

Contextual Visual Recognition from Images and Videos

Georgia Gkioxari
Jitendra Malik

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-132

<http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-132.html>

July 19, 2016



Copyright © 2016, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Contextual Visual Recognition from Images and Videos

by

Georgia Gkioxari

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jitendra Malik, Chair
Professor Alexei (Alyosha) Efros
Professor Bruno Olshausen

Summer 2016

Contextual Visual Recognition from Images and Videos

Copyright 2016
by
Georgia Gkioxari

Abstract

Contextual Visual Recognition from Images and Videos

by

Georgia Gkioxari

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Jitendra Malik, Chair

Object recognition from images and videos has been a topic of great interest in the computer vision community. Its success directly impacts a wide variety of real-world applications; from surveillance and health care to self-driving cars and online shopping.

Objects exhibit organizational structure in their real-world setting (Biederman *et al.*, 1982). Contextual reasoning is part of human’s visual understanding and has been modeled by various efforts in computer vision in the past (Torralba, 2001). Recently, object recognition has reached a new peak with the help of deep learning. State-of-the-art object recognition systems use convolutional neural networks (CNNs) to classify regions of interest in an image. The visual cues extracted for each region are limited to the content of the region and ignore the contextual information from the scene. So the question remains, how can we enhance convolutional neural networks with contextual reasoning to improve recognition?

Work presented in this manuscript shows how contextual cues conditioned on the scene and the object can improve CNNs’ ability to recognize difficult, highly contextual objects from images. Turning to the most interesting object of all, *people*, contextual reasoning is a key for the fine-grained tasks of action and attribute recognition. Here, we demonstrate the importance of extracting cues in an instance-specific and category-specific manner tied to the task in question. Finally, we study motion which captures the change in shape and appearance in time and is a way to extract dynamic contextual cues. We show that coupling motion with the complementary signal of static visual appearance leads to a very effective representation for action recognition from videos.

To my parents and sister.

Contents

Contents	ii
List of Figures	iv
List of Tables	vii
1 Introduction	1
1.1 Contextual Object Recognition	2
1.2 Part-based Object Recognition	3
1.3 Visual Recognition with Convolutional Neural Networks	5
2 Contextual Object Recognition	6
2.1 Related Work on Object Detection	6
2.2 The Spotlight Network	7
2.2.1 Spotlight Channel	7
2.2.2 The Tower Network	9
2.3 Experiments	9
2.3.1 Variants of Spotlight Networks	10
2.3.2 Tower Network	11
2.3.2.1 Classifying Ground Truth Regions	11
2.3.2.2 Object Detection on MS COCO	13
3 Action and Attribute Recognition	15
3.1 Related Work on Action and Attribute Recognition	15
3.2 Deep Part-based Action and Attribute Recognition	17
3.2.1 Deep Part Detectors	18
3.2.1.1 Feature pyramid	18
3.2.1.2 Designing Part models	19
3.2.1.3 Learning Part Models	20
3.2.2 Experiments	22
3.2.2.1 System Variations	23
3.2.2.2 Action Classification	23

3.2.2.3	Attribute Classification	25
3.3	R*CNN	29
3.3.1	Architecture	29
3.3.2	Learning	31
3.3.3	Experiments	31
3.3.3.1	Control Experiments	31
3.3.3.2	Performance on PASCAL VOC	33
3.3.3.3	Performance on MPII Human Pose Dataset	35
3.3.3.4	Performance on Stanford 40 Actions Dataset	38
3.3.3.5	Performance on Berkeley Attributes of People Dataset	39
4	Action Recognition From Videos	42
4.1	Related Work on Action Recognition from Videos	42
4.2	Action Tubes	44
4.2.1	Regions of Interest	44
4.2.2	Action Specific Classifiers	45
4.2.2.1	CNNs for Action Detection	45
4.2.2.2	Training Action Specific SVM Classifiers	47
4.2.3	Linking Action Detections	48
4.3	Experiments	48
4.3.1	Results on UCF Sports	49
4.3.2	Results on J-HMDB	51
5	Conclusion	56
	Bibliography	58

List of Figures

1.1	The ideal recognition system. Left: An image of a scene Right: The desired output of an object recognition system. Source: Jitendra Malik	1
1.2	An example of a positional violation of objects in a scene. Source: Biederman <i>et al.</i> [1]	3
1.3	Measuring dependencies between objects. The distribution of locations, sizes and shapes of a target object (shown in yellow) conditioned on the presence of a reference object (shown in red), drawn from the LabelMe dataset. Source: Oliva & Torralba [2].	4
1.4	Part-based Object Recognition. Left: Generalized cylinders are used to model the anatomic parts of an object in order to explain its shape and appearance. Right: A part-based description of a face. The various parts, e.g. eyes, nose, and their spatial configuration are explicitly defined in order to holistically describe the face.	4
1.5	CNNs for digit recognition. The input image is being processed through a hierarchy of local convolutions, non-linear activation functions and pooling. A classifier is trained to predict whether the input image belongs to one of 10 classes. . . .	5
2.1	Tower architecture for recognizing objects in context. The Spotlight Network (top stream) takes as input the whole image along with the <i>spotlight channel</i> , which indicates the region of interest. The Region Network (bottom stream) is network, such as fast R-CNN, which classifies the region solely from its content. The two networks are combined at the penultimate layer, which is followed by the final classification layer.	8
2.2	Activations of a Spotlight Network. The first column shows the input image. The second column shows the spotlight channel of a region in question. The channel is injected at three different layers of the network. The third column shows the top 3 predictions made by the network.	12
2.3	Predictions made by our Tower Network compared to R-CNN. The first column shows the image and the region in question (shown with a red rectangle). The second and third column show the top-3 predictions made by R-CNN and the tower, respectively.	14

3.1	Schematic overview of our overall approach. (a) Given an R-CNN person detection (red box), we detect parts using a novel, deep version of poselets. (b) The detected whole-person and part bounding boxes are input into a fine-grained classification engine to produce predictions for actions and attributes.	18
3.2	Schematic overview of our part detectors. (a) A gaussian pyramid is build from an input image. (b) Each level of the pyramid is fed into a truncated SuperVision CNN. (c) The output is a pyramid of pool5 feature maps. (d) Each level of the feature pyramid is convolved with the part models. (e) The output is a pyramid of part model scores	19
3.3	Examples of clusters for the three body areas, head, torso and legs (left) and their top few detections on PASCAL VOC val 2009 (right). The first two rows correspond to cluster examples for head, the following two for torso and the last two for legs.	21
3.4	Schematic overview of our approach for fine grained classification using parts. (a) We consider regions of part activations. (b) Each part is forward propagated through a CNN. (c) The output is the fc ₇ feature vector for each input. (d) The features are concatenated and fed into linear SVM classifiers. (e) The classifiers produce scores for each class.	24
3.5	Top action predictions on the test set. Different blocks correspond to different actions.	26
3.6	Top errors of classification for two of the attribute categories, <i>Has Glasses</i> (top) and <i>Has T-Shirt</i> (bottom).	28
3.7	Top attribute predictions on the test set. Each block corresponds to a different attribute	28
3.8	Schematic overview of our approach. Given image I , we select the primary region to be the bounding box containing the person (red box) while region proposals define the set of candidate secondary regions (green boxes). For each action α , the most informative secondary region is selected (max operation) and its score is added to the primary. The <i>softmax</i> operation transforms scores into probabilities and forms the final prediction.	30
3.9	Top predictions on the PASCAL VOC Action test set. The instance in question is shown with a red box, while the selected secondary region with a green box. The nature of the secondary regions depends on the action and the image itself. Even within the same action category, the most informative cue can vary.	34
3.10	Top mistakes on the PASCAL VOC Action val set. The misclassified instance is shown in red, while the selected secondary region in green.	36
3.11	Performance on MPII val for RCNN (blue) and R*CNN (brown). Left: AP (%) for all actions as a function of their training size (x -axis). Right: Mean AP (%) for three discrete ranges of training size (x -axis).	37
3.12	Mean AP (%) on MPII test for R*CNN across actions in descending order of their training size. A direct comparison with published results, as shown in Figure 1(b) in [3], can be drawn.	38

3.13	Predictions on the MPII test set. We highlight the person in question with a red box, and the secondary region with a green box. The predicted action label is overlaid.	39
3.14	AP (%) of R*CNN on the Stanford 40 dataset per action. Performance varies from 70.5% for <i>texting message</i> to 100% for <i>playing violin</i> . The average AP across all actions achieved by our model is 90.9%.	40
3.15	Results on the Berkeley Attributes of People test set. We highlight the person in question with a red box, and the secondary region with a green box. The predicted attribute is overlaid.	41
4.1	An outline of our approach. (a) Candidate regions are fed into action specific classifiers, which make predictions using static and motion cues. (b) The regions are linked across frames based on the action predictions and their spatial overlap. <i>Action tubes</i> are produced for each action and each video.	45
4.2	We use action specific SVM classifiers on spatio-temporal features. The features are extracted from the fc_7 layer of two CNNs, <i>spatial-CNN</i> and <i>motion-CNN</i> , which were trained to detect actions using static and motion cues, respectively.	46
4.3	Performance on UCF Sports. Left: ROC curves on UCF Sports for an intersection-over-union threshold of $\sigma = 0.2$. Red shows our approach. We manage to reach a high true positive rate at a much smaller false positive rate, compared to the other approaches shown on the plot. Right: AUC on UCF Sports for various values of intersection-over-union threshold of σ (x -axis). Red shows our approach. We consistently outperform other approaches, with the biggest improvement being achieved at high values of overlap ($\sigma \geq 0.4$).	50
4.4	Examples from UCF Sports. Each block corresponds to a different video. We show the highest scoring action tube detected in the video. The red box indicates the region and the predicted label is overlaid. We show 4 frames from each video. The top example on the right shows the problem of tracking, while the 4th example on the right is a wrong prediction, with the true label being <i>Skate Boarding</i>	51
4.5	AUC on J-HMDB for different values of intersection-over-union threshold (averaged over the three splits).	52
4.6	The confusion matrix on J-HMDB for the classification task, when using action tubes to predict a label for each video.	54
4.7	Examples from J-HMDB. Each block corresponds to a different video. We show the highest scoring action tube detected in the video. The red box indicates the region and the predicted label is overlaid. We show 4 frames from each video. The 2nd example on the left and the two bottom ones on the right are wrong predictions, with true labels being <i>catch</i> , <i>sit</i> and <i>run</i> respectively.	55
5.1	Towards home robots. Visual recognition is a key component towards home robots in order for robots to efficiently assist users.	57

List of Tables

2.1	Mean AP on ground truth regions of the validation set of MS COCO for three different variants of spotlight networks. <i>binary</i> and <i>dt</i> denote the binary and distance transform version of the spotlight channels.	11
2.2	Mean AP on ground truth regions of the validation set of MS COCO for fast R-CNN [4], R-CNN [5] and our tower network.	12
2.3	AP (%) for object detection on the validation set of MS COCO. We compare our tower network with fast R-CNN and report the AP for the categories that improved the most compared to fast R-CNN.	13
3.1	AP for each part type on PASCAL VOC val 2009. We evaluate the part activations and measure AP for different thresholds of intersection-over-union.	22
3.2	AP on the PASCAL VOC 2012 Actions test set. The first three rows show results of two other methods. Action Poselets [6] is a part based approach using HOG features, while Oquab <i>et al.</i> [7], Hoai [8] and Simonyan & Zisserman [9] are CNN based approaches. <i>Ours (no parts)</i> is the baseline approach of our method, when only the ground truth box is considered, while <i>Ours</i> is the full approach, including parts. All approaches use ground truth boxes at test time.	25
3.3	AP on the PASCAL VOC 2012 Actions val set of our approach. <i>Ours (no parts)</i> is our approach without parts. <i>Ours (3-way split)</i> is our approach when parts are defined as the three horizontal splits comprising an instance box. <i>Ours (joint fine-tuning)</i> uses a CNN fine-tuned jointly on the instances and the parts, while <i>Ours (instance fine-tuning)</i> uses a single CNN fine-tuned just on the instance box. All the above variations of our approach use ground truth information at test time as the object bound. <i>Ours (R-CNN bbox)</i> uses R-CNN detections for person.	26

3.4	AP on the test set of the Berkeley Attributes of People Dataset. All approaches on the top use ground truth boxes for evaluation. <i>Ours (no parts)</i> is the baseline approach with no parts. <i>Ours (3-way split)</i> is a variant of our approach, where parts are defined as the three horizontal splits comprising an instance box. <i>Ours (instance fine-tuning)</i> uses a CNN fine-tuned on instance boxes, while <i>Ours (joint fine-tuning)</i> uses a CNN fine-tuned jointly on instances and parts. We also show the effectiveness of our approach <i>Ours (R-CNN bbox)</i> , when no ground truth boxes are given at test time.	27
3.5	AP on the PASCAL VOC Action 2012 val set. <i>RCNN</i> is the baseline approach, with the ground-truth region being the primary region. <i>Random-RCNN</i> is a network trained with primary the ground-truth region and secondary a random region. <i>Scene-RCNN</i> is a network trained with primary the ground-truth region and secondary the whole image. <i>R*CNN</i> (l, u) is our system where l, u define the lower and upper bounds of the allowed overlap of the secondary region with the ground truth. <i>R*CNN</i> (l, u, n_S) is a variant in which n_S secondary regions are used, instead of one.	33
3.6	AP on the PASCAL VOC Action 2012 test set. Oquab <i>et al.</i> [7] train an 8-layer network on ground-truth boxes. Gkioxari <i>et al.</i> [10] use part detectors for <i>head</i> , <i>torso</i> , <i>legs</i> and train a CNN. Hoai [11] uses an 8-layer network to extract fc7 features from regions at multiple locations and scales. Simonyan and Zisserman [12] combine a 16-layer and a 19-layer network and train SVMs on fc7 features from the image and the ground-truth box. R*CNN (with ($l = 0.2, u = 0.75$)) outperforms all other approaches by a significant margin.	33
3.7	AP on the Berkeley Attributes of People test set. PANDA [13] uses CNNs trained for each poselet type. Gkioxari <i>et al.</i> [10] detect parts and train a CNN jointly on the whole and the parts. RCNN is our baseline approach based on FRCN. Both RCNN and R*CNN do not use any additional part annotations at training time. [10] and R*CNN perform equally well, with the upside that R*CNN does not need use keypoint annotations during training.	39
4.1	AP on the UCF Sports dataset for an intersection-over-union threshold of $\sigma = 0.5$. <i>frame-AP</i> measures AP of the action detections at the frame level, while <i>video-AP</i> measures AP of the predicted action tubes.	50
4.2	Results and ablation study on J-HMDB (averaged over the three splits). We report <i>frame-AP</i> (top) and <i>video-AP</i> (bottom) for the spatial and motion component and their combination (full). The combination of the spatial- and motion-CNN performs significantly better under both metrics, showing the significance of static and motion cues for the task of action recognition.	52

4.3	Classification accuracy on J-HMDB (averaged over the three splits). CNN (third column) shows the result of the weighted average of spatial and motion-CNN on the whole frames, while Action Tubes (fourth column) shows the result after using the scores of the predicted action tubes to make decisions for the video's label.	53
-----	--	----

Acknowledgments

I would like to thank my advisor and mentor, Jitendra Malik, for the wonderful 6 years in grad school. I really enjoyed working with you, Jitendra! Also, I would like to thank my amazing lab mates, who were always there when I had questions about all sorts of gradients. Especially, I would like to thank Bharath and Saurabh for being the kindest lab mates one could ever have. I am very glad I met you! I would like to thank Ross G. for teaching me how science is done. And of course, the whole Berkeley Artificial Intelligence Research lab for the very fun retreats full of wine and laughter.

I want to thank my mom, the only one in a family of 7 to leave the farm, go to school and become a math professor. Thank you for pushing me to be good in science, even in liberal arts! And thank you for always putting us above yourself.

I would like to thank my friends, who were there with me at all times (Ross B., Katerina, Evan, Alexoukos, Vaggelis, Niki, Armin, Shiry, Ana, Cory, Sergey, Rob, Jon, Karl). Grad school without friends (and tequila) is not fun. Finally, I want to thank my two favorite people in the world, Aristeia and Argyro.

Chapter 1

Introduction

Consider Figure 1.1 (left). The grand goal of object recognition in computer vision is to build a system that accurately recognizes the scene, the objects that participate in it, their state and attributes, their 3D configuration as well as their affordances. The desired output is shown in Figure 1.1 (right).

Visual recognition is a popular task among computer vision researchers. Its success is tied to a variety of real-world applications, from surveillance and health care to self-driving cars and e-commerce. To enable its progress, the computer vision community has defined gold standards for visual recognition, including *image classification*, where the task is to predict whether an object is present in the image, and *object detection*, where the task is to predict the location and type of all objects in the image. Beyond generic objects, there has been a lot of focus on tasks directly related to people, such as action recognition, where here the goal is to recognize the actions performed by people in images or videos, and attribute classification, which aims to characterize people appearance features.

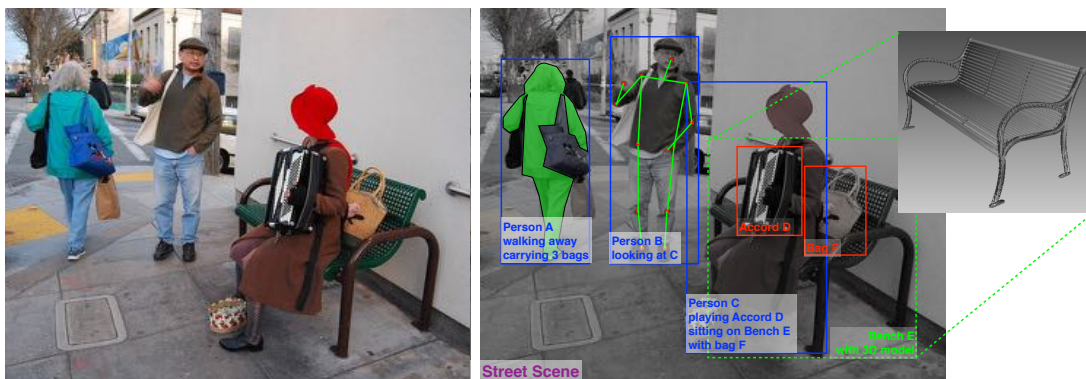


Figure 1.1: The ideal recognition system. **Left:** An image of a scene **Right:** The desired output of an object recognition system. Source: Jitendra Malik

1.1 Contextual Object Recognition

Objects exhibit organizational structure within their real world settings. Biederman *et al.* [1] identify five classes of relations among an object and its environment. Those classes are

- Interposition. Objects interrupt their background
- Support. Objects rest on surfaces
- Probability. Objects tend to be found in some scenes and not others
- Position. Given an object is probable in a scene, it often is found in some positions and not others
- Familiar size. Objects have a limited set of size relations with other objects

In their work, Biederman *et al.* quantified the importance of those relations by showing subjects examples of scenes which violated one or more of the afore mentioned classes. An example is shown in Figure 1.2, which is a scene showing a positional violation.

In computer vision, there has been a lot of effort to build object recognition systems which incorporate contextual reasoning by modelling the relationships of objects with their environment. The most prominent of such works is by Torralba [14]. In [14], an object is described as $O = \{o, \mathbf{x}, \sigma\}$ while the contextual information is captured by \mathbf{v}_C . The object likelihood $P(O | \mathbf{v}_C)$ is decomposed into three terms

$$P(O | \mathbf{v}_C) = P(\sigma | \mathbf{x}, o, \mathbf{v}_C) \cdot P(\mathbf{x} | o, \mathbf{v}_C) \cdot P(o | \mathbf{v}_C) \quad (1.1)$$

The meaning of these three terms is

- $P(o | \mathbf{v}_C)$. The most likely object types given the context information
- $P(\mathbf{x} | o, \mathbf{v}_C)$. The most likely locations of object type o given the context information
- $P(\sigma | \mathbf{x}, o, \mathbf{v}_C)$. The most likely scales of object type o in location \mathbf{x} given the context information

The location and size relations of objects in a scene are visualized in the work by Oliva and Torralba [2] as reproduced in Figure 1.3. Each square shows a reference object (in red) and its typical occurrences of neighboring objects of a target category (in yellow) drawn from the LabelMe dataset of annotated objects [15]. From these figures, it is very obvious that objects tend to co-occur with other objects under a strict set of rules regarding their relative size and location.



FIG. 1. An example of a Position violation for the fire hydrant. The camouflage rating for the fire hydrant was 5.5.

Figure 1.2: An example of a positional violation of objects in a scene. Source: Biederman *et al.* [1]

1.2 Part-based Object Recognition

Orthogonal to the context derived from the scene and in an effort to explain objects and their intricate shapes, objects have been regarded to be comprised of a set of simpler and smaller parts. Nevatia and Binford [16] model the anatomic parts of an object with generalized cylinders to explain the shape and appearance of the object, as seen in Figure 1.4 (left).

For object recognition, part-based models aim at recognizing the parts, a presumably easier task, and their spatial configuration in order to predict the location and identity of the whole. Fischler and Elschlager [17] introduce pictorial structures, which aim to recognize an object in an image given its part-based description and a metric of *goodness* of matching. Figure 1.4 (right) shows an example of a part-based description for the face. Felzenszwalb and Huttenlocher [18] presented a probabilistic framework for the same problem, which they called Pictorial Structure Model (PSM). In its original formulation, body parts were represented as rectangles and their spatial relations were captured by tree-structured graphs. The tree-structure of PSM makes inference on location, scale and orientation of the body parts exact and efficient.

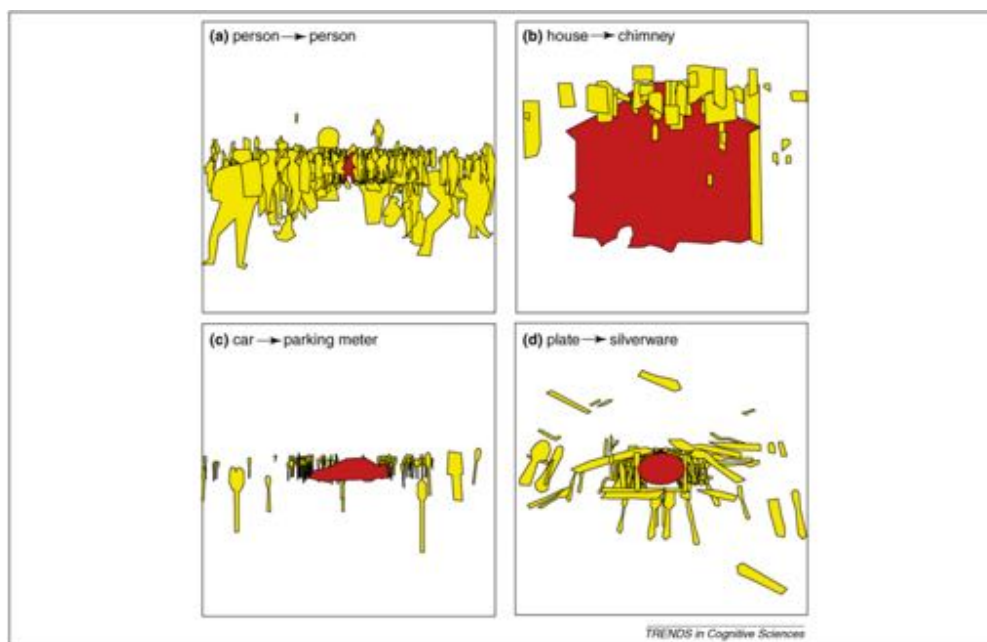


Figure 1.3: Measuring dependencies between objects. The distribution of locations, sizes and shapes of a target object (shown in yellow) conditioned on the presence of a reference object (shown in red), drawn from the LabelMe dataset. Source: Oliva & Torralba [2].

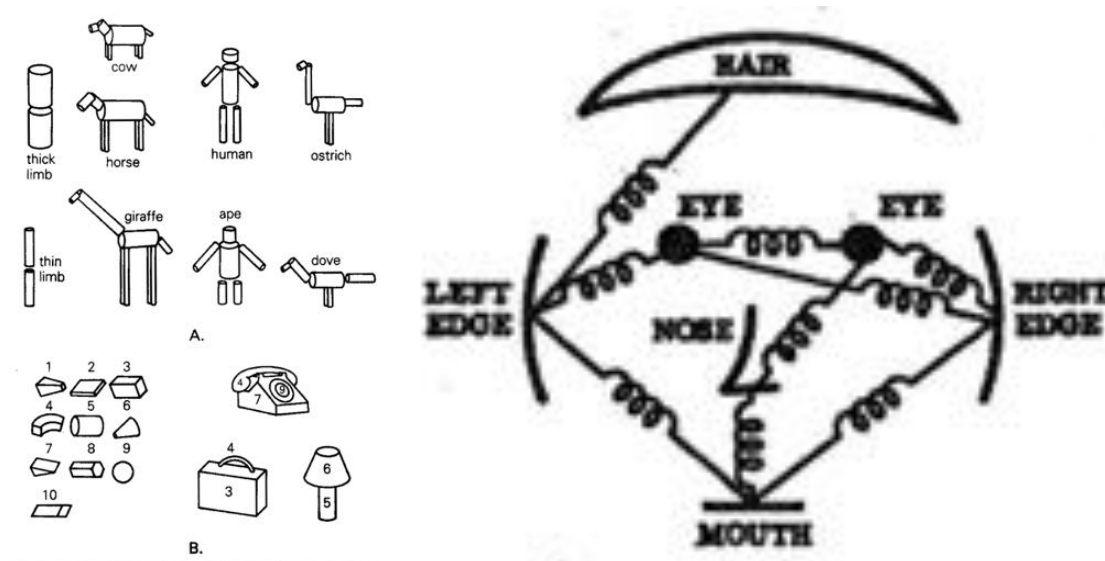


Figure 1.4: Part-based Object Recognition. **Left:** Generalized cylinders are used to model the anatomic parts of an object in order to explain its shape and appearance. **Right:** A part-based description of a face. The various parts, e.g. eyes, nose, and their spatial configuration are explicitly defined in order to holistically describe the face.

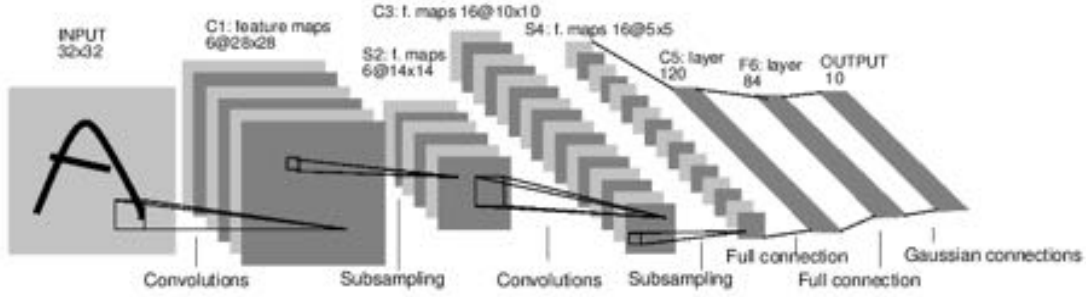


Figure 1.5: CNNs for digit recognition. The input image is being processed through a hierarchy of local convolutions, non-linear activation functions and pooling. A classifier is trained to predict whether the input image belongs to one of 10 classes.

1.3 Visual Recognition with Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have dominated the field of object recognition today. CNNs apply a hierarchy of operations to an input image. These operations are comprised of local convolutions and non-linear activation functions. As a result, CNNs embed an image into a feature space which is learned by optimizing a task-specific objective function.

LeCun *et al.* [19] applied CNNs for the task of digit recognition back, as shown in Figure 1.5. Since then, and with the arrival of big data, CNNs have shown unprecedented performance for a variety of computer vision tasks, such as image classification [20], object detection [5] and semantic segmentation [21].

In this thesis, we show how contextual and part-based reasoning can be coupled with CNNs in order to build recognition engines which detect objects in complex real-world scenes and predict people’s actions and attributes. We will show the effectiveness of using contextual cues, information that is not explicitly captured by feed forward CNNs, and how those cues can be mined automatically from the data.

Chapter 2

Contextual Object Recognition

In natural scenes, objects do not occur by themselves. They exhibit great organizational structure, both in how they interact with each other as well as the scene. There is considerable evidence that humans make use of these regularities and that context is a very important cue for visual recognition in natural settings [1, 2]. Current object detection systems fail to capture such cues and rely solely on local appearance. In this chapter, we present an effort to derive the necessary contextual cues, using *spotlight channels*, conditioned on an object in question.

2.1 Related Work on Object Detection

There have been several studies in computer vision attempting to formalize context [14, 22, 23, 24, 25, 26]. Currently the dominant approaches to visual recognition are based on neural networks, so we turn specifically to the use of context in such frameworks. In a classification setting, where one just has to declare what objects are present in an image, not where they are, this is relatively straightforward. On the ImageNet classification challenge, ever since 2012, the leading approaches [20, 9, 27] compute features over the entire image, and then output a set of labels corresponding to objects present in the image. Implicitly context is being used in the features computed at the different layers of the network, and if the network declares that a boat is present in the image it could do so using both features from the boat or the surrounding water. The network draws no distinction between object and context, as it is simply trained to minimize a loss function based on the label prediction error. Context is being exploited, without ever being made explicit.

However in the case of detection, where it is not enough for the computer to declare that the image contains a boat, but also needs to output a bounding box around the boat, the use of context has proved to be much more challenging. One of the earliest efforts in the post 2012 era to do object detection based on neural networks was OverFeat [28], where the authors tried to regress to the bounding box coordinates from the features computed on the whole image. This obviously implicitly uses context, but it does not work as com-

petitively as the R-CNN approach [5] which has since come to dominate the field of object detection. In R-CNN (Region-based Convolutional Neural Network), the starting point is a set of region proposals (to be precise, the rectangular bounding boxes associated with these proposals), which are putative guesses of the pixel supports of the different objects. Features are computed on these bounding boxes and the softmax score output by the network is used to decide whether the box corresponds to one of the objects of interest or background. Typically this is accompanied by a regression stage where a (hopefully) better bounding box is predicted, also based on neural network features. Since the original R-CNN paper, various refinements have been introduced, notably SPPNet [29] and Fast R-CNN [4]. In MultiBox [30], an approach contemporary to R-CNN, a network is trained to predict the locations of the bounding boxes which are subsequently classified, instead of using precomputed bottom-up region proposals. However, what is common in all those approaches is that the features on the object are responsible for the assignment of the object category. One may ask why this works better than OverFeat, which in principle has more information. It seems that region proposals provide an attention focusing mechanism, directly telling the network about the most relevant features - those on the object - which help localize the object accurately and thus improve performance on the detection task.

Various ad hoc ways have been proposed to add contextual features in an R-CNN like pipeline. For example features computed on the whole scene could be used, or features on a box which is centered on the object proposal box but is k times its size (e.g. [31]). One might add several such boxes displaced by varying amounts to provide for some diversity in the context. Indeed several such approaches have been tried in the literature. However we find these approaches intellectually unsatisfying, as they make arbitrary choices. One really should be learning rather than hand designing the support regions on which to collect context. But how?

2.2 The Spotlight Network

Spotlight networks operate at the image level and condition on a region's shape and location within the image. This is complementary to R-CNN based approaches, which *zoom-in* at the objects and disregard the rest of the image. However, resolution is important for tasks like object recognition and is explicitly used by R-CNN models, unlike Spotlight Networks. In order to make most of the information provided, we couple spotlight networks with R-CNN. In particular, we build a two stream network comprised of a spotlight network and a R-CNN, as shown in Figure 2.1. The two streams are fused at the penultimate layer, whose activations are combined and fed into a softmax classifier.

2.2.1 Spotlight Channel

Spotlight networks are designed to exploit the contextual information conditioned on a region in question. Assume r is a region (a bounding box in our case) of image I . We define a

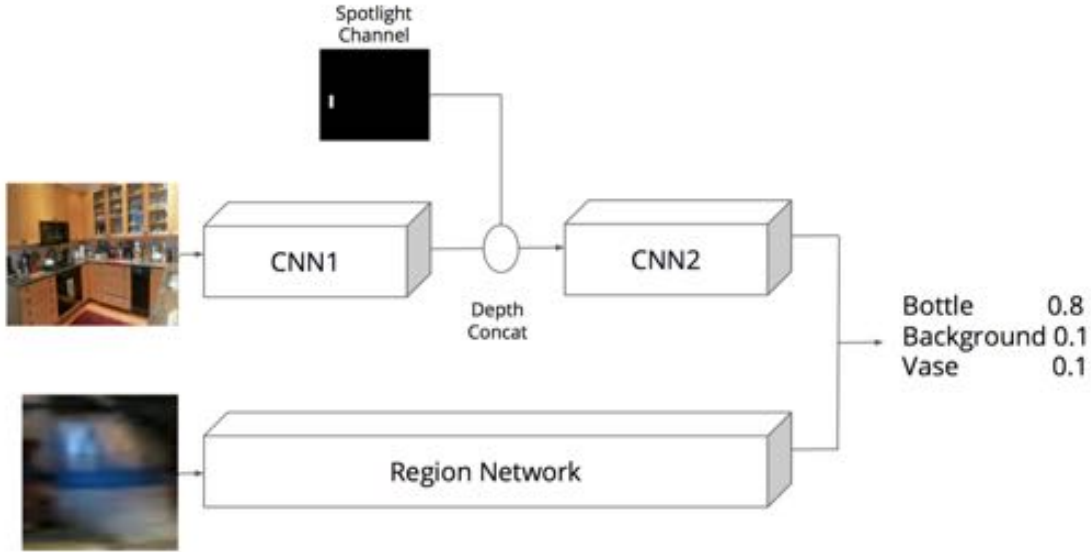


Figure 2.1: Tower architecture for recognizing objects in context. The Spotlight Network (top stream) takes as input the whole image along with the *spotlight channel*, which indicates the region of interest. The Region Network (bottom stream) is network, such as fast R-CNN, which classifies the region solely from its content. The two networks are combined at the penultimate layer, which is followed by the final classification layer.

binary *spotlight channel* for region r as

$$S_r^b(x, y) = s \cdot [(x, y) \in r] \quad (2.1)$$

where $[P]$ is the Iverson bracket notation for a boolean statement P and s is a scalar (The value of s is chosen to ensure that the features computed from spotlight channels are in the same dynamic range as the output of the previous neural network layer). This channel highlights the region of reference, while it masks out the background. Another variant used in our experiments is the distance transform of S_r , defined as

$$S_r^{dt}(x, y) = s \cdot d((x, y), r) \quad (2.2)$$

where d is a signed distance function. A variety of choices are possible, but we use two channels corresponding to distances along the x- and y-axes. The channel is zero inside the region of reference and provides explicit information of the distance of the background pixels from the region.

Spotlight networks are neural networks which use the spotlight channels in various pre-defined stages of their architecture. Assume l is such a layer somewhere in a network and $i(l; I)$ is the activation of the previous layer with image I as the input to the network. The

output activation of layer l is given by $o(l; I) = f_l(i(l; I))$ where f_l is the transformation in layer l . In a spotlight network the output of layer l is conditioned on the region, as follows

$$o(l; I, r) = f_l(\{i(l; I), S_r\}) \quad (2.3)$$

where the spotlight channels S_r are appended to the channels of activation $i(l; I)$.

To be more concrete, assume $l = \text{conv4_1}$ of the VGG16 network as defined in [9]. According to the VGG16 architecture, the input to conv4_1 is a $28 \times 28 \times 256$ matrix. The convolutional layer has 512 weight filters each of shape $3 \times 3 \times 256$. Subsequently, the output of conv4_1 is a $28 \times 28 \times 512$ matrix, since the 2D convolutions are of stride 1. Assume we inject a binary spotlight channel as well as distance transform channels in both x- and y-direction. Thus the spotlight channels to be injected are of shape $28 \times 28 \times 3$. The resulting spotlight network at conv4_1 has an input shape of $28 \times 28 \times 259$ and 512 weight filters of shape $3 \times 3 \times 259$, while the shape of the output is unchanged. If l is convolutional this is equivalent to adding a bias image to the output activations, one for each output channel. The bias image is the result of a 2D convolution between the spotlight channels and the corresponding weight filter. When convolutional layers are followed by ReLU layers (as is the case in most network architectures), adding a bias image allows for the threshold of the ReLUs to vary at every location and this allows the activations to pass depending on their location as well as their value.

2.2.2 The Tower Network

Spotlight networks process an image conditioned on a region by using the spotlight channels, as explained above. This allows them to exploit the context provided by the whole image with reference to the region in question. On the other hand, R-CNN focuses exclusively on the content of the region. The input to R-CNN is a cropped and warped region, with some predefined context padding. Zooming in the object is a property that spotlight networks do not share since the classification happens at the image level. There is a way to couple spotlight networks with R-CNN, as shown in Figure 2.1. The *Tower Network* combines the activations from the spotlight network and the R-CNN at the penultimate layer, where the activations are concatenated and fed into the final scoring layer.

2.3 Experiments

In this section we quantify the effectiveness of spotlight networks. We report results on the task of object detection on the very challenging MS COCO dataset and compare different variants of spotlight networks, in an effort to better understand the different choices of the architecture. Finally, we show examples of detections and directly compare them to a pure region network.

Dataset and Metrics. We evaluate the spotlight networks on the very challenging MS COCO dataset [32], which consists of 120K images labelled with 80 object categories. For a more detailed analysis of the dataset, we refer readers to the dataset’s [webpage](#). We report Average-Precision (AP) for each category at an intersection-over-union threshold of 0.5; a prediction is correct if it overlaps more than 0.5 with a ground truth object.

Architecture. For our experiments, we use as a base network a 16-layer CNN (VGG16) as defined in [9]. This network achieves great performance on image classification [9] and object detection [4]. For region proposals, we use MCG object proposals [33] and only evaluate the top 2000 regions per image.

2.3.1 Variants of Spotlight Networks

The variants of spotlight networks include different choices of injection layers as well as the type of spotlight channels to be injected. We compare the performance of different variants on the task of object classification.

Learning Details. We initialize the main layers with a pre trained model for the ImageNet classification task, while the weights corresponding to the spotlight channels are randomly sampled from a zero-mean gaussian distribution with standard deviation of 0.1. We use a learning rate of 0.001, and a batch size of 60. We sample 50% of the regions to be positive, with an overlap more than 0.5 with a ground truth region. The input image is resized to 224x224.

We quantify the performance of spotlight networks on the task of object classification. In this task we consider the ground truth object boxes, as provided by [32], and classify them as belonging to an object or the background. Table 2.1 shows the average AP achieved by variants of spotlight networks on ground truth regions on the MS COCO validation set. The first column indicates the type of spotlight channels used, the second column the layers of VGG16 in which those channels are injected, the third column reports mean AP. We noticed that an injection early on in the network caused the learning to ignore the spotlight channel, thus regressing to image classification, while an injection at the final convolutional block had a faster convergence rate. In terms of performance, an injection of all types of spotlight channels at three different layers of the network seemed to perform best.

Figure 2.2 shows examples of predictions made by the spotlight network, where spotlight channels are injected in three different stages of the network, namely *conv3_1*, *conv4_1*, *conv5_1*. The first column shows the image fed into the network. The second column shows the binary spotlight channel which is injected in the above mentioned layers, indicating the region of interest. The third column shows the top-3 predictions made by the spotlight network. Notice that even though the visual input into the network is the same for all three examples, the network conditions on the spotlight channels to predict the correct object categories, which can be regarded as a proof of concept that the spotlight representation is

type of spotlight channels	layers of injection	mAP (%)
binary	conv5_1	55.9
dt	conv5_1	52.7
binary, dt	conv3_1, conv4_1, conv5_1	58.9

Table 2.1: Mean AP on ground truth regions of the validation set of MS COCO for three different variants of spotlight networks. *binary* and *dt* denote the binary and distance transform version of the spotlight channels.

used successfully by the system. If the network was to ignore the spotlight channels identical predictions would be produced for all three cases.

For the rest of our experiments, we use the spotlight network which utilizes both binary and distance transform spotlight channels injected in three intermediate layers, i.e. *conv3_1*, *conv4_1*, *conv5_1*, of the VGG16 network.

2.3.2 Tower Network

For the task of object detection, we train a tower network as shown in Figure 2.1. We use the VGG16 architecture for both streams of the network. The fc_7 feature maps from the two streams are concatenated and fed into the final fully connected layer, which outputs scores for every class.

Learning Details The spotlight network takes as input the whole image, resized to 224x224, and the corresponding spotlight channels constructed for the object proposal r . Fast R-CNN takes as input the image and the region r , which is used to pool the corresponding feature activations in $pool_5$. We initialize the main layers with weights from an ImageNet model, while the weight filters for the spotlight channels are drawn from a gaussian distribution. We use a learning rate of 0.001 and a batch size of 30. We select 25% of positive examples per batch.

2.3.2.1 Classifying Ground Truth Regions

We first evaluate our tower network on ground truth boxes to show its effectiveness in classifying highly contextual objects and compare our approach to purely region based techniques, in an effort to disentangle from the hard task of object localization.

Table 2.2 shows the average AP achieved by various networks on ground truth regions on the validation set of MS COCO. Our tower architecture classifies ground truth regions with 78% mean AP, improving over the fast R-CNN baseline by 9%. There are a few useful observations that emerge from this experiment. Noticeably, R-CNN performs quite poorly. The reason for this might be that the context padding (= 16 pixels) or the square warping







Input Image	Spotlight Channel	Predictions
		dining table 0.88 background 0.108 couch 0.008
		refrigerator 1.00 background 0.000 oven 0.000
		orange 0.38 background 0.282 apple 0.220

Figure 2.2: Activations of a Spotlight Network. The first column shows the input image. The second column shows the spotlight channel of a region in question. The channel is injected at three different layers of the network. The third column shows the top 3 predictions made by the network.

Method	Fast R-CNN [4]	R-CNN [5]	Tower Network (%)
mAP (%)	68.7	48.7	78.0

Table 2.2: Mean AP on ground truth regions of the validation set of MS COCO for fast R-CNN [4], R-CNN [5] and our tower network.

is not appropriate for the MS COCO dataset. There is a variety of parameter choices made in the original R-CNN work that should be revisited for the MS COCO dataset, since it is a dataset of drastically different statistics than the PASCAL VOC dataset. On the contrary, fast R-CNN performs significantly better than R-CNN. We believe this is due to the fact that fast R-CNN implicitly uses context through the receptive fields of the activations, from which many categories on MS COCO can benefit.

Method	Frisbee	Tie	Bas. Glove	Cup	Fork	Keyboard	Mouse	Microwave	Toaster	mAP (%)
Fast R-CNN [4]	59.1	28.0	33.1	30.3	18.7	54.5	39.8	60.5	15.4	39.2
Tower Network	63.2	30.5	39.8	38.6	21.2	57.3	46.2	65.3	20.6	40.1

Table 2.3: AP (%) for object detection on the validation set of MS COCO. We compare our tower network with fast R-CNN and report the AP for the categories that improved the most compared to fast R-CNN.

Figure 2.3 shows predictions made by R-CNN and our tower network for the same region side by side. The first column shows the image and the region in question. The second column shows the top-3 predictions made by R-CNN, while the third column from the tower. The examples showcase how objects which are hard to classify based purely on the content of the region, become easier when context is used through spotlight channels.

2.3.2.2 Object Detection on MS COCO

The task of object detection requires to accurately localize all objects in an image. We evaluate our architecture on this task and directly compare to fast R-CNN.

Learning Details. Fast R-CNN was trained with a batch size of 500, with 25% of the examples belonging to the foreground. Our tower network was trained with 30 examples per batch (out of which 25% are foreground regions).

Table 2.3 shows the results on the validation set of MS COCO. The average improvement in AP is small. This is due to the fact that many categories, such as animals and vehicles do not show an improved performance. When the performance is broken down into categories, we observe big gains on the order of 6% for *cup*, *mouse*, *toaster*, *microwave*, *baseball glove* and of the order of 3% for *tie*, *suitcase*, *fork*, *knife*, *potted plant*, *keyboard*. All of those categories have strong ties with the context that surrounds them, and thus benefit from a contextual model, as expected. We show the AP for a few categories for proof.





















Object Proposal	R-CNN predictions	Tower predictions	Object Proposal	R-CNN predictions	Tower predictions
	background 0.3 toilet 0.2 tie 0.106	tie 0.98 background 0.015 snowboard 0.000		toothbrush 0.3 background 0.236 person 0.079	toothbrush 0.27 spoon 0.149 bottle 0.125
	background 0.94 book 0.006 cup 0.005	bottle 0.79 book 0.064 background 0.056		chair 0.27 background 0.264 cup 0.226	cup 0.45 background 0.216 chair 0.135
	background 0.95 potted plant 0.015 umbrella 0.003	potted plant 0.98 background 0.016 broccoli 0.000		background 0.48 knife 0.136 scissors 0.125	knife 0.57 background 0.216 spoon 0.129
	background 0.58 potted plant 0.336 vase 0.034	vase 0.35 potted plant 0.295 background 0.253		potted plant 0.47 background 0.412 vase 0.069	potted plant 0.74 background 0.259 vase 0.000
	elephant 0.29 potted plant 0.165 background 0.132	horse 0.89 cow 0.068 elephant 0.019		sink 0.23 car 0.224 background 0.194	car 0.99 truck 0.006 background 0.001
Object Proposal	R-CNN predictions	Tower predictions	Object Proposal	R-CNN predictions	Tower predictions
	person 0.94 background 0.049 refrigerator 0.004	person 0.99 background 0.008 chair 0.001		background 0.99 bench 0.006 boat 0.002	background 0.73 sink 0.111 bench 0.072
	background 0.55 toothbrush 0.172 bottle 0.099	toothbrush 0.52 bottle 0.164 knife 0.078		vase 0.5 background 0.211 potted plant 0.123	vase 0.97 background 0.021 potted plant 0.005
	toilet 0.44 background 0.38 sink 0.078	toilet 0.58 background 0.3 sink 0.061		background 0.49 person 0.216 cup 0.038	person 0.76 background 0.214 dining table 0.016
	person 0.76 bed 0.114 teddy bear 0.047	person 0.91 bed 0.044 teddy bear 0.019		bowl 0.74 cup 0.116 background 0.047	bowl 0.91 cup 0.058 background 0.020
	background 0.57 book 0.175 truck 0.110	bus 0.6 truck 0.366 background 0.032		background 0.90 kite 0.022 umbrella 0.012	background 0.4 car 0.271 truck 0.091

Figure 2.3: Predictions made by our Tower Network compared to R-CNN. The first column shows the image and the region in question (shown with a red rectangle). The second and third column show the top-3 predictions made by R-CNN and the tower, respectively.

Chapter 3

Action and Attribute Recognition

When people interact with their environment certain common patterns appear that are very distinctive for that particular type of interaction. For example, when people *play the guitar*, the way they interact with the instrument and their pose is very characteristic for that action. Or when people are *running*, their placement in the scene, their interaction with other people in the scene and their pose and appearance are indicative of that action. This is more evident for attributes. People who *wear glasses* all share the same mode of appearance resulting from the presence of glasses. Same holds true for people who *wear shorts*, they share the same appearance in their upper leg area resulting from the presence of the same clothing item.

In order to successfully predict people’s actions or attributes, recognition engines need to capture class-characteristic modes of appearance. One way to do so is to hand-define regions of interest which are likely to contain informative cues. An example of such areas are parts, such as *head* or *legs*. Localizing areas which are relevant for the task in question should make it easier to identify patterns of appearance necessary for recognition.

In this chapter, we design *deep* part-based models for the task of action recognition and attribute classification to quantify the importance of parts in fine-grained recognition tasks when using CNNs. Subsequently, we move beyond parts and focus on building a system which identifies areas of interest in an instance-specific and class-specific manner, which we call *R*CNN*. R*CNN does not rely on hand-designed areas, such as parts, but it is able to recover them automatically if necessary. The success of R*CNN lies on Multiple-Instance Learning with CNNs. ¹

3.1 Related Work on Action and Attribute Recognition

Part-based Methods. Part-based approaches using low-level features have been successful for a variety of computer vision tasks. DPMs [36] capture different aspects of an object

¹The work presented here is based on published work in [34, 10, 35].

using mixture components and deformable parts, leading to good performance on object detection and attribute classification [37]. Similarly, poselets [38, 39, 40, 41, 6, 42] are an ensemble of models that capture parts of an object under different viewpoints and have been used for object detection, action and attribute classification and pose estimation. Pictorial structures and its variants [18, 17, 43] explicitly model parts of objects and their geometric relationship in order to accurately predict their location.

Even more recently, a number of methods incorporate HOG-based parts into deep models, showing significant improvements. Zhang *et al.* [13] use HOG-poselet activations and train CNNs, one for each poselet type, for the task of attribute classification. They show a large improvement on the task compared to HOG-based approaches. However, their approach includes a number of suboptimal choices. They use pre-trained HOG poselets to detect parts and they train a “shallow” CNN (by today’s standards) from scratch using a relatively small dataset of 25k images.

In the same vein, Branson *et al.* [44] tackle the problem of bird species categorization by first detecting bird parts with a HOG-DPM and then extracting CNN features from the aligned parts. They experimentally show the superiority of CNN-based features to hand-crafted representations. However, they work from relatively weak HOG-DPM part detections, using CNNs solely for classification purposes. Switching to the person category, HOG-DPM does not generate accurate part/keypoint predictions as shown by [41], and thus cannot be regarded as a source for well aligned body parts.

Zhang *et al.* [45] introduce part-based R-CNNs for the task of bird species classification. They discover parts of birds from region proposals and combine them for classification. They gain from using parts and also from fine-tuning a CNN for the task starting from ImageNet weights. However, region proposals are not guaranteed to produce parts. Most techniques, such as [46], are designed to generate candidate regions that contain whole objects based on bottom-up cues. While this approach works for birds, it may fail in general as parts can be defined arbitrarily in an object and need not be of distinct color and texture with regard to the rest of the object.

Action Recognition. Maji *et al.* [6] train action specific poselets and for each instance create a poselet activation vector that is classified using SVMs. They capture contextual cues in two ways: they explicitly detect objects using pre-trained models for the *bicycle*, *motorbike*, *horse* and *tvmonitor* categories and exploit knowledge of actions of other people in the image. Hoai *et al.* [8] use body-part detectors and align them with respect to the parts of a similar instance, thus aligning their feature descriptors. They combine the part based features with object detection scores and train non-linear SVMs. Khosla *et al.* [47] densely sample image regions at arbitrary locations and scales with reference to the ground-truth region. They train a random forest classifier to discriminate between different actions. Prest *et al.* [48] learn human-object interactions using only action labels. They localize the action object by finding recurring patterns on images of actions and then capture their relative spatial relations. The aforementioned approaches are based on hand-engineered features

such as HOG [49] and SIFT [50].

Turning to deep architectures, Oquab *et al.* [7] use a CNN on ground-truth boxes for the task of action classification, but observe a small gain in performance compared to previous methods. Hoai [11] uses a geometrical distribution of regions placed in the image and in the ground-truth box and weights their scores to make a single prediction, using fc_7 features from a network trained on the ImageNet-1k dataset [51].

Attribute Classification. Bourdev *et al.* [52] use HOG-based poselets to localize parts of instances in images. They train a SVM classifiers on poselet activations and more hand-crafted features (e.g. skin color) to predict appearance features of people, such as *wears hat* or *is female*. Zhang *et al.* [13] take a similar approach, using HOG-based poselets to localize people’s parts, but train CNN classifiers, one for each poselet, for the task. They combine the output of all CNNs externally by training a linear SVM classifier to make a single prediction.

3.2 Deep Part-based Action and Attribute Recognition

We develop a part-based system, leveraging convolutional network features, and apply it to attribute and action classification. Figure 3.1 gives an outline of our approach. We compute CNN features on a set of bounding boxes associated with the instance to classify. One of these bounding boxes corresponds to the whole instance and is either provided by an oracle or comes from a person detector. The other bounding boxes (three in our implementation) come from poselet-like part detectors.

Our part detectors are a novel “deep” version of poselets. We define three human body parts (head, torso, and legs) and cluster the keypoints of each part into several distinct poselets. Traditional poselets [38, 39] would then operate as sliding-window detectors on top of low-level gradient orientation features, such as HOG [49]. Instead, we train a sliding-window detector for each poselet on top of a deep feature pyramid, using the implementation of [53]. Unlike HOG-based poselets, our parts are capable of firing on difficult to detect structures, such as sitting versus standing legs. Also, unlike recent deep parts based on bottom-up regions [45], our sliding-window parts can span useful, but inhomogeneous regions, that are unlikely to group together through a bottom-up process (e.g. bare arms and a t-shirt).

Another important aspect of our approach is task-specific CNN fine-tuning. We show that a fine-tuned holistic model (i.e. no parts) is capable of matching the attribute classification performance of the part-based PANDA system [13]. Then, when we add parts our system outperforms PANDA. This result indicates that PANDA’s dramatic improvement from parts comes primarily from the weak holistic classifier baseline used in their work, rather than from the parts themselves. While we also observe an improvement from adding parts, our marginal

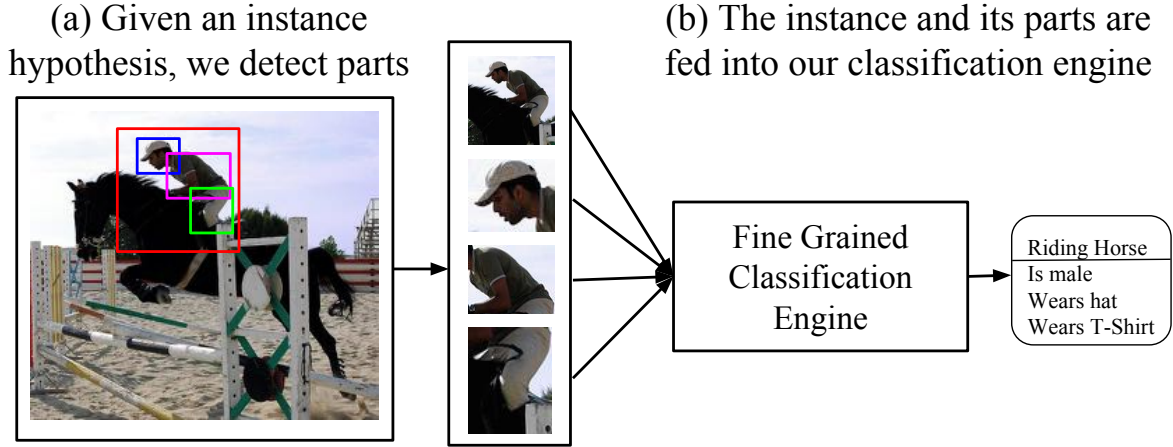


Figure 3.1: Schematic overview of our overall approach. (a) Given an R-CNN person detection (red box), we detect parts using a novel, deep version of poselets. (b) The detected whole-person and part bounding boxes are input into a fine-grained classification engine to produce predictions for actions and attributes.

gain over the holistic model is smaller, and the gain becomes even smaller as our network becomes deeper. This observation suggests a possible trend: as more powerful convolutional network architectures are engineered, the marginal gain from explicit parts may vanish.

As a final contribution, we show that our system can operate “without training wheels.” In the standard evaluation protocol for benchmarking attributes and actions [52, 54], an oracle provides a perfect bounding box for each test instance. While this was a reasonable “cheat” a couple of years ago, it is worth revisiting. Due to recent substantial advances in detection performance, we believe it is time to drop the oracle bounding box at test time. We show, for the first time, experiments doing just this; we replace ground-truth bounding boxes with person detections from a state-of-the-art R-CNN person detector [5]. Doing so only results in a modest drop in performance compared to the traditional oracle setting.

3.2.1 Deep Part Detectors

Figure 3.2 schematically outlines the design of our deep part detectors, which can be viewed as a multi-scale fully convolutional network. The first stage produces a feature pyramid by convolving the levels of the gaussian pyramid of the input image with a 5-layer CNN, similar to Girshick *et al.* [53] for training DeepPyramid DPMs. The second stage outputs a pyramid of part scores by convolving the feature pyramid with the part models.

3.2.1.1 Feature pyramid

Feature pyramids allow for object and part detections at multiple scales while the corresponding models are designed at a single scale. This is one of the oldest “tricks” in computer vision

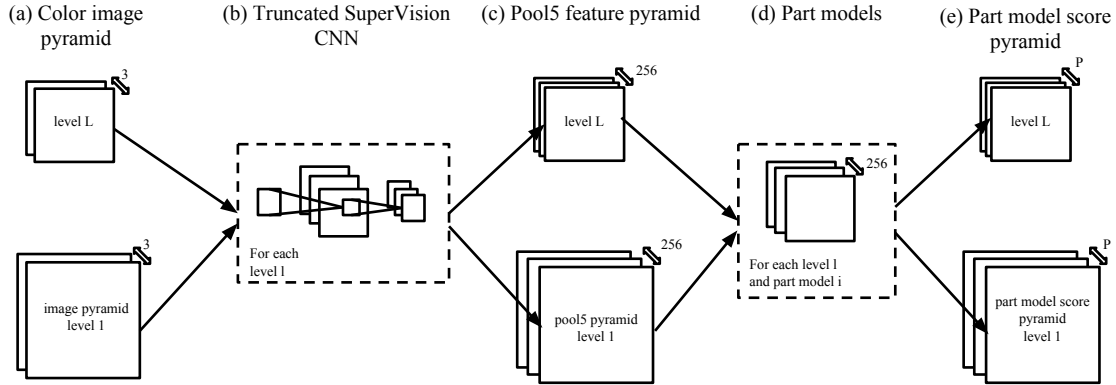


Figure 3.2: Schematic overview of our part detectors. (a) A gaussian pyramid is build from an input image. (b) Each level of the pyramid is fed into a truncated SuperVision CNN. (c) The output is a pyramid of pool5 feature maps. (d) Each level of the feature pyramid is convolved with the part models. (e) The output is a pyramid of part model scores

and has been implemented by sliding-window object detection approaches throughout the years [38, 36, 28, 55].

Given an input image, the construction of the feature pyramid starts by creating the gaussian pyramid for the image for a fixed number of scales and subsequently extracting features from each scale. For feature extraction, we use a CNN and more precisely, we use a variant of the single-scale network proposed by Krizhevsky *et al.* [20]. More details can be found in [53]. Their software is publicly available and we build on their implementation.

3.2.1.2 Designing Part models

We design models to capture parts of the human body under a particular viewpoint and pose. Ideally, part models should be (a) pose-sensitive, i.e. produce strong activations on examples of similar pose and viewpoint, (b) inclusive, i.e. cover all the examples in the training set, and (c) discriminative, i.e. score higher on the object than on the background. To achieve all the above properties, we build part models by clustering the keypoint configurations of all the examples in the training set and train linear SVMs on pool5 features with hard negative mining.

We model the human body with three high-level parts: the head, the torso and the legs. Even though the pose of the parts is tied with the global pose of the person, each one has its own degrees of freedom. In addition, there is a large, yet not infinite due to the kinematic constraints of the human body, number of possible part combinations that cover the space of possible human poses.

We design parts defined by the three body areas, head (H), torso (T) and legs (L). Assume $t \in \{H, T, L\}$ and $K_t^{(i)}$ the set of 2D keypoints of the i -th training example corresponding to part t . The keypoints correspond to predefined landmarks of the human

body. Specifically, $K_H = \{Eyes, Nose, Shoulders\}$, $K_T = \{Shoulders, Hips\}$ and for $K_L = \{Hips, Knees, Ankles\}$.

For each t , we cluster the set of $K_t^{(i)}, i = 1, \dots, N$, where N is the size of the training set. The output is a set of clusters $C_t = \{c_j\}_{j=1}^{P_t}$, where P_t is the number of clusters for t , and correspond to distinct part configurations

$$C_t = cluster\left(\{K_t^{(i)}\}_{i=1}^N\right). \quad (3.1)$$

We use a greedy clustering algorithm, similar to [40]. Examples are processed in a random order. An example is added to an existing cluster if its distance to the center is less than ϵ , otherwise it starts a new cluster. The distance of two examples is defined as the euclidean distance of their normalized keypoint distributions. For each cluster $c \in C_t$, we collect the M closest cluster members to its center. Those form the set of positive examples that represent the cluster. From now on, we describe a part by its body part type t and its cluster index j , with $c_j \in C_t$, while $S_{t,j}$ represents the set of positive examples for part (t, j) .

Figure 3.3 (left) shows examples of clusters as produced by our clustering algorithm with $\epsilon = 1$ and $M = 100$. We show 4 examples for each cluster example. We use the PASCAL VOC 2012 train set, along with keypoint annotations as provided by [38], to design and train the part detectors. In total we obtain 30 parts, 13 for head, 11 for torso and 6 for legs.

3.2.1.3 Learning Part Models

For each part (t, j) , we define the part model to be the vector of weights $\mathbf{w}_{t,j}$ which when convolved with a feature pyramid gives stronger activations near the ground-truth location and scale (right most part of Figure 3.2).

One could view the whole pipeline shown in Figure 3.2 as a fully convolutional model and thus one could train it end-to-end, optimizing the weights of the CNN for the pool5 feature extraction and the weights of the part models jointly. We choose to simplify the problem by decoupling it. We use the publicly available ImageNet weights of the CNN [5] to extract pool5 feature pyramids. Subsequently, we train linear SVMs for the part models. For each part (t, j) we train a linear SVM with positives from $S_{t,j}$ to obtain model weights $\mathbf{w}_{t,j} \in \mathbb{R}^{8 \times 8 \times 256}$. We use hard negative mining from images of no people to train the model.

Figure 3.3 (right) shows the top few detections of a subset of parts on PASCAL VOC val 2009 set. Each row shows activations of a different part, which is displayed at the left side of the same row.

Evaluation of part models. We quantify the performance of our part detectors by computing the average precision (AP) - similar to object detection PASCAL VOC - on val 2009. For every image, we detect part activations at all scales and locations which we non-maximum suppress with a threshold of 0.3 across all parts of the same type. Since there are available keypoint annotations on the val set, we are able to construct ground-truth part boxes. A detection is marked as positive if the intersection-over-union with a ground-truth part box is

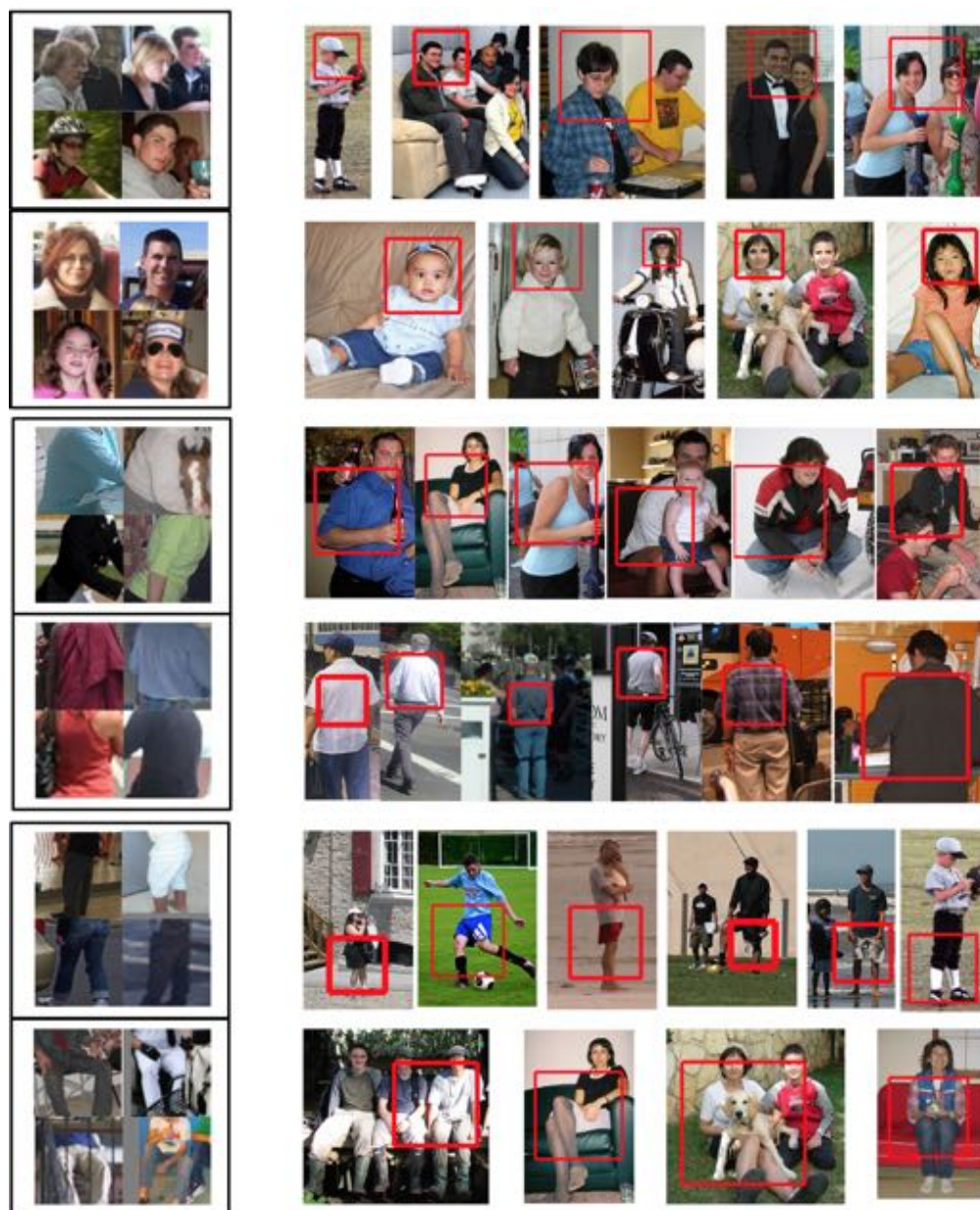


Figure 3.3: Examples of clusters for the three body areas, head, torso and legs (left) and their top few detections on PASCAL VOC val 2009 (right). The first two rows correspond to cluster examples for head, the following two for torso and the last two for legs.

AP (%)	$\sigma = 0.2$	$\sigma = 0.3$	$\sigma = 0.4$	$\sigma = 0.5$
Head	55.2	51.8	45.2	31.6
Torso	42.1	36.3	23.6	9.4
Legs	34.9	27.9	20.0	10.4

Table 3.1: AP for each part type on PASCAL VOC val 2009. We evaluate the part activations and measure AP for different thresholds of intersection-over-union.

more than σ . In PASCAL VOC, σ is set to 0.5. However, this threshold is rather strict for small objects, such as our parts. We report AP for various values of σ for a fair assessment of the quality of our parts. Table 3.1 shows the results.

Mapping parts to instances. Since our part models operate independently, we need to group part activations and link them to an instance in question. Given a candidate region box in an image I , for each part t we keep the highest scoring part within box

$$j^* = \arg \max_j \max_{(x,y) \in box} \mathbf{w}_{t,j} * F_{(x,y)}(I), \quad (3.2)$$

where $F_{(x,y)}(I)$ is the point in feature pyramid for I corresponding to the image coordinates (x, y) . This results in three parts being associated with each box , as shown in Figure 3.1. A part is considered absent if the score of the part activation is below a threshold, here the threshold is set to -0.1 .

In the case when an oracle gives ground-truth bounding boxes at test time, one can refine the search of parts even further. If box is the oracle box in question, we retrieve the k nearest neighbor instances $i = \{i_1, \dots, i_k\}$ from the training set based on the L_2 -norm of their pool5 feature maps $F(\cdot)$, i.e. $\frac{F(box)^T F(box_{i_j})}{\|F(box)\| \cdot \|F(box_{i_j})\|}$. If K_{i_j} are the keypoints for the nearest examples, we consider the average keypoint locations $K_{box} = \frac{1}{K} \sum_{j=1}^k K_{i_j}$ to be an estimate of the keypoints for the test instance box . Based on K_{box} we can reduce the regions of interest for each part within box by only searching for them in the corresponding estimates of the body parts.

3.2.2 Experiments

In this section we investigate the role of parts for fine-grained classification tasks. We focus on the tasks of action classification (e.g. running, reading, etc.) and attribute classification (e.g. male, wears hat, etc.). Figure 3.4 schematically outlines our approach at test time. We start with the part activations mapped to an instance and forward propagate the corresponding part and instance boxes through a CNN. The output is a fc_7 feature vector for each part

as well as the whole instance. We concatenate the feature vectors and classify the example with a linear SVM, which predicts the confidence for each class (action or attribute).

3.2.2.1 System Variations

For each task, we consider four variants of our approach in order to understand which design factors are important.

No parts. This approach is our baseline and does not use part detectors. Instead, each instance is classified according to the fc_7 feature vector computed from the instance bounding box. The CNN used for this system is fine-tuned from an ImageNet initialization, as in [5], on jittered instance bounding boxes.

Instance fine-tuning. This method uses our part detectors. Each instance is classified based on concatenated fc_7 feature vectors from the instance and all three parts. The CNN used for this system is fine-tuned on instances, just as in the “no parts” system. We note that because some instances are occluded, and due to jittering, training samples may resemble parts, though typically only the head and torso (since occlusion tends to happen from the torso down).

Joint fine-tuning. This method also uses our part detectors and concatenated fc_7 feature vectors. However, unlike the previous two methods we fine-tune the CNN jointly using instance and part boxes from each training sample. During fine-tuning the network can be seen as a four-stream CNN, with one stream for each bounding box. Importantly, we tie weights between the streams so that the number of CNN parameters is the same in all system variants. This design explicitly forces the CNN to see each part box during fine-tuning.

3-way split. To test the importance of our part detectors, we employ a baseline that vertically splits the instance bounding box into three (top, middle, and bottom) in order to simulate crude part detectors. This variation uses a CNN fine-tuned on instances.

3.2.2.2 Action Classification

We focus on the problem of action classification as defined by the PASCAL VOC action challenge. The task involves predicting actions from a set of predefined action categories.

Learning details. We train all networks with backpropagation using Caffe [56], starting from the ImageNet weights, similar to the fine-tuning procedure introduced in [5]. A small learning rate of 10^{-5} and a dropout ratio of 50% were used. During training, and at test time, if a part is absent from an instance then we use a box filled with the ImageNet mean image values (i.e. all zeros after mean subtraction). Subsequently, we train linear SVMs, one for each action, on the concatenated fc_7 feature vectors.

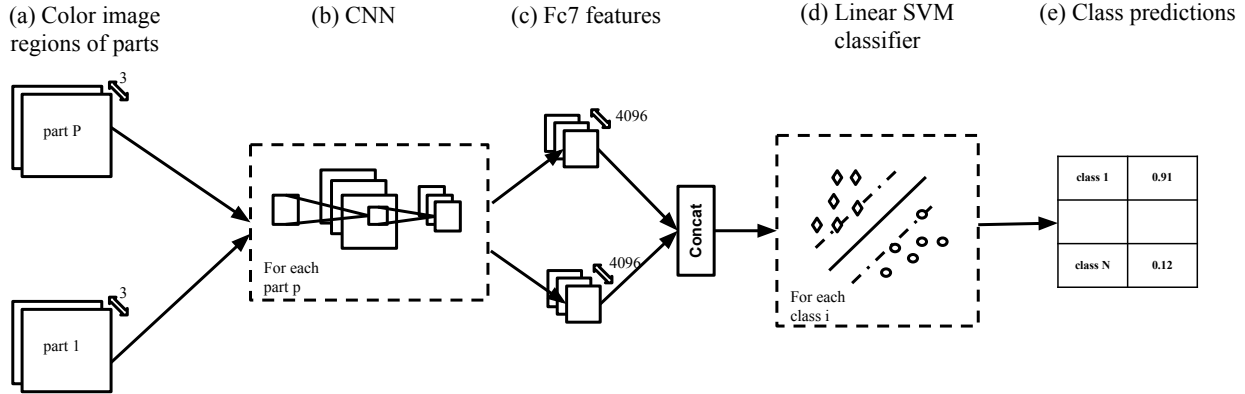


Figure 3.4: Schematic overview of our approach for fine grained classification using parts. (a) We consider regions of part activations. (b) Each part is forward propagated through a CNN. (c) The output is the fc7 feature vector for each input. (d) The features are concatenated and fed into linear SVM classifiers. (e) The classifiers produce scores for each class.

Context. In order to make the most of the context in the image, we rescore our predictions by using the output of R-CNN [5] for the 20 PASCAL VOC object categories and the presence of other people performing actions. We train a linear SVM on the action score of the test instance, the maximum scores of other instances (if any) and the object scores, to obtain a final prediction. Context rescoring is used for all system variations on the *test* set.

Results. Table 3.2 shows the result of our approach on the PASCAL VOC 2012 *test* set. These results are in the standard setting, where an oracle gives ground-truth person bounds at test time. We conduct experiments using two different network architectures: a 8-layer CNN as defined in [20], and a 16-layer as defined in [9]. *Ours (no parts)* is the baseline approach, with no parts. *Ours* is our full approach when we include the parts. For the 8-layer network, we use the CNN trained on instances, while for the 16-layer network we use the CNN trained jointly on instances and their parts based on results on the *val* set (Table 3.3). For our final system, we also present results when we add features extracted from the whole image, using a 16-layer network trained on ImageNet-1k (*Ours (w/ image features)*). We show results as reported by action poselets [6], a part-based approach, using action specific poselets with HOG features, Oquab *et al.* [7], Hoai [8] and Simonyan and Zisserman [9], three CNN-based approaches on the task. The best performing method by [9] uses a 16- and 19-layer network. Their 16-layer network is equivalent to *Ours (no parts)* with 16 layers, thus the additional boost in performance comes from the 19-layer network. This is not surprising, since deeper networks perform better, as is also evident from our experiments. From the comparison with the baseline, we conclude that parts improve the performance. For the 8-layer CNN, parts contribute 3% of mAP, with the biggest improvement coming from *Phoning*, *Reading* and *Taking Photo*. For the 16-layer CNN, the improvement from

AP (%)	CNN layers	Jumping	Phoning	Playing Instrument	Reading	Riding Bike	Riding Horse	Running	Taking Photo	Using Computer	Walking	mAP
Action Poselets [6]	-	59.3	32.4	45.4	27.5	84.5	88.3	77.2	31.2	47.4	58.2	55.1
Oquab <i>et al.</i> [7]	8	74.8	46.0	75.6	45.3	93.5	95.0	86.5	49.3	66.7	69.5	70.2
Hoai [8]	8	82.3	52.9	84.3	53.6	95.6	96.1	89.7	60.4	76.0	72.9	76.3
Simonyan & Zisserman [9]	16 & 19	89.3	71.3	94.7	71.3	97.1	98.2	90.2	73.3	88.5	66.4	84.0
Ours (no parts)	8	76.2	47.4	77.5	42.2	94.9	94.3	87.0	52.9	66.5	66.5	70.5
Ours	8	77.9	54.5	79.8	48.9	95.3	95.0	86.9	61.0	68.9	67.3	73.6
Ours (no parts)	16	84.7	62.5	86.6	59.0	95.9	96.1	88.7	69.5	77.2	70.2	79.0
Ours	16	83.7	63.3	87.8	64.2	96.0	96.7	88.9	75.2	80.0	71.5	80.7
Ours (w/ image features)	16	84.7	67.8	91.0	66.6	96.6	97.2	90.2	76.0	83.4	71.6	82.6

Table 3.2: AP on the PASCAL VOC 2012 Actions test set. The first three rows show results of two other methods. Action Poselets [6] is a part based approach using HOG features, while Oquab *et al.* [7], Hoai [8] and Simonyan & Zisserman [9] are CNN based approaches. *Ours (no parts)* is the baseline approach of our method, when only the ground truth box is considered, while *Ours* is the full approach, including parts. All approaches use ground truth boxes at test time.

parts is smaller, 1.7 % of mAP, and the actions benefited the most are *Reading*, *Taking Photo* and *Using Computer*. The image features capture cues from the scene and give an additional boost to our final performance.

Table 3.3 shows results on the PASCAL VOC action *val* set for a variety of different implementations of our approach. *Ours (no parts)* is the baseline approach, with no parts, while *Ours (3-way split)* uses as parts the three horizontal splits comprising the instance box. *Ours (joint fine-tuning)* shows the results when using a CNN fine-tuned jointly on instances and parts, while *Ours (instance fine-tuning)* shows our approach when using a CNN fine-tuned on instances only. We note that all variations that use parts significantly outperform the *no-parts* system.

We also show results of our best system when ground-truth information is *not* available at test time *Ours (R-CNN bbox)*. In place of oracle boxes we use R-CNN detections for person. For evaluation purposes, we associate a R-CNN detection to a ground-truth instance as following: we pick the highest scoring detection for person that overlaps more than 0.5 with the ground truth. Another option would be to define object categories as “person+action” and then just follow the standard detection AP protocol. However, this is not possible because not all people are marked in the dataset (this is true for the attribute dataset as well). We report numbers on the val action dataset. We observe a drop in performance, as expected due to the imperfect person detector, but our method still works reasonably well under those circumstances. Figure 3.5 shows the top few predictions on the test set. Each block corresponds to a different action.

3.2.2.3 Attribute Classification

We focus on the problem of attribute classification, as defined by [52]. There are 9 different categories of attributes, such as *Is Male*, *Has Long Hair*, and the task involves predicting attributes, given the location of the people. Our approach is shown in Figure 3.4. We use the Berkeley Attributes of People Dataset as proposed by [52].

AP (%)	CNN layers	Jumping	Phoning	Playing Instrument	Reading	Riding Bike	Riding Horse	Running	Taking Photo	Using Computer	Walking	mAP
Ours (no parts)	8	76.5	44.7	75.0	43.3	90.0	91.6	79.2	53.5	66.5	61.4	68.2
Ours (3-way split)	8	79.0	44.4	77.8	46.9	91.5	93.1	83.1	59.3	67.3	64.4	70.7
Ours (instance fine-tuning)	8	77.2	48.4	79.0	49.5	92.4	93.8	80.9	60.4	68.9	64.0	71.5
Ours (joint fine-tuning)	8	75.2	49.5	79.5	50.2	93.7	93.6	81.5	58.6	64.6	63.6	71.0
Ours (no parts)	16	85.4	58.6	84.6	60.9	94.4	96.6	86.6	68.7	74.9	67.3	77.8
Ours (instance fine-tuning)	16	85.1	60.2	86.6	63.1	95.6	97.4	86.4	71.0	77.6	68.3	79.1
Ours (joint fine-tuning)	16	84.5	61.2	88.4	66.7	96.1	98.3	85.7	74.7	79.5	69.1	80.4
Ours (R-CNN bbox)	8	67.8	46.6	76.9	47.3	85.9	81.4	71.5	53.1	61.2	53.9	64.6
Ours (R-CNN box)	16	79.4	63.3	86.1	64.4	93.2	91.9	80.2	71.2	77.4	63.4	77.0

Table 3.3: AP on the PASCAL VOC 2012 Actions val set of our approach. *Ours (no parts)* is our approach without parts. *Ours (3-way split)* is our approach when parts are defined as the three horizontal splits comprising an instance box. *Ours (joint fine-tuning)* uses a CNN fine-tuned jointly on the instances and the parts, while *Ours (instance fine-tuning)* uses a single CNN fine-tuned just on the instance box. All the above variations of our approach use ground truth information at test time as the object bound. *Ours (R-CNN bbox)* uses R-CNN detections for person.

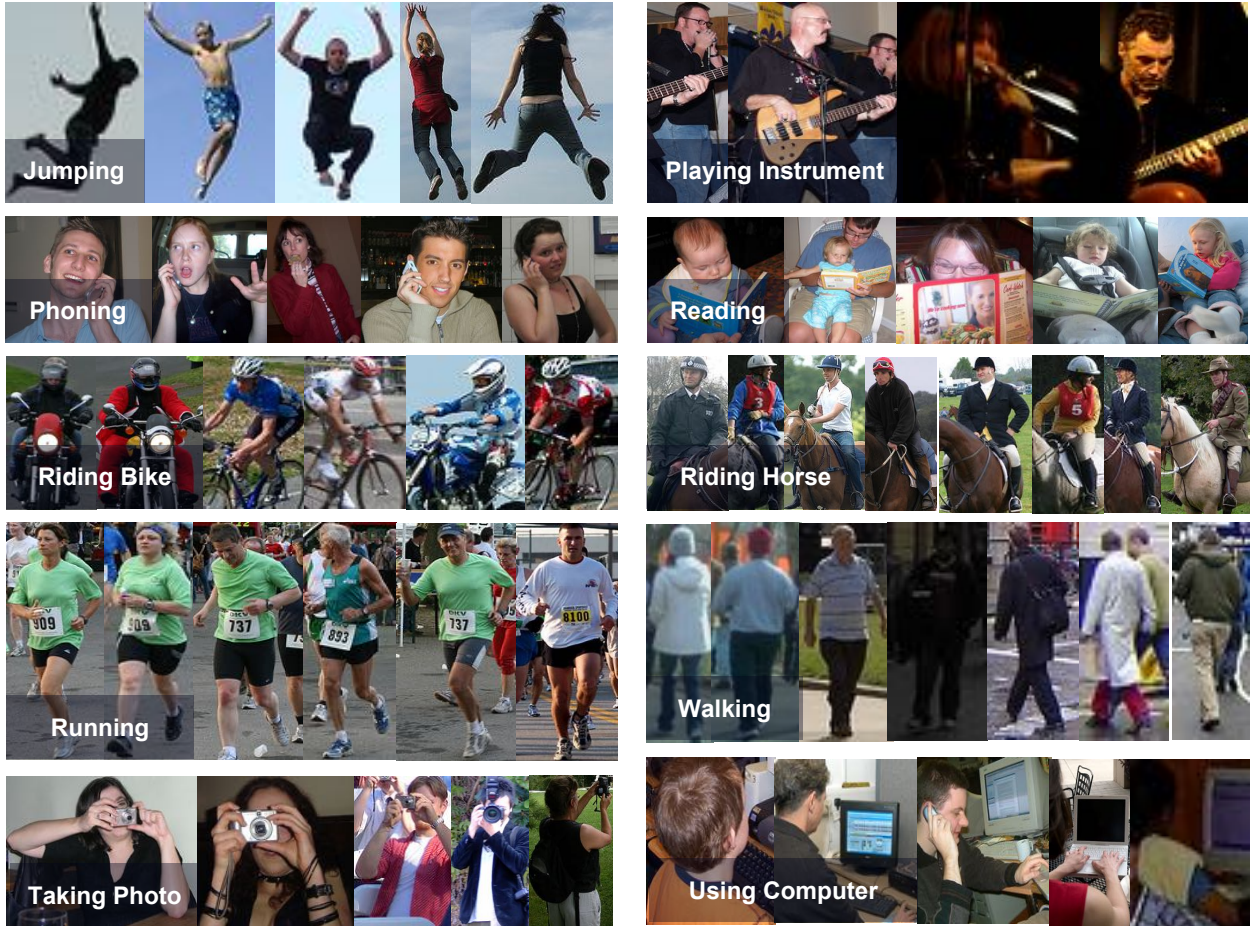


Figure 3.5: Top action predictions on the test set. Different blocks correspond to different actions.

AP (%)	CNN layers	Is Male	Has Long Hair	Has Glasses	Has Hat	Has T-Shirt	Has Long Sleeves	Has Shorts	Has Jeans	Has Long Pants	mAP
PANDA [13]	5	91.7	82.7	70.0	74.2	49.8	86.0	79.1	81.0	96.4	79.0
Ours (no parts)	8	87.5	80.4	43.3	77.0	61.5	86.4	88.5	88.7	98.2	79.1
Ours (3-way split)	8	89.3	82.2	51.2	84.0	60.1	87.4	88.3	89.2	98.2	81.1
Ours (instance fine-tuning)	8	89.9	83.5	60.5	85.2	64.3	89.0	88.6	89.1	98.2	83.1
Ours (joint fine-tuning)	8	91.7	86.3	72.5	89.9	69.0	90.1	88.5	88.3	98.1	86.0
Ours (no parts)	16	93.4	88.7	72.5	91.9	72.1	94.1	92.3	91.9	98.8	88.4
Ours (instance fine-tuning)	16	93.8	89.8	76.2	92.9	73.3	94.4	92.3	91.8	98.7	89.3
Ours (joint fine-tuning)	16	92.9	90.1	77.7	93.6	72.6	93.2	93.9	92.1	98.8	89.5
Ours (R-CNN bbox)	8	84.1	77.9	62.7	84.5	66.8	84.7	80.7	79.2	91.9	79.2
Ours (R-CNN bbox)	16	90.1	85.2	70.2	89.8	63.2	89.7	83.4	84.8	96.3	83.6

Table 3.4: AP on the test set of the Berkeley Attributes of People Dataset. All approaches on the top use ground truth boxes for evaluation. *Ours (no parts)* is the baseline approach with no parts. *Ours (3-way split)* is a variant of our approach, where parts are defined as the three horizontal splits comprising an instance box. *Ours (instance fine-tuning)* uses a CNN fine-tuned on instance boxes, while *Ours (joint fine-tuning)* uses a CNN fine-tuned jointly on instances and parts. We also show the effectiveness of our approach *Ours (R-CNN bbox)*, when no ground truth boxes are given at test time.

Learning details. Similar to the task of action classification, we separately learn the parameters of the CNN and the linear SVM. Again, we fine-tune a CNN for the task in question with the difference that the softmax layer is replaced by a cross entropy layer (sum of logistic regressions).

Results. Table 3.4 shows AP on the *test* set. We show results of our approach with and without parts, as well as results as reported by Zhang *et al.* [13], the state-of-the-art on the task, on the same test set. With an 8-layer network, parts improve the performance of all categories, indicating their impact on attribute classification. Also, a network jointly fine-tuned on instances and parts seems to work significantly better than a CNN trained solely on the instance boxes. In the case of a 16-layer network, joint fine-tuning and instance fine-tuning seem to work equally well. The gain in performance from adding parts is less significant in this case. This might be because of the already high performance achieved by the holistic network. Interestingly, our 8-layer holistic approach matches the current state-of-the-art on this task, PANDA [13] showcasing the importance of deeper networks and good initialization.

Table 3.4 also shows the effectiveness of our best model, namely the jointly fine-tuned 16-layer CNN, when we use R-CNN detections instead of ground truth boxes on the Berkeley Attributes of People test set. Figure 3.7 shows the top few predictions on the test set. Each block corresponds to a different attribute. Figure 3.6 shows top errors for two of our lowest performing attribute classes.



Figure 3.6: Top errors of classification for two of the attribute categories, *Has Glasses* (top) and *Has T-Shirt* (bottom).



Figure 3.7: Top attribute predictions on the test set. Each block corresponds to a different attribute

3.3 R*CNN

Part-based models predefine the regions of interest a model attends to. However, when people perform actions there is a variety of cues, spanning outside of the appearance of the human body, that are informative about the action. For example, they might include the interaction with objects, or with other subjects or even with the scene. A visual recognition system should be able to attend to the cues necessary to make a prediction, by learning to mine those informative cues in an instance-specific and class-specific manner.

In order to achieve this, we define a model that attends to two regions: one is the whole instance, the person that is to be classified, and the other one is the auxiliary cue, which the model selects automatically from the image to assist with the prediction. Formally, we adapt the Region-based Convolutional Network method (RCNN) [5] to use more than one region when making a prediction. We call our method *R*CNN*.

3.3.1 Architecture

In R*CNN, we have a *primary* region that contains the person in question and a *secondary* region that automatically discovers contextual cues. How do we select the secondary region? In other words, how do we decide which region contains information about the action being performed? Inspired by multiple-instance learning (MIL) [57, 58] and Latent SVM [36], if I is an image and r is a region in I containing the target person, we define the score of action α as

$$\text{score}(\alpha; I, r) = \mathbf{w}_P^\alpha \cdot \phi(r; I) + \max_{s \in R(r; I)} \mathbf{w}_S^\alpha \cdot \phi(s; I), \quad (3.3)$$

where $\phi(r; I)$ is a vector of features extracted from region r in I , while \mathbf{w}_P^α and \mathbf{w}_S^α are the primary and secondary weights for action α respectively. $R(r; I)$ defines the set of candidates for the secondary region. For example, $R(r; I)$ could be the set of regions in the proximity of r , or even the whole set of regions in I . Given scores for each action, we use a softmax to compute the probability that the person in r is performing action α :

$$P(\alpha|I, r) = \frac{\exp(\text{score}(\alpha; I, r))}{\sum_{\alpha' \in A} \exp(\text{score}(\alpha'; I, r))}. \quad (3.4)$$

The feature representation $\phi(\cdot)$ and the weight vectors \mathbf{w}_P^α and \mathbf{w}_S^α in Equation 3.3 are learned *jointly* for all actions $\alpha \in A$ using a CNN trained with stochastic gradient descent (SGD). We build on the Fast RCNN implementation [4], which efficiently processes a large number of regions per image. Figure 3.8 shows the architecture of our network.

Given an image I , we select the primary region to be the bounding box containing the person (knowledge of this box is given at test time in all datasets). Bottom up region proposals form the set of candidate secondary regions. For each action α , the most informative region is selected through the *max* operation and its score is added to the primary (Equa-

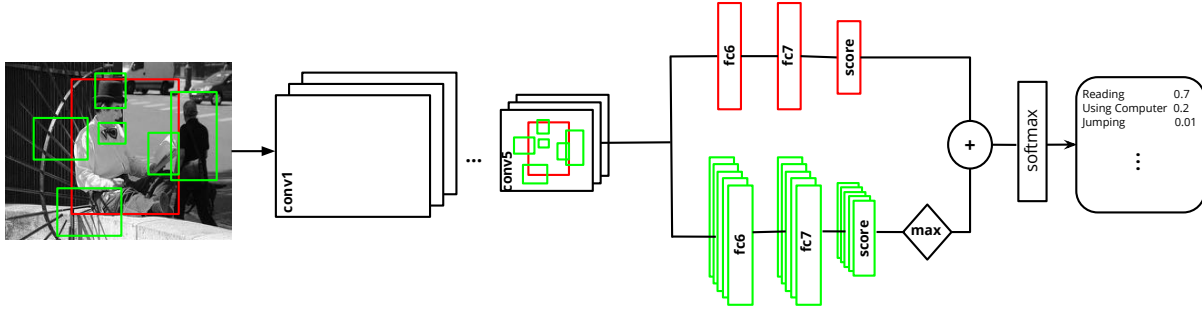


Figure 3.8: Schematic overview of our approach. Given image I , we select the primary region to be the bounding box containing the person (red box) while region proposals define the set of candidate secondary regions (green boxes). For each action α , the most informative secondary region is selected (\max operation) and its score is added to the primary. The *softmax* operation transforms scores into probabilities and forms the final prediction.

tion 3.3). The *softmax* operation transforms scores into estimated posterior probabilities (Equation 3.4), which are used to predict action labels.

We build on Fast RCNN (FRCN) [4]. In FRCN, the input image is up-sampled and passed through the convolutional layers. An adaptive max pooling layer takes as input the output of the last convolutional layer and a list of regions of interest (ROIs). It outputs a feature map of fixed size (e.g. 7×7 for the 16-layer CNN by [9]) specific to each ROI. The ROI-pooled features are subsequently passed through the fully connected layers to make the final prediction. This implementation is efficient, since the computationally intense convolutions are performed at an image-level and are subsequently being reused by the ROI-specific operations.

The test-time operation of FRCN is similar to SPPnet [29]. However, the training algorithm is different and enables fine-tuning all network layers, not just those above the final ROI pooling layer, as in [29]. This property is important for maximum classification accuracy with very deep networks.

In our implementation, we extend the FRCN pipeline. Each primary region r of an image I predicts a score for each action $\alpha \in A$ (top stream in Figure 3.8). At the same time, each region within the set of candidate secondary regions $R(r; I)$ independently makes a prediction. These scores are combined, for each primary region r , by a \max operation over r 's candidate regions (bottom stream in Figure 3.8).

We define the set of candidate secondary regions $R(r; I)$ as

$$R(r; I) = \{s \in S(I) : \text{overlap}(s, r) \in [l, u]\}, \quad (3.5)$$

where $S(I)$ is the set of region proposals for image I . In our experiments, we use Selective

Search [46]. The lower and upper bounds for the overlap, which here is defined as the intersection over union between the boxes, defines the set of the regions that are considered as secondary for each primary region. For example, if $l = 0$ and $u = 1$ then $R(r; I) = S(I)$, for each r , meaning that all bottom up proposals are candidates for secondary regions.

3.3.2 Learning

We train R*CNN with stochastic gradient descent (SGD) using backpropagation. We adopt the 16-layer network architecture from [12], which has been shown to perform well for image classification and object detection.

During training, we minimize the log loss of the predictions. If $P(\alpha | I, r)$ is the softmax probability that action α is performed in region r in image I computed by Equation 3.4, then the loss over a batch of training examples $B = \{I_i, r_i, l_i\}_{i=1}^M$ is given by

$$\text{loss}(B) = -\frac{1}{M} \sum_{i=1}^M \log P(\alpha = l_i | I_i, r_i), \quad (3.6)$$

where l_i is the true label of example r_i in image I_i .

Rather than limiting training to the ground-truth person locations, we use all regions that overlap more than 0.5 with a ground-truth box. This condition serves as a form of data augmentation. For every primary region, we randomly select N regions from the set of candidate secondary regions. N is a function of the GPU memory limit (we use a Nvidia K40 GPU) and the batch size.

We fine-tune our network starting with a model trained on ImageNet-1K for the image classification task. We tie the weights of the fully connected primary and secondary layers (fc_6, fc_7), but not for the final scoring models. We set the learning rate to 0.0001, the batch size to 30 and consider 2 images per batch. We pick $N = 10$ and train for 10K iterations. Larger learning rates prevented fine-tuning from converging.

Due to the architecture of our network, most computation time is spent during the initial convolutions, which happen over the whole image. Computation does not scale much with the number of boxes, contrary to the original implementation of RCNN [5]. Training takes 1s per iteration, while testing takes 0.4s per image.

3.3.3 Experiments

We demonstrate the effectiveness of R*CNN on action recognition from static images on the PASCAL VOC Actions dataset [54], the MPII Human Pose dataset [59] and the Stanford 40 Actions dataset [60].

3.3.3.1 Control Experiments

We experiment with variants of our system to show the effectiveness of R*CNN on the PASCAL VOC Action dataset. The PASCAL VOC Action dataset consists of 10 different

actions, *Jumping*, *Phoning*, *Playing Instrument*, *Reading*, *Riding Bike*, *Riding Horse*, *Running*, *Taking Photo*, *Using Computer*, *Walking* as well as examples of people not performing some of the above action, which are marked as *Other*. The ground-truth boxes containing the people are provided both at train and test time. During test time, for every example we estimate probabilities for all actions and compute AP.

The variants, which are baselines for our final system R*CNN, are as follows:

- **RCNN**. As a baseline approach we train Fast R-CNN for the task of action classification. This network exploits only the information provided from the primary region, which is defined as the ground-truth region.
- **Random-RCNN**. We use the ground-truth box as a primary region and a box randomly selected from the secondary regions. We train a network for this task similar to R*CNN with the *max* operation replaced by *rand*
- **Scene-RCNN**. We use the ground-truth box as the primary region and the whole image as the secondary. We jointly train a network for this task, similar to R*CNN, where the secondary model learns action specific weights solely from the scene (no *max* operation is performed in this case)
- **R*CNN** (l, u). We experiment with various combinations of values for the only free parameters of our pipeline, namely the bounds (l, u) of the overlaps used when defining the secondary regions $R(r; I)$, where r is the primary region
- **R*CNN** (l, u, n_S). In this setting, we use $n_S > 1$ secondary regions instead of one. The secondary regions are selected in a greedy manner. First we select the secondary region s_1 exactly as in R*CNN. The i -th secondary region s_i is selected via the *max* operation from the set $R(r; I) \cap R(s_1; I) \cap \dots \cap R(s_{i-1}; I)$, where r is the primary region.

The Random- and Scene- settings show the value of selecting the most informative region, rather than forcing the secondary region to be the scene or a region selected at random, respectively

Table 3.5 shows the performance of all the variants on the val set of the PASCAL VOC Actions. Our experiments show that R*CNN performs better across all categories. In particular, *Phoning*, *Reading*, *Taking Photo* perform significantly better than the baseline approach and Scene-RCNN. *Riding Bike*, *Riding Horse* and *Running* show the smallest improvement, probably due to scene bias of the images containing those actions. Another interesting observation is that our approach is not sensitive to the bounds of overlap (l, u). R*CNN is able to perform very well even for the unconstrained setting where all regions are allowed to be picked by the secondary model, ($l = 0, u = 1$). In our basic R*CNN setting, we use one secondary region. However, one region might not be able to capture all the modes of contextual cues present in the image. Therefore, we extend R*CNN to include n_S secondary regions. Our experiments show that for $n_S = 2$ the performance is the same as with R*CNN for the optimal set of parameters of ($l = 0.2, u = 0.75$).

AP (%)	Jumping	Phoning	Playing Instrument	Reading	Riding Bike	Riding Horse	Running	Taking Photo	Using Computer	Walking	mAP
RCNN	88.7	72.6	92.6	74.0	96.1	96.9	86.1	83.3	87.0	71.5	84.9
Random-RCNN	89.1	72.7	92.9	74.4	96.1	97.2	85.0	84.2	87.5	70.4	85.0
Scene-RCNN	88.9	72.5	93.4	75.0	95.6	98.1	88.6	83.2	90.4	71.5	85.7
R*CNN (0.0, 0.5)	89.1	80.0	95.6	81.0	97.3	98.7	85.5	85.6	93.4	71.5	87.8
R*CNN (0.2, 0.5)	88.1	75.4	94.2	80.1	95.9	97.9	85.6	84.5	92.3	71.6	86.6
R*CNN (0.0, 1.0)	89.2	77.2	94.9	83.7	96.7	98.6	87.0	84.8	93.6	70.1	87.6
R*CNN (0.2, 0.75)	88.9	79.9	95.1	82.2	96.1	97.8	87.9	85.3	94.0	71.5	87.9
R*CNN (0.2, 0.75, 2)	87.7	80.1	94.8	81.1	95.5	97.2	87.0	84.7	94.6	70.1	87.3

Table 3.5: AP on the PASCAL VOC Action 2012 val set. *RCNN* is the baseline approach, with the ground-truth region being the primary region. *Random-RCNN* is a network trained with primary the ground-truth region and secondary a random region. *Scene-RCNN* is a network trained with primary the ground-truth region and secondary the whole image. *R*CNN* (l, u) is our system where l, u define the lower and upper bounds of the allowed overlap of the secondary region with the ground truth. *R*CNN* (l, u, n_S) is a variant in which n_S secondary regions are used, instead of one.

AP (%)	CNN layers	Jumping	Phoning	Playing Instrument	Reading	Riding Bike	Riding Horse	Running	Taking Photo	Using Computer	Walking	mAP
Oquab <i>et al.</i> [7]	8	74.8	46.0	75.6	45.3	93.5	95.0	86.5	49.3	66.7	69.5	70.2
Hoai [11]	8	82.3	52.9	84.3	53.6	95.6	96.1	89.7	60.4	76.0	72.9	76.3
Gkioxari <i>et al.</i> [10]	16	84.7	67.8	91.0	66.6	96.6	97.2	90.2	76.0	83.4	71.6	82.6
Simonyan & Zisserman [12]	16 & 19	89.3	71.3	94.7	71.3	97.1	98.2	90.2	73.3	88.5	66.4	84.0
R*CNN	16	91.5	84.4	93.6	83.2	96.9	98.4	93.8	85.9	92.6	81.8	90.2

Table 3.6: AP on the PASCAL VOC Action 2012 test set. Oquab *et al.* [7] train an 8-layer network on ground-truth boxes. Gkioxari *et al.* [10] use part detectors for *head*, *torso*, *legs* and train a CNN. Hoai [11] uses an 8-layer network to extract fc7 features from regions at multiple locations and scales. Simonyan and Zisserman [12] combine a 16-layer and a 19-layer network and train SVMs on fc7 features from the image and the ground-truth box. R*CNN (with ($l = 0.2, u = 0.75$)) outperforms all other approaches by a significant margin.

3.3.3.2 Performance on PASCAL VOC

We compare R*CNN to other approaches on the PASCAL VOC Action test set. Table 3.6 shows the results. Oquab *et al.* [7] train an 8-layer network on ground-truth boxes. Gkioxari *et al.* [10] use part detectors for *head*, *torso*, *legs* and train a CNN on the part regions and the ground-truth box, as described in Section 3.2. Hoai [11] uses an 8-layer network to extract fc7 features from regions at multiple locations and scales inside the image and the box and accumulates their scores to get the final prediction. Simonyan and Zisserman [12] combine a 16-layer and a 19-layer network and train SVMs on fc7 features from the image and the ground-truth box. R*CNN (with ($l = 0.2, u = 0.75$)) outperforms all other approaches by a substantial margin. R*CNN seems to be performing significantly better for actions which involve small objects and action-specific pose appearance, such as *Phoning*, *Reading*, *Taking Photo*, *Walking*. More interestingly, R*CNN outperforms our own part-based approach described in Section 3.2, validating the claim that predefining the regions of interests for fine-grained tasks is suboptimal.



Figure 3.9: Top predictions on the PASCAL VOC Action test set. The instance in question is shown with a **red box**, while the selected secondary region with a **green box**. The nature of the secondary regions depends on the action and the image itself. Even within the same action category, the most informative cue can vary.

Visualization of secondary regions Figure 3.9 shows examples from the top predictions for each action on the test set. Each block corresponds to a different action. Red highlights the person to be classified while green the automatically selected secondary region. For actions *Jumping*, *Running* and *Walking* the secondary region is focused either on body parts (e.g. legs, arms) or on more instances surrounding the instance in question (e.g. joggers). For *Taking Photo*, *Phoning*, *Reading* and *Playing Instrument* the secondary region focuses almost exclusively on the object and its interaction with the arms. For *Riding Bike*, *Riding Horse* and *Using Computer* it focuses on the object, or the presence of similar instances and the scene.

Interestingly, the secondary region seems to be picking different cues depending on the instance in question. For example in the case of *Running*, the selected region might highlight the scene (e.g. road), parts of the human body (e.g. legs, arms) or a group of people performing the action, as shown in Figure 3.9.

Figure 3.10 shows erroneous predictions for each action on the val set (in descending score). Each block corresponds to a different action. The misclassified instance is shown in red and the corresponding secondary region with green. For *Riding Bike* and *Riding Horse*, which achieve a very high AP, the mistakes are of very low score. For *Jumping*, *Phoning* and *Using Computer* the mistakes occur due to confusions with instances of similar pose. In addition, for *Playing Instrument* most of the misclassifications are people performing in concert venues, such as singers. For *Taking Photo* and *Playing Instrument* the presence of the object seems to be causing most misclassifications. For *Running* and *Walking* they seem to often get confused with each other as well as with standing people (an action which is not present explicitly in the dataset).

3.3.3.3 Performance on MPII Human Pose Dataset

The MPII Human Pose dataset contains 400 actions and consists of approximately 40,000 instances and 24,000 images. The images are extracted from videos from YouTube. The training set consists of 15,200 images and 22,900 instances performing 393 actions. The number of positive training examples per category varies drastically [3]. The amount of training data ranges from 3 to 476 instances, with an average of 60 positives per action. The annotations do not include a ground-truth bounding box explicitly, but provide a point (anywhere in the human body) and a rough scale of the human. This information can be used to extract a rough location of the instance, which is used as input in our algorithm.

R*CNN vs. RCNN We split the training set into train and val sets. We make sure that frames of the same video belong to the same split to avoid overfitting. This results in 12,500 instances in train and 10,300 instances in val. We train the baseline RCNN network and R*CNN. We pick ($l = 0.2, u = 0.5$) due to the large number of region proposals generated by [46] (on average 8,000 regions per image).

On the val set, RCNN achieves 16.5% mean AP while R*CNN achieves 21.7% mean AP, across all actions. Figure 3.11 shows the performance on MPII val for RCNN and R*CNN. On the **left**, we show a scatter plot of the AP for all actions as a function of their training size. On the **right**, we show the mean AP across actions belonging to one out of three categories, depending on their training size.

The performance reported in Figure 3.11 is instance-specific. Namely, each instance is evaluated. One could evaluate the performance at the frame-level (as done in [3]), i.e. classify the frame and not the instance. We can generate frame-level predictions by assigning for each action the maximum score across instances in the frame. That yields 18.2% mean AP for RCNN and 23% mean AP for R*CNN.

Comparison with Published Results In [3], various approaches for action recognition are reported on the test set. All the approaches mentioned use motion features, by using frames in the temporal neighborhood of the frame in question. The authors test variants of Dense Trajectories (DT) [61] which they combine with pose specific features. The best

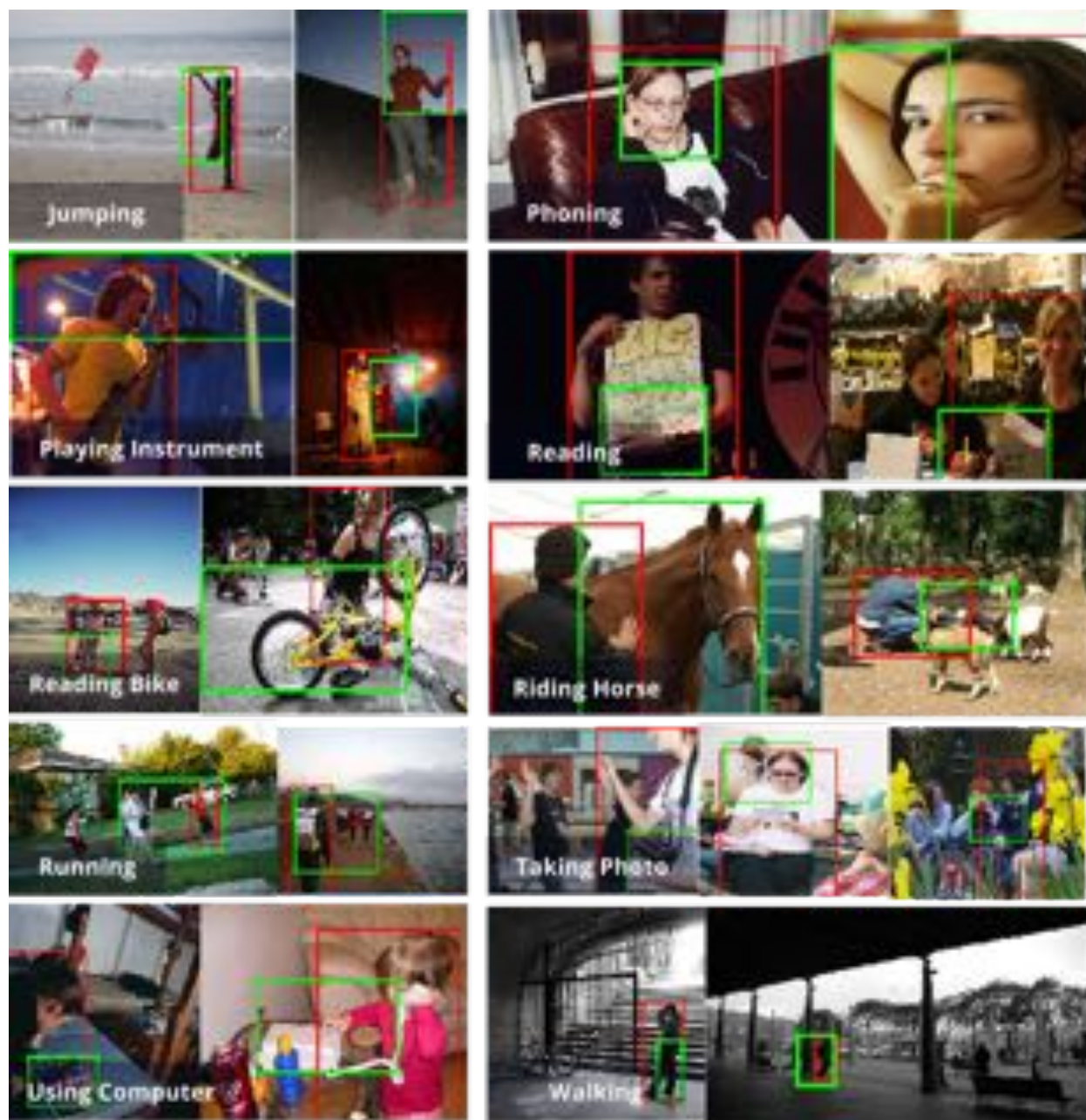


Figure 3.10: Top mistakes on the PASCAL VOC Action val set. The misclassified instance is shown in **red**, while the selected secondary region in **green**.

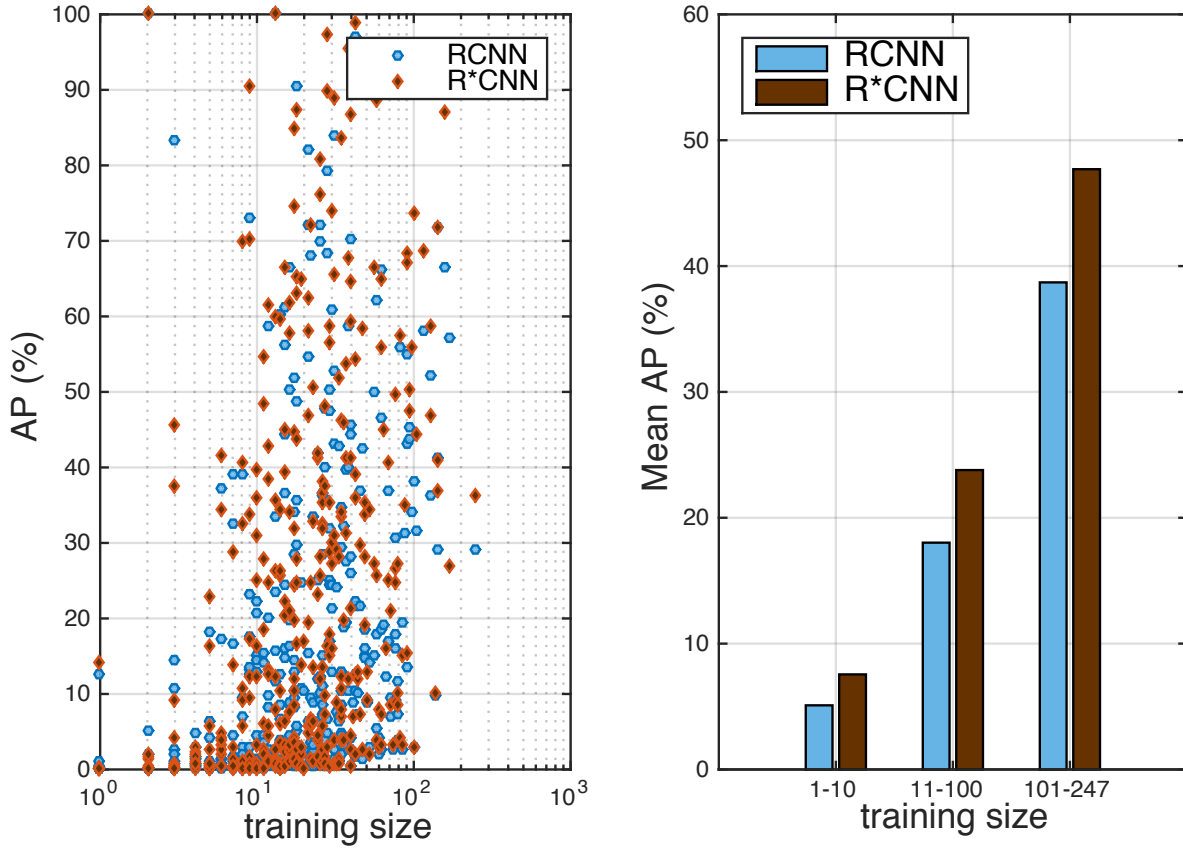


Figure 3.11: Performance on MPII val for RCNN (blue) and R*CNN (brown). **Left:** AP (%) for all actions as a function of their training size (x -axis). **Right:** Mean AP (%) for three discrete ranges of training size (x -axis).

performance on the test set is 5.5% mean AP (frame-level) achieved by the DT combined with a pose specific approach.

We evaluate R*CNN on the test set and achieve 26.7% mAP for frame-level recognition. Our approach does not use motion, which is a strong cue for action recognition in video, and yet manages to outperform DT by a significant margin. Evaluation on the test set is performed only at the frame-level.

Figure 3.12 shows the mean AP across actions in a descending order of training size. This figure allows for a direct comparison with the published results, as shown in Figure 1(b) in [3].

Figure 3.13 shows some results on the test set. We highlight the instance in question with red, and the secondary box with green. The boxes for the instances were derived from the point annotations (some point on the person) and the rough scale provided at train and

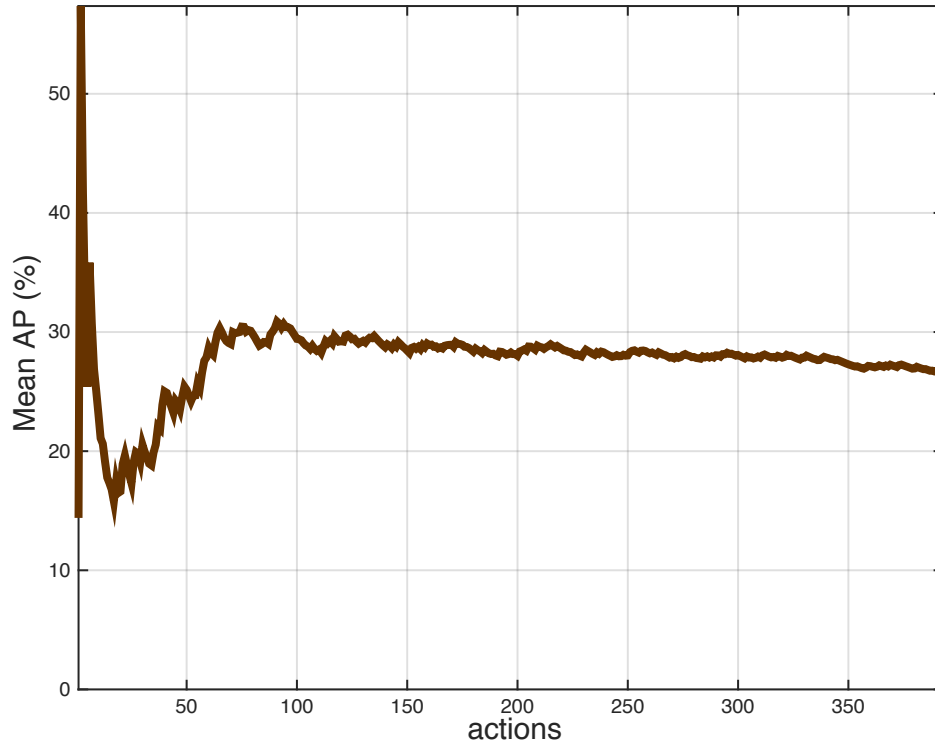


Figure 3.12: Mean AP (%) on MPII test for R*CNN across actions in descending order of their training size. A direct comparison with published results, as shown in Figure 1(b) in [3], can be drawn.

test time. The predicted action label is overlaid in each image.

Even though R*CNN outperforms DT, there is still need of movement to boost performance for many categories. For example, even though the MPII dataset has a many examples for actions such as *Yoga*, *Cooking or food preparation* and *Video exercise workout*, R*CNN performs badly on those categories (1.1% mean AP). We believe that a hybrid approach which combines image and motion features, similar to [62, 63], would perform even better.

3.3.3.4 Performance on Stanford 40 Actions Dataset

We run R*CNN on the Stanford 40 Actions dataset [60]. This dataset consists of 9532 images of people performing 40 different actions. The dataset is split in half to comprise the training and test split. Bounding boxes are provided for all people performing actions. R*CNN achieves an average AP of 90.9% on the test set, with performance varying from 70.5% for *texting message* to 100% for *playing violin*. Figure 3.14 shows the AP performance per action on the test set. Training code and models are publicly available.



Figure 3.13: Predictions on the MPII test set. We highlight the person in question with a red box, and the secondary region with a green box. The predicted action label is overlaid.

AP (%)	CNN layers	Is Male	Has Long Hair	Has Glasses	Has Hat	Has T-Shirt	Has Long Sleeves	Has Shorts	Has Jeans	Has Long Pants	mAP
PANDA [13]	5	91.7	82.7	70.0	74.2	49.8	86.0	79.1	81.0	96.4	79.0
Gkioxari <i>et al.</i> [10]	16	92.9	90.1	77.7	93.6	72.6	93.2	93.9	92.1	98.8	89.5
RCNN	16	91.8	88.9	81.0	90.4	73.1	90.4	88.6	88.9	97.6	87.8
R*CNN	16	92.8	88.9	82.4	92.2	74.8	91.2	92.9	89.4	97.9	89.2

Table 3.7: AP on the Berkeley Attributes of People test set. PANDA [13] uses CNNs trained for each poselet type. Gkioxari *et al.* [10] detect parts and train a CNN jointly on the whole and the parts. RCNN is our baseline approach based on FRCN. Both RCNN and R*CNN do not use any additional part annotations at training time. [10] and R*CNN perform equally well, with the upside that R*CNN does not need use keypoint annotations during training.

3.3.3.5 Performance on Berkeley Attributes of People Dataset

Finally, we show that R*CNN can also be used for the task of attribute classification. On the Berkeley Attributes of People dataset [52], which consists of images of people and their attributes, e.g. *wears hat*, *is male* etc, we train R*CNN as described above. The only difference is that our loss is no longer a log loss over softmax probabilities, but the cross entropy over independent logistics because attribute prediction is a multi-label task. Table 3.7 reports the performance in AP of our approach, as well as other competing methods. Figure 3.15 shows results on the test set. From the visualizations, the secondary regions learn to focus on the parts that are specific to the attribute being considered. For example, for the *Has Long Sleeves* class, the secondary regions focus on the arms and torso of the instance in question, while for *Has Hat* focus is on the face of the person.

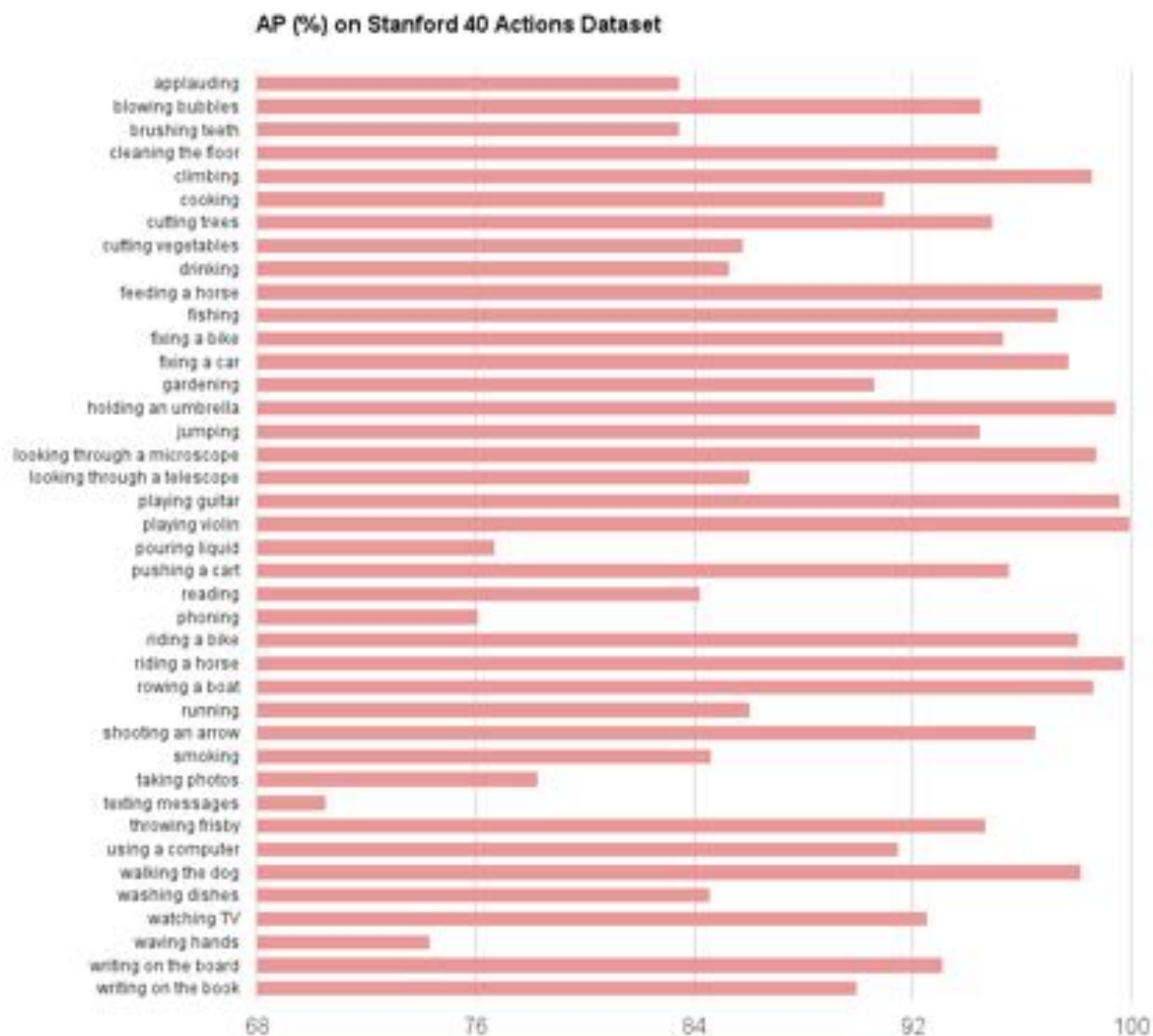


Figure 3.14: AP (%) of R*CNN on the Stanford 40 dataset per action. Performance varies from 70.5% for *texting message* to 100% for *playing violin*. The average AP across all actions achieved by our model is 90.9%.

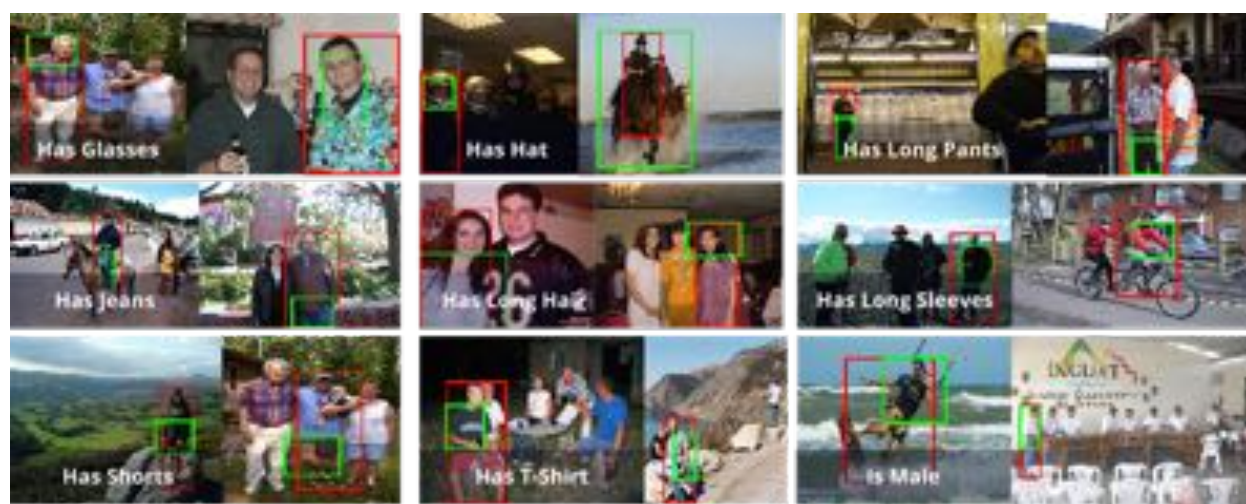


Figure 3.15: Results on the Berkeley Attributes of People test set. We highlight the person in question with a red box, and the secondary region with a green box. The predicted attribute is overlaid.

Chapter 4

Action Recognition From Videos

Action recognition from still images is an ill-defined task. For example, it is very difficult to accurately recognize if a person is *walking* vs. *standing* from a single image. Actions take their meaning *in time*. *Motion* captures the patterns and change in shape of objects as they interact with their environment, thus providing more information about the scene.

In this chapter, we will explore the impact of motion for the task of action recognition. Our effort focuses on combining motion and spatial cues with CNNs, inspired by the human vision system and, in particular, the two-streams hypothesis [64]. The ventral pathway (“*what pathway*”) in the visual cortex responds to shape, color and texture while the dorsal pathway (“*where pathway*”) responds to spatial transformations and movement. We use convolutional neural networks to computationally simulate the two pathways. The first network, *spatial-CNN*, operates on static cues and captures the appearance of the actor and the environment. The second network, *motion-CNN*, operates on motion cues and captures patterns of movement of the actor and the object (if any) involved in the action. Both networks are trained to discriminate between the actors and the background as well as between actors performing different actions.

In particular, we focus on the task of *action detection* from videos. Here the goal is to localize the action in space across all frames of the video. This is contrary to *action classification*, where the task is to classify a video with an action label, without localizing it. The distinction between action detection and classification is similar to the one between object detection and image classification, in the single image case.¹

4.1 Related Work on Action Recognition from Videos

There has been a fair amount of research on action recognition. We refer to [65, 66, 67] for recent surveys in the field. For the task of action classification, recent approaches use features based on shape (e.g. HOG [49], SIFT [50]) and motion (e.g. optical flow, MBH [68]) with high order encodings (e.g. Bag of Words, Fischer vectors) and train classifiers (e.g. SVM,

¹The work presented here is based on published work in [63].

decision forests) to make action predictions. More specifically, Laptev *et al.* [69] extract local features at spatio-temporal interest points which they encode using Bag of Words and train SVM classifiers. Wang *et al.* [70] use dense point trajectories, where features are extracted from regions which are being tracked using optical flow across the frames, instead of fixed locations on a grid space. Recently, the authors improved their approach [61] using camera motion to correct the trajectories. They estimate the camera movement by matching points between frames using shape and motion cues after discarding those that belong to the humans in the frame. The big relative improvement of their approach shows that camera motion has a significant impact on the final predictions, especially when dealing with real world video data. Jain *et al.* [71] make a similar observation.

Following the impressive results of deep architectures, such as CNNs, on the task of handwritten digit recognition [19] and more recently image classification [20] and object detection in images [5], attempts have been made to train deep networks for the task of action classification. Jhuang *et al.* [72] build a feedforward network which consists of a hierarchy of spatio-temporal feature detectors of predesigned motion and shape filters, inspired by the dorsal stream of the visual cortex. Taylor *et al.* [73] use convolutional gated RBMs to learn features for video data in an unsupervised manner and apply them for the task of action classification. More recently, Ji *et al.* [74] build 3D CNNs, where convolutions are performed in 3D feature maps from both spatial and temporal dimensions. Karpathy *et al.* [75] explore a variety of network architectures to tackle the task of action classification on 1M videos. They show that operating on single frames performs equally well than when considering sequences of frames. Simonyan & Zisserman [62] train two separate CNNs to explicitly capture spatial and temporal features. The spatial stream operates on the RGB image while the temporal stream on the optical flow signal. The two stream structure in our network for action detection is similar to their work, but the crucial difference is that their network is for full image classification while our system works on candidate regions and can thus localize the action. Also, the way we do temporal integration is quite different since our work tackles a different problem.

Approaches designed for the task of action classification use feature representations that discard any information regarding the location of the action. However, there are older approaches which are figure centric. Efros *et al.* [76] combine shape and motion features to build detectors suitable for action recognition at low resolution and predict the action using nearest neighbor techniques, but they assume that the actor has already been localized. Schüldt *et al.* [77] build local space-time features to recognize action patterns using SVM classifiers. Blank *et al.* [78] use spatio-temporal volume silhouettes to describe an action assuming in addition known background. More recently, per-frame human detectors have been used. Prest *et al.* [79] propose to detect humans and objects and then model their interaction. Lan *et al.* [80] learn spatio-temporal models for actions using figure-centric visual word representation, where the location of the subject is treated as a latent variable and is inferred jointly with the action label. Raptis *et al.* [81] extract clusters of trajectories and group them to predict an action class using a graphical model. Tian *et al.* [82] extend the deformable parts model, introduced by [36] for object detection in 2D images, to video

using HOG3D feature descriptors [83]. Ma *et al.* extract segments of the human body and its parts based on color cues, which they prune using motion and shape cues. These parts serve as regions of interest from which features are extracted and subsequently are encoded using Bag of Words. Jain *et al.* [84] produce space-time bounding boxes, starting from super-voxels, and use motion features with Bag of Words to classify the action within each candidate. Wang *et al.* [85] propose a unified approach to discover effective action parts using dynamical poselets and model their relations.

4.2 Action Tubes

Our goal is to build models which can localize and classify actions in video. Figure 4.1 outlines our approach. Inspired by the recent advances in the field of object detection in images [5], we start by selecting candidate regions and use convolutional networks (CNNs) to classify them. Motion is a valuable cue for action recognition and we utilize it in two ways. We use motion saliency to eliminate regions that are not likely to contain the action. This leads to a big reduction in the number of regions being processed and subsequently in compute time. Additionally, we incorporate kinematic cues to build powerful models for action detection. Figure 4.2 shows the design of our action models. Given a region, appearance and motion cues are used with the aid of convolutional neural networks to make a prediction. Predictions from all the frames of the video are linked to produce consistent detections in time. We call the linked predictions in time *action tubes*.

4.2.1 Regions of Interest

Given a frame, the number of possible regions that contain the action is enormous. However, the majority of these candidates are not descriptive and can be eliminated without loss in performance. There has been a lot of work on generating useful region proposals based on color, texture, edge cues ([46, 86]). We use selective search [46] on the RGB frames to generate approximately 2K regions per frame. Given that our task is to localize the actor, we discard the regions that are void of motion, using the optical flow signal. As a result, the final regions we consider are those that are salient in shape and motion. One could use more complicated techniques, such as action saliency detectors trained on human eye fixations and low level cues [87].

Our motion saliency algorithm is extremely simple. We view the normalized magnitude of the optical flow signal f_m as a heat map at the pixel level. If R is a region, then $f_m(R) = \frac{1}{|R|} \sum_{i \in R} f_m(i)$ is a measure of how motion salient R is. R is discarded if $f_m(R) < \alpha$.

For $\alpha = 0.3$, approximately 85% of boxes are discarded, with a loss of only 4% in recall on J-HMDB, for an overlap threshold of 0.5. Despite the small loss in recall, this step is of great importance for the algorithm’s time complexity. It takes approximately 11s to process an image with 2K boxes, with the majority of the time being consumed in extracting features for the boxes (for more details see [5]). This means that a video of 100 frames

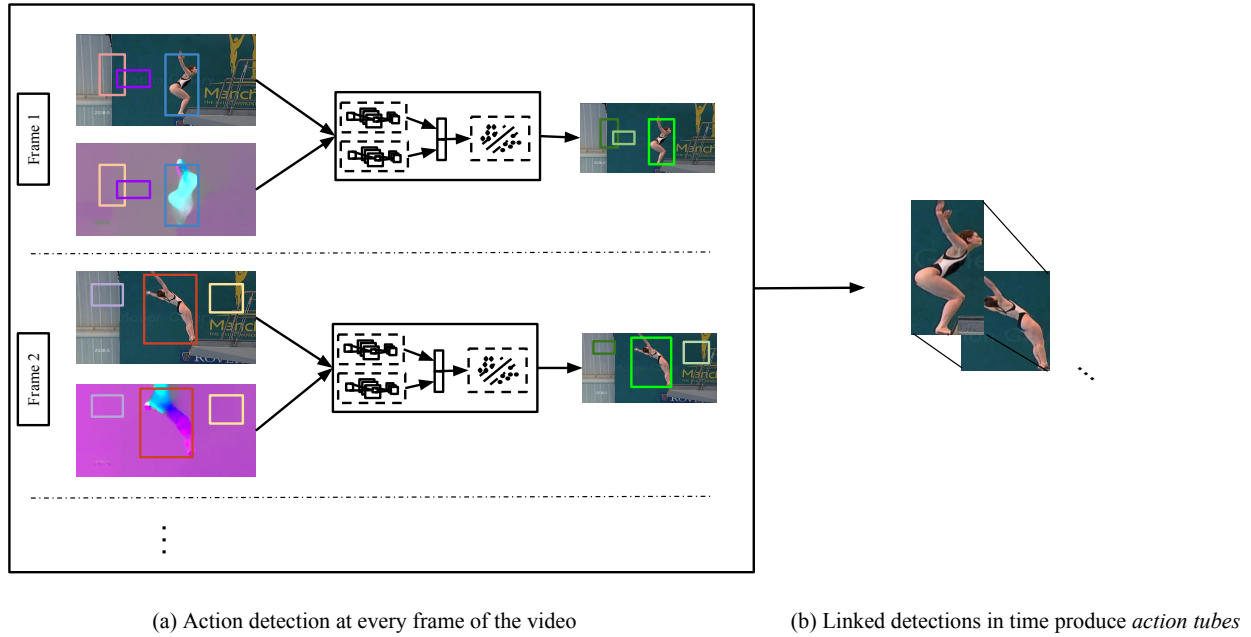


Figure 4.1: An outline of our approach. (a) Candidate regions are fed into action specific classifiers, which make predictions using static and motion cues. (b) The regions are linked across frames based on the action predictions and their spatial overlap. *Action tubes* are produced for each action and each video.

would require 18min to process! This is prohibitive, especially for a dataset of thousands of videos. Eliminating regions which are unlikely to contain the action reduces the compute time significantly.

4.2.2 Action Specific Classifiers

We use discriminative action classifiers on spatio-temporal features to make predictions for each region. The features are extracted from the final layer of the CNNs which are trained to discriminate among different actions as well as between actions and the background. We use a linear SVM with hard negative mining to train the final classifiers. Figure 4.2 shows how spatial and motion cues are combined and fed into the SVM classifier.

4.2.2.1 CNNs for Action Detection

We train two Convolutional Neural Networks for the task of action detection. The first network, *spatial-CNN*, takes as input RGB frames and captures the appearance of the actor as well as cues from the scene. The second network, *motion-CNN*, operates on the optical flow signal and captures the movement of the actor. Spatio-temporal features are extracted

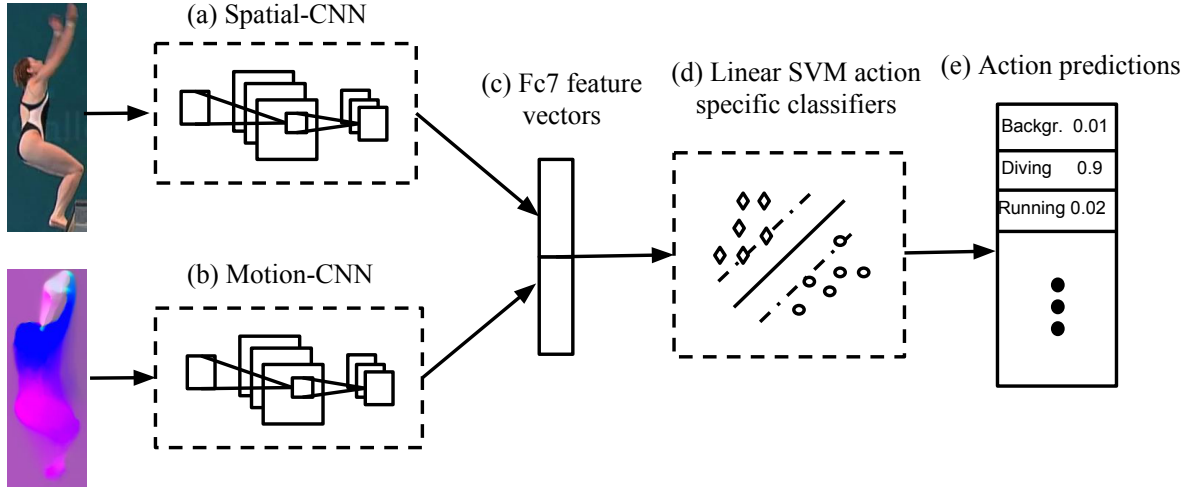


Figure 4.2: We use action specific SVM classifiers on spatio-temporal features. The features are extracted from the fc_7 layer of two CNNs, *spatial-CNN* and *motion-CNN*, which were trained to detect actions using static and motion cues, respectively.

by combining the output from the intermediate layers of the two networks. Action specific SVM classifiers are trained on the spatio-temporal features and are used to make predictions at the frame level. Figure 4.2 schematically outlines the procedure. Subsequently, we link the detections in time to produce temporarily consistent action predictions, which we call *action tubes*.

We train spatial-CNN and motion-CNN similar to R-CNN [5]. Regions of interest are computed at every frame of the video, as described above. At train time, the regions which overlap more than 50% with the ground truth are considered as positive examples, and the rest are negatives. The networks are carefully initialized to avoid overfitting.

The architecture of spatial-CNN and motion-CNN is identical and follows [20] and [88]. Assume $C(k, n, s)$ is a convolutional layer with kernel size $k \times k$, n filters and a stride of s , $P(k, s)$ a max pooling layer of kernel size $k \times k$ and stride s , N a normalization layer, RL a rectified linear unit, $FC(n)$ a fully connected layer with n filters and $D(r)$ a dropout layer with dropout ratio r . The architecture of our networks follows: $C(7, 96, 2) - RL - P(3, 2) - N - C(5, 384, 2) - RL - P(3, 2) - N - C(3, 512, 1) - RL - C(3, 512, 1) - RL - C(3, 384, 1) - RL - P(3, 2) - FC(4096) - D(0.5) - FC(4096) - D(0.5) - FC(|A| + 1)$. The final fully connected layer has number of outputs as many as the action classes plus one for the background class. During training a softmax loss layer is added at the end of the network.

Network details. The architecture of our CNNs is inspired by two different network designs, [20] and [88]. Our network achieves 17% top-5 error on the ILSVRC-2012 validation set for the task of classification.

Weight initialization. Proper initialization is a key for training CNNs, especially in the absence of data.

spatial-CNN: We want spatial-CNN to accurately localize people performing actions in 2D frames. We initialize spatial-CNN with a model that was trained on the PASCAL VOC 2012 detection task, similar to [5]. This model has learned feature representations necessary for accurately detecting people under various appearance and occlusion patterns, as proven by the high person detection AP reported on the VOC2012 test set.

motion-CNN: We want motion-CNN to capture motion patterns. We train a network on single frame optical flow images for the task of action classification. We use the UCF101 dataset (split 1) [89], which contains 13320 videos of 101 different actions. Our single frame optical flow model achieves an accuracy of 72.2% on split 1, similar to 73.9% reported by [62]. The 1.7% difference can be attributed to the differences in the network’s architectures. Indeed, the network used in [62] yields 13.5% top-5 error on the ILSVRC-2012 validation set, compared to the 17% top-5 error achieved by our network. This model is used to initialize motion-CNN when trained on smaller datasets, such as UCF Sports and J-HMDB.

Processing of input data. We preprocess the input for each of the networks as follows

spatial-CNN: The RGB frames are cropped to the bounds of the regions of interest, with a padding of 16 pixels, which is added in each dimension. The average RGB values are subtracted from the patches. During training, the patches are randomly cropped to 227×227 size, and are flipped horizontally with a probability of 0.5.

motion-CNN: We compute the optical flow signal for each frame, according to [90]. We stack the flow in the x-, y-direction and the magnitude to form a 3-dimensional image, and scale it by a constant ($s = 16$). During training, the patches are randomly cropped and flipped.

Parameters. We train spatial-CNN and motion-CNN with backpropagation, using Caffe [56]. We use a learning rate of 0.001, a momentum of 0.9 and a weight decay of 0.0005. We train the networks for 2K iterations. We observed more iterations were unnecessary, due to the good initialization of the networks.

4.2.2.2 Training Action Specific SVM Classifiers

We train action specific SVM classifiers on spatio-temporal features, which are extracted from an intermediate layer of the two networks. More precisely, given a region R , let $\phi_s(R)$ and $\phi_m(R)$ be the feature vectors computed after the 7th fully connected layer in spatial-CNN and motion-CNN respectively. We combine the two feature vectors $\phi(R) = [\phi_s(R)^T \phi_m(R)^T]^T$

to obtain a spatio-temporal feature representation for R . We train SVM classifiers \mathbf{w}_α for each action $\alpha \in A$, where ground truth regions for α are considered as positive examples and regions that overlap less than 0.3 with the ground truth as negative. During training, we use hard negative mining.

At test time, each region R is associated with a score vector $score(R) = \{\mathbf{w}_\alpha^T \phi(R) : \alpha \in A\}$, where each entry is a measure of confidence that action α is performed within the region.

4.2.3 Linking Action Detections

Actions in videos are being performed over a period of time. Our approach makes decisions on a single frame level. In order to create temporally coherent detections, we link the results from our single frame approach into unified detections along time.

Assume two consecutive frames at times t and $t + 1$, respectively, and assume R_t is a region at t and R_{t+1} at $t + 1$. For an action α , we define the linking score between those regions to be

$$s_\alpha(R_t, R_{t+1}) = \mathbf{w}_\alpha^T \phi(R_t) + \mathbf{w}_\alpha^T \phi(R_{t+1}) + \lambda \cdot ov(R_t, R_{t+1}) \quad (4.1)$$

where $ov(R, \hat{R})$ is the intersection-over-union of two regions R and \hat{R} and λ is a scalar. In other words, two regions are strongly linked if their spatial extent significantly overlaps and if they score high under the action model.

For each action in the video, we seek the optimal path

$$\bar{R}_\alpha^* = \arg \max_{\bar{R}} \frac{1}{T} \sum_{t=1}^{T-1} s_\alpha(R_t, R_{t+1}) \quad (4.2)$$

where $\bar{R}_\alpha = [R_1, R_2, \dots, R_T]$ is the sequence of linked regions for action α . We solve the above optimization problem using the Viterbi algorithm. After the optimal path is found, the regions in \bar{R}_α^* are removed from the set of regions and Equation 4.2 is solved again. This is repeated until the set of regions is empty. Each path from Equation 4.2 is called an *action tube*. The score of an action tube \bar{R}_α is defined as $S_\alpha(\bar{R}_\alpha) = \frac{1}{T} \sum_{t=1}^{T-1} s_\alpha(R_t, R_{t+1})$.

4.3 Experiments

We evaluate our approach on two widely used datasets, namely UCF Sports [91] and J-HMDB [92]. On UCF sports we compare against other techniques and show substantial improvement from state-of-the-art approaches. We present an ablation study of our CNN-based approach and show results on action classification using our action tubes on J-HMDB, which is a substantially larger dataset than UCF Sports.

Datasets. UCF Sports consists of 150 videos with 10 different actions. There are on average 10.3 videos per action for training, and 4.7 for testing (The split was proposed by [80]). J-HMDB contains about 900 videos of 21 different actions. The videos are extracted from the larger HMDB dataset [93], consisting of 51 actions. Contrary to J-HMDB, UCF Sports has been widely used by scientists for evaluation purposes. J-HMDB is more interesting and should receive much more attention than it has in the past.

Metrics. To quantify our results, we report Average-Precision at a frame level, *frame-AP*, and at the video level, *video-AP*. We also plot ROC curves and measure AUC, a metric commonly used by other approaches. None of the AP metrics have been used by other methods on this task. However, we feel they are informative and provide a direct link between the tasks of action detection and object detection in images.

- **frame-AP** measures the area under the precision-recall curve of the detections for each frame (similar to the PASCAL VOC detection challenge [54]). A detection is correct if the intersection-over-union with the ground truth at that frame is greater than σ and the action label is correctly predicted.
- **video-AP** measures the area under the precision-recall curve of the action tubes predictions. A tube is correct if the mean per frame intersection-over-union with the ground truth across the frames of the video is greater than σ and the action label is correctly predicted.
- **AUC** measures the area under the ROC curve, a metric previously used on this task. An action tube is correct under the same conditions as in *video-AP*. Following [82], the ROC curve is plotted until a false positive rate of 0.6, while keeping the top-3 detections per class and per video. Consequently, the best possible AUC score is 60%.

4.3.1 Results on UCF Sports

In Figure 4.3 (left) we plot the ROC curve for $\sigma = 0.2$ (red). In Figure 4.3 (right) we plot the average AUC for different values of σ . We plot the curves as produced by the recent approaches, Jain *et al.* [84], Wang *et al.* [85], Tian *et al.* [82] and Lan *et al.* [80]. Our approach outperforms all other techniques by a significant margin for all values of σ , showing the most improvement for high values of overlap, where other approaches tend to perform poorly. In particular, for $\sigma = 0.6$, our approach achieves an average AUC of 41.2% compared to 22.0% by [85].

Table 4.1 shows frame-AP (second row) and video-AP (third row) for an intersection-over-union threshold of $\sigma = 0.5$. Our approach achieves a mean AP of 68.1% at the frame level and 75.8% at the video level, with excellent performance for most categories. *Running* is the only action for which the action tubes fail to detect the actors (11.7 % video-AP), even though our approach is able to localize them at the frame level (54.9% frame-AP).

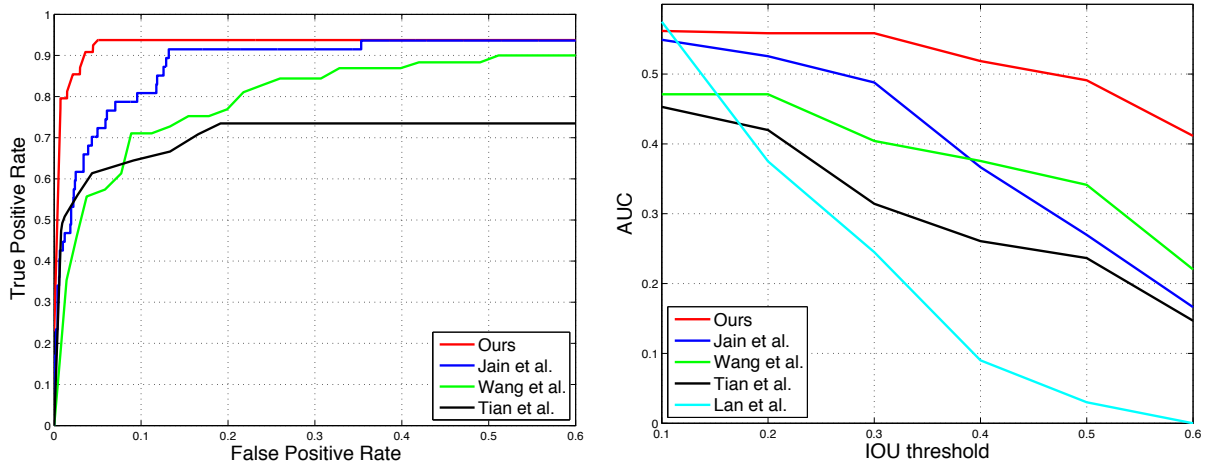


Figure 4.3: Performance on UCF Sports. **Left:** ROC curves on UCF Sports for an intersection-over-union threshold of $\sigma = 0.2$. Red shows our approach. We manage to reach a high true positive rate at a much smaller false positive rate, compared to the other approaches shown on the plot. **Right:** AUC on UCF Sports for various values of intersection-over-union threshold of σ (x -axis). Red shows our approach. We consistently outperform other approaches, with the biggest improvement being achieved at high values of overlap ($\sigma \geq 0.4$).

AP (%)	Diving	Golf	Kicking	Lifting	Riding	Running	Skateboarding	Swing1	Swing2	Walking	mAP
frame-AP	75.8	69.3	54.6	99.1	89.6	54.9	29.8	88.7	74.5	44.7	68.1
video-AP	100	91.7	66.7	100	100	11.7	41.7	100	100	45.8	75.8

Table 4.1: AP on the UCF Sports dataset for an intersection-over-union threshold of $\sigma = 0.5$. *frame-AP* measures AP of the action detections at the frame level, while *video-AP* measures AP of the predicted action tubes.

This is due to the fact that the test videos for *Running* contain multiple actors next to each other and our simple linking algorithm fails to consistently associate the detections with the correct actors, because of the proximity of the subjects and the presence of camera motion. In other words, the action tubes for *Running* contain the action but the detections do not always correspond to the same person. Indeed, if we make our evaluation agnostic to the instance, video-AP for *Running* is 83.8%. Tracking objects in a video is a very interesting but rather orthogonal problem to action localization and is beyond the scope of this work.

Figure 4.4 shows examples of detected action tubes on UCF sports. Each block corresponds to a different video. The videos were selected from the test set. We show the highest scoring action tube for each video. Red boxes indicate the detections in the corresponding frames. The predicted label is overlaid.

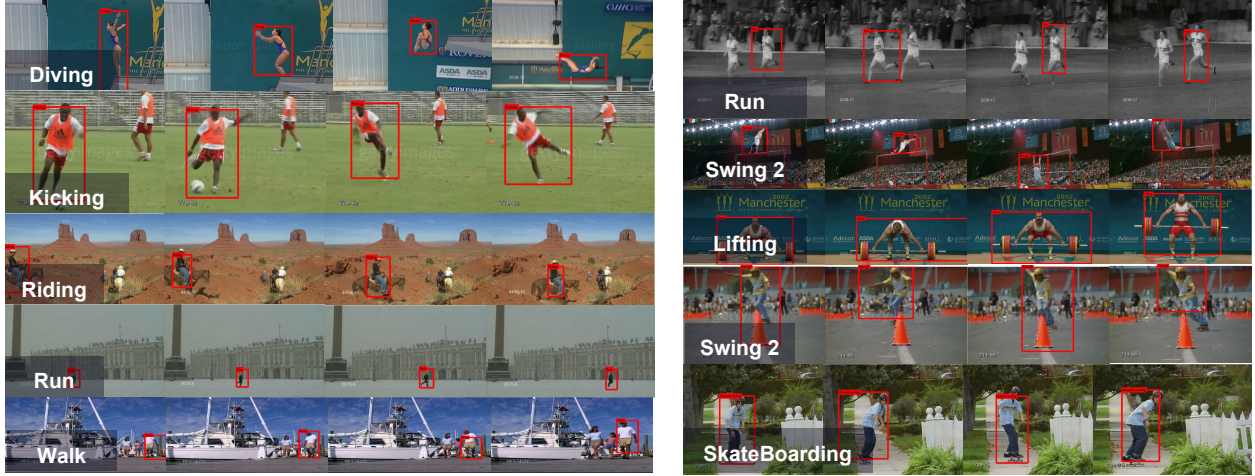


Figure 4.4: Examples from UCF Sports. Each block corresponds to a different video. We show the highest scoring action tube detected in the video. The red box indicates the region and the predicted label is overlaid. We show 4 frames from each video. The top example on the right shows the problem of tracking, while the 4th example on the right is a wrong prediction, with the true label being *Skate Boarding*.

4.3.2 Results on J-HMDB

We report frame-AP and video-AP for the 21 actions of J-HMDB. We present an ablation study of our approach by evaluating the performance of the two networks, *spatial-CNN* and *motion-CNN*. Table 4.2 shows the results for each method and for each action category.

As shown in the ablation study, it is apparent that the combination of spatial and motion-CNN performs significantly better for almost all actions. In addition, we can make some very useful observations. There are specific categories for which one signal matters more than the other. In particular, motion seems to be the most important for actions such as *Clap*, *Climb Stairs*, *Sit*, *Stand* and *Swing Baseball*, while appearance contributes more for actions such as *Catch*, *Shoot Gun* and *Throw*. Also, we notice that even though motion-CNN performs on average a bit worse than spatial-CNN at the frame level (24.3% vs. 27.0% respectively), it performs significantly better at the video level (45.7% vs. 37.9% respectively). This is due to the fact that the flow frames are not very informative when considered separately, however they produce a stronger overall prediction after the temporal smoothing provided by our linking algorithm.

Figure 4.5 shows the AUC for different values of the intersection-over-union threshold, averaged over the three splits on J-HMDB. Unfortunately, comparison with other approaches is not possible on this dataset, since no other approaches report numbers or have source code available.

Figure 4.7 shows examples of action tubes on J-HMDB. Each block corresponds to a different video. The videos are selected from the split 1 test set. We show the highest

frame-AP (%)	brush_hair	catch	clap	climb_stairs	golf	jump	kick_ball	pick	pour	pullup	push	run	shoot_ball	shoot_bow	shoot_gun	sit	stand	swing_baseball	throw	walk	wave	<i>mAP</i>
spatial-CNN	55.8	25.5	25.1	24.0	77.5	01.9	05.3	21.4	68.6	71.0	15.4	06.3	04.6	41.1	28.0	09.4	08.2	19.9	17.8	29.2	11.5	27.0
motion-CNN	32.3	05.0	35.6	30.1	58.0	07.8	02.6	16.4	55.0	72.3	08.5	06.1	03.9	47.8	07.3	24.9	26.3	36.3	04.5	22.1	7.6	24.3
full	65.2	18.3	38.1	39.0	79.4	07.3	09.4	25.2	80.2	82.8	33.6	11.6	05.6	66.8	27.0	32.1	34.2	33.6	15.5	34.0	21.9	36.2
video-AP (%)																						
spatial-CNN	67.1	34.4	37.2	36.3	93.8	07.3	14.4	29.6	80.2	93.9	17.4	10.0	08.8	71.2	45.8	17.7	11.6	38.5	20.4	40.5	19.4	37.9
motion-CNN	66.3	16.0	60.0	51.6	88.6	18.9	10.8	23.9	83.4	96.7	18.2	17.2	14.0	84.4	19.3	72.6	61.8	76.8	17.3	46.7	14.3	45.7
full	79.1	33.4	53.9	60.3	99.3	18.4	26.2	42.0	92.8	98.1	29.6	24.6	13.7	92.9	42.3	67.2	57.6	66.5	27.9	58.9	35.8	53.3

Table 4.2: Results and ablation study on J-HMDB (averaged over the three splits). We report *frame-AP* (top) and *video-AP* (bottom) for the spatial and motion component and their combination (full). The combination of the spatial- and motion-CNN performs significantly better under both metrics, showing the significance of static and motion cues for the task of action recognition.

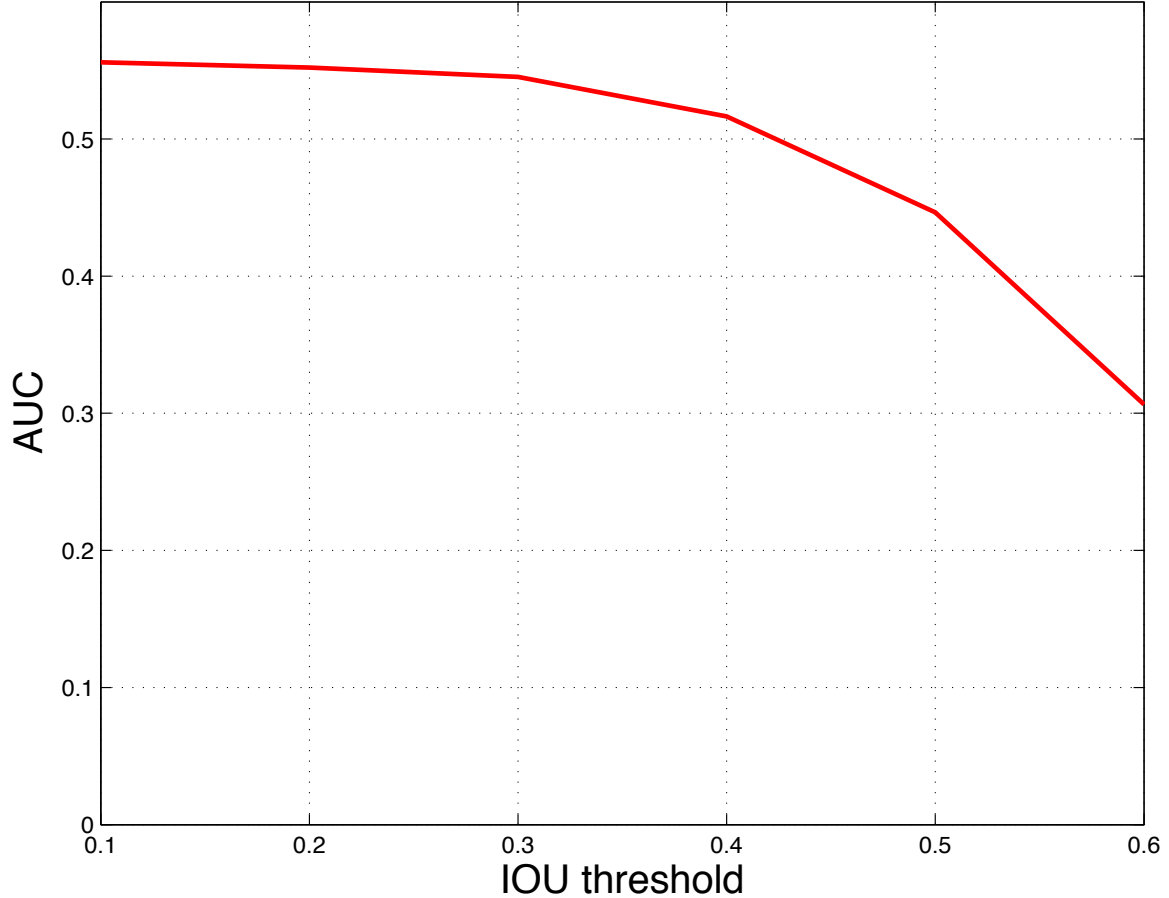


Figure 4.5: AUC on J-HMDB for different values of intersection-over-union threshold (averaged over the three splits).

Accuracy (%)	Wang <i>et al.</i> [70]	CNN (1/3 spatial, 2/3 motion)	Action Tubes
J-HMDB	56.6	56.5	62.5

Table 4.3: Classification accuracy on J-HMDB (averaged over the three splits). CNN (third column) shows the result of the weighted average of spatial and motion-CNN on the whole frames, while Action Tubes (fourth column) shows the result after using the scores of the predicted action tubes to make decisions for the video’s label.

scoring action tube for each video. Red boxes indicate the detections in the corresponding frames. The predicted label is overlaid.

Action Classification Our approach is not limited to action detection. We can use the action tubes to predict an action label for the whole video. In particular, we can predict the label l for a video by picking the action with the maximum action tube score

$$l = \arg \max_{\alpha \in A} \max_{\bar{R} \in \{\bar{R}_\alpha\}} S_\alpha(\bar{R}) \quad (4.3)$$

where $S_\alpha(\bar{R})$ is the score of the action tube \bar{R} as defined by Equation 4.2.

If we use Equation 4.3 as the prediction, our approach yields an accuracy of 62.5%, averaged over the three splits of J-HMDB. Figure 4.6 shows the confusion matrix.

In order to show the impact of the action tubes in the above result, we create a baseline model for action classification, similar to [62]. We use spatial and motion-CNNs in a classification setting, where full frames are used as input instead of regions. The weights of the CNNs are initialized from networks trained on UCF 101 (split1) for the task of action classification. We average the class probabilities as produced by the softmax layers of the CNNs, instead of training SVM classifiers (We observed major overfitting problems when training SVM classifiers on top of the combined fc_7 features). We average the outputs of spatial- and motion-CNNs, with weights 1/3 and 2/3 respectively, and pick the action label with the maximum score after averaging the frames of the videos. This approach yields an accuracy of 56.5% averaged over the three splits of J-HMDB. This compares to 56.6% achieved by [70]. Table 4.3 summarizes the results for action classification on J-HMDB. It is quite evident that focusing on the actor is beneficial for the task of video classification, while a lot of information is being lost when the whole scene is analyzed in an orderless fashion.

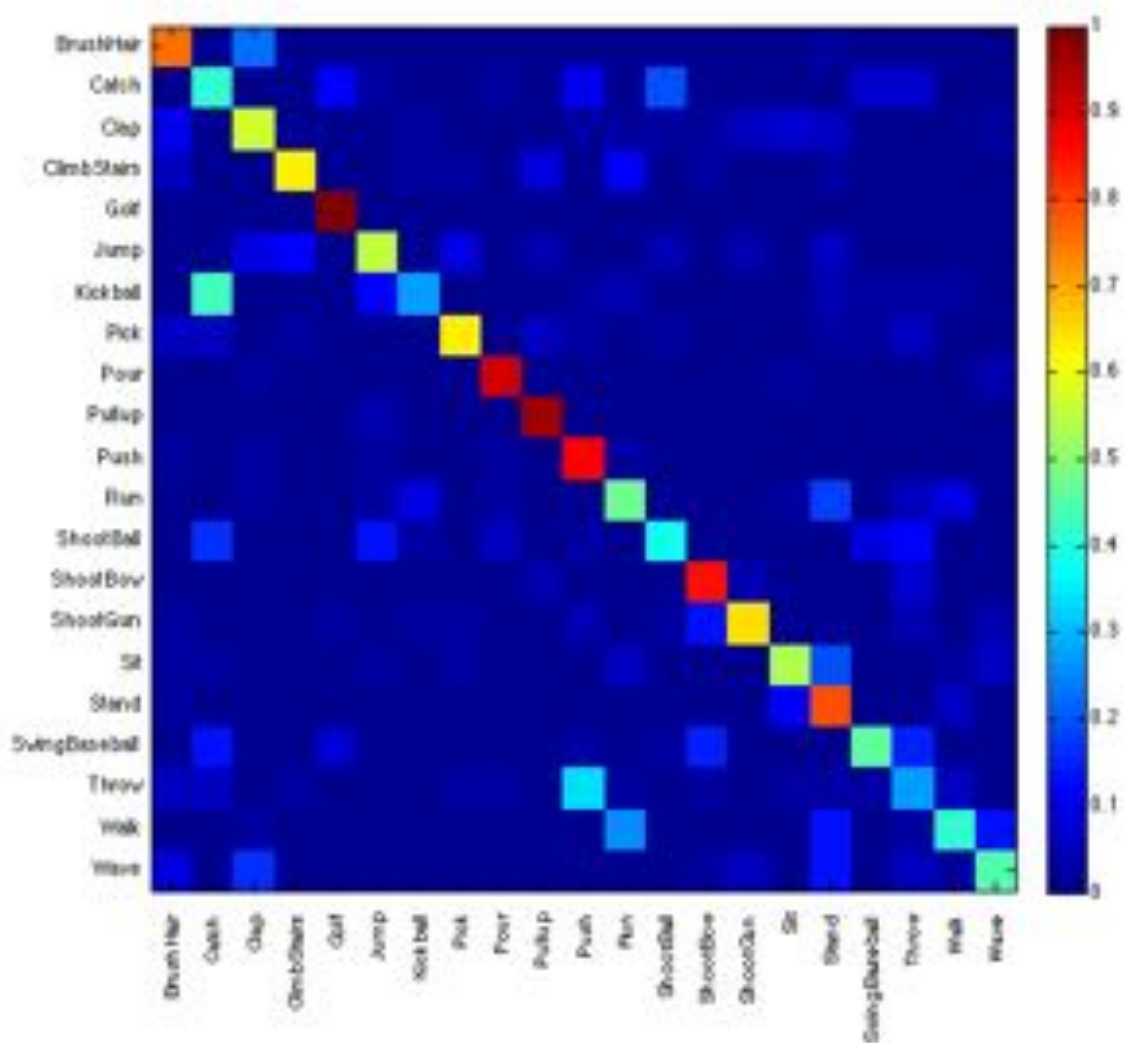


Figure 4.6: The confusion matrix on J-HMDB for the classification task, when using action tubes to predict a label for each video.

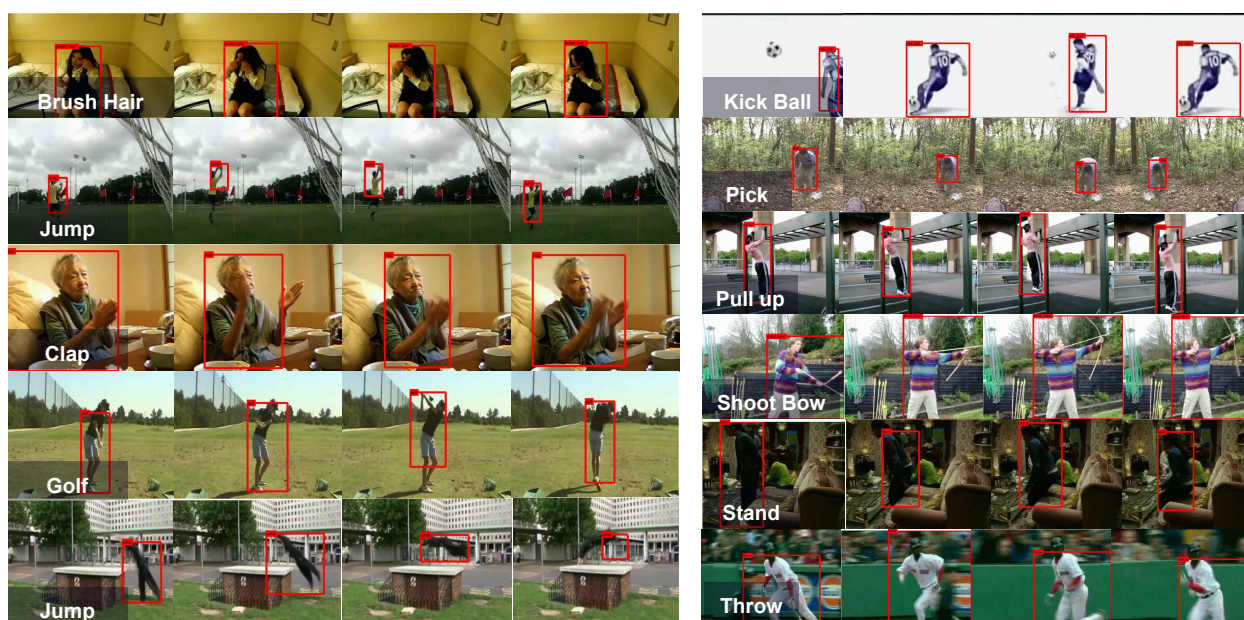


Figure 4.7: Examples from J-HMDB. Each block corresponds to a different video. We show the highest scoring action tube detected in the video. The red box indicates the region and the predicted label is overlaid. We show 4 frames from each video. The 2nd example on the left and the two bottom ones on the right are wrong predictions, with true labels being *catch*, *sit* and *run* respectively.

Chapter 5

Conclusion

In this thesis, we highlight the importance of contextual reasoning for visual recognition and propose models which rely on contextual cues in space as well as in time. The basis of our models is in all cases a convolutional neural network. We show the impact of contextual information for object detection, action recognition and attribute classification, bringing us one next closer to the idea recognition engine as shown in Figure 1.1.

Advances in computer vision and object recognition can impact many practical applications. For example, users should be able to query a system with a question and an image, e.g. “*Where can I buy this shirt?*” along with an image of a person wearing the shirt. The system should be able to parse the question and correctly identify and localize the shirt in order to retrieve the exact same one from a large database, such as the internet. Similarly, personalized recommendations can be improved with the help of computer vision. A user’s pictures in social media, such as Facebook, say a lot about that individual’s interests. A system that identifies users’ activities and hobbies through their images should be able to recommend events of interest. Health care and smart surveillance are another example where the use of computer vision can be impactful.

More importantly, visual recognition is a key component towards home robots. The goal of home robots is to assist humans in various indoor activities, such as cooking or cleaning. For instance, in the test case shown in Figure 5.1 the robot is being asked to fetch the mug. In order for the robot to complete the task successfully it needs to localize the mug in the scene.

Regarding home robots and personal assistance, teaching robots to perform new tasks is vital. One possible but rather impractical way would be to collect data and train models for each task at a time. However, imitation learning or learning by demonstration seem more promising. For example, there are a few acceptable ways to grasp a mug full of coffee or pass a knife to a person, while maintaining a safe environment for everybody involved. Observing people perform such actions and interact with others seem the best way to teach a robot the norms of acceptable motions.

Last but not least, the marriage of robotics and computer vision can lead to even greater scientific breakthrough far beyond deploying vision for robotic tasks. Today, visual recogni-

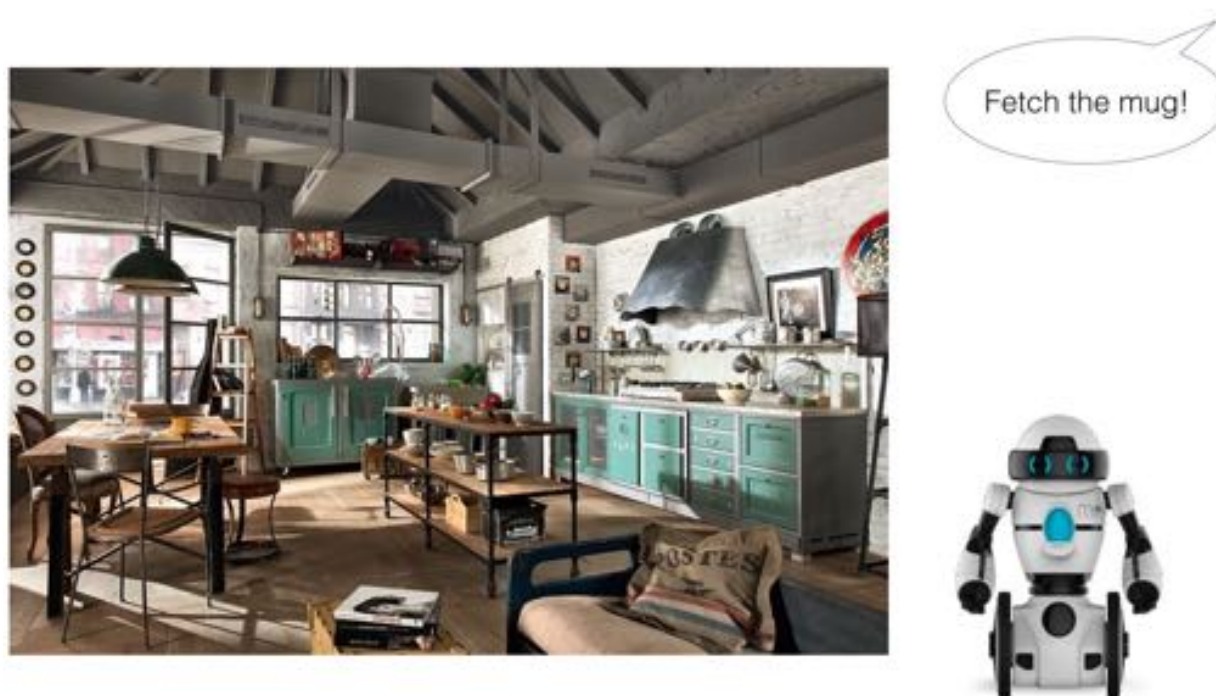


Figure 5.1: Towards home robots. Visual recognition is a key component towards home robots in order for robots to efficiently assist users.

tion relies heavily on a huge set of detailed annotated data, while evaluation occurs in the same domain as training. Moreover, systems learn by merely observing and analyzing data (*passive perception*). This is contrary to the way humans develop their visual understanding of the world. We, as humans, get to interact with the objects in front of us and we are able to learn new concepts with a few training instances. Robotics offers an environment that could support *active perception*. It might be through robotics, that we might achieve to bridge the gap between human and computational visual recognition.

Bibliography

- [1] Irving Biederman, Robert J. Mezzanotte, and Jan C. Rabinowitz. Scene perception detecting and judging objects undergoing relational violations. *Cognitive Psychology*, 1982.
- [2] Aude Oliva and Antonio Torralba. The role of context in object recognition. *Trends in cognitive sciences*, 2007.
- [3] Leonid Pishchulin, Mykhaylo Andriluka, and Bernt Schiele. Fine-grained activity recognition with holistic and pose based features. In *GCPR*, 2014.
- [4] Ross Girshick. Fast R-CNN. In *ICCV*, 2015.
- [5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.
- [6] Subhransu Maji, Lubomir Bourdev, and Jitendra Malik. Action recognition from a distributed representation of pose and appearance. In *CVPR*, 2011.
- [7] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.
- [8] Minh Hoai, Lubor Ladicky, and Andrew Zisserman. Action recognition from weak alignment of body parts. In *BMVC*, 2014.
- [9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [10] G. Gkioxari, R. Girshick, and J. Malik. Actions and attributes from wholes and parts. In *ICCV*, 2015.
- [11] Minh Hoai. Regularized max pooling for image categorization. In *BMVC*, 2014.
- [12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
- [13] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev. PANDA: Pose aligned networks for deep attribute modeling. In *CVPR*, 2014.

- [14] A. Torralba. Contextual priming for object detection. *IJCV*, 53(2), July 2003.
- [15] B. Russell, A. Torralba, K. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 2007.
- [16] R. Nevatia and T. Binford. Description and recognition of curved objects. *Artif. Intell.*, 1977.
- [17] M.A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *IEEE Trans. on Computer*, 22(1), 1973.
- [18] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 2005.
- [19] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989.
- [20] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.
- [21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. 2015.
- [22] D. Hoiem, A.A. Efros, and M. Hebert. Putting objects in perspective. *IJCV*, 80(1), October 2008.
- [23] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S.J. Belongie. Objects in context. In *ICCV*, 2007.
- [24] G Heitz and D Koller. Learning spatial context: Using stuff to find things. In *ECCV*, 2008.
- [25] J Yao, S Fidler, and R Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012.
- [26] Josep M. Gonfaus, Xavier Boix, Joost van de Weijer, Andrew D. Bagdanov, Joan Serrat, and Jordi Gonzalez. Harmony potentials for joint classification and segmentation. In *CVPR*, 2010.
- [27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
- [28] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

- [29] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.
- [30] Dumitru Erhan, Christian Szegedy, Alexander Toshev, , and Dragomir Anguelov. Scalable object detection using deep neural networks. In *CVPR*, 2014.
- [31] Yukun Zhu, Raquel Urtasun, Ruslan Salakhutdinov, and Sanja Fidler. segdeepm: Exploiting segmentation and context in deep neural networks for object detection. In *CVPR*, 2015.
- [32] 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*, 2014.
- [33] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition*, 2014.
- [34] G. Gkioxari, R. Girshick, and J. Malik. Contextual action recognition with r*cnn. 2015.
- [35] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. R-cnns for pose estimation and action detection. 2014.
- [36] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010.
- [37] N. Zhang, R. Farrell, F. Iandola, and T. Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. In *ICC*, 2013.
- [38] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *ECCV*, 2010.
- [39] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, 2009.
- [40] Georgia Gkioxari, Pablo Arbelaez, Lubomir Bourdev, and Jitendra Malik. Articulated pose estimation using discriminative armlet classifiers. In *CVPR*, 2013.
- [41] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. Using k-poselets for detecting people and localizing their keypoints. In *CVPR*, 2014.
- [42] Y. Wang, D. Tran, and Z. Liao. Learning hierarchical poselets for human parsing. In *CVPR*. IEEE, 2011.
- [43] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures-of-parts. *TPAMI*, 2012.

- [44] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. In *BMVC*, 2014.
- [45] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based R-CNNs for fine-grained category detection. In *ECCV*, 2014.
- [46] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 2013.
- [47] Bangpeng Yao, Aditya Khosla, and Li Fei-Fei. Combining randomization and discrimination for fine-grained image categorization. In *CVPR*, 2011.
- [48] A. Prest, C. Schmid, and V. Ferrari. Weakly supervised learning of interactions between humans and objects. *PAMI*, 2012.
- [49] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
- [50] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.
- [51] J. Deng, A. Berg, S. Satheesh, H Su, A. Khosla, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012). <http://www.image-net.org/challenges/LSVRC/2012/>.
- [52] Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Describing people: Poselet-based attribute classification. In *ICCV*, 2011.
- [53] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. *arXiv preprint arXiv:1409.5403*, 2014.
- [54] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*, 2010.
- [55] Rgis Vaillant, Christophe Monrocq, and Yann Le Cun. An original approach for the localization of objects in images. *IEE Proc on Vision, Image, and Signal Processing*, 1994.
- [56] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
- [57] P. Viola, J.C. Platt, and C. Zhang. Multiple instance boosting for object detection. In *NIPS*, 2005.
- [58] O. Maron and T. Lozano-Pérez. A framework for multiple instance learning. In *NIPS*, 1998.

- [59] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014.
- [60] Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas Guibas, and Li Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *ICCV*, 2011.
- [61] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.
- [62] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.
- [63] G. Gkioxari and J. Malik. Finding action tubes. 2015.
- [64] Melvyn A. Goodale and A. David. Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15:20–25, 1992.
- [65] J.K. Aggarwal and M.S. Ryoo. Human activity analysis: A review. *ACM Computing Surveys*, 2011.
- [66] R. Poppe. A survey on vision-based human action recognition. *Image Vision Computing*, 2010.
- [67] D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 2011.
- [68] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006.
- [69] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.
- [70] H. Wang, A. Kläser, C. Schmid, and C. Liu. Action recognition by dense trajectories. In *CVPR*, 2011.
- [71] M. Jain, H. Jegou, and P. Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013.
- [72] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *ICCV*, 2007.
- [73] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *ECCV*, 2010.
- [74] S. Ji, W. Hu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. In *PAMI*, 2013.

- [75] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.
- [76] A. A. Efros, A. C. Berg, G., and J. Malik. Recognizing action at a distance. In *ICCV*, 2003.
- [77] C. Schödl, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICPR*, 2004.
- [78] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *ICCV*, 2005.
- [79] A. Prest, V. Ferrari, and C. Schmid. Explicit modeling of human-object interactions in realistic videos. 2012.
- [80] T. Lan, Y. Wang, and G. Mori. Discriminative figure-centric models for joint action localization and recognition. In *ICCV*, 2011.
- [81] M. Raptis, I. Kokkinos, and S. Soatto. Discovering discriminative action parts from mid-level video representations. In *CVPR*, 2012.
- [82] Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. In *CVPR*, 2013.
- [83] A. Klaser, M. Marszalek, C. Schmid, and A. Zisserman. Human Focused Action Localization in Video. In *ECCV*, 2010.
- [84] M. Jain, J.v. Gemert, H. Jegou, P. Bouthemy, and C. G. M. Snoek. Action localization with tubelets from motion. In *CVPR*, 2014.
- [85] L. Wang, Y. Qiao, and X. Tang. Video action detection with relational dynamic-poselets. In *ECCV*, 2014.
- [86] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.
- [87] S. Mathe and C. Sminchisescu. Dynamic Eye Movement Datasets and Learned Saliency Models for Visual Action Recognition. In *ECCV*, 2012.
- [88] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.
- [89] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human action classes from videos in the wild. In *CRCV-TR-12-01*, 2012.
- [90] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004.

- [91] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008.
- [92] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. Black. Towards understanding action recognition. In *ICCV*, 2013.
- [93] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.