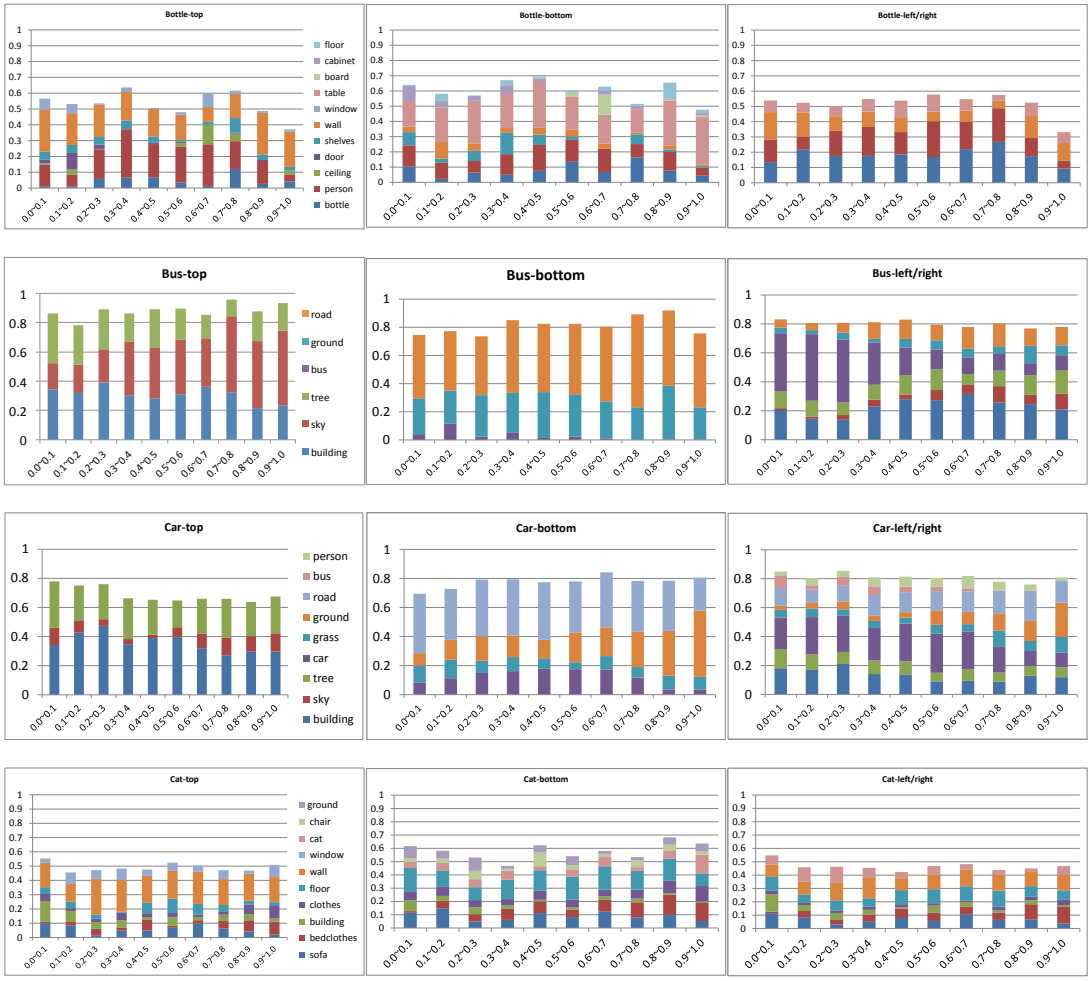


Figure 2.6: Pixel-wise frequency of context classes in top, bottom, and left/right contextual parts. The x-axis corresponds to size percentile and the y-axis represents the frequency of appearance. Only the most correlated classes are shown.

Figure 2.7: Pixel-wise frequency of context classes in top, bottom, and left/right contextual parts. The x-axis corresponds to size percentile and the y-axis represents the frequency of appearance. Only the most correlated classes are shown.

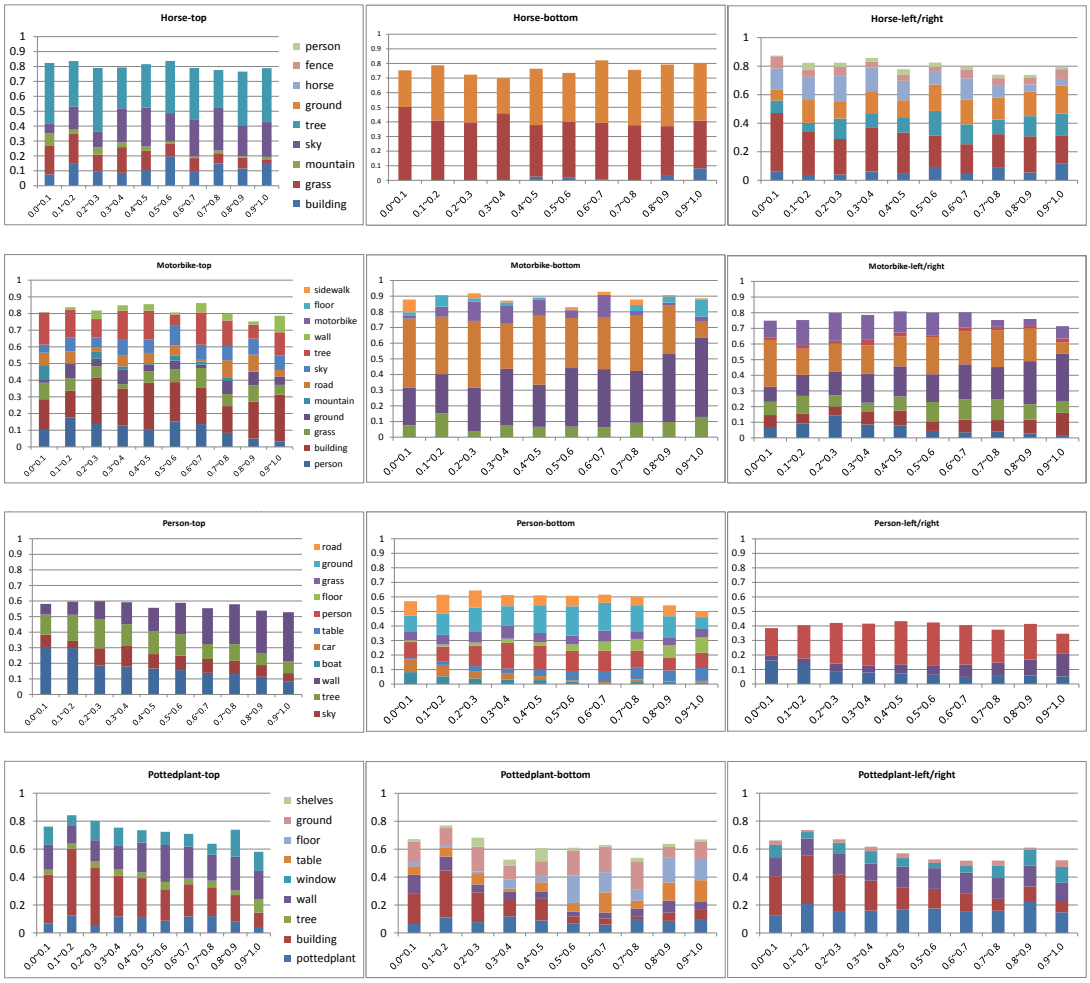


Figure 2.8: Pixel-wise frequency of context classes in top, bottom, and left/right contextual parts. The x-axis corresponds to size percentile and the y-axis represents the frequency of appearance. Only the most correlated classes are shown.

Figure 2.9: Pixel-wise frequency of context classes in top, bottom, and left/right contextual parts. The x-axis corresponds to size percentile and the y-axis represents the frequency of appearance. Only the most correlated classes are shown.

CHAPTER 3

Parsing Semantic Parts of Cars Using Graphical Models and Segment Appearance Consistency

3.1 Introduction

This chapter addresses the two goals of parsing an object into its semantic parts and performing object part segmentation, so that each pixel within the object is assigned to one of the parts (i.e. all pixels in the object are labeled). More specifically, we attempt to parse cars into wheels, lights, windows, license plates and body, as illustrated in Figure 3.1. This is a fine-scale task, which differs from the classic task of detecting an object by estimating a bounding box.

We formulate the problem as landmark identification. We first select representative locations on the boundaries of the parts to serve as landmarks. They are selected so that locating them yields the silhouette of the parts, and hence enables us to do object part segmentation. We use a mixture of graphical models to deal with different viewpoints so that we can take into account how the visibility and appearance of parts alter with viewpoint (see Figure 3.2).

A novel aspect of our graphical model is that we couple the landmarks with the segmentation of the image to exploit the image contents when modeling the pairwise relation between neighboring landmarks. In the ideal case where part boundaries of the cars are all preserved by the segmentation, we can assume that the landmarks lie near the boundaries between different segments. Each landmark is then associated to the appearance of its two closest segments. This enables us to associate appearance information to the landmarks and

Figure 3.1: The goal of car parsing is to detect the locations of semantic parts and to perform object part segmentation. The inputs (left) are images of a car taken from different viewpoints. The outputs (right) are the locations of the car parts – the wheels, lights, windows, license plates and bodies – so that each pixel within the car is assigned to a part.

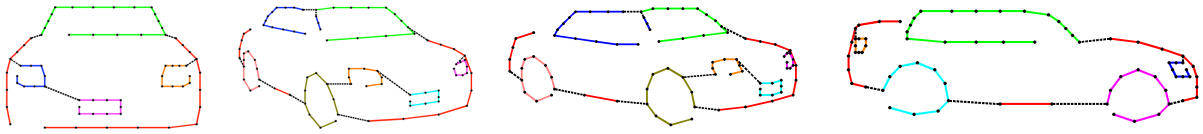


Figure 3.2: The proposed mixture-of-trees model. Models of left-front, right-back and right views are not shown due to the symmetry. The landmarks connected by the solid lines of same colors belong to the same semantic parts. The black dashed lines show the links between different parts. Best view in color.

to introduce pairwise coupling terms which enforce that the appearance is similar within parts and different between parts. We call this segmentation appearance consistency (SAC) between segments of neighboring landmarks. This is illustrated in Figure 3.3, where both of the two neighboring landmarks (the red and green squares) on the boundary between the window and the body have two segments (belonging to window and body respectively) close to them. Segments from the same part tend to have homogeneous color and texture appearance (*e.g.*, a and c, b and d in the figure), while segments from different parts usually do not (*e.g.*, a and b, c and d in the figure). The four blue dashed lines in the figure correspond to the SAC terms whose strengths will be learnt.

However, in practice, it is always impossible to capture all part boundaries using single level segmentation. Instead, people try to use a pool of segmentations [GFK09, CJG11, KK10] or segmentation trees [AMF11b, LVZ11, Vek00]. Inspired by those, we couple the landmarks to a hierarchical segmentation of the image. However, the difference of the sizes

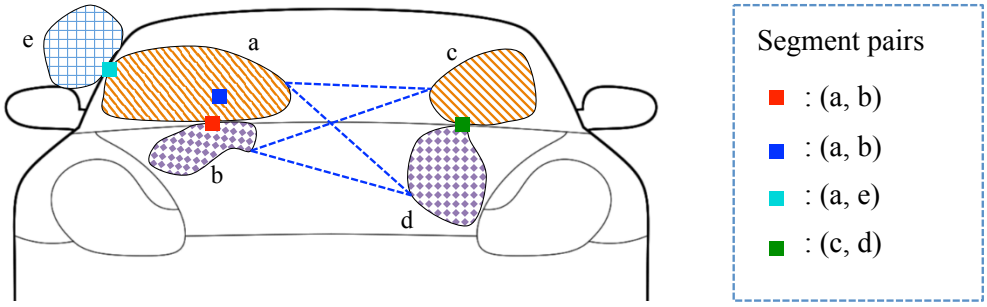


Figure 3.3: Illustration of segmentation appearance consistency (SAC) and segment pairs. Red and green squares represent two neighboring landmarks lying on the boundary between window and body. Each landmark has two segments (a and b for the red landmark, c d for the green landmark) close to it. Our method models and learns the SACs for every pair of neighboring landmarks (blue dashed lines) and uses them to enhance the reliability of landmark localization. For the blue landmark, its segment pair is the same as the red landmark, which is the closest one on the boundary.

of the parts (*e.g.*, the license plate is much smaller than the body) and the variability of the images mean that the optimal segmentation level for each part also varies. Therefore the level of the hierarchy used in this coupling must be chosen *dynamically* during inference/parsing. This leads us to treat the level of the hierarchy for each part as a hidden variable. By doing this, our model is able to automatically select the most suitable segmentation level for each part while parsing the image.

In the next section, we review related work. Then we describe the details of our method in Sections 3.3. Experimental results are given in Section 3.4, which is followed by the conclusions in Section 3.5.

3.2 Related Work

There is an extensive literature dating back to Fischler and Elschlager [FE73] which represents objects using graphical models. Nodes of the graphs typically represent distinctive regions or landmark points. These models are typically used for detecting objects

[FMR08, FGM10b] but they can also be used for parsing objects by using the positions of the nodes to specify the locations of different parts of the object. For example, Zhu *et al.* [ZCL08, ZCY10] uses a compositional AND/OR graph to parse baseball players and horses. More recently, in Zhu and Ramanan's graphical model for faces [ZR12] there are nodes which correspond to the eyes and mouth of the face. But we note that these types of models typically only output a parse of the object and are not designed to perform object part segmentation. They do not exploit the SAC either.

Recently, a very similar graphical model for cars has been proposed by Hejrati and Ramanan [HR12], which cannot do part segmentation since each part is represented by only one node. The more significant difference is that the binary terms do not consider the local image contents.

There are, however, some recent graphical models that can perform object part segmentation. Bo and Fowlkes [BF11b] use a compositional model to parse pedestrians, where the semantic parts of pedestrians are composed of segments generated by the UCM algorithm [AMF11b] (they select high scoring segments to form semantic parts and use heuristic rules for pruning the space of parses). Thomas *et al.* [TFL08a] use Implicit Shape Models to determine the semantic part label of every pixel. Eslami and Williams [EW12] extend the Shape Bolzmann Machine to model semantic parts and enable object part segmentation. [TFL08a] and [EW12] did car part segmentation on ETHZ car dataset [TFL08a], which contains non-occluded cars of a single view (semi-profile view).

Image labeling is a related problem since it requires assigning labels to pixels, such as [SWR09, MBH10, LYT11, EF12, FCN12, TL13]. But these methods are applied to labeling all the pixels of an image, and are not intended to detect the position of objects or perform object part segmentation.

3.3 The Method for Parsing Cars

The silhouette, and hence the segment of a part is defined by the polygon formed by the landmarks of the part. This allows us to formulate the segmentation problem as landmark

localization. We model the landmark points and their spatial configuration as a mixture of tree-structured graphical models, one model for each viewpoint. The model is represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The nodes $\mathcal{V}$ correspond to landmark points. They are divided into subsets $\mathcal{V} = \bigcup_{p=1}^{N} \mathcal{V}_p$, where N is the number of parts and $\mathcal{V}_p$ consists of landmarks lying at the boundaries of semantic part p. The edge structures $\mathcal{E}$ are manually designed (see Figure 3.2).

We define an energy function for each graphical model, which consists of unary terms at the landmarks and binary terms at the edges. The binary terms not only model the spatial deformations as in[ZR12, FGM10b], but also utilize local image contents, *i.e.*,the segment appearance consistency (SAC) between neighboring landmarks.

To do that, we couple the landmarks to a hierarchical segmentation of the image which is obtained by the SWA algorithm [SGS06] (see Figure 3.4 for a typical SWA segmentation hierarchy). Then we associate with each image location at every segmentation level a pair of nearby segments: If a location is on the segment boundary, the two segments are on either sides of the boundary, otherwise it shares the same segment pairs with the nearest boundary location. Then SAC terms are used to model the four pairings of segments from neighboring landmarks (blue dashed lines in Figure 3.3 for example). The strengths of SAC terms are learnt from data. In order to do the learning, the four pairings need to be ordered, or equivalently, the two segments of each location need to be represented in the form of an ordered tuple (s^1, s^2). In practice, choosing two segments for a segment boundary location and ordering them is not straightforward (*e.g.*, a location on T-junction where there are more than two segments nearby). We put technical details about segment pairs in Section 3.3.3.

3.3.1 Score Function

In this section we describe the score function for each graphical model, which is the sum of unary potentials defined at the graph nodes, representing the landmarks, and binary potentials defined over the edges connecting neighboring landmarks.

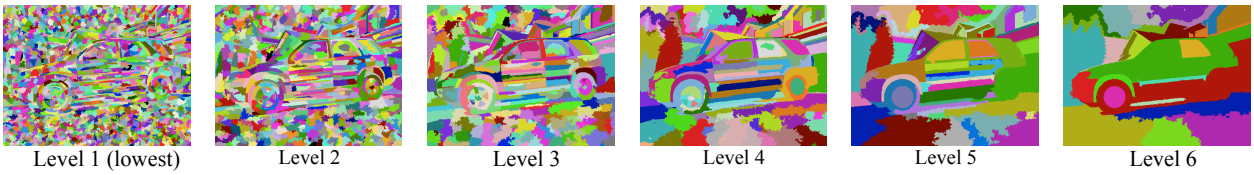


Figure 3.4: The segments output by SWA at six levels. Note how the segments covering the semantic parts change from level 1 to level 6 (*e.g.*, left windows and left wheels). This illustrates that different parts need different levels of segmentation. For example, the best level for the left-back wheel is level 4 and the best level for the left windows is level 5. Best view in color.

We first define the variables of the graph. Each node has pixel position of landmark $l_i = (x_i, y_i)$. The set of all positions is denoted by $\mathbf{L} = \{l_i\}_{i=1}^{|\mathcal{V}|}$. We denote by p_i the indicator specifying which part landmark i belongs to, and by $h(p)$ the segmentation level of part p. Then the segment pair of node i, $\mathbf{s}_i$, can be seen as the function of $h(p_i)$, which we denote by $\mathbf{s}_{i,h}$ for simplicity. Similar to the definitions of $\mathbf{L}$, we have $\mathbf{H} = \{h(p_i)\}_{i=1}^{N}$ and $\mathbf{S}(\mathbf{H}) = \{\mathbf{s}_{i,h}\}_{i=1}^{|\mathcal{V}|}$. The score function of the model for viewpoint v is

$$S(\mathbf{L}, \mathbf{H}, v \mid \mathbf{I}) = \phi(\mathbf{L}, \mathbf{H}, v \mid \mathbf{I}) + \psi(\mathbf{L}, \mathbf{H}, v \mid \mathbf{I}) + \beta_v \tag{3.1}$$

In the following we omit v for simplicity. The unary terms $\phi(\mathbf{L}, \mathbf{H} \mid \mathbf{I})$ is expressed as:

$$\phi(\mathbf{L}, \mathbf{H} \mid \mathbf{I}) = \sum_{i \in \mathcal{V}} \Big[\mathbf{w}_i^f \cdot f(l_i \mid \mathbf{I}) + w_i^e e(h(p_i), l_i \mid \mathbf{I}) \Big] \tag{3.2}$$

The first term in the bracket of Equation 3.2 measures the appearance evidence for landmark i at location l_i. We write $f(l_i \mid \mathbf{I})$ for the HOG feature vector extracted from l_i in image $\mathbf{I}$. In the second term, the term $e(h(p_i), l_i \mid \mathbf{I})$ is equal to one minus the distance between l_i and the closest segment boundary at segmentation level $h(p_i)$. This function penalizes landmarks being far from edges. The unary terms encourage locations with distinctive local appearances and with segment boundaries nearby to be identified as landmarks. The binary term $\psi(\mathbf{L}, \mathbf{H} \mid \mathbf{I})$ is:

$$\psi(\mathbf{L}, \mathbf{H} \mid \mathbf{I}) = \sum_{(i,j) \in \mathcal{E}} \mathbf{w}_{i,j}^d \cdot d(l_i, l_j) + \sum_{\substack{(i,j) \in \mathcal{E} \\ p_i = p_j}} \mathbf{w}_{i,j}^A \cdot A(\mathbf{s}_{i,h}, \mathbf{s}_{j,h} \mid \mathbf{I}) \tag{3.3}$$

Figure 3.5: The landmark annotations for typical images. Yellow dots are the annotated landmark locations. Please refer to Section 3.3.3 for landmark selection criteria.

$d(l_i, l_j) = (-|x_i - x_j - \bar{x}_{ij}|, -|y_i - y_j - \bar{y}_{ij}|)$ measures the deformation cost for connected pairs of landmarks, where $\bar{x}_{ij}$ and $\bar{y}_{ij}$ are the anchor (mean) displacement of landmark i and j. We adopt L1 norm to enhance our model's robustness to deformation. In the second term of Equation 3.3, $A(\mathbf{s}_{i,h}, \mathbf{s}_{j,h} \mid \mathbf{I}) = (\alpha(s^1_{i,h}, s^1_{j,h} \mid \mathbf{I}), \alpha(s^1_{i,h}, s^2_{j,h} \mid \mathbf{I}), \alpha(s^2_{i,h}, s^1_{j,h} \mid \mathbf{I}), \alpha(s^2_{i,h}, s^2_{j,h} \mid \mathbf{I}))$ is a vector storing the pairwise similarity between segments of nodes i and j. This, together with the strength term $\mathbf{w}^A_{ij}$, models the SAC. The computation of $\alpha(\mathbf{s}_{i,h}, \mathbf{s}_{j,h} \mid \mathbf{I})$ is given in Section 3.3.3. Finally, β is a mixture-specific scalar bias.

The parameters of the score function are $\mathcal{W} = \{\mathbf{w}^f_i\} \cup \{w^e_i\} \cup \{\mathbf{w}^d_{ij}\} \cup \{\mathbf{w}^A_{ij}\} \cup \{\beta\}$. Note that the score function is linear in $\mathcal{W}$, therefore similar to [FGM10b] we can express the model more simply by

$$S(\mathbf{L}, \mathbf{H} \mid \mathbf{I}) = \mathbf{w} \cdot \Phi(\mathbf{L}, \mathbf{H} \mid \mathbf{I}) \tag{3.4}$$

where $\mathbf{w}$ is formed by concatenating the parameters $\mathcal{W}$ into a vector.

Special case Without the second term in the bracket of Equation (3.2) and the second term of Equation (3.3) (*i.e.* setting $w^e_i = 0$ and $\mathbf{w}^A_{i,j} = 0$), the model is equivalent to [ZR12], which we will compare our performance with.

3.3.2 Inference and Learning

Inference. The viewpoint v, the positions of the landmarks $\mathbf{L}$ and the segmentation levels $\mathbf{H}$ are unobserved. Our model detects the landmarks and searches for the optimal viewpoint and segmentation levels of parts simultaneously, as expressed by the following equation,

$$S(\mathbf{I}) = \max_v [\max_{\mathbf{H},\mathbf{L}} S(\mathbf{L}, \mathbf{H}, v \mid \mathbf{I})] \tag{3.5}$$

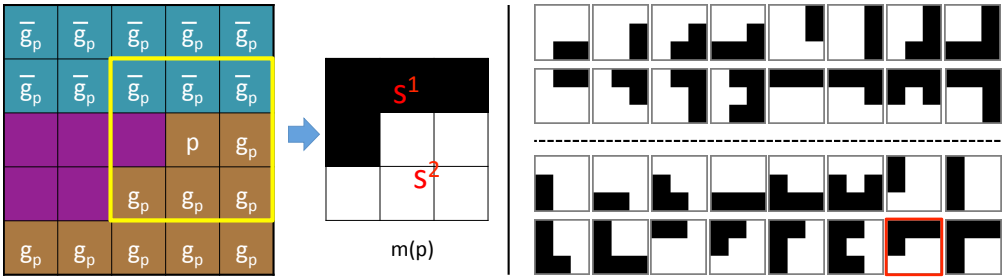


Figure 3.6: Illustration of segment pair assignment. Right: The look-up table for segment pair assignment, which is divided into two parts (separated by the dashed line). White represents 1 and black represents 0. Left: an example of how to construct the binary matrix $m(p)$ for location p and how to determine its segment pair. The hit of $m(p)$ in the look-up table is marked by the red rectangle. Best view in color.

The outer maximizing is done by enumerating all mixtures. Within each mixture, we apply dynamic programming to estimate the segmentation levels and landmark positions of parts. Then the silhouette of each part can be directly inferred from its landmarks. In our experiment, it took a half to one minute to do the inference on an image about 300-pixel height.

Learning. We learn the model parameters by training our method for car detection (this is simpler than training it for part segmentation). We use a set of image windows as training data, where windows containing cars are labeled as positive examples and windows not containing cars are negative examples. A loss function is specified as:

$$\mathcal{J}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \max(0, 1 - t_i \cdot \max_{\mathbf{L}_i,\mathbf{H}_i} \mathbf{w} \cdot \Phi(\mathbf{L}_i, \mathbf{H}_i \mid \mathbf{I}_i)) \tag{3.6}$$

where $t_i \in \{1, -1\}$ is the class label of the object in the training image and C is a constant. Let's take a closer look at the inner maximization. The segmentation levels of the semantic parts $\mathbf{H}$ are hidden and need to be estimated. The CCCP algorithm [YR03] is used to estimate the parameters by minimizing the loss function through alternating inference and optimization.

3.3.3 Implementation Details

Landmarks. The landmarks are specified manually for each viewpoint. They are required to lie on the boundaries between the car and background (contour landmarks) or between parts (inner landmarks), so that the silhouettes of parts and the car itself can be identified from landmarks. For front/back view, we use 69 landmarks; for left and right side views, we use 74 landmarks; for the other views, we use 88 landmarks. The assignment of landmarks to parts is determined by the following rule: contour landmarks are assigned to parts they belong to (*e.g.*, landmarks of the lower half of wheels), and inner landmarks are assigned to parts that they are surrounding (*e.g.*, landmarks around license plates). See Figure 3.5 for some examples. It took about two minutes to label one image.

Appearance features at landmarks. The appearance features $\mathbf{f}$ at the landmarks are HOG features. More specifically, we calculate the HOG descriptor of an image patch centered at the landmark. The patch size is determined by the 80% percentile of the distances between neighboring landmarks in training images.

Appearance similarity between segments. The similarity $\alpha(\cdot, \cdot)$ is a two dimensional vector, whose components are the χ^2 distances of two types of features of the segments: color histograms and the grey-level co-occurrence matrices (GLCM) [HSD73]. The color histograms are computed in the HSV space. They have 96 bins, 12 bins in the hue plane and 8 bins in the saturation plane. The GLCM is computed as follows: We choose 8 measurements of the co-occurrence matrix, including HOM, ASM, MAX and means (variances and covariance) of x and y (please refer to [HSD73] for details); The GLCM feature is computed in the R, G and B channels in 4 directions (0, 45, 90, 135 degrees); As a result, the final feature length is 96 (8 measurements $\times$ 3 channels $\times$ 4 directions).

Segment pair assignment. For each location on boundaries, we need to choose two segments from its neighborhood and assign s^1 or s^2 to them. In order to do this, we build a look-up table which consists of 32 3-by-3 binary matrices, as shown in the right of Figure 3.6. At each boundary location p we first construct a 3-by-3 binary matrix $m(p)$ according to the segmentation pattern of its 3-by-3 neighborhood. Then we search for $m(p)$ in the look-up

table. If $m(p)$ matches to one of the upper 16 matrices, g_p will be s^1 and $\bar{g}_p$ be s^2 of p; If it matches to one of the lower 16 matrices, $\bar{g}_p$ will be s^1 and g_p be s^2 of p. In the following, we will explain the design of the look-up table, describe how the segment assignment is done, and at the end demonstrate how the look-up table makes segment assignments consistent.

- The first design criterion of the look-up table is that the assignment should be consistent, which is twofold: 1) Moving a contour point in its vicinity should not change its segment pair assignment; 2) For two nodes in the graphical model whose landmarks are from the same part, their segment pairs should have the same order (*e.g.*, boths s_1 are assigned to segments inside the part and s_2 assigned to segments outside the part) across different images. This criterion guarantees that learning the parameters ($w^A_{i,j}$ in Eq 3.3) of SAC terms are statistically meaningful. The second criterion is that the look-up table should be able to identify locations with jagged edges for our model to ignore considering them as potential landmark locations, as in such locations it is very hard to guarantee consistency.

  We use 3-by-3 binary matrices to index the local segmentation patterns around contour locations. Figure 3.7 shows 70 out of all 256 possible matrices (the center is fixed to one), the rest of which are obtained by rotating these 70 prototypes. Not all of the 256 matrices are suitable for indexing. Some of them correspond to jagged edges and some of them will not occur on the contours in practice. We discard those that are not suitable and pick the remaining 8 matrices as shown on the first row in figure 3.7. The we rotate them to generate a set of 32 matrices which compose of the look-up table as shown in figure 3.8.

- At each boundary location p we construct a 3-by-3 binary matrix m according to the segmentation pattern of its 3-by-3 neighborhood: locations covered by the same segment which covers p are given value 1 and other locations are given value 0. We denote the segment which p belongs to by g_p, and denote by $\bar{g}_p$ the segment which most 0-valued locations in m belong to. Then we search in the look-up table for the same binary matrix as m. If there is a hit from one of matrices 1 to 16 in figure 3.8,

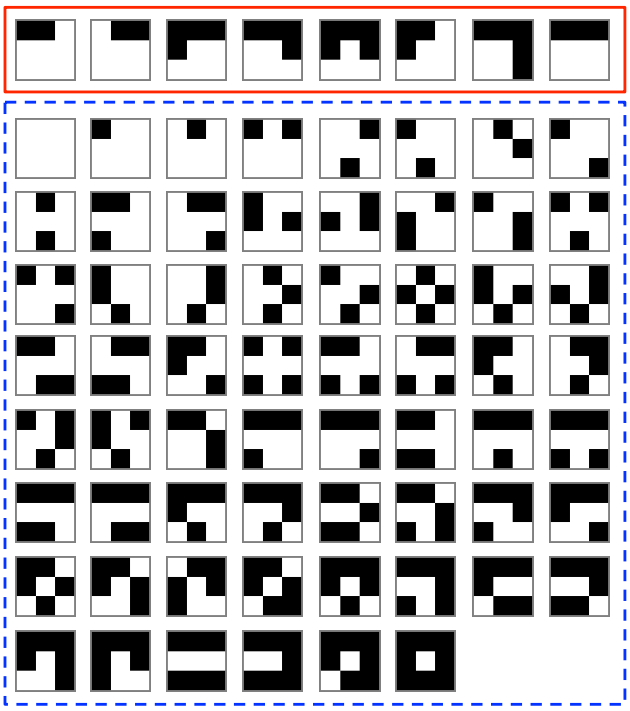


Figure 3.7: 70 out of all 256 3-by-3 binary matrices (black indicates "0" and white indicates "1"), with the center fixed to one. Matrices in red rectangle are used to generated the 32 binary matrices of the look-up table. Matrices in the blue dashed rectangle are considered not suitable for indexing.

g_p will be s^1 and $\bar{g}_p$ will be s^2; if there is a hit from one of matrices 17 to 32, g_p will be s^2 of p and $\bar{g}_p$ will be s^1 of p; otherwise, we will not apply SAC terms to p in the score function.

As an example, Figure 3.9 shows how to compute the binary matrices for contour locations. The green rectangle marks the 3-by-3 neighborhood with p in the center. The bronze segment is g_p and the cyan segment is $\bar{g}_p$. According to the above rule, the bronze region is given value 1, and the rest is given value 0; then we got a hit in the look-up table with matrix 31, and hence assign s^2 to the bronze segment and s^1 to the cyan segment.

- On Figure 3.10, we show two segmentation patterns from two locations p and p' not far from each other. Although different from each other, they all assign s^1 to the bronze

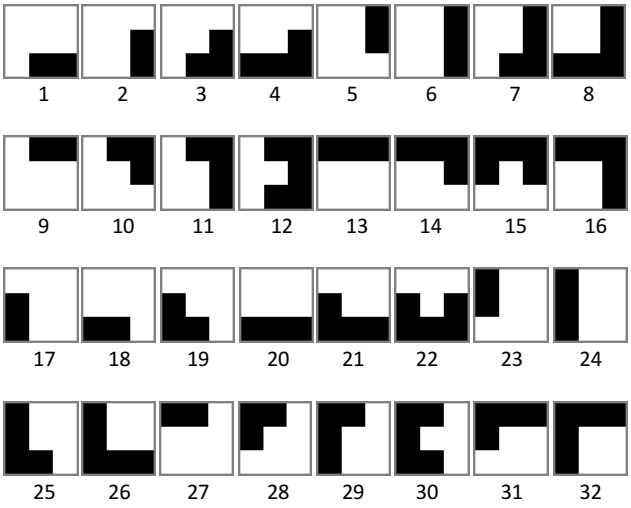


Figure 3.8: The 32 binary matrices in the look-up table, separated by a dashed line.

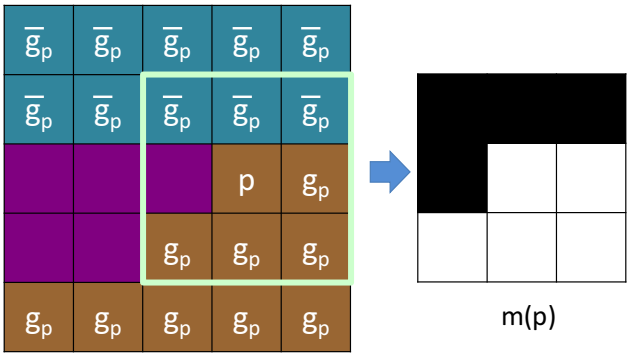


Figure 3.9: Example of how segment pair assignment rule works.

segment and s^2 to the violet segment. In fact, in this example, all points along the segment boundaries have the same assignment (*i.e.*,bronze segment to s^1 and violet segment to s^2). This shows the consistency of the assignment algorithm.

3.4 Experiments

3.4.1 Dataset

We validate our approach on two datasets, PASCAL VOC 2010 (VOC10) [EVW] and 3D car (CAR3D) [SL07]. VOC10 is a hard dataset because the variations of the cars (e.g,

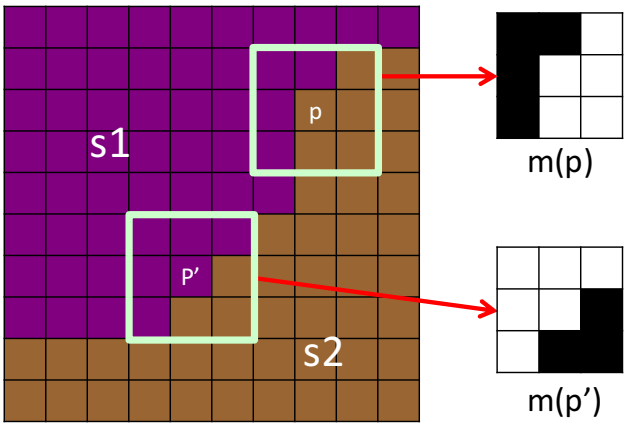
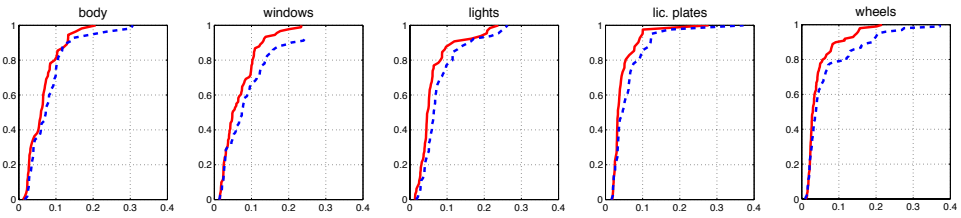
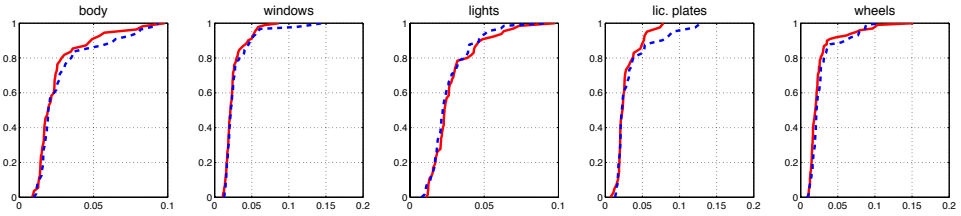


Figure 3.10: Illustration of the consistency of the segment assignment algorithm.



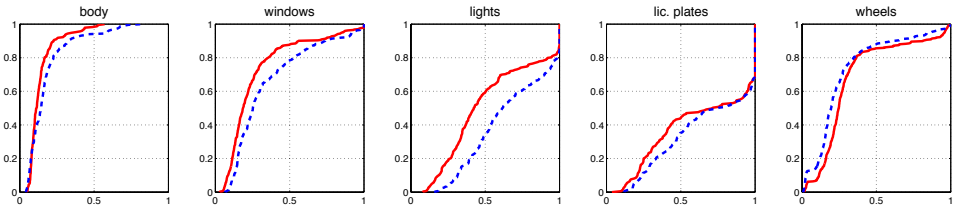
(a) PASCAL VOC 2010



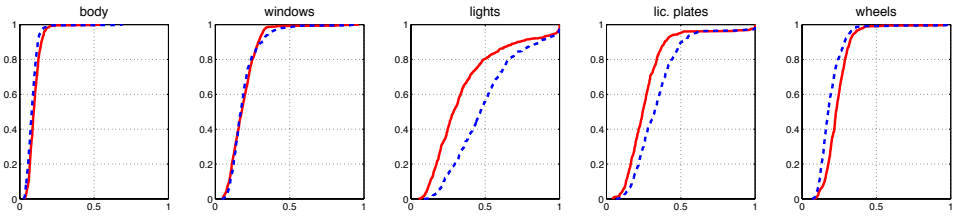
(b) CAR3D

Figure 3.11: Cumulative localization error distribution for parts. X-axis is the average localization error normalized by image width, and Y-axis is the fraction of the number of testing images. The red solid lines are the performance using SAC and the blue dashed lines are the performance of [ZR12].

appearance and shape) are very large. From VOC10, we choose car images whose sizes are greater than 80×80. This ensures that the semantic parts are big enough for inference and learning. Currently our method cannot handle occlusion, so we remove images where cars are occluded by other objects or truncated by image border. We augment the image set



(a) PASCAL VOC 2010



(b) CAR3D

Figure 3.12: Cumulative segmentation error distribution for parts. X-axis is the average segmentation error normalized by image width, and Y-axis is the fraction of the number of testing images. The red solid lines are the performance using SAC and the blue dashed lines are the performance of [ZR12].

by flipping the cars in the horizontal direction. This yields a dataset containing 508 cars. Then we divide images into seven viewpoints spanning over 180° spacing at 30°. CAR3D provides 960 non-occluded cars. We also divide them into seven viewpoints (instead of using the original eight viewpoints). We collect 300 negatives images by randomly sampling from non-car images of PASCAL VOC 2010 using windows of the sizes of training images. These 300 negative images are used for both datasets. In our experiments, for each dataset, we randomly select half of the images as training data and test the trained model on the other half.

3.4.2 Baseline

We compare our method with the model proposed by Zhu and Ramanan [ZR12] on landmark localization and semantic part segmentation. We simply use their code to localize landmarks and assume the regions surrounded by certain landmarks are the semantic parts. Note that

Figure 3.13: Visualized comparison of our method with [ZR12] on car part segmentation. In each pair of results, the lower one is produced by our method.

we use the same landmark and part definitions for both the baseline and our methods.

3.4.3 Evaluation

We first evaluate our method on landmark localization. We normalize the localization error as Zhu and Ramanan did in [ZR12]. In this and the following experiments, we consider parts of same category as a single part (*e.g.*, two lights of a front-view car are treated as one part). Figure 3.11 shows the cumulative error distribution curves on both datasets. We can see that by using SAC we had a big improvement of the landmark localization performance of all semantic parts on VOC10. We achieved better or comparable performance on CAR3D. Images in CAR3D are relatively easier than those in VOC10 and therefore SAC cannot bring big performance gain.

Then we evaluate our method on semantic part segmentation. The segmentation error of a part is computed by $(1 - IOU)$, where IOU is the intersection of detected segments and ground truth segments over the union of them. Figure 3.12 shows the cumulative error distribution curves on both datasets. Again, using SAC our method improves the performance on almost all parts (improvement on lights and license plate is significant). However, we got slightly worse result on wheels. The errors occurred when SWA produces

Figure 3.14: More segmentation results of our method on VOC10 (upper) and CAR3D (lower).

segments that are crossing the boundaries of wheels and the nearby background at all levels. The reason is that due to illumination and shading, it is difficult to separate wheels and background by appearance.

Figure 3.13 shows the visualization comparison, from which we can see that our method works better on part boundaries, especially for lights and license plates. Figure 3.14 shows more segmentation results on VOC10 and CAR3D.

3.5 Conclusion

In this chapter, we address the novel task of car parsing, which includes obtaining the positions and the silhouettes of the semantic parts (e.g., windows, lights and license plates). We propose a novel graphical model which integrates the SAC coupling terms between neighboring landmarks, including using hidden variables to specify the segmentation level for each part. This allows us to exploit the appearance similarity of segments within different parts of the car. The experimental results on two datasets demonstrate the advances of using segment appearance cues. Currently, the model cannot handle large occlusion and truncation,

which is our future direction.

CHAPTER 4

Video Action Recognition Using Attention

It has been noted in visual cognition literature that humans do not have a detailed and coherent representation of the whole world around them [Ren00]. Instead, there is a focused attention mechanism in their visual perception which can detect the change in an scene. Then attention is coordinated so that a sparse set of stable structures are created to form a stable and detailed scene representation.

Attention mechanisms are not employed in most previous computer vision algorithms and hence information lying within local regions of images or sub-volume of videos is not fully exploited. With the recent success of deep neural networks, attention based models have been shown to achieve promising results on several challenging tasks, including caption generation [XBK15] and image recognition (*e.g.*, Street View House Numbers dataset [BSG15]).

There are two types of attention models: hard attention and soft attention models. Hard attention models usually produce a set of locations that need to be attended and they can be trained by reinforcement learning methods [MHG14] or by maximizing a variational lower bound [BMK15]. On the other hand, soft attention models estimate the "importance" of all locations, which is usually present as a heat map (*i.e.*,attention map). The numerical values of the heap map intuitively indicate how much attention is needed for each location.

In this chapter, we propose a video action recognition framework that is able to estimate the action critical places of a video and infer the action happening in the video by attending only to relevant places in each frame. The framework consists of two parts: a classification module and a soft attention module, as illustrated in Fig. 4.1. The classification module applies a fully convolutional neural network (FCN) [LSD15], which produces a dense classification score map for the input video. On the other hand, the attention module, computes

an dense attention map with the same size as the classification score map, based on the same input video. The final output of our model is produced by the weighted sum of classification scores across all locations. Intuitively, each value at a location of the attention map indicates how much "attention" we should pay to the classification result at that location.

In Fig. 4.1, inputs of the attention module can come from the intermediate outputs of the classification module (type A) or directly from the input video (type B). By taking inputs from outputs of a intermediate layer of the classification module, the two modules of type A network share parameters of early-stage layers, which leads to a model of less complexity and easier learning. On the other hand, type B network can have its attention module exploit a different input modality than its classification module (*e.g.*, the classification module takes RGB image as input and the attention module takes optical flows), so that the network can benefit from the complementary information from different modalities [WXW16]. We will evaluate both type of networks in Section 4.3.

We jointly train the classification module and the attention model. We demonstrate the effectiveness of our model on two challenging datasets, UCF101 and HMDB51. Experimental results show that our proposed method consistently improves over strong baselines. The attention module also outperforms average and max-pooling methods. In addition, the proposed attention module can be visualized, unveiling the insights of what the black box network has learned.

The rest of the paper is organized as following: In Section 4.1 we briefly review the literature and related work. Then we describe the proposed method in Section 4.2. Evaluation are given in Section 4.3 followed by conclusions in Section 4.4.

4.1 Related Work

Computational limitations have received much attention in the computer vision literature. For instance, for object detection, much work has been dedicated to reducing the cost of the widespread sliding window paradigm, focusing primarily on reducing the number of windows for which the full classifier is evaluated, e.g. via classifier cascades (e.g. [FGM10a, VJ01]),

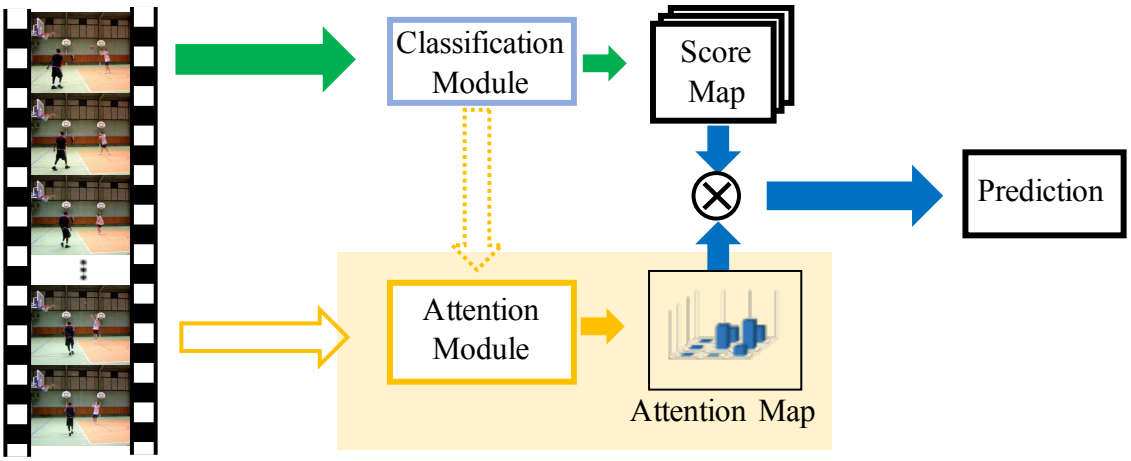


Figure 4.1: The overall framework of the proposed video action recognition model. The feature extraction module takes a single frame and applies a deep convolutional neural network (DCNN) on the the frame image in a sliding window manner, yielding for each location a deep convolutional feature. The attention module computes an attention map for those locations, based on a short clip (*i.e.*,a sequence of frames) centered at the frame.

removing image regions from consideration via a branch and bound approach on the classifier output (e.g. [LBH08]), or by proposing candidate windows that are likely to contain objects (e.g. [SUG11]). Even though substantial speedups may be obtained with such approaches, and some of these can be combined with or used as an add-on to CNN classifiers [GDD14], they remain firmly rooted in the window classifier design for object detection and only exploit past information to inform future processing of the image in a very limited way.

A second class of approaches that has a long history in computer vision and is strongly motivated by human perception are saliency detectors (e.g. [IKN98]). These approaches prioritize the processing of potentially interesting ("salient) image regions which are typically identified based on some measure of local low-level feature contrast. Saliency detectors indeed capture some of the properties of human eye movements, but they typically do not to integrate information across fixations, their saliency computations are mostly hardwired, and they are based on low-level image properties only, usually ignoring other factors such as semantic content of a scene and task demands (but see [TOC06]).

In computer vision, attention models have been widely used for image classification [CLY15,

KDG15, XXY15] and object detection [BMK15, CL15, YPL15]. Mnih *et al.* learn an attention model that adaptively selects image regions for processing [MHG14]. However, their attention model is not differentiable. On the other hand, Gregor *et al.* employ a differentiable attention model to specify where to read/write image regions for image generation [KDG15]. Bahdanau *et al.* propose an attention model that softly weights the importance of input words in a source sentence when predicting a target word for machine translation [BMK15]. Following this, Xu *et al.* [XBK15] and Yao *et al.* [YTC15] use attention models for image captioning and video captioning respectively. These methods apply attention in the 2D spatial and/or temporal dimension. Jaderberg *et al.* have proposed a soft-attention mechanism called the Spatial Transformer module [JSZ15] which they add between the layers of CNNs. Instead of weighting locations using a softmax layer, they apply affine transformations to multiple layers of their CNN to attend to the relevant part.

Some work uses long short-term memory (LSTM) to model the spatial or temporal attention of videos. Yeung *et al.* do dense action labelling using a soft temporal attention, which is essentially a set of weights. The weights are computed based on the context [YRJ15]. Sharma *et al.* models spatial attention of videos, as does our work. Unlike their method, which uses LSTM to determine one frame's attention based on its predecessor's attention and content, our method learns a 3D DCNN for that and yields much better performance.

Chen *et al.* proposed a very similar attention module to learn the attention to multiple scales for image segmentation. The attention module uses the second to the last convolution layer of the segmentation network as input.

4.2 Approach

In this section, we first briefly introduces some commonly-adopted input modalities for DCNN and some state-of-the-art network architectures in video analysis, then we describe in details the proposed video recognition framework.

4.2.1 Classification Module

The classification module applies a 2D deep convolutional neural network to single-frame images, which gives dense predictions across locations. In the past several years, many famous network structures have been proposed, such as AlexNet [KSH12], GoogLeNet [SLJ15], VGGNet [SZ14a], ResNet [HZR16] etc. Several works have shown that deeper structures improve object recognition performance [SLJ15, SZ14a, HZR16]. In the application of video recognition, Simonyan *et al.* proposed a two-stream network with 16-layer VGGNets (VGG16), which is the first structure that outperforms the combination of hand-crafted features and shallow classifiers [SZ14b]. Our proposed classification module is built based on the two-stream model, which plays essential roles in state-of-the-art action recognition models [WXW15, WFG15, ZHS16, WXW16].

As its name indicates, two-stream architecture has two streams: spatial stream and temporal stream. Each stream is implemented using a DCNN. Spatial stream DCNN operates on individual video frames, performing action recognition from still images. Since this is essentially a image classification task, our classification module is built upon the VGG16 network pre-trained a large image classification dataset, such as the ImageNet challenge dataset.

Temporal stream DCNN is for capturing motion information and therefore use inputs from consecutive frames. First, dense optical flows of L consecutive frames are computed. A dense optical flow in frame t has a horizontal component d_x^t and a vertical component d_y^t. At location (u, v), $d_x^t(u, v)$ and $d_y^t(u, v)$ together represents the displacement at (u, v) from frame t to frame $t+1$. Then to represent the motion across a sequence of L frames starting at frame t, the optical flow components are stacked to form a volume of $2L$ channels. Formally, the input of the temporal stream DCNN $I_t \in \mathbb{R}^{h \times w \times 2L}$ is constructed as follows:

$$\begin{aligned} I_t(u, v, 2k-1) &= d_x^{t+k-1}(u, v), \\ I_t(u, v, 2k) &= d_y^{t+k-1}(u, v), \end{aligned} \tag{4.1}$$

where h and w are the height and the width of a video respectively.

We ends this section by briefly introducing the architecture of VGG16 and how to trans-

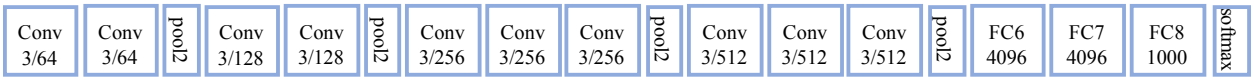


Figure 4.2: Architecture of VGG16 in [SZ14a], which adopts small convolution kernels of size 3×3 and stride 1×1, and small pooling window of size 2×2.

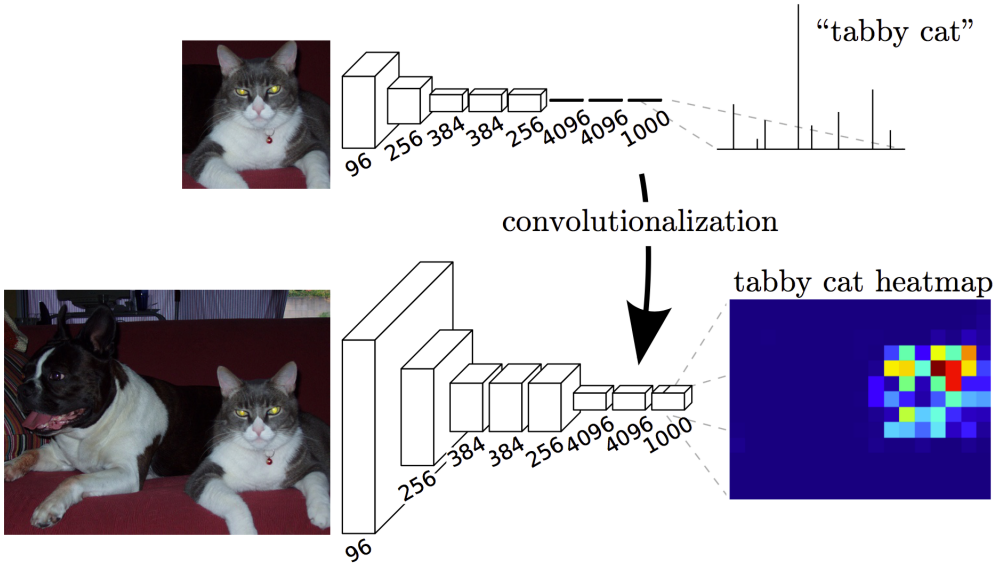


Figure 4.3: Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. The figure is borrowed from [LSD15].

form it into a FCN. In [SZ14a], the authors systematically investigated the influence of network depth on the recognition performance, by building and pre-training deeper architectures based on the shallower ones. One of the best architectures they came up with is VGG16, which is composed of 13 convolutional layers and 3 fully-connected layers. VGG16 adopts small convolution kernels of size 3×3 and stride 1×1, and small pooling window of size 2×2. Figure 4.2 shows the network architecture. Please refer to [SZ14a] for more details.

Typical DCNNs like VGG16 take fixed-sized inputs and produce non-spatial outputs. The fully connected layers of these nets have fixed dimensions and throw away spatial coordinates. However, these fully connected layers can also be viewed as convolutions with kernels that cover their entire input regions. Doing so casts them into fully convolutional

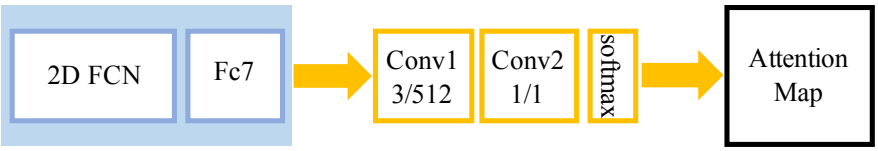


Figure 4.4: Type A attention module (orange rectangles), which takes inputs from the outputs of fc7 layer in the classification module (the blue shaded part).

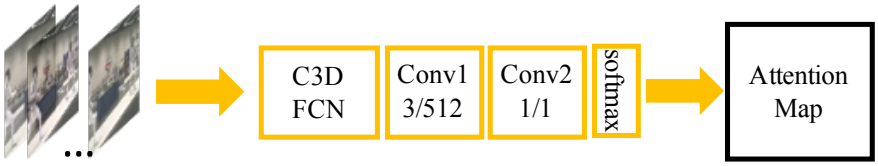


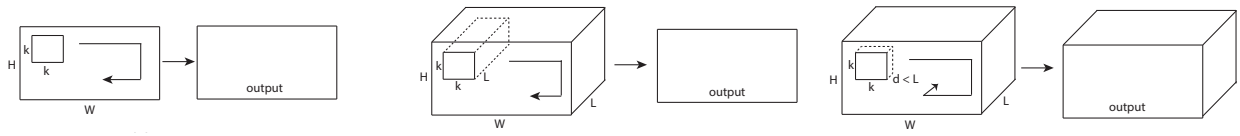
Figure 4.5: Type B attention module (orange rectangles), which takes multiple frames as input.

networks that take input of any size and output classification maps [LSD15]. This transformation is illustrated in Figure 4.3. Furthermore, while the resulting maps are equivalent to the evaluation of the original net on particular input patches, the computation is highly amortized over the overlapping regions of those patches.

4.2.2 Attention Module

As previously introduced, the attention module can take inputs from either some intermediate layer of the classification module (type A) or the input video directly (type B). Type A modules, as illustrated in Figure share the same input modality. In [CYW16], Chen *et al.* directs the output of fc7 of a FCN into an attention module which learns attentions from different scales for image segmentation. Such design is reasonable for image tasks. However, in videos of actions, most of the time the critical information lies within motions, and therefore the attention module should cover the temporal dimension.

In our proposed model, we solve this issue in two ways. One way is using type A module with a temporal stream network as the classification module, whose input is stacked optical flows. Another way is to adopt type B module, where we can choose any modalities that



(a) 2D convolution (b) 2D convolution with multiple frames (c) 3D convolution

Figure 4.6: Illustration of 2D and 3D convolutions. (a) Applying 2D convolution on an image results in an image. (b) Applying 2D convolution on a video volume (multiple frames as multiple channels) also results in an image. (c) Applying 3D convolution on a video volume results in another volume, preserving temporal information of the input signal. The figure is borrowed from [TBF14].

preserve temporal information. Straightforward choices are stacked frames or optical flows. The former choice has been shown not ideal in [SZ14b], while the latter one is essentially a two-stream model whose performance we will evaluate in Section 4.3.

A drawback of stack multiple frames or optical flows as input to a traditional 2D DCNN is that temporal information is lost after the first convolution by doing so, as shown in Figure 4.6a. The Slow Fusion model in [KTS14] extends the connectivity of all convolutional layers in time and carrying out temporal convolutions in addition to spatial convolutions to compute activations 4.6b, but its performance is not very promising.

Tran *et al.* proposed a 3D DCNN (C3D) [TBF14], which originally has 8 3D convolution, 5 3D max-pooling, and 2 fully connected layers, followed by a softmax output layer. 3D convolution and pooling have a temporal dimension for kernels, and hence preserves the temporal information of the input signals resulting in an output volume. All 3D convolution kernels of C3D are $3 \times 3 \times 3$ with stride 1 in both spatial and temporal dimensions. All the 3D pooling layers, except for the first one, are using pooling kernels size $2 \times 2 \times 2$. The first pooling layer's kernel size is $1 \times 2 \times 2$.

We apply C3D as our type B attention module due to its ability to model appearance and motion information simultaneously. In order to get a attention map out of C3D, we transform the first fully connected layer into a 2D convolutional layer as in the VGGNet, and replace

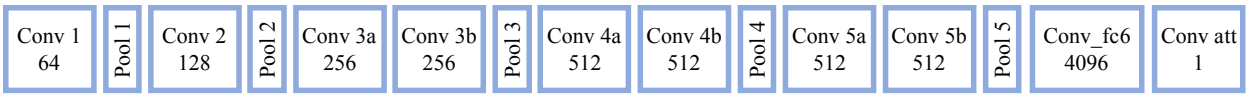


Figure 4.7: Attention module. The network inherits from C3D net 8 convolution, 5 max-pooling layers. The first fully connected layer is transformed into a 2D convolutional layer "Conv_fc6", and the last fully connected layer and the softmax output layer are replaced with a 2D convolutional layer "Conv_att". All 3D convolution kernels are $3 \times 3 \times 3$ with stride 1 in both spatial and temporal dimensions. Number of filters are denoted in each box. The 3D pooling layers are denoted from pool1 to pool5. All pooling kernels are $2 \times 2 \times 2$, except for pool1 which is $1 \times 2 \times 2$.

the last fully connected layer and the softmax output layer with a 2D convolutional layer "Conv_att" which has kernel size 1×1. The network structure is illustrated in Figure 4.7.

In both types, we use a convolutional layer of 512 filters with kernel size 3×3, followed by another convolutional layer of kernel size 1×1 to reduce the dimension to 1, to produce the attention map.

Each short clip is passed through the attention module which then produces a score map v. The score map has the same spatial dimension as the feature map produced by the feature extraction module. The final score s_c for category c at location l is

$$s_c = \sum_{i \in \mathcal{L}} a_l \cdot h_{l,c} \tag{4.2}$$

where $\mathcal{L}$ is a set of all locations, and a_l is computed as

$$a_l = \frac{\exp(v_l)}{\sum_{i \in \mathcal{L}}^{L} \exp(v_i)} \tag{4.3}$$

4.3 Experiment

In this section, we first introduce the details of our experimental settings. Then we study the effectiveness of the components of the proposed models through a series of diagnostic experiments. After this, we compare the performance of our method with the state of the art.

4.3.1 Datasets

The evaluation is performed on UCF101 [SRS12] and HMDB51 [KJG11] benchmarks. UCF101 contains $13,320$ videos of 101 action classes; HMDB51 includes $6,766$ videos of 51 actions. In both datasets, the videos of the same action class are divided into several groups; The videos from the same group may share some common features, such as similar background, similar viewpoint, etc. Both datasets provide three official splits into training and test data. The performance is measured by the average classification accuracy across the splits.

We begin by conducting diagnostic experiments on the first splits of UCF101 dataset (UCF101_split1) and HMDB51 dataset (HMDB51_split1). For comparison with the state of the art, we follow the standard evaluation protocol on both UCF101 and HMDB51.

4.3.2 Implementation Details

4.3.2.1 Classification Module

For UCF101 dataset, we initialize the parameters of the classification module with VGG16 [SZ14c] pre-trained on ImageNet dataset. The last fully connected layer is initialized with random weights. When the inputs are stacked optical flows, where in this paper we set $L = 10$, there is a dimension mismatch between the input channel and the input size of the first convolutional kernel (20 vs 3). We modify the kernel by first averaging it across the channel dimension and copying the results 20 times, as is done in [WXW15]. For HMDB51 dataset, as the number of training videos are relatively small (around 3.7K), we fine-tune the classification module trained on all videos of UCF101 dataset, a strategy that is commonly adopted, *e.g.*, [SZ14b, WFG15].

4.3.2.2 Attention Module

For type A attention module, we initialize the parameters with random weights. For type B module using the temporal network, we use the same initialization as in classification module. For the one using C3D network, the C3D part is pre-trained on Sports-1M dataset,

		None	Type A	Type B (temporal)	Type B (C3D)
Spatial	split 1	79.8%	80.0%	80.3%	80.6%
	split 2	77.3%	77.4%	77.8%	78.2%
	split 3	77.8%	77.9%	78.1%	78.4%
	average	78.4%	78.4%	78.7.0%	79.1%
Temporal	split 1	85.7%	85.8%	85.8%	86.7%
	split 2	88.2%	88.3%	88.4%	89.1%
	split 3	87.4%	87.6%	87.7%	88.0%
	average	87.0%	87.2%	87.3%	87.9%

Table 4.1: Comparison of different variants of the proposed framework on UCF101.

as in [TBF14]. The rest parameters (*i.e.*,two convultional layers) are initialized with random weights.

4.3.2.3 Training

The corner cropping and the multi-scale cropping suggested in [WXW15] are used on video frames for data augmentation. For the spatial network and the temporal network, we fix the input video size as 256×340 and randomly sample the cropping width and height from $\{256, 224, 192, 168\}$ and the cropping location from the center and the four corners. After that, we resize the cropped regions to 224×224. Horizontal flipping is applied with probability 50%. The cropped regions are resized to 112×112 for C3D network. The learning rate starts with 0.001, decreases to its 1/10 every $4,000$ iterations and stops at $10,000$ iterations. The dropout ratios for the fully connected layers are set to be 0.5, as we observed performance degradation with higher dropout ratios.

		None	Type A	Type B (temporal)	Type B (C3D)
Spatial	split 1	54.3%	54.6%	54.8%	55.2%
	split 2	50.3%	50.5%	50.8%	51.0%
	split 3	50.1%	50.4%	50.5%	50.9%
	average	51.6%	51.8%	52.0%	52.4%
Temporal	split 1	65.6%	65.8%	65.9%	66.4%
	split 2	62.4%	62.5%	62.6%	63.2%
	split 3	62.0%	62.2%	62.4%	62.7%
	average	63.3%	63.5%	63.6%	64.1%

Table 4.2: Comparison of different variants of the proposed framework on HMDB51.

4.3.2.4 Testing

We sample 25 frames of a video with equal temporal intervals. From each of these sampled frames, we obtain 10 regions, i.e. 4 corners, 1 center, and their horizontal flippings as in [WXW15]. The final prediction score is obtained by averaging across the cropped regions from all sampled frames.

4.3.3 Experimental Results

Tables 4.1 and 4.2 show the performance of different combinations of classification modules (spatial network and temporal network) and attention modules (no attention, type A attention, type B attention using the temporal network and type B attention using C3D) on UCF101 and HMDB51. Using attention generally improves the performance than not using attention. In the first half of both table, where the spatial network is used in the classification module, two Type B models perform better than the Type A one. This is because temporal inputs in this case bring complementary information to single frames. Within Type B attention modules, using C3D achieves better results than using the temporal network. Same thing happens in the second half of both tables, where the temporal

network is used in the classification module. However, this time "Type B (temporal)" only shows marginally improvement over "Type A", which is because they both use the same input modality, *i.e.*,stacked optical flows. On the other hand, as it preserves the temporal dimension in convolution layers which essentially uses a different input modality, "Type B (C3D)" improves over "Type A" and "Type B (temporal)" more significantly.

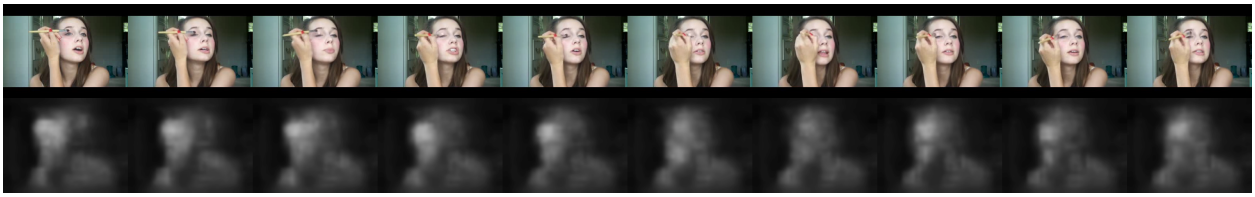
4.3.4 Visualization

In this section, we visualize the attention map learned by our model in Figure 4.8. The model we use here is temporal network with C3D as type B attention module. Although our model is estimating latent attention as no relevant supervision is available, the learned attention map can find the actual important regions in the video. For example, look at how our model can focus on the right facial parts in action "Apply Eye Makeup" and "Apply lipstick" that distinguish these two actions. Also, in other examples in the figure, our model can locate the people doing the actions, and also the objects related to the actions (*e.g.*, basketboard in "Basketball" and the dumbell in "Bench Press").

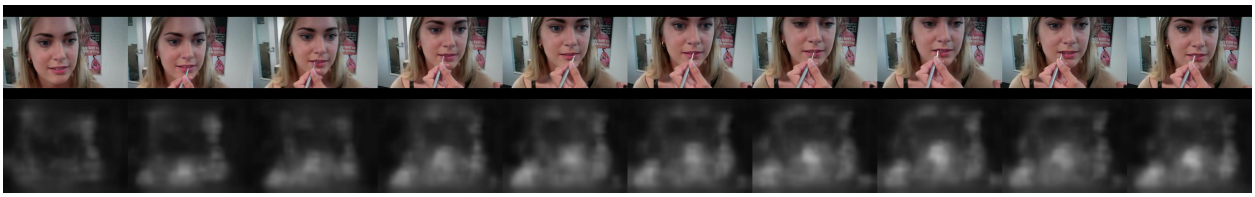
4.4 Conlusion

In this work, we introduce a novel deep action recognition method with a structure that learns latent attention on videos. By exploiting the complementary information brought by different input modalities, the learned attention maps can help the classification task. Extensive experiments on UCF 101 and HMDB51 benchmarks demonstrate this.

Ice Dancing



Apply Eye Makeup



Apply Lipstick

Basketball

Bench Press

Figure 4.8: Visualization of the attention map learned by our model. For each video, the upper row are video frames, and the bottom row is the attention map, where brighter means more attention.

CHAPTER 5

Mining Spatial and Spatio-Temporal ROIs for Action Recognition

Recognition of human actions in realistic videos is a challenging problem because of complex content, cluttered backgrounds, and large intra-class variations caused by scale and location variations and viewpoint changes [AR11].

Recently, convolutional neural networks (CNNs) have been explored for video action classification. A problem with most of these works is that they are indifferent to various parts of videos. Oftentimes, action-related information exists in certain spatial and spatio-temporal regions of interests (ROIs). For example, regions of single video frames (*i.e.*,static ROIs) include not only the people performing the action but also the objects that people interact with (*e.g.*, bicycles in *Biking*) or which often co-occur with the actions (*e.g.*, basket board in *Basketball*). Similarly, the spatio-temporal ROIs can track motion of the entire body, the motion of body parts, the movements of objects (*e.g.*, barbell in *Clean and Jerk*), and background motion (*e.g.*, sea waves in *Surfing*). Previous studies also have shown a promising direction of using the ROIs of videos to better understand the video content [SCT14, JGR13, ZNH15, ZHS16].

These considerations motivate us to propose a video action recognition method that attends to regions of the videos, instead of to the entire videos. Figure 5.1 illustrates the pipeline of our method, which consists of two models: the Static Model and the Motion Model. Both models attend to ROIs in the video to obtain discriminative action cues. The Static Model works on single video frames, and mines static ROIs (*i.e.*,2D bounding boxes). The Motion Model works on video clips (*i.e.*,a short sequence of frames), and

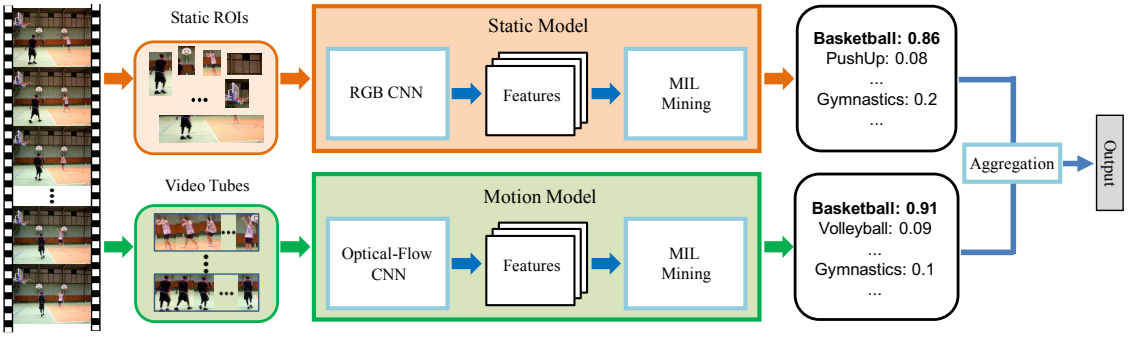


Figure 5.1: The pipeline of the proposed approach. The Static Model mines static ROIs of individual video frames, and the Motion Model mines spatio-temporal ROIs (*i.e.*,video tubes) of short video clips. The two models are fused at the end.

mines spatio-temporal ROIs which we call *video tubes*. Each video tube links a sequence of 2D bounding boxes from a sequence of video frames based on motion smoothness and appearance consistence. The Static and the Motion Models classify image frames and video clips respectively, and are combined in the final step.

Information in ROIs could be noisy, as some of ROIs are irrelevant to the actions or even causing confusions. This issue becomes worse when only video-level labeling is available. To solve this, we use multiple instance learning (MIL), where a video frame or a video clip is a "bag" and the ROIs are its "instances". In the mining component, we propose a novel aggregation module that learns to robustly combine instance features. We combine MIL with deep convolutional neural networks (CNNs) to enable joint learning of ROI mining and deep features.

In summary, main contributions of this work are:

1. We propose an unsupervised algorithm to generate spatial and spatio-temporal ROIs of videos as candidates regions that contain discriminative cues for action recognition.

2. We design a novel instance feature aggregation module in MIL and integrate it into CNN structure, enabling unified learning of deep features and the aggregation.

3. Our model achieves state of the art performance on two action recognition datasets:

UCF101 [SRS12] and HMDB51 [KJG11].

The rest of the paper is organized as following: In Section 5.1 we briefly review the literature and related work. Then we describe the proposed method in Section 5.2. Evaluation are given in Section 5.3 followed by conclusions in Section 5.4.

5.1 Related Works

Action recognition on videos has been extensively studied in computer vision community, and it is beyond the scope to review the entire literature. We refer readers to [AR11] for a detailed survey .

Recently, motivated by the great success of deep learning techniques in image-based tasks [KSH12], there have been attempts to learn deep representations for video action recognition, *e.g.*, CNN [KTS14], 3D CNN [JXY13, TBF14, VLS16], and Two-Stream CNN [SZ14b, FPZ16, WXW15]. The first deep learning framework with matching performance to the hand-crafted features [WS13] is Two-Stream network [SZ14b], which uses two separate CNNs to model color and motion. Different from previous two-stream models where features are extracted on the whole spatial extent, our model utilizes local regions (in the Static Model) and flexible video tubes (in the Motion Model). Wang *et al.* applied the trajectory-based pooling on the convolutional descriptors output by the Two-Stream network and encoded them using Fisher vector [WQT15]. Their pooling strategy shares some similarities with our motion tubes.

Action localization and spatio-temporal action proposals are popular topics in action video analysis. Some of existing methods maximize a temporal classification path of 2D boxes through static frames and hence require strong supervision (*e.g.*, locations of human actors) for learning models [WHS15, GM15, WQT16, SSS16]. In contrast to them, our video tube proposal method is unsupervised and can be applied to any videos to detect subvolumes with motions. Some other methods generate proposals based on dense trajectories [GJG15] or by grouping segments produced video segmentation [JGJ14, ORV14]. Comparing with

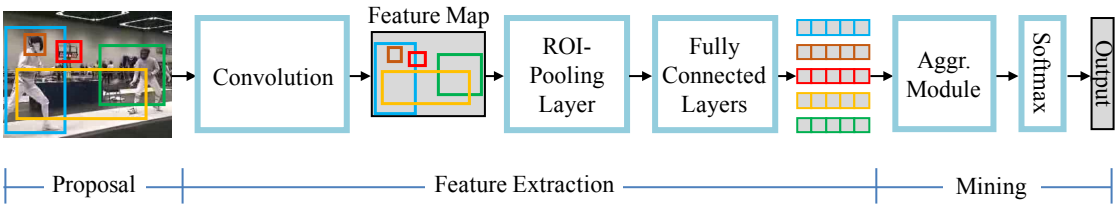


Figure 5.2: The network architecture of the Static Model. Given an image frame I, a set of 2D bounding boxes (indicated by colors) are selected as candidate static ROIs. The deep features of ROIs with the dimension equal to the number of categories are computed, which are passed to the mining component, which is composed of an aggregation module and a softmax layer that transforms the aggregated feature into final scores of actions.

these methods, our method does not need to compute dense trajectories or do video segmentation, which are usually time-consuming. In [YY15], Yu and Yuan proposed a real-time action proposal methods which does not require learning. However, their method requires human detection and dense trajectory computation. In this paper, we modify their method to remove such requirement. The existing methods are usually designed to detect human motions, while our method is able to find subvolumes of videos containing motions not only from human, which enables our models to discover more action-related cues in the videos.

Multiple instance learning (MIL) has been recently integrated within deep learning frameworks. [WYH15], [GGM15] and [XVR15] used max to combine instance-level scores [WYH15, GGM15, XVR15]. [ZHS16] proposed a stochastic sampling method to replace the deterministic max operator. Instead of picking only one of a few values, our aggregation module learns to combine all values, which is less sensitive to outliers.

There are a lot of studies about exploiting spatio-temporal subvolumes of videos for video action recognition [SCT14, JGR13, ZNH15]. We implement this idea into a deep learning framework. The work most similar to ours is by [ZHS16], which shares the same motivation and also mines discriminative spatio-temporal ROIs using MIL framework. Code for this method is not yet available so we cannot compare all the technical details, but we do compare the final performance and obtain better results.

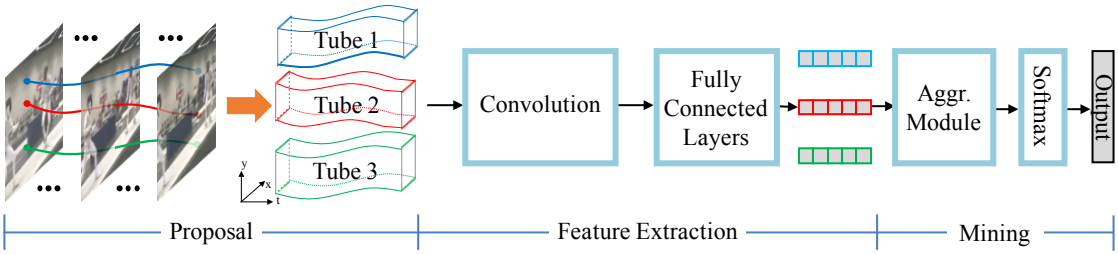


Figure 5.3: The network architecture of the Motion Model). Given a video clip, a set of video tubes (indicated by colors) are selected as candidates spatio-temporal ROIs. The deep features of ROIs with the dimension equal to the number of categories are computed, which are passed to the mining component, which is composed of an aggregation module and a softmax layer that transforms the aggregated feature into final scores of actions.

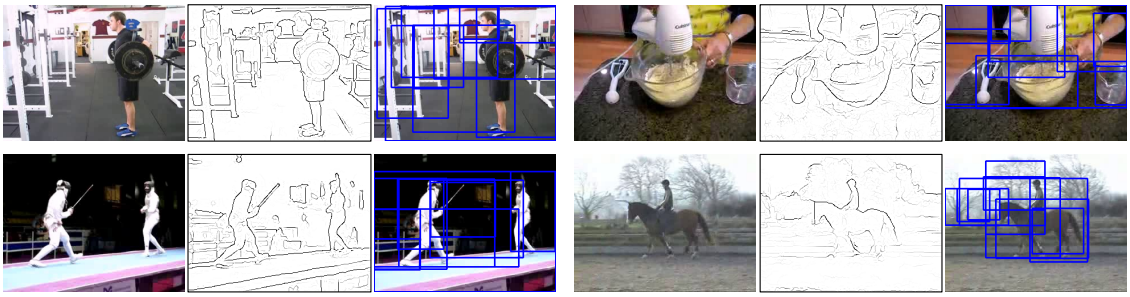


Figure 5.4: Four examples of our region proposals for Static Model. For each example, the left is the original frame image, the middle is the edge map, the right shows top 10 bounding box ROIs.

5.2 Approach

In this section, we describe in details the Static Model and the Motion Model. Figure 5.2 and Figure 5.3 shows the architectures of the two models, both of which have three steps: ROI proposal generation, ROI feature extraction, and ROI mining. First a set of candidate ROIs are generated. Then deep convolutional features of these ROIs are computed. Finally, in the mining step, we apply the MIL framework to discover discriminative information from ROIs, using video labels as bag labels, the ROIs as instances, and the features of these ROIs as the instance-level features. In the reset of this section, we will describe each step in details.

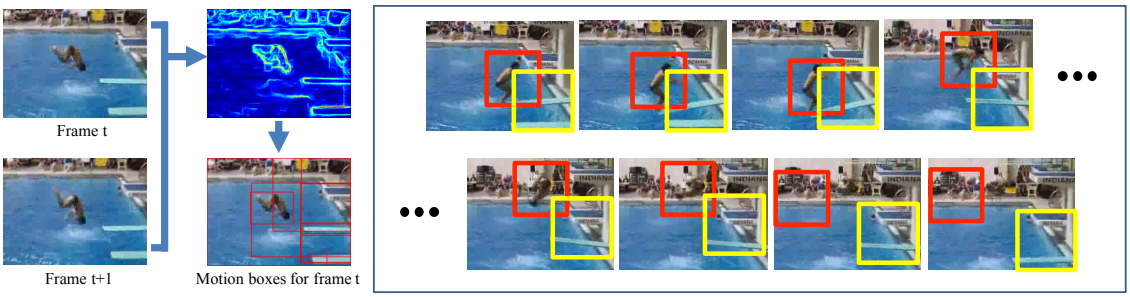


Figure 5.5: Left: Motion box generation on a single frame: two consecutive frames are used to estimation the motion boundaries which is then used as edge map input for Edge Boxes to produce motion boxes (red bounding boxes). Right: Two video tubes proposals on the first four and last four frames of a 16-frame video clips. Boxes with same color belong to the same video tube. The red tubes localizes the diver and the yellow one finds the diving board.

5.2.1 ROI Proposal Generation

Our ROI proposal algorithm is class-agnostic, since no labels of ROIs are provided. It is able to score the proposals, so that our models only need to process a few, top scored, ROIs. This saves computation time and simplifies learning discriminative classifiers.

5.2.1.1 Static ROIs

To obtain a list of K ROIs $R(I) = \{r_1, \ldots, r_K\}$ from frame I, we use the formulation of Edge Boxes [ZD14], which estimates bounding boxes for objects based on the amount of contours wholly within the box, together with an "objectiveness" score. We remove small boxes (*i.e.*,with shorter side less than 50 pixels), and keep K boxes with highest "objectiveness" scores. We also include the whole frame region in case the full background context is needed. Figure 5.4 shows some examples.

5.2.1.2 Video Tubes

Given a video clip of L frames $V = (I^1, \ldots, I^L)$, the goal is to propose a set of K video tubes $T = \{t_1, \ldots, t_K\}$, where each tube $t_k = (r_k^1, \ldots, r_k^L)$ is a temporal series of 2D bounding boxes r_k^l that localize motions. We call these 2D bounding boxes "motion boxes". Our algorithm build up a video tube by generating motion boxes on individual frames in V and then linking the boxes across frames to form video tubes.

The left part of Figure 5.5 illustrates motion box generation on a single frame I. Unlike the object boxes in the Static Model, motion boxes are intended to capture moving parts in the video. We apply Edge Boxes again, but use the motion boundaries [WRH15] detected based on two consecutive image frames as edge map. In this case, the objectiveness score estimated by Edge Boxes actually reflects the amount of motion contours within in a motion box b, which we call the "motionness" score $m(b)$.

Once we have motion boxes on individual frames, we produce a set of video tubes by linking boxes across frames. A common strategy is defining an energy function with unary and binary potentials and applying Viterbi Algorithm to maximize the energy, *e.g.*, [GM15, WHS15, YY15]. In our case, a good video tube proposal t_k should have a high motionness score, *i.e.*,$m(t_k) = \sum_{l=1}^{L} m(r_k^l)$ is large, and satisfy the spatio-temporal smoothness constraint

$$\frac{r_k^l \cap r_k^{l+1}}{r_k^l \cup r_k^{l+1}} \geq \sigma_o, \quad l = 1, \ldots, L-1 \tag{5.1}$$

and appearance consistency constraint

$$\parallel A(r_k^l) - A(r_k^{l+1}) \parallel_2 \leq \sigma_a, \quad l = 1, \ldots, L-1 \tag{5.2}$$

where σ_o and σ_a are thresholds, and $A(\cdot)$ compute the color histogram within a box. In this paper, we use $\sigma_o = 0.5$, $\sigma_a = 0.2$ and divide R, G and B channels into 16 bins when computing color histogram.

Now for each motion box b_i^L in the last frame I^L of V, we compute the best tube ending at b_i^L, using dynamic programming

$$f(b_i^l) = \max_{b_j^{l-1} \in I^{l-1}} f(b_j^{l-1}) + m(b_i^l) + d(b_i^l, b_j^{l-1}) \tag{5.3}$$

where $d(b_i^l, b_j^{l-1})$ is $-\infty$ if b_i^l and b_i^{l-1} do not satisfy the constraints in Eq. 5.1 and Eq. 5.2, and is equal to 0 otherwise. Then we can back-trace from every $b_i^L \in M(I^L)$ to recover a video tube. For each video tube, say t_k, we first crop from l-th frame a square patch p_k^l with its center at the center of r_k^l and size

$$\begin{aligned} a = \max(&\text{median}(h(r_k^1), \ldots, h(r_k^l)), \\ &\text{median}(w(r_k^1), \ldots, w(r_k^l))) \end{aligned} \tag{5.4}$$

where $h(\cdot)$ and $w(\cdot)$ returns the height and the width of a bounding box respectively. We then update t_k by replacing r_k^l with p_k^l and obtain the final video tube t_k. Finally, we apply non-maximum suppress to prune out highly overlapping video tubes, according to their motionness scores. The right part of Figure 5.5 shows an example video tubes.

5.2.2 ROI Feature Extraction

We apply CNN to compute the deep features of ROIs. In the Static Model, the inputs to CNN are the RGB images within ROIs of I. In practice, we run forward pass once for a frame image I, and use ROI Pooling layer [Gir15] to get all the features of the ROIs. In the Motion Model, before applying CNN, we stack the optical flows of r_k^l in t_k as is done in [SZ14b].

In both models, the output of the last fully connected layer is used as feature. The dimension of the output is set to be the number of categories, denoted by C, hence the output can be seen as the unnormalized category scores.

5.2.3 ROI Mining

Our model utilizes the MIL framework to learn mining discriminative information from ROIs. Suppose the features extracted by the previous step are $\{\mathbf{s}_k\}_{k=1}^K$, where $\mathbf{s} \in \mathbb{R}^C$. In ROI mining step, they are used as instance features and are mapped to one bag-level feature by an aggregation module g:

$$\mathbf{h} = g(\mathbf{s}_1, \ldots, \mathbf{s}_k; \mathbf{w}) \tag{5.5}$$

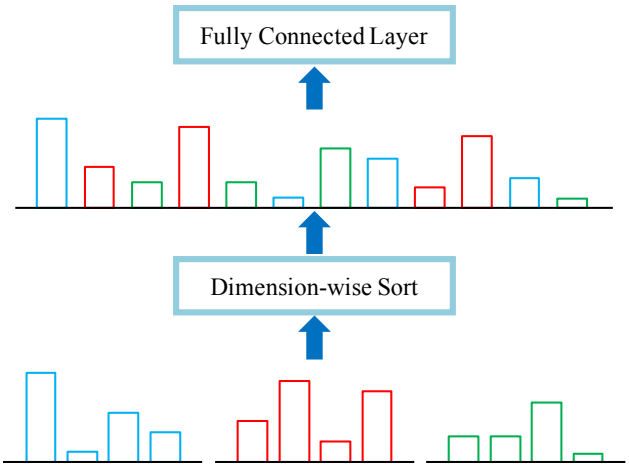


Figure 5.6: Illustration of how the proposed sort aggregation module works on a case of 3 4-D instance features (*i.e.*,$K = 3$ and $C = 4$).

where $\mathbf{w}$ is the parameters of $g(\cdot)$ and $\mathbf{h} \in \mathbb{R}^C$ is the bag-level feature. The loss we use is cross-entropy classification loss

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log p_i \tag{5.6}$$

where $\mathbf{y} = [y_1, \ldots, y_c]^t$ is the one-hot vector representing the ground truth category label and $\mathbf{p} = [p_1, \ldots, p_c]^t$ is the softmax output of $\mathbf{h}$.

5.2.3.1 Instance Aggregation

The aggregation module g in our model can be any operation that maps multiple features into one feature regardless of the order of them and that can be blended into the gradient descent optimization mechanism. An example of how our aggregation module works is given in Figure 5.6. Suppose we have a set of K instances with features $\mathbf{s}_k \in \mathbb{R}^C$, and let $\hat{\mathbf{s}}_c \in \mathbb{R}^K$ be the set of values on the c-th dimension of all $\mathbf{s}_k$. The aggregation module first sorts all values of $\hat{\mathbf{s}}_c$ in non-ascending order, yielding sorted arrays denoted by $\mathbf{o}_c (c = 1, \ldots, C)$. Then it concatenates all o_c into a single vector

$$\mathbf{O} = o_1 \oplus o_2 \cdots \oplus o_C \tag{5.7}$$

This re-arrangement can be represented by a mapping $\tau : \mathbb{Z}_K \times \mathbb{Z}_C \mapsto \mathbb{Z}_{KC}$ from components of instance features to components of $\mathbf{O}$, *i.e.*,$\tau(k, c) = t$ means the c-th component of $\mathbf{s}_k$, s_{kc}, is placed at the t-th component of $\mathbf{O}$, O_t. Finally linear projection is applied to $\mathbf{O}$, *i.e.*,

$$\mathbf{h} = \mathbf{w} \times \mathbf{O} \tag{5.8}$$

where $\mathbf{w} \in \mathbf{R}^{C \times KC}$ is the projection matrix. We name our module *sort aggregation*.

The parameters of the aggregation module, *i.e.*,$\mathbf{w}$, are learned through SGD together with the parameters of the rest models:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{w}} = (\mathbf{p} - \mathbf{y})\mathbf{O}^t. \tag{5.9}$$

To do backpropagation, we need to compute

$$\frac{\partial \mathcal{L}}{\partial \mathbf{O}} = \frac{\partial \mathbf{h}}{\partial \mathbf{O}}^t \frac{\partial \mathcal{L}}{\partial \mathbf{h}} = \mathbf{w}^t(\mathbf{p} - \mathbf{y}). \tag{5.10}$$

Then the errors can be back propagated from $\mathbf{O}$ to $\mathbf{s_k}$ by reversing the mapping τ

$$\frac{\partial \mathcal{L}}{\partial s_{kc}} = \frac{\partial \mathcal{L}}{\partial O_t} \quad \text{for } \tau(k, c) = t. \tag{5.11}$$

5.2.3.2 Discussion

What the sort aggregation does is organizing the instance features $\{\mathbf{s}_k\}_{k=1}^K$ from K ROIs into a new feature $\mathbf{O}$ which is invariant to the order of ROIs. By doing so, the linear projection can be learned to capture the discriminative patterns in $\mathbf{O}$, *i.e.*,the category scores of all ROIs.

In previous work [ZHS16, GGM15], max and avg function have been used jointly with a CNN to aggregate instance features. Our method can be seen as a generalization of these functions, *e.g.*, a $\mathbf{w}$ with all ones in the first column and zero elsewhere performs max operation.

[ZHS16] proposed a stochastic sampling operation called "stochastic-out", which randomly chooses a number from a vector with a probability proportional to the value of this number. Compared to this method, our method takes all values of $\{s_k\}_{k=1}^K$ into account, which is less sensitive to the outliers.

UCF101_split1	K	max	avg	sort
Static Model	3	76.5%	76.3%	**84.0**%
	6	78.1%	78.0%	**85.2**%
	12	79.0%	79.0%	**85.7**%
Motion Model	3	85.4%	85.3%	**88.5**%
	6	86.4%	86.4%	**89.3**%
	12	86.6%	86.5%	**89.4**%

Table 5.1: Average accuracy of different aggregation methods and ROI numbers for the Static and the Motion Models on UCF101_split1.

5.3 Experiments

In this section, we first introduce the details of our experimental settings. Then we study the effectiveness of the components of the proposed models through a series of diagnostic experiments. After this, we compare the performance of our method with the state of the art.

5.3.1 Datasets

The evaluation is performed on UCF101 [SRS12] and HMDB51 [KJG11] benchmarks. UCF101 contains $13,320$ videos of 101 action classes; HMDB51 includes $6,766$ videos of 51 actions. In both datasets, the videos of the same action class are grouped into several groups; The videos from the same group may share some common features, such as similar background, similar viewpoint, etc. Both datasets provide three official splits into training and test data. The performance is measured by the average classification accuracy across the splits.

We begin by conducting diagnostic experiments on the first splits of UCF101 dataset (UCF101_split1) and HMDB51 dataset (HMDB51_split1). For comparison with the state of the art, we follow the standard evaluation protocol on both UCF101 and HMDB51.

HMDB51_split1	K	max	avg	sort
Static Model	3	55.1%	55.1%	**55.9**%
	6	56.5%	56.4%	**57.1**%
	12	56.7%	56.7%	**57.4**%
Motion Model	3	65.6%	65.5%	**66.2**%
	6	65.7%	65.6%	**66.8**%
	12	65.7%	65.7%	**66.9**%

Table 5.2: Average accuracy of different aggregation methods and ROI numbers for the Static and the Motion Models on HMDB51_split1.

5.3.2 Implementation Details

5.3.2.1 Static Model

For UCF101 dataset, we initialize the parameters of the Static Model (*i.e.*,13 convolutional layers and the first two fully connected layers) with VGG-16 [SZ14c] pre-trained on ImageNet dataset. The last fully connected layer is initialized with random weights. For HMDB51 dataset, as the number of training videos are relatively small (around 3.7K), we fine-tune the Static Model trained on all videos of UCF101 dataset, a strategy that is commonly adopted, *e.g.*, [SZ14b, WFG15]. The learning rate starts with 0.001, decreases to its 1/10 every $4,000$ iterations and stops at $10,000$ iterations. The dropout ratios for the fully connected layers are set to be 0.5, as we observed performance degradation with higher dropout ratios. The corner cropping and the multi-scale cropping suggested in [WXW15] are used on video frames of size 256×340 to get cropped frames of size 224×224, which are later horizontally flipped with probability 50%. When test, we sample 25 frames of a video with equal temporal intervals. From each of these sampled frames, we obtain 10 regions, i.e. 4 corners, 1 center, and their horizontal flippings as in [WXW15]. The final prediction score is obtained by averaging across the cropped regions from all sampled frames.

5.3.2.2 Motion Model

When training, the videos are split into non-overlapped 10-frame clips. For each clip, the video tube proposals with fixed length $L = 10$ are generated and resized to 224×224, hence the channel of the network input is 20. For UCF101 dataset, we initialize the parameters of the Motion Model also by a pre-trained ImageNet model. To solve the problem of dimension mismatch between the input channel and input size of the first convolution kernel (20 vs. 3), we modify the kernel by first averaging it across the channel dimension and copying the results 20 times, as is done in [WXW15]. The last fully connected layer is initialized with random weights. For HMDB51 dataset, we again fine-tune the Motion Model trained on UCF101 dataset due to the smaller size of the dataset. The learning rate starts with 0.0001, decreases to its 1/10 every $10,000$ iterations and stops at $20,000$ iterations. The dropout ratios for the fully connected layers are set to be 0.5. We use horizontal flipping as data augmentation. When test, we again sample 25 temporal locations and the 10 regions, and average the prediction scores across the cropped regions from all sampled temporal locations.

5.3.2.3 Model Fusion

We perform the inference with the two models separately. For each video, we use a weighted linear combination of the prediction scores produced by the two models. We randomly choose two groups of videos from the training partition of UCF101 split1 as validation set and repeat the process three times. The weight is determined as 1/3 for the Static Model and 2/3 for the Motion Model.

5.3.3 Diagnostic Experiments

All the experiments in this subsection are conducted on UCF101_split1 and HMDB51_split1.

UCF101_split1	K	stochastic out	sort
Static Model	3	**84.1**%	84.0%
	6	84.8%	**85.2**%
	12	85.3%	**85.7**%
Motion Model	3	**88.5**%	**88.5**%
	6	89.1%	**89.3**%
	12	89.2%	**89.4**%

Table 5.3: Comparisons between stochastic out [ZHS16] and sort aggregation on UCF101_split1.

5.3.3.1 Aggregation Methods

We first compare the proposed sort aggregation ("sort") with two baseline aggregation methods, *i.e.*,max and avg in the Static and the Motion models. We try different number of ROIs K, *i.e.*,$K = 3$, $K = 6$ and $K = 12$ for each methods. The results on UCF101_split1 and HMDB51_split1 are shown in Table 5.1 and Table 5.2 respectively. We can see that in all cases, our proposed sort aggregation performs better than max and avg.

Next we compoare sort aggregation with stochastic out [ZHS16] on UCF101_split1. The method in [ZHS16] differs with ours in many parts, not just the ROI mining ("key volume mining" in the original paper). In [ZHS16], 3D volumes instead of tubes are used to represent ROIs, and one stage of preliminary training is conducted with all ROIs are assigned with video-level labels. In order to make a direct comparison of aggregation methods, we implement the stochastic out in [ZHS16] and replace the sort aggregation with it in both the Static and the Model Models. The results are shown in Table 5.3. At $K = 3$, sort aggregation performs slightly worse than (in the Static Model) or the same as (in the Motion Model) stochastic out. However, at $K = 6, 12$, ours outperform theirs with larger margins in both models. This shows when K is large, it is more beneficial to consider all values instead of a single value.

	methods	avg. accuracy
UCF101_split1	SNet-Box(6)-sort	79.8%
	S-Box(6)-sort	**85.2**%
	TNet-Tube(6)-sort	87.7%
	M-Tube(6)-sort	**89.3**%
HMDB51_split1	SNet-Box(6)-sort	56.3%
	S-Box(6)-sort	**57.1**%
	TNet-Tube(6)-sort	65.2%
	M-Tube(6)-sort	**66.8**%

Table 5.4: Comparison between joint and separate learning of deep feature and ROI mining on UCF101_split1 and HMDB51_split1.

In the following experiments, we will use sort aggregation and 6 ROIs per frame for both models, which are denoted by S-Box(6)-sort and M-Tube(6)-sort respectively. [1].

5.3.3.2 Combining Deep Feature Learning and ROI Mining

In this part, we study the benefits of combining deep feature learning and ROI mining. We use the spatial net (denoted by SNet) and the temporal net (denoted by TNet) trained in [WXW15] to replace the CNN parts of the Static Model and the Motion Model, fix their parameters and just learn the mining components. We call the two baselines SNet-Box(6)-sort and TNet-Tube(6)-sort respectively. The results are shown in Table 5.4, from which we can see that the performance drop a lot when learning the deep feature and ROI mining separately

[1] We choose 6 instead of 12 ROIs as a trade-off of accuracy and efficiency, and for the sake of fair comparison with [ZHS16]

	methods	avg. accuracy
UCF101_split1	M-[SSS16](6)-sort	87.7%
	M-APT(6)-sort	89.2%
	M-Cubic(6)-sort	89.1%
	M-Tube(6)-sort	**89.3**%
HMDB_split1	M-[SSS16](6)-sort	62.4%
	M-APT(6)-sort	66.6%
	M-Cubic(6)-sort	66.5%
	M-Tube(6)-sort	**66.8**%

Table 5.5: Comparison with alternative ROI proposals in the Motion Model.

5.3.3.3 ROI Proposal Methods

In this part, we test serveral alternative spatio-temporal proposal methods in the Motion Model. The first one is [SSS16], the state-of-the-art action localization method on several benchmarks. [SSS16] detects human actors, and is not for capturing the motion of human parts, objects and background. We select 6 top scored proposals of length 10 (if less than 6 proposals available, we just duplicate the one with the highest score) using the code provided by the authors of [SSS16]. We call this M-[SSS16](6)-sort. From Table 5.5 we can see that [SSS16] is outperformed by our video tube method. This shows the advantage of using a variety of motions not limited to human motion. The second alternative is APT [GJG15], which produces a large amount of un-scored video tubes. In order to use these proposals in our model, we score each of them with the summation of motionness scores (given by the edge box scoring function) of all boxes in the tube, and then apply non-maximum suppression to prune out highly overlapping ones. The method is denoted by M-APT(6)-sort in Table 5.5, whose performance is slightly worse than ours at the cost of the computation of dense trajectories. The last alternative is the "3D volumes" in [ZHS16], which are essentially 3D bounding boxes with edge-box scores [ZD14]. We call it M-Cubic(6)-sort. From Table 5.5

methods	avg. accuracy
IDT+FV [WS13]	85.9%
Hybrid [PWW14]	87.9%
Two-Stream [SZ14b]	88.0%
LSTM+Two-Stream [YHV15]	88.6%
C3D+iDT+SVM [TBF14]	90.4%
Hybrid LSTM [WWJ15]	91.3%
Two Stream [WXW15]	91.4%
Two-Stream Siamese [WFG15]	92.4%
Two-Stream Fusion [FPZ16]	92.5%
Key-Volume [ZHS16]	93.1%
Ours	**93.6**%

Table 5.6: State-of-the-art results on UCF101.

we can see that by using more flexible and tighter video tubes, our method is better than using 3D bounding boxes.

5.3.4 Comparison with The State of The Art

After exploring different settings of our models and comparing with the alternatives, we are ready to test our models with the best configuration on the full benchmarks of UCF101 and HMDB51. The results are summarized in Table 5.6 for UCF101 and Table 5.7 for HMDB51. We compare our methods with hand-crafted features, shallow models, and deep learning methods. On both datasets, our methods obtain the state of the art result.

5.3.5 Visualization

After the ROI feature extraction step, each ROI obtains a vector of category scores. For a video of action category c, we visualize the ROIs with top scores for action c. Figure 5.7,

method	avg. accuracy
IDT+FV [WS13]	57.2%
Two-Stream [SZ14b]	59.4%
H-VLAD [PWQ14]	59.8%
Hybrid [PWW14]	61.1%
TDD+FV [WQT15]	63.2%
Two Stream Siamese [WFG15]	63.4%
SFV [PZQ14]	66.8%
Two-Stream by us	68.0%
Key-Volume [ZHS16]	63.3%
Two-Stream Fusion [FPZ16]	65.4%
Ours	**69.6**%

Table 5.7: State-of-the-art results on HMDB51.

shows the top two static ROIs from S-Box(6)-sort, using videos from the test partition of UCF101_split1. From the figure we can see that the Static Model is able to give high scores to body parts and objects which are related to the actions. Note that in the first two frames of third row, the second scored ROIs are on the door of the building. The reason may be the person is blending into the background bushes. However, our model manages to give the basketboard top score when it cannot locate the human player. Figure 5.8 shows the top two scored video tubes from M-Tube(6)-sort, from which we can see that the Motion Model is able to acknowledge action-related spatio-temporal ROIs.

5.4 Conclusion

In this work, we introduce a novel deep action recognition method with ROIs. By exploiting video benchmarks, we find that critical representations occur with in sub-regions of videos. Based on this observation, we extract static and spatio-temporal regions of interest (ROI)

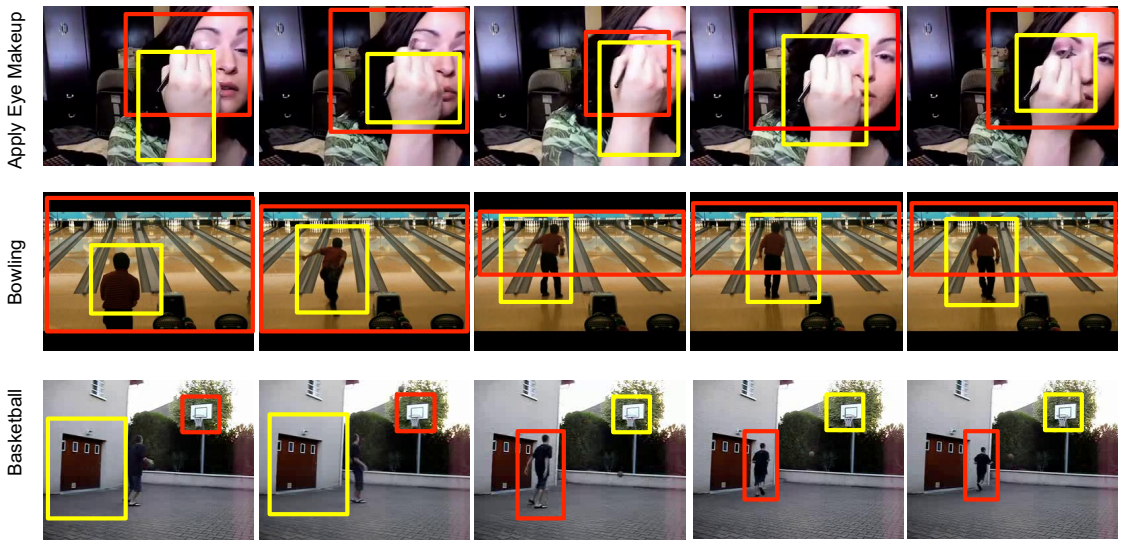


Figure 5.7: Visualization of the top two static ROIs from S-Box(6)-sort. Each row corresponds to a video from the test partition of UCF101_split1. Red box corresponds to the top score one, and the yellow is the second best one. For each video we display five frames with equal temporal intervals.

to enhance the performance of deep network. Features from different instances are naturally integrated into our MIL framework to adaptively select the most discriminative ROIs to enable end-to-end learning. Extensive experiments on UCF 101 and HMDB51 benchmarks demonstrate that our algorithm outperforms existing methods.

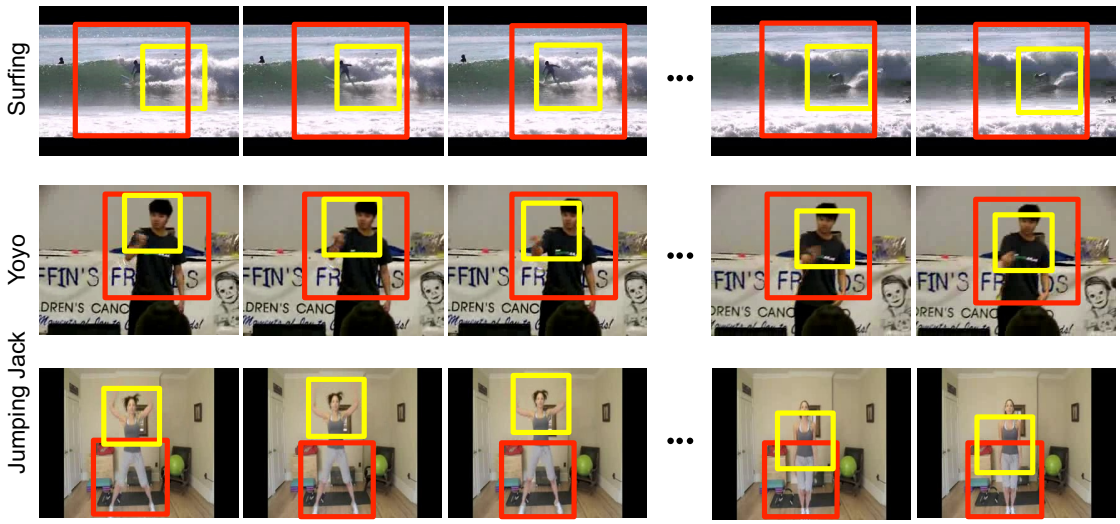


Figure 5.8: Visualization of the top two scored spatio-temporal ROIs from M-Tube(6)-sort. Each row corresponds to a video clip from the test partition of UCF101 split1. For each video clip we display first three and last two frames and omit the between. The red boxes correspond to the video tube with best action score, and the yellow is the one with second best score.

REFERENCES

[AL12] Hossein Azizpour and Ivan Laptev. "Object detection using strongly-supervised deformable part models." In *ECCV*, pp. 836–849. Springer, 2012.

[AMF11a] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. "Contour detection and hierarchical image segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, **33**(5):898–916, 2011.

[AMF11b] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. "Contour Detection and Hierarchical Image Segmentation." *IEEE Trans. Pattern Anal. Mach. Intell.*, **33**(5):898–916, 2011.

[AR11] Jake K Aggarwal and Michael S Ryoo. "Human activity analysis: A review." *ACM Computing Surveys*, **43**(3):16, 2011.

[BF11a] Yihang Bo and Charless C Fowlkes. "Shape-based pedestrian parsing." In *CVPR*, pp. 2265–2272. IEEE, 2011.

[BF11b] Yihang Bo and Charless C. Fowlkes. "Shape-based pedestrian parsing." In *CVPR*, pp. 2265–2272, 2011.

[BM09a] Lubomir Bourdev and Jitendra Malik. "Poselets: Body part detectors trained using 3d human pose annotations." In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1365–1372. IEEE, 2009.

[BM09b] Lubomir Bourdev and Jitendra Malik. "Poselets: Body Part Detectors Trained Using 3D Human Pose Annotations." In *ICCV*, 2009.

[BMK15] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. "Multiple object recognition with visual attention." *International Conference on Learning Representations*, 2015.

[BP98] Kevin Bowyer and P Jonathon Phillips. *Empirical evaluation techniques in computer vision*. IEEE Computer Society Press, 1998.

[BSG15] Jimmy Ba, Ruslan R Salakhutdinov, Roger B Grosse, and Brendan J Frey. "Learning wake-sleep recurrent attention models." In *Advances in Neural Information Processing Systems*, pp. 2593–2601, 2015.

[Can86] John Canny. "A computational approach to edge detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 679–698, 1986.

[CJG11] Xi Chen, Arpit Jain, Abhinav Gupta, and Larry S Davis. "Piecing together the segmentation jigsaw using context." In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 2001–2008. IEEE, 2011.

[CL15] Juan C Caicedo and Svetlana Lazebnik. "Active object localization with deep reinforcement learning." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2488–2496, 2015.

[CLY15] Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, et al. "Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2956–2964, 2015.

[CML14] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Nam-Gyu Cho, Sanja Fidler, and Alan Yuille Raquel Urtasun. "Detect What You Can: Detecting and Representing Objects using Holistic Models and Body Parts." In *CVPR*, 2014.

[CPK15] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Semantic image segmentation with deep convolutional nets and fully connected crfs." *ICLR*, 2015.

[CYW16] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. "Attention to Scale: Scale-aware Semantic Image Segmentation." In *CVPR*, 2016.

[DCS14] Jian Dong, Qiang Chen, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. "Towards unified human parsing and pose estimation." In *CVPR*, pp. 843–850. IEEE, 2014.

[DCX13] Jian Dong, Qiang Chen, Wei Xia, Zhongyang Huang, and Shuicheng Yan. "A deformable mixture parsing model with parselets." In *ICCV*, pp. 3408–3415. IEEE, 2013.

[DT05] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection." In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pp. 886–893. IEEE, 2005.

[DTB06] Piotr Dollar, Zhuowen Tu, and Serge Belongie. "Supervised learning of edges and object boundaries." In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR),*, volume 2, pp. 1964–1971. IEEE, 2006.

[DZ14] Piotr Dollár and C Lawrence Zitnick. "Fast edge detection using structured forests." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[EF12] David Eigen and Rob Fergus. "Nonparametric image parsing using adaptive neighbor sets." In *CVPR*, pp. 2799–2806, 2012.

[EH10] Ian Endres and Derek Hoiem. "Category independent object proposals." In *Computer Vision–ECCV 2010*, pp. 575–588. Springer, 2010.

[EVW] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. "The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results." http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

[EVW10a] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." *IJCV*, **88**(2):303–338, 2010.

[EVW10b] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." *International journal of computer vision*, **88**(2):303–338, 2010.

[EW12] S. M. Ali Eslami and Chris Williams. "A Generative Model for Parts-based Object Segmentation." In *NIPS*, pp. 100–107, 2012.

[FCN12] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. "Scene parsing with multiscale feature learning, purity trees, and optimal covers." *ICML*, 2012.

[FE73] Martin A Fischler and Robert A Elschlager. "The representation and matching of pictorial structures." *IEEE Transactions on Computers*, **100**(1):67–92, 1973.

[FGM10a] Pedro F Felzenszwalb, Ross B Girshick, and David McAllester. "Cascade object detection with deformable part models." In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pp. 2241–2248. IEEE, 2010.

[FGM10b] Pedro F. Felzenszwalb, Ross B. Girshick, David A. McAllester, and Deva Ramanan. "Object Detection with Discriminatively Trained Part-Based Models." *IEEE Trans. Pattern Anal. Mach. Intell.*, **32**(9):1627–1645, 2010.

[FH05] Pedro F Felzenszwalb and Daniel P Huttenlocher. "Pictorial structures for object recognition." *International Journal of Computer Vision*, **61**(1):55–79, 2005.

[FMR08] Pedro F. Felzenszwalb, David A. McAllester, and Deva Ramanan. "A discriminatively trained, multiscale, deformable part model." In *CVPR*, 2008.

[FPZ16] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. "Convolutional Two-Stream Network Fusion for Video Action Recognition." In *CVPR*, 2016.

[GDD14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587. IEEE, 2014.

[GFK09] Stephen Gould, Richard Fulton, and Daphne Koller. "Decomposing a scene into geometric and semantically consistent regions." In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1–8. IEEE, 2009.

[GGM15] Georgia Gkioxari, Ross Girshick, and Jitendra Malik. "Contextual action recognition with r* cnn." In *ICCV*, pp. 1080–1088, 2015.

[Gir15] Ross Girshick. "Fast r-cnn." In *ICCV*, pp. 1440–1448, 2015.

[GJG15] Jan C van Gemert, Mihir Jain, Ella Gati, and Cees GM Snoek. "APT: Action localization Proposals from dense Trajectories." In *BMVC*, volume 2, p. 4, 2015.

[GM15] Georgia Gkioxari and Jitendra Malik. "Finding action tubes." In *CVPR*, pp. 759–768, 2015.

[HAB11] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. "Semantic contours from inverse detectors." In *IEEE International Conference on Computer Vision (ICCV),*, pp. 991–998. IEEE, 2011.

[HAG15] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. "Hypercolumns for object segmentation and fine-grained localization." In *CVPR*, 2015.

[HR12] Mohsen Hejrati and Deva Ramanan. "Analyzing 3D Objects in Cluttered Images." In *NIPS*, pp. 602–610, 2012.

[HSD73] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. "Textural features for image classification." *IEEE Transactions on Systems, Man and Cybernetics*, **3**(6):610–621, 1973.

[HYK13] Xiaodi Hou, Alan Yuille, and Christoph Koch. "Boundary detection benchmarking: Beyond f-measures." In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 2123–2130. IEEE, 2013.

[HZR06] Xuming He, Richard S Zemel, and Debajyoti Ray. "Learning and incorporating top-down cues in image segmentation." In *ECCV*, pp. 338–351. Springer, 2006.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[IKN98] Laurent Itti, Christof Koch, and Ernst Niebur. "A model of saliency-based visual attention for rapid scene analysis." *IEEE Transactions on pattern analysis and machine intelligence*, **20**(11):1254–1259, 1998.

[JG06] Ya Jin and Stuart Geman. "Context and hierarchy in a probabilistic image model." In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pp. 2145–2152. IEEE, 2006.

[JGJ14] Mihir Jain, Jan van Gemert, Herve Jegou, Patrick Bouthemy, and Cees G.M. Snoek. "Action Localization with Tubelets from Motion." In *CVPR*, June 2014.

[JGR13] Arpit Jain, Abhinav Gupta, Mikel Rodriguez, and Larry Davis. "Representing videos using mid-level discriminative patches." In *CVPR*, pp. 2571–2578, 2013.

[JSZ15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks." In *Advances in Neural Information Processing Systems*, pp. 2017–2025, 2015.

[JXY13] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. "3D convolutional neural networks for human action recognition." *TPAMI*, **35**(1):221–231, 2013.

[KDG15] G Karol, I Danihelka, A Graves, D Rezende, and D Wierstra. "DRAW: a recurrent neural network for image generation." In *ICML*, pp. 1462–1471, 2015.

[KJG11] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. "HMDB: a large video database for human motion recognition." In *ICCV*, pp. 2556–2563. IEEE, 2011.

[KK10] M Pawan Kumar and Daphne Koller. "Efficiently selecting regions for scene understanding." In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3217–3224. IEEE, 2010.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[KTS14] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. "Large-scale video classification with convolutional neural networks." In *CVPR*, pp. 1725–1732, 2014.

[KY00] Scott Konishi and Alan L Yuille. "Statistical cues for domain specific image segmentation with performance analysis." In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pp. 125–132. IEEE, 2000.

[KYC99] Scott Konishi, Alan L Yuille, James Coughlan, and Song Chun Zhu. "Fundamental bounds on edge detection: An information theoretic evaluation of different edge cues." In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1. IEEE, 1999.

[KYC03] Scott Konishi, Alan L Yuille, James M Coughlan, and Song Chun Zhu. "Statistical edge detection: Learning and evaluating edge cues." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **25**(1):57–74, 2003.

[LBH08] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. "Beyond sliding windows: Object localization by efficient subwindow search." In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8. IEEE, 2008.

[LLL10] Xiao-Chen Lian, Zhiwei Li, Bao-Liang Lu, and Lei Zhang. "Max-margin dictionary learning for multiclass image categorization." In *European Conference on Computer Vision*, pp. 157–170. Springer, 2010.

[LLY14] Wenhao Lu, Xiaochen Lian, and Alan Yuille. "Parsing Semantic Parts of Cars Using Graphical Models and Segment Appearance Consistency." In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[LMB14a] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick. "Microsoft COCO: Common Objects in Context." In *ECCV*, Zrich, 2014.

[LMB14b] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick. "Microsoft COCO: Common Objects in Context." In *European Conference on Computer Vision (ECCV)*, 2014.

[LMS08] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. "Learning realistic human actions from movies." In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8. IEEE, 2008.

[Low04] David G Lowe. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision*, **60**(2):91–110, 2004.

[LRM15] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. "Bilinear cnn models for fine-grained visual recognition." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1449–1457, 2015.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In *CVPR*, pp. 3431–3440, 2015.

[LVZ11] Victor S Lempitsky, Andrea Vedaldi, and Andrew Zisserman. "Pylon Model for Semantic Segmentation." In *NIPS*, volume 24, pp. 1485–1493, 2011.

[LWT13] Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Pedestrian parsing via deep decompositional network." In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pp. 2648–2655. IEEE, 2013.

[LYT09] Ce Liu, Jenny Yuen, and Antonio Torralba. "Nonparametric Scene Parsing via Label Transfer." In *CVPR*, 2009.

[LYT11] Ce Liu, Jenny Yuen, and Antonio Torralba. "Nonparametric Scene Parsing via Label Transfer." *IEEE Trans. Pattern Anal. Mach. Intell.*, **33**(12):2368–2382, 2011.

[LZD13] Jasmine J Lim, C Lawrence Zitnick, and Piotr Dollár. "Sketch tokens: A learned mid-level representation for contour and object detection." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),*, pp. 3158–3165. IEEE, 2013.

[MBH10] Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. "Stacked Hierarchical Labeling." In *ECCV (6)*, pp. 57–70, 2010.

[MCL14a] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. "The Role of Context for Object Detection and Semantic Segmentation in the Wild." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[MCL14b] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. "The role of context for object detection and semantic segmentation in the wild." In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 891–898. IEEE, 2014.

[MFM04] David R Martin, Charless C Fowlkes, and Jitendra Malik. "Learning to detect natural image boundaries using local brightness, color, and texture cues." *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, **26**(5):530–549, 2004.

[MFT01] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics." In *International Conference on Computer Vision (ICCV)*, volume 2, pp. 416–423. IEEE, 2001.

[MHG14] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. "Recurrent models of visual attention." In *Advances in neural information processing systems*, pp. 2204–2212, 2014.

[ORV14] Dan Oneata, Jerome Revaud, Jakob Verbeek, and Cordelia Schmid. "Spatio-Temporal Object Detection Proposals." In *ECCV*. Springer, 2014.

[PWQ14] Xiaojiang Peng, Limin Wang, Yu Qiao, and Qiang Peng. "Boosting vlad with supervised dictionary learning and high-order statistics." In *ECCV*, pp. 660–674. Springer, 2014.

[PWW14] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice." *arXiv preprint arXiv:1405.4506*, 2014.

[PZQ14] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. "Action recognition with stacked fisher vectors." In *ECCV*, pp. 581–595. Springer, 2014.

[Ren00] Ronald A Rensink. "The dynamic representation of scenes." *Visual cognition*, **7**(1-3):17–42, 2000.

[RTM08] B. Russell, A. Torralba, K. Murphy, and W. T. Freeman. "LabelMe: a database and web-based tool for image annotation." *IJCV*, 2008.

[SBB10] Leonid Sigal, Alexandru O Balan, and Michael J Black. "Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion." *IJCV*, **87**(1-2):4–27, 2010.

[SCT14] Michael Sapienza, Fabio Cuzzolin, and Philip HS Torr. "Learning discriminative space–time action parts from weakly labelled videos." *IJCV*, **110**(1):30–47, 2014.

[SGS06] Eitan Sharon, Meirav Galun, Dahlia Sharon, Ronen Basri, and Achi Brandt. "Hierarchy and adaptivity in segmenting visual scenes." *Nature*, **442**(7104):719–846, June 2006.

[SJC08] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. "Semantic texton forests for image categorization and segmentation." In *CVPR*. IEEE, 2008.

[SKH12] Nathan Silberman, Pushmeet Kohli, Derek Hoiem, and Rob Fergus. "Indoor Segmentation and Support Inference from RGBD Images." In *ECCV*, pp. 746–760, 2012.

[SL07] Silvio Savarese and Fei-Fei Li. "3D generic object categorization, localization and pose estimation." In *ICCV*, pp. 1–8, 2007.

[SLJ15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[SRS12] K. Soomro, A. Roshan Zamir, and M. Shah. "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild." In *CRCV-TR-12-01*, 2012.

[SSS16] Suman Saha, Gurkirt Singh, Michael Sapienza, Philip HS Torr, and Fabio Cuzzolin. "Deep Learning for Detecting Multiple Space-Time Action Tubes in Videos." In *BMVC*, 2016.

[SUG11] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. "Segmentation as selective search for object recognition." In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 1879–1886. IEEE, 2011.

[SWR09] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context." *International Journal of Computer Vision*, **81**(1):2–23, 2009.

[SZ14a] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." *CoRR*, **abs/1409.1556**, 2014.

[SZ14b] Karen Simonyan and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." In *NIPS*, pp. 568–576, 2014.

[SZ14c] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556*, 2014.

[TBF14] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "Learning spatiotemporal features with 3d convolutional networks." *ICCV*, 2014.

[TFL08a] Alexander Thomas, Vittorio Ferrari, Bastian Leibe, Tinne Tuytelaars, and Luc J. Van Gool. "Using Recognition to Guide a Robot's Attention." In *Robotics: Science and Systems*, 2008.

[TFL08b] Alexander Thomas, Vittorio Ferrari, Bastian Leibe, Tinne Tuytelaars, and Luc Van Gool. "Using Recognition to Guide a Robot's Attention." 2008.

[TKP15] Stavros Tsogkas, Iasonas Kokkinos, George Papandreou, and Andrea Vedaldi. "Semantic Part Segmentation with Deep Learning." *arXiv preprint arXiv:1505.02438*, 2015.

[TL10] Joseph Tighe and Svetlana Lazebnik. "SuperParsing: Scalable Nonparametric Image Parsing with Superpixels." In *ECCV*, 2010.

[TL13] Joseph Tighe and Svetlana Lazebnik. "Finding things: Image parsing with regions and per-exemplar detectors." In *CVPR*, pp. 3001–3008. IEEE, 2013.

[TOC06] Antonio Torralba, Aude Oliva, Monica S Castelhano, and John M Henderson. "Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search." *Psychological review*, **113**(4):766–786, 2006.

[Vek00] Olga Veksler. "Image segmentation by nested cuts." In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pp. 339–344. IEEE, 2000.

[VJ01] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features." In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001*, volume 1, pp. 511–518. IEEE, 2001.

[VLS16] Gül Varol, Ivan Laptev, and Cordelia Schmid. "Long-term Temporal Convolutions for Action Recognition." *arXiv:1604.04494*, 2016.

[WFG15] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. "Actions˜ Transformations." *arXiv preprint arXiv:1512.00795*, 2015.

[WHS15] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. "Learning to Track for Spatio-Temporal Action Localization." In *ICCV)*, December 2015.

[WQT15] Limin Wang, Yu Qiao, and Xiaoou Tang. "Action recognition with trajectory-pooled deep-convolutional descriptors." In *CVPR*, pp. 4305–4314, 2015.

[WQT16] Limin Wang, Yu Qiao, Xiaoou Tang, and Luc Van Gool. "Actionness Estimation Using Hybrid Fully Convolutional Networks." In *CVPR*, pp. 2708–2717, 2016.

[WRH15] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. "Learning to detect motion boundaries." In *CVPR*, pp. 2578–2586, 2015.

[WS13] Heng Wang and Cordelia Schmid. "Action recognition with improved trajectories." In *ICCV*, pp. 3551–3558, 2013.

[WSS07] Liming Wang, Jianbo Shi, Gang Song, and I-fan Shen. "Object detection combining recognition and segmentation." In *ACCV*, pp. 189–199. Springer, 2007.

[WWJ15] Zuxuan Wu, Xi Wang, Yu-Gang Jiang, Hao Ye, and Xiangyang Xue. "Modeling spatial-temporal clues in a hybrid deep learning framework for video classification." In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pp. 461–470. ACM, 2015.

[WXW15] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. "Towards Good Practices for Very Deep Two-Stream ConvNets." *ArXiv e-prints*, July 2015.

[WXW16] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. "Temporal segment networks: towards good practices for deep action recognition." In *European Conference on Computer Vision*, pp. 20–36. Springer, 2016.

[WY15] Jianyu Wang and Alan L. Yuille. "Semantic Part Segmentation Using Compositional Model Combining Shape and Appearance." In *CVPR*, June 2015.

[WYH15] Jiajun Wu, Yu Yinan, Chang Huang, and Yu Kai. "Deep multiple instance learning for image classification and auto-annotation." In *CVPR*, pp. 3460–3469. IEEE, 2015.

[XBK15] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." In *International Conference on Machine Learning*, volume 14, pp. 77–81, 2015.

[XHE10] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. "SUN Database: Large-scale Scene Recognition from Abbey to Zoo." In *CVPR*, 2010.

[XT15] Saining Xie and Zhuowen Tu. "Holistically-Nested Edge Detection." In *International Conference on Computer Vision (ICCV),*, 2015.

[XVR15] Huijuan Xu, Subhashini Venugopalan, Vasili Ramanishka, Marcus Rohrbach, and Kate Saenko. "A Multi-scale Multiple Instance Video Description Network." *arXiv preprint arXiv:1505.05914*, 2015.

[XXY15] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. "The application of two-level attention models in deep convolutional neural network for fine-grained image classification." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 842–850, 2015.

[YHV15] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. "Beyond short snippets: Deep networks for video classification." In *CVPR*, pp. 4694–4702, 2015.

[YKO12] Kota Yamaguchi, M Hadi Kiapour, Luis E Ortiz, and Tamara L Berg. "Parsing clothing in fashion photographs." In *CVPR*, pp. 3570–3577. IEEE, 2012.

[YPL15] Donggeun Yoo, Sunggyun Park, Joon-Young Lee, Anthony S Paek, and In So Kweon. "Attentionnet: Aggregating weak directions for accurate object detection." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2659–2667, 2015.

[YR03] Alan L Yuille and Anand Rangarajan. "The concave-convex procedure." *Neural Computation*, **15**(4):915–936, 2003.

[YRJ15] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. "Every moment counts: Dense detailed labeling of actions in complex videos." *arXiv preprint arXiv:1507.05738*, 2015.

[YTC15] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. "Describing videos by exploiting temporal structure." In *Proceedings of the IEEE international conference on computer vision*, pp. 4507–4515, 2015.

[YY15] Gang Yu and Junsong Yuan. "Fast action proposals for human action detection and search." In *CVPR*, pp. 1302–1311, 2015.

[ZCL08] Long Zhu, Yuanhao Chen, Yifei Lu, Chenxi Lin, and Alan Yuille. "Max margin and/or graph learning for parsing the human body." In *CVPR*, pp. 1–8. IEEE, 2008.

[ZCY10] Long Zhu, Yuanhao Chen, and Alan Yuille. "Learning a hierarchical deformable template for rapid deformable object parsing." *IEEE transactions on pattern analysis and machine intelligence*, **32**(6):1029–1043, 2010.

[ZD14] C Lawrence Zitnick and Piotr Dollár. "Edge boxes: Locating object proposals from edges." In *Computer Vision–ECCV 2014*, pp. 391–405. Springer, 2014.

[ZDG14] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. "Part-based R-CNNs for fine-grained category detection." In *CVPR*, pp. 834–849. Springer, 2014.

[ZHS16] Wangjiang Zhu, Jie Hu, Gang Sun, Xudong Cao, and Yu Qiao. "A key volume mining deep framework for action recognition." In *CVPR*, pp. 1991–1999, 2016.

[ZM07] Song-Chun Zhu, David Mumford, et al. "A stochastic grammar of images." *Foundations and Trends® in Computer Graphics and Vision*, **2**(4):259–362, 2007.

[ZNH15] Yang Zhou, Bingbing Ni, Richang Hong, Meng Wang, and Qi Tian. "Interaction part mining: A mid-level approach for fine-grained action recognition." In *CVPR*, pp. 3323–3331, 2015.

[ZR12] Xiangxin Zhu and Deva Ramanan. "Face detection, pose estimation, and landmark localization in the wild." In *CVPR*, pp. 2879–2886, 2012.

[ZTM15] Yan Zhu, Yuandong Tian, Dimitris Mexatas, and Piotr Dollár. "Semantic Amodal Segmentation." *arXiv preprint arXiv:1509.01329*, 2015.

[ZWY13] Jun Zhu, Baoyuan Wang, Xiaokang Yang, Wenjun Zhang, and Zhuowen Tu. "Action recognition with actons." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3559–3566, 2013.