



# Learning visual models for person detection and action prediction

Tuan-Hung Vu

## ► To cite this version:

Tuan-Hung Vu. Learning visual models for person detection and action prediction. Computer Vision and Pattern Recognition [cs.CV]. Ecole Normale Supérieure de Paris - ENS Paris, 2018. English. NNT: . tel-01861455

**HAL Id: tel-01861455**

**<https://inria.hal.science/tel-01861455>**

Submitted on 24 Aug 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences Lettres  
PSL Research University

Préparée à l'École normale supérieure

## Learning visual models for person detection and action prediction

Apprentissage de modèles visuels pour la détection de personnes et la prédiction  
d'actions

**École doctorale n°386**

ÉCOLE DOCTORALE DE SCIENCES MATHÉMATIQUES DE PARIS CENTRE

**Spécialité** INFORMATIQUE

### COMPOSITION DU JURY :

M Patrick Pérez  
VALEO AI Research, Rapporteur

M Iasonas Kokkinos  
University College London  
Facebook AI Research, Rapporteur

M Ivan Laptev  
Inria Paris, Directeur de thèse

M Matthieu Cord  
Sorbonne University, Membre du Jury

M Josef Sivic  
Inria Paris, Membre du Jury

M Manmohan Chandraker  
University of California San Diego  
NEC Labs America, Membre du Jury

Soutenue par Tuan-Hung Vu  
le 21.09.2018

Dirigée par Ivan Laptev





If I have seen further it is by standing on  
the shoulders of giants.

---

—ISAAC NEWTON

Experience life in all possible ways —  
good-bad, bitter-sweet, dark-light,  
summer-winter. Experience all the  
dualities. Don't be afraid of experience,  
because the more experience you have,  
the more mature you become.

---

—OSHO





Dedicated to my family



# Abstract

In this thesis, we address person detection and action prediction in visual data. We develop models that learn representations for visual data and the structure in the output space while making use of contextual cues and temporal consistency. We also propose a predictive model to anticipate person’s attention in given static scenes.

In the first part of the thesis, we explore the strong association between scene categories and actions. Based on that understanding, we formulate a new task of predicting human actions in static scenes. To train and evaluate the proposed model, we collect a new dataset of scene-action associations, named SUN Action dataset. The success of this task enables potential applications such as affordance geo-localization.

The second part of the thesis is focused on person and generic object detection in videos. First, we construct contextual models to enhance person detection in individual frames. We train and evaluate our method on our new HollywoodHeads dataset with annotated human heads in movies. Our models consistently improve detection performance over baseline detectors. Second, we introduce a novel convolutional neural network architecture operating on short clips of frames to leverage temporal consistency and to learn spatio-temporal representations. By empirical experiments, we demonstrate the benefit of our spatio-temporal representations for object detection in videos. Last, we learn video representations that incorporate multiscale information on coarse time scales and design practical frameworks that achieve accuracy, efficiency and predictive power. Compared to per-frame features, our video representations show best detection improvement on frames degraded by fast motions.

**Keywords:** Scene understanding, action prediction, interaction analysis, spatio-temporal visual representation, object detection in videos, deep convolutional neural networks, representation learning.



# Acknowledgements

Parents, sister, wife

Supervisors: Ivan, Anton

Friends: Rafael, Guilhem, Vadim, Antoine, Suha, Minsu, Julia, JB, Gul, Andrei, others.

Thank sequoia and meleze for suffering tvu

Thank github for the codes

Thank coffee for all the energies

Thank Paris for the never-ending cold



# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Goals . . . . .	5
1.2	Contributions and outline . . . . .	9
<b>2</b>	<b>Related Work</b>	<b>13</b>
2.1	Action Recognition and Prediction . . . . .	13
2.1.1	Action Recognition . . . . .	13
2.1.2	Action Prediction. . . . .	15
2.2	Person and Object Detection . . . . .	16
2.2.1	Face Detection . . . . .	16
2.2.2	Early Object Detection Frameworks . . . . .	18
2.2.3	Object Detection using Convolutional Neural Networks . . . . .	18
<b>3</b>	<b>Background</b>	<b>23</b>
3.1	Support Vector Machines for Image Classification . . . . .	23
3.1.1	Image Representation . . . . .	23
3.1.2	SVM Classification . . . . .	25
3.2	Convolutional Neural Networks . . . . .	26
3.2.1	Architectures . . . . .	27
3.2.2	Object Detection with CNN . . . . .	29
	<b>Contributions Part I</b>	<b>35</b>
<b>4</b>	<b>Action Prediction in Images</b>	<b>37</b>
4.1	Introduction . . . . .	37
4.2	SUN Action dataset . . . . .	39
4.3	Analysis of scene-action correlation . . . . .	41
4.3.1	Predicting scenes from actions . . . . .	42
4.3.2	Action-based scene clustering . . . . .	45
4.4	Visual Action Prediction . . . . .	46
4.4.1	Implementation Details . . . . .	47
4.4.2	Experimental results . . . . .	47
4.5	Image-based Geo-Mapping of Actions . . . . .	50
4.6	Conclusion . . . . .	53



<b>Contributions Part II</b>	<b>55</b>
<b>5 Person detection with context-aware CNNs</b>	<b>57</b>
5.1 Context-aware CNN model . . . . .	59
5.1.1 Local model . . . . .	59
5.1.2 Global model . . . . .	60
5.1.3 Pairwise model . . . . .	61
5.2 Implementation details . . . . .	66
5.2.1 Local model . . . . .	66
5.2.2 Global model . . . . .	67
5.2.3 Pairwise model . . . . .	67
5.2.4 Combining models . . . . .	68
5.3 Datasets . . . . .	69
5.3.1 HollywoodHeads dataset . . . . .	69
5.3.2 TVHI dataset . . . . .	70
5.3.3 Casablanca dataset . . . . .	70
5.4 Experiments . . . . .	71
5.4.1 Results of context-aware models . . . . .	71
5.4.2 Comparison with the state-of-the-art methods . . . . .	71
5.4.3 Architectures of the Local model. . . . .	74
5.4.4 Size of the training set . . . . .	75
5.4.5 Complexity reduction with the Global model . . . . .	76
5.5 Conclusion . . . . .	77
<b>6 Tube-CNN: Modeling temporal evolution of appearance for object detection in video</b>	<b>79</b>
6.1 Tube-CNN for object detection . . . . .	81
6.1.1 Architecture . . . . .	82
6.1.2 Supervision . . . . .	83
6.2 Generating tube proposals . . . . .	83
6.2.1 Tube proposal network . . . . .	84
6.2.2 Tube proposals by tracking box proposals . . . . .	86
6.3 Experiments . . . . .	87
6.3.1 Datasets . . . . .	87
6.3.2 Training details . . . . .	88
6.3.3 Evaluation of tube proposals . . . . .	90
6.3.4 Detection results . . . . .	92
6.3.5 Ablation study . . . . .	95
6.4 Conclusion . . . . .	99
<b>7 Long-Term Representation Learning for Efficient Object Detection in Videos</b>	<b>101</b>
7.1 Feature propagation via memory networks . . . . .	104
7.1.1 Aggregating features over time . . . . .	104
7.1.2 Extending the temporal scale . . . . .	107

7.1.3	Discussion . . . . .	107
7.2	Detection, Propagation and Anticipation in Videos . . . . .	109
7.2.1	Object detection in videos . . . . .	109
7.2.2	Real-time detection by propagating strong features . . . . .	110
7.2.3	Anticipating features . . . . .	111
7.3	Experiments . . . . .	112
7.3.1	Object detection in videos . . . . .	113
7.3.2	Propagating and anticipating features . . . . .	117
7.3.3	Fast detection by propagating strong features . . . . .	120
7.4	Conclusions . . . . .	122
<b>8</b>	<b>Conclusion and Future Work</b>	<b>123</b>







# Chapter 1

## Introduction

Computer vision aims to automatically understand and interpret visual scenes around us. The field has come a long way since early attempts to address the problem [Papert, 1966]. Nowadays, many applications of computer vision have found their way out of research labs to numerous on-line services and consumer devices. Vision technologies are gradually shaping a new future where people are relieved from tedious, repetitive tasks. Moreover, intelligent devices are now able to provide human-like assistances in our daily activities. For example, face recognition has matured into a reliable security technology on smart phones and computers, replacing the passcode. Driver assistance systems have evolved from simple warnings and speed control, to the so called "Level 3" where human intervention is only required in rare difficult situations. Intelligent systems, like Amazon Echo Look<sup>1</sup>, can now be placed in the living room not only to answer questions but also to understand emotions.

People are arguably most important objects in images and video. Indeed, about 35% of pixels in movies and YouTube videos as well as about 25% of pixels in photographs belong to people [Laptev, 2013]. Not surprisingly, human-centric tasks usually serve as core components in many computer vision pipelines. Analyzing people in images and videos has many practical applications in security, entertainment, education and other domains. Airport security can benefit from face recognition to

---

1. <http://money.cnn.com/2017/04/26/technology/amazon-echo-look/index.html>

prevent attacks and violent behavior<sup>2</sup>. Video surveillance systems require reliable detection of malicious human activities like robbery, burglary or violence<sup>3</sup>. Aside from security purposes, person detection/tracking and action analysis can be used to assist sport coaches in planning strategies. Human gaze detectors in smart TVs can capture human attention and help generating targeted advertisements. Gaze detection and emotion recognition is also relevant for designing more efficient education schemes [Asteriadis et al., 2009, Rodrigo and Baker, 2009, D’Mello et al., 2012, Sümer et al., 2018]. Analyzing student behaviors and interactions gives more insight on how students teach each other which helps to optimize peer instruction [Jermann et al., 2011, Chong et al., 2017, Hayashi, 2018]. At a larger scale, crowd counting and crowd motion analysis are useful in scenarios such as demonstrations or mass evacuations.

Among a large range of human-related vision tasks, *Action Prediction* attempts to anticipate human behaviors based on available visual cues. Action prediction can be addressed in images [Delaitre et al., 2010, Yao and Fei-Fei, 2010a, Delaitre et al., 2011, Vu et al., 2014] or videos [Kitani et al., 2012, Walker et al., 2014], in settings that contain people [Ikizler et al., 2008, Ikizler-Cinbis et al., 2009, Ferrari et al., 2009, Yang et al., 2010] or without people [Vu et al., 2015]. Importantly, action prediction can provide reliable priors for other action analysis tasks such as action recognition or action localization. Besides helping to improve recognition performance, action priors are useful to detect abnormal behavior.

*Person Detection* is another human-centric computer vision task aiming to localize people in images and video. It often serves as a backbone for many other human analysis tasks, e.g. human verification, action recognition, behavior understanding, crowd counting and others. Given the needs of time-critical applications, the performance of person detectors is important both in terms of speed and accuracy [Viola and Jones, 2001, Dalal and Triggs, 2005, Girshick, 2015]. Similar to action prediction, the task can be addressed in the context of still images [Girshick, 2015, Vu et al., 2015, Ren et al., 2015] and videos [Zhu et al., 2017b].

---

2. <http://www.chinadaily.com.cn/a/201804/08/WS5ac9bfe9a3105cdcf6516b7b.html>

3. <https://www.theverge.com/2018/6/26/17479068/ai-guardman-security-camera-shoplifter-japan-automated-surveillance>

This thesis addresses action prediction and person detection in visual data. In the scope of these two problems, our study spans a range of topics from human intentions, interactions and appearance to visual recognition in the spatio-temporal domain. Below we identify the goals of the thesis in Section 1.1 and outline main contributions of the thesis in Section 1.2.

## 1.1 Goals

Our first goal is to study and leverage correlation between human intentions and static scenes. We formulate a prediction task and introduce a predictive model to anticipate human actions given natural scene images. Our second goal is to improve person detection by exploring spatial interactions of people in images and the spatio-temporal evolution of appearance in video. We also aim to improve the speed and accuracy of general object detection.

**Action Prediction in Static Scenes.** As illustrated in Figure 1-1, scenes are frequently correlated with typical actions. Action prediction in static scenes could therefore provide scene-specific priors when recognizing human actions. In this work, we want to explore and make use of the strong action-scene associations using statistical methods. Based on such associations, we address a new task of predicting human actions for images of static scenes. The success of this task is expected to improve general scene understanding, reasoning about affordance and action recognition. This work also enables potential applications such as affordance geo-localization.

**Person Detection in Videos.** Understanding complex visual scenarios like movies is a challenging task where person detection plays an important role. Standard detection approaches often perform local search on the object hypothesis space, generated by techniques like exhaustive sliding windows [Fleuret and Geman, Amit et al., 2004, Dalal and Triggs, 2005, Lampert et al., 2008, Dollár et al., 2010, Kokkinos, 2011, Felzenszwalb et al., 2010a] or heuristic object proposals [Uijlings et al., 2013, Krähenbühl and Koltun, 2014, Zitnick and Dollár, 2014, Cheng et al., 2014, Girshick et al.,



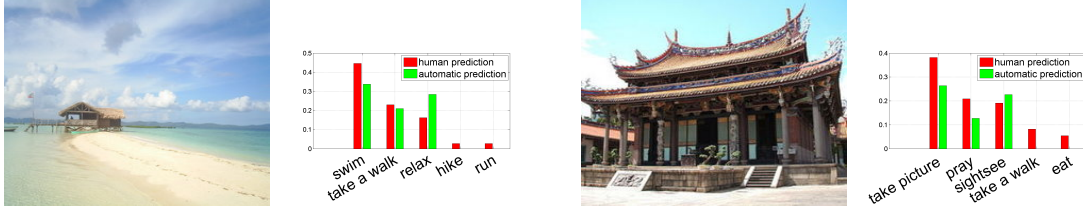


Figure 1-1 – Illustration of scene-action correlation [Vu et al., 2014]: Images of scene classes `sandbar` and `temple_east_asia` from the SUN dataset [Xiao et al., 2010] together with probabilities for the five most likely actions, predicted manually by people (red) and by an automatic method (green).

2014]. Recent advances in object detection are strongly related to the progress in learning deep visual representations, i.e. features, extracted from deep convolutional neural networks. Even equipped with powerful features, state-of-the-art detection methods are still behind human-level performance. We argue that a visual scene is not a random composition of objects, therefore independently recognizing objects while ignoring the whole context is sub-optimal. Indeed, there exists an underlying latent structure that bind things together to make the scene "look-real" under human perception. Such an argument becomes even stronger in the case of videos where the temporal continuity is also a critical recognition cue [Ellis, 1938].

We aim at modeling human dynamics in movies in both spatial and spatio-temporal domains. Detection in videos is the testbed for our models where we demonstrate performance improvement over state-of-the-art baselines. In the spatial domains, i.e. individual frames, we try to exploit human-specific cues, e.g. human interactions, as additional evidence for detecting people. In the spatio-temporal domain, i.e. videos, we design a framework that can efficiently make use of temporal consistency to improve detection performance.

In the first line of work, we exploit *structural contexts* to enhance person head detection on single frames. Two types of structural context are considered: holistic information and interactions between people. First, we observe that head locations can be predicted from other image parts. In particular, other parts of the human body and the rest of the scene constrain locations of heads in the video frame. In addition, interactions between people put constraints on the relative locations and



(a) Predicting person head position and scale from holistic information. Columns from the left to the right correspond to the original image and the output of the context model at different resolutions. Red color corresponds to cells with high scores for the “head” class, blue color indicates cells with low scores for the “head” class.



(b) Filtering head hypotheses with the pairwise interaction cues. Yellow boxes correspond to correct detections, red – to false positives. Pairwise links are plotted with either yellow (attractive) or red (repulsive).

Figure 1-2 – Illustration of contextual models [Vu et al., 2015].

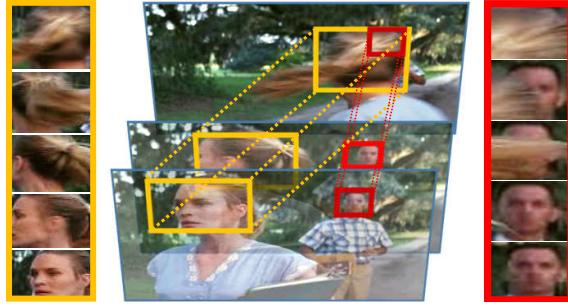


Figure 1-3 – Space-time object tubes used in our work to capture evolution of object appearance over short sequences of video frames.

appearance of heads. For example, heads appear differently while people are talking and kissing. We exploit such constraints by explicitly modeling relations between pairs of people. Figure 1-2 illustrates examples of the two contextual cues.

In the second line of work, we build a temporal model for changes of head appearance in consecutive video frames. Our model can be applied to a more generic case of detecting multiple object classes. Unlike images, video provides additional cues which can help to disambiguate the detection problem. For example, sequences

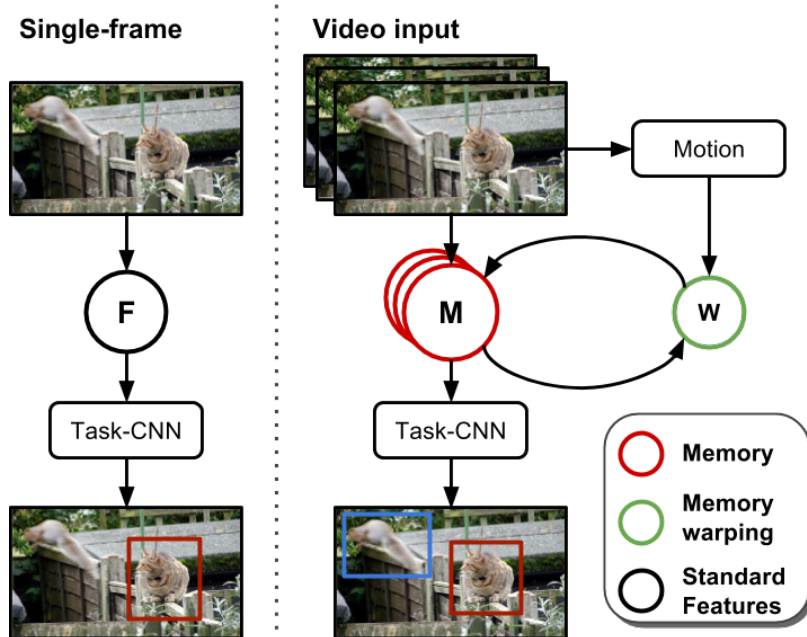


Figure 1-4 – A schematic comparison between a typical per-frame method (left) and the memory-based video representation learning approach (right) for object detection in videos.

with objects undergoing dynamic occlusions and rotations are structured and contain space-time patterns with object-specific properties, see Figure 1-3. In this work, we argue that dynamic changes of appearance provide discriminative information that can improve the detection performance. Our goal is to learn discriminative models for the temporal evolution of object appearance and to use such models for detection. The learned spatio-temporal features are robust in difficult cases, for example, limited lighting, fast motions and occlusions.

In the third line of work, we learn video representations that incorporate multi-scale information on longer time horizons and design practical frameworks that achieve accuracy, efficiency and predictive power. Analogous to the Gestalt principle of common fate [Wertheimer, 1938, Ellis, 1938], we hypothesize that temporal coherence by accounting for motion across frames allows learning powerful representations while achieving greater invariance to blur, lighting, pose and occlusions. Our goal is to design a memory-based causal framework that can encapsulate video representations over time. At each time step, memories in the past can be propagated and aggre-

gated to enhance the current video feature, improving object detection in videos, see Figure 1-4. Compared to per-frame features, our video representations show best detection improvement on frames degraded by fast motions.

## 1.2 Contributions and outline

The rest of the thesis is organized in seven chapters. In Chapter 2, we review related work. We then present prior methods and models that are used in this thesis in Chapter 3. The remaining chapters are split into two parts corresponding to our two main goals as follows:

**Action Prediction in Static Scenes.** In Chapter 4, we present the following three contributions. First, we introduce a new dataset with manual annotations of typical actions for 397 scene classes and use it to analyze action-scene correlations. Second, based on the discovered correlations, we demonstrate successful action prediction for images of static scenes. Finally, we propose a new task of geo-localized action prediction. We apply our method to geo-tagged images on the web and show encouraging results of searching maps for locations suitable for particular activities. This work corresponds to the publication presented in ECCV 2014 [Vu et al., 2014].

### **Person Detection in Videos.**

- Chapter 5 presents our use of two contextual cues to improve person detection in videos. We build on a state-of-the-art CNN model for object detection and extend it to contextual reasoning. First, we leverage person-scene relations and propose a Global CNN model trained to predict positions and scales of heads directly from a full image. Second, we explicitly model pairwise relations among objects and train a Pairwise CNN model using a structured-output surrogate loss. To train and test our model, we introduce a new large dataset with 369,846 human heads annotated in 224,740 video frames from 21 movies. We show the importance of our large dataset for training and evaluate our method on the

new and two existing datasets. The results demonstrate improvements of the proposed contextual CNN model compared to other recent baselines including R-CNN [Girshick et al., 2014] on all three datasets. We also demonstrate a speed-up of object detection provided by our Global model. Our new dataset and the code are publicly available from the project web-page<sup>4</sup>. This work was presented at ICCV 2015 [Vu et al., 2015].

- To model the temporal evolution of appearance, we introduce space-time *tubes* corresponding to sequences of object bounding boxes in consecutive frames. In Chapter 6, we propose two CNN architectures for generating and classifying tubes, respectively. Our tube proposal network (TPN) first generates a large number of spatio-temporal tube proposals maximizing the objectness score and guaranteeing high recall for the ground truth object tubes. The Tube-CNN implements a tube-level object detector in the video using tube proposals as input. Our tube models show notable improvement in comparison with frame-level detector baselines on HollywoodHeads, a large-scale person detection dataset. We achieve comparable results with the winner of ImageNet VID challenge of object detection in videos [Kang et al., 2016a]. On the Youtube-Objects (YTO) dataset [Prest et al., 2012], a weakly annotated dataset closely related to ImageNet VID dataset, our framework surpasses state-of-the-art localization performance by a large margin.
- In Chapter 7, we propose a novel network structure, termed MemNet, that encodes a memory of the feature representation, which is updated at every frame based on image observations and warped from one frame to the next to account for observer and scene motions. In order to encapsulate a long-term video representation, we introduce a hierarchical architecture, named ClockNet, operating on multiple temporal scales with minimal memory consumption. Compared to baseline methods on object detection in videos, our proposed methods achieve state-of-the-art results on ImageNet VID dataset, while running at faster speed. In addition, our networks can propagate memories over time, which leads to the

---

4. <http://www.di.ens.fr/willow/research/headetection>

two potential applications: (i) fast object detection with two-stream architecture on multiple GPUs and (ii) feature anticipation.

Chapter 8 concludes the thesis and discusses some possible future works.



# Chapter 2

## Related Work

In this chapter we review previous works that is closely related to this thesis. The chapter is comprised of two sections: Section 2.1 discusses some works on action recognition and action prediction; Section 2.2 gives a brief overview of person and object detection methods.

### 2.1 Action Recognition and Prediction

We give a brief overview of the literature related to our problem of action prediction in static scenes. The section is organized in two parts: action recognition and action prediction.

#### 2.1.1 Action Recognition

There is a large body of research addressing human action recognition in videos. Early methods mainly involve human body parts tracking [Rohr, 1994] and human motion analysis [Bobick and Davis, 2001, Efros et al., 2003]. Follow up methods focused on statistical representations for action recognition. Laptev [2005] represented motion patterns with space-time local features. The idea is to localize spatio-temporal interest points corresponding to characteristic events. Using such interest points, Bag-of-Words approach has been used to represent actions in the video. Schuldt et al.



[2004], [Laptev et al. \[2008\]](#) classified actions by applying Support Vector Machines (SVMs) on the occurrence histograms. [Wang et al. \[2011\]](#) proposed an action recognition framework with dense trajectory descriptors. Feature points are first localized and then tracked with optical flow to densely produce point trajectories. Each trajectory is represented by descriptors, e.g. HOG, HOF and MOH, within its neighborhood space-time volume. Action recognition is performed with the standard bag-of-features approach.

Deep convolutional neural networks have been applied for action recognition. [Simonyan and Zisserman \[2014\]](#) designed a two-stream architecture separately processing RGB images and optical flows. Late fusion is applied on the  $L_2$ -normalized softmax outputs of the two streams. The network achieved comparable performance with state-of-the-art methods using "hand-crafted" features. Despite relatively small improvements, this work showed promising potential of CNNs for action recognition. More recently, [Tran et al. \[2015\]](#) introduces C3D, a 3D convolutional neural networks for action recognition. C3D architecture extends 2D CNNs to videos. The learned C3D features computed from RGB input have been used for video representation, followed by SVM for action classification. [Varol et al. \[2016\]](#) extended C3D to learn long-term video representation and confirmed the advantage of using optical flows for human action recognition. Like how CNN models for recognition tasks on images benefit from the pretraining phase on the ImageNet dataset, CNN models for videos considerably benefit in pretraining on big datasets such as Sport-1M [[Karpathy et al., 2014](#)] and Kinetics [[Kay et al., 2017](#)]. The "Two-Stream Inflated 3D ConvNets" (I3D) extends state-of-the-art architectures on image classification to handle spatio-temporal 3D information in videos. I3D models pretrained on the Kinetics dataset and finetuned on HMDB-51 [[Kuehne et al., 2011](#)] and UCF-101 [[Soomro et al., 2012](#)] datasets achieve state-of-the-art performance on the both action recognition benchmarks. In general, training CNNs for videos is a challenging task due to the difficulty to collect data annotation and high memory consumption of the deep networks.

Action recognition in stills images received less attention compared to videos. The work of [Ikizler et al. \[2008\]](#) was one of the first attempts to recognize actions in static

images using human poses. The authors argued that poses often characterize actions; so one can extract and classify poses to derive action labels of images. Yao and Fei-Fei [2010a,b] have shown how an explicit model of human-object interaction can also benefit action recognition in images. Human-object interactions are represented by stochastic AND/OR structures. Training is done with a modified Apriori mining algorithm. Delaitre et al. [2010] classified Bag-of-Features representations to action categories. In that work, the authors also explore the use of latent SVM (LSVM) to learn deformable part-based representations of actions. The combination of the two types of action representation showed significant improvement in action recognition performance in still images.

### 2.1.2 Action Prediction.

Recognition of functional properties of objects and scenes is an interesting but less explored area of computer vision. Action prediction has been addressed in [Kitani et al., 2012] and Walker [Walker et al., 2014], where the authors want to model future motion of people and cars using priors derived from the scene. Yuen and Torralba [2010] predict motion for images of static scenes by searching and transferring motion cues from video scenes with similar appearance. Relations between people and objects as well as between human poses and scene geometry have been investigated in Grabner et al. [2011], Gupta et al. [2011], Delaitre et al. [2012]. Patterson and Hays [2012] annotate scene images with a set of global attributes of various types (i.e: material, surface property, affordance and spatial envelope), and recognize attributes from scene images. Unlike any previous work, our work in Chapter 4 aims to model functional properties for a wide range of scene classes. Our work is similar in spirit to [Arietta et al., 2014] and [Khosla et al., 2014] who aim to predict non-observed scene properties such as crime rate in the area.

Follow-up works show an increasing interest for human intentions prediction in complex settings. Koppula and Saxena [2016] model human activities and object affordances in the past using conditional random field (CRF), and show that action anticipation can improve performance of event detection. Alahi et al. [2016] pro-

pose a Long short-term memory network (LSTM) to model human movement and predict their future trajectories in crowded scenes. [Yoo et al. \[2016\]](#) introduces an unsupervised Hierarchical Topic-Gaussian Mixture Model to learn moving dynamics of co-occurring objects for path prediction in a scene that includes crowded moving objects. In [\[Gupta et al., 2018\]](#), the authors propose Social-GAN combining a recurrent sequence-to-sequence model and generative adversarial networks (GANs) to predict socially plausible human trajectories in crowded scenes.

## 2.2 Person and Object Detection

Person is an important object class for many computer vision tasks. Indeed, early detection frameworks mostly focused on finding human faces or bodies [\[Fischler and Elschlager, 1973, Papageorgiou and Poggio, 2000, Viola and Jones, 2001, Dalal and Triggs, 2005\]](#). Localizing people plays an important role in higher-level tasks such as behavior and interaction understanding. In this section, we review previous work on person and object detection in videos. We start by giving an overview of some early face detection methods. We continue by reviewing more general approaches that could detect human bodies as well as other object classes. Most of this section is dedicated to the related literature of object detection built upon deep convolutional neural networks, more details of which will be given in Chapter 3.

### 2.2.1 Face Detection

During the last decades, many methods have been focused on detecting human faces in images and videos. We review three categories of face detection methods: low-level features analysis, template matching and appearance feature learning.

**Low-level feature analysis.** Edge and color are two common low-level features used in face detection. Edge-detection-based approaches [\[Govindaraju, 1996, Sirohey, 1998\]](#) try to assign labels, e.g. left/right sides, to edges then fit them to a face model using the golden ratio or elliptic hough transform. Edges were extracted by

classical techniques such as Canny filters [Canny, 1987]. When it comes to color, most approaches focus on localizing skin areas, which most-likely belongs to human parts like faces. On color spaces like HSV, YIQ and YCrCb, skin-color pixels are tightly clustered. Some methods simply set a threshold on color histogram to determine skin-color [Sobottka and Pitas, 1996, Cai and Goshtasby, 1999, Yoo and Oh, 1999]. More complex methods use statistical models, e.g. mixture of Gaussians, to parameterize skin-color clusters [Yang and Waibel, 1996, McKenna et al., 1998, Oliver et al., 2000]. In general, low-level-feature-based detection methods are susceptible to illumination change and noise.

**Template matching.** In template-matching-based methods, a template, i.e. face pattern, is manually predefined [Sakai et al., 1969, Craw et al., 1987, 1992] or parameterized [Yuille et al., 1992, Lanitis et al., 1995, da Vitoria Lobo and Kwon, 1998, Cootes et al., 2001]. Object of interest, i.e. human faces, is determined based on the correlation scores between the template and local regions in the image. Correlation score represents the likelihood of having face appearance on the local region.

**Learning methods.** Such methods use statistical analysis and machine learning to learn face appearance feature [Kirby and Sirovich, 1990, Turk and Pentland, 1991], instead of manually designing face templates. Most popular techniques are Eigen-faces [Turk and Pentland, 1991], Support Vector Machine [Osuna et al., 1997, Papa-georgiou et al., 1998], Naive Bayes Classifier [Schneiderman and Kanade, 1998, 2000] and Hidden Markov Model [Samaria and Young, 1994, Nefian and Hayes, 1998]. Feature learning with Neural Networks [Feraund et al., 2001], or Deep Convolutional Neural Network [Girshick et al., 2014, Girshick, 2015, Ren et al., 2015], also falls into this category. Such approaches often require a large amount of annotated training data to achieve good performance.

### 2.2.2 Early Object Detection Frameworks

Here we extend our review to more general approaches that can detect arbitrary objects. Viola-Jones [Viola and Jones, 2001] is an efficient detection framework exploiting a cascading architecture with multiple stages, built upon fast Haar features and Adaboost. The framework works well for rigid objects which appearance variants are low, e.g. frontal faces.

In cases of non-rigid deformations and large variations in appearance due to changing viewpoints or intra-class variability, more complicated frameworks using pictorial structures [Fischler and Elschlager, 1973, Felzenszwalb and Huttenlocher, 2005] like Deformable Part Models (DPM) [Felzenszwalb et al., 2010a] are more effective. DPM considers object as a combination of parts in a pre-defined star-like structure. Detection and parts discovery are done simultaneously by solving a joint optimization problem. DPM’s main bottleneck is the speed. Follow-up works accelerate DPM using different techniques such as cascade [Felzenszwalb et al., 2010b], coarse-to-fine [Pedersoli et al., 2011], branch-and-bound [Kokkinos, 2011], Fast Fourier Transform (FFT) [Dubout and Fleuret, 2012]

### 2.2.3 Object Detection using Convolutional Neural Networks

Prior to 2011, most computer vision models relied on hand-crafted features, e.g. Haar, HOG, SIFT. Since 2011, the success of deep convolutional neural networks (CNNs) [Krizhevsky et al., 2012] has opened a new epoch for computer vision. Combining the massive amounts of data and the powerful computing hardwares, i.e. GPUs, CNNs are able to learn strong appearance features that can be well adapted to multiple vision tasks, including object detection. In the following, we give an overview of the most recent object detection frameworks that are built on Convolutional Neural Networks.

**Object Detection in Images.** R-CNN [Girshick et al., 2014] presents a successful object detector based on recent CNN features. The main difference between R-CNN

and traditional frameworks, e.g. Viola-Jones, is the use of CNN features instead of hand-crafted ones. In the original work, the authors used CNN features produced by AlexNet [Krizhevsky et al., 2012] pretrained on image classification task. To reduce the search space, R-CNN uses object proposals generated by Selective Search [Uijlings et al., 2013]. Each regional proposal is cropped, resized and forwarded through a CNN to produce an appearance feature. The pipeline ends with SVM classifiers, taking the CNN feature as input and producing object scores. With a relatively simple design, R-CNN outperformed other contemporary methods by a large margin.

The limitation of R-CNN is its heavy computations and high storage-demand. Inspired by the idea of the Spatial Pyramid Pooling network (SPPnet) [He et al., 2014], Fast-RCNN [Girshick, 2015] improves R-CNN speed and performance with the efficient Region-Of-Interest (ROI) pooling layer. Features of region proposals are extracted from a shared feature map. Moreover, the SVM classifier is replaced by a neural network, which not only removes the high storage-demand but also unifies the detection pipeline.

In the following work, Faster-RCNN [Ren et al., 2015] improves Fast-RCNN speed and performance with the Region Proposal Networks (RPN). The RPN replaces the only non-CNN component in the fast-RCNN pipeline, i.e. region proposal generator, and results in the end-to-end CNN-based detection pipeline. Compared to Fast-RCNN, Faster-RCNN is both faster and more accurate.

All aforementioned methods use image features produced by pre-trained CNNs on image classification task. Detection performance thus heavily depends on the quality of features generated by the base-networks. In practice, the stronger the base-network is, the better performance we get, i.e. Faster-RCNN based on ResNet-101 [He et al., 2016] outperforms Faster-RCNN based on AlexNet [Krizhevsky et al., 2012] by a large margin. R-CNN, Fast-RCNN and Faster-RCNN are designed in such a way that we can easily ‘upgrade’ the feature extraction part. However, when it comes to Residual Networks (ResNet) [He et al., 2016], such a simple modification is suboptimal. In [Dai et al., 2016], the authors argue that detection frameworks based on ResNet should be fully-convolutional in order to achieve the optimal performance. R-FCN is designed

in such a spirit by using Position-Sensitive ROI pooling layer. ROI pooling is no more a class-agnostic operation but a class- and position-dependent operation. That modification allows a fully-convolutional design for the object classification part of the R-FCN framework. R-FCN is later improved with deformable convolution and deformable pooling layers in [Dai et al., 2017]. Deformable R-FCN learns where to aggregate and pool information in the image and demonstrates better robustness to variations in scale and rotation. Deformable R-FCN currently shows best performance for object detection in still images on standard benchmarks.

While performance of recent object detection methods is getting close to human-level, such methods often require expensive computational power. For real-time applications, along side with performance, speed is an important factor. Works like Single-Shot-Detector (SSD) [Liu et al., 2016] and You-Only-Look-Once (YOLO) [Redmon et al., 2016, Redmon and Farhadi, 2017, 2018] endeavour to reduce detection time without losing much performance.

**Object Detection in Videos** Object detection in video is closely related to object detection in images, as one can apply image object detectors to each frame of a video. Video, however, provides additional cues, such as temporal continuity. A common practice for video object detection is to post-process detections by means of a tracker [Bojanowski et al., 2013, Ordonez et al., 2011, Everingham et al., 2006, Pirsiavash et al., 2011]. The tracker first links high-confident detections on consecutive frames into tracks. As object detector often produces unstable scores along an object track, the linking step helps to find missing detections. An additional smoothing step could be done to stabilize object position on the track. Recently, Kang *et al.* [Kang et al., 2016b] use this practice to generate object tracks. In their work, the tracker output is named *tube proposal*. It then becomes input for the next post-processing steps including box-perturbation, max pooling and rescore. To avoid possible confusion, we stress that the term *tube proposal* is later used in Chapter 6 with a different meaning. Unlike other video detection methods [Pirsiavash et al., 2011, Tang et al., 2015], which separate appearance and motion processing phases, our models treat

video signal as an entire combination of the two factors.

In [Kang et al., 2017], the authors propose a Tubelet Proposal Networks for generating tubelet proposals, which are then scored by an encoder-decoder CNN-LSTM. Regression is done on every frame to adjust box location along the tubelet. The authors show that their proposal network performs best with temporal window size of 5 frames.

Object detection in video and *tracking-by-detection* have a tight relation. Graph-based methods [Pirsiavash et al., 2011, Tang et al., 2015] formulate object tracking as a global optimization problem on association graph. The association graph is constructed by measuring appearance similarity between objects. In [Leal-Taixé et al., 2016], the authors propose a Siamese CNN to learn such appearance similarity efficiently.

**Video representation learning for object detection in videos.** Learning representations for videos has been a long-standing goal in computer vision and many directions have been explored. Donahue *et al.* [Donahue et al., 2015] or Srivastava *et al.* [Srivastava et al., 2015] rely on recurrent neural networks (RNNs) like LSTMs [Hochreiter and Schmidhuber, 1997] to propagate feature representation from still frames over time. However, unlike our approach, features are propagated without explicit knowledge of motion in the scene. 3D CNNs provide more freedom to learn motion-specific kernels (although motion is also not explicitly used) and were successfully used in [Tran et al., 2015, Yao et al., 2015] for tasks like video captioning or action recognition, but come with considerably more parameters to learn and typically more computational costs. Recent works have also considered unsupervised learning with pretext tasks [Wang and Gupta, 2015], however, we focus on efficient learning of supervised representations.

Two-stream architectures like [Simonyan and Zisserman, 2014, Feichtenhofer et al., 2016, 2017a] combine features extracted from both images and motion (optical flow) to boost the representational power. While 3D CNNs learn motion-specific features implicitly, two-stream architectures explicitly take optical flow as input. In both cases,



however, this information is not used to transform features over time to compensate for observer and scene motions.

The recently proposed flow-guided feature aggregation framework (FGFA) [Zhu et al., 2017a,b], on the other hand, explicitly warps convolutional feature maps between frames for better alignment when aggregating them. The warping function is triggered by a learned displacement field initialized with FlowNet [Dosovitskiy et al., 2015]. FGFA [Zhu et al., 2017b] uses a fixed temporal window of nearby frames, requires as many warp computations as the length of the window, and compromises causality, *i.e.*, integrates features from future frames.

Object detection in videos has recently attracted much interest, partly due to the introduction of the ImageNet VID challenge [Russakovsky et al., 2015]. Besides [Feichtenhofer et al., 2017b, Zhu et al., 2017b] with a focus on the feature level, most recent approaches for detection in video leverage the temporal data on a higher level, *e.g.*, by tracking objects or specifically designed post-processing mechanisms [Han et al., 2016, Kang et al., 2016b], which are orthogonal to contributions in this thesis. Besides object detection, the proposed feature representations can also be used for other tasks that operate on videos like semantic segmentation [Kundu et al., 2016, Shelhamer et al., 2016] or action recognition [Feichtenhofer et al., 2016, 2017a, Simonyan and Zisserman, 2014]. Visual anticipation has also been shown useful for the tasks of segmentation [Luc et al., 2017] and action recognition [Vondrick et al., 2016]. Our work targets object detection and explicitly accounts for motion in the learned representation.

# Chapter 3

## Background

This chapter introduces terminology and methods that will be used in the rest of the thesis. In Section 3.1 we review SVM-based image classification used for action prediction in Chapter 4. Section 3.2 gives an overview of CNN architectures used in our work for object detection in Chapters 5-7.

### 3.1 Support Vector Machines for Image Classification

*Image classification* is a task of assigning input images with one or more labels from a given set of categories. A traditional approach to this problem includes a construction of image representation in terms of image features followed by a classifier. In this section, we first outline two common pipelines for image representation: Bag of visual Words (BoW) and Fisher Vector (FV) in Section 3.1.1. We then revisit Support Vector Machines (SVM), one of the most common classification technique in Section 3.1.2.

#### 3.1.1 Image Representation

A common image representation pipeline is composed of three steps: extracting local image descriptors, learning a visual dictionary and encoding descriptors. In

the first step, local descriptors like SIFT [Lowe, 2004], CSIFT [Abdel-Hakim and Farag, 2006] and HOG [Dalal and Triggs, 2005] are extracted from the input image. In practice, we often use a dimensionality reduction technique like PCA to reduce the size of those descriptive vectors. This step helps decreasing the computational complexity as well as removing noises. Bag of visual words (BoW) [Fei-Fei and Perona, 2005, Avila et al., 2013] and Fisher Vector (FV) [Perronnin and Dance, 2007, Sanchez et al., 2013] pipelines are different at dictionary learning and feature encoding steps. While BoW uses K-means clustering and Histogram Encoding, FV uses Gaussian Mixture Models and Fisher Encoding.

**Quantization.** The main goal of quantization is to construct a base for descriptors, usually named as *visual dictionary*. A visual dictionary is built by grouping descriptors into clusters, called *visual words*. Two most used clustering techniques are K-mean and Gaussian Mixture Models (GMM).

**Encoding.** Extracted descriptors are encoded into a single feature vector. The two common encoding techniques are Histogram Encoding and Fisher Encoding.

*Histogram Encoding.* Assumed that we already constructed a visual dictionary with K-means, consisting of  $K$  visual centers  $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K$ . On a given image,  $n$  local descriptors  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$  are densely extracted and are assigned to their closest centers. Considering  $s^{\text{th}}$  descriptor, we have the  $s^{\text{th}}$  assignment determined as:  $a_s = \text{argmin}_{i \in \mathbf{T}} d(\mathbf{x}_s, \boldsymbol{\mu}_i)$  with  $d(\cdot)$  is the distance function. Finally, the set of descriptors is encoded into a histogram vector  $h \in \mathbb{R}^K$ , with:  $h_i = |\{s \mid a_s = i\}|$

*Fisher Encoding.* Fisher Vector (FV) is a popular encoding technique for scene classification [Perronnin and Dance, 2007]. FV is the gradients of the log-likelihood function  $\log P(\mathbf{X}|\boldsymbol{\theta})$  w.r.t the GMM parameters  $\boldsymbol{\theta} = (\boldsymbol{\pi}_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\pi}_K, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K)$ . In other words, FV stands for the direction to which the parameters should change to move the input descriptor toward the generative model's modes. Because FV is the concatenation of different types of gradients, a normalization step is often needed. One way is using the Fisher information matrix  $F_{\boldsymbol{\theta}} = E_{\mathbf{X}}[\log P(\mathbf{X}|\boldsymbol{\theta}) \log P(\mathbf{X}|\boldsymbol{\theta})']$ .

The normalized gradient is then  $F_{\theta}^{-1/2} \nabla_{\theta} \log P(\mathbf{X}|\theta)$ . In practice, to achieve better performance, there are other good practices such as  $l_2$  and power normalizations [Peronin et al., 2010].

### 3.1.2 SVM Classification

Support Vector Machines (SVM) is a supervised machine learning technique which is used in classification and regression tasks. In this part, we revisit the use of SVM in binary classification. In a nutshell, SVM is a discriminative classifier trying to separate data points lying on a  $p$ -dimensional feature space with an optimal  $(p-1)$ -dimensional hyperplane. The optimal hyperplane maximizes the distances, called *margins*, between nearest data points of either class and the hyperplane. Originated from the work of Cortes and Vapnik [1995], SVM is formulated as a linear classifier. To solve nonlinear tasks, we can use the *kernel trick* [Boser et al., 1992].

**Linear SVM.** We formulate the binary classification problem as follows: Given a dataset  $\mathcal{D} = \{x_i, y_i | i = 1..N\}$  with data points  $x_i \in \mathbb{R}^M$  and labels  $y_i \in \{1, -1\}$ . The SVM classifier is written as:  $h(x) = \text{sign}(w^T x + b)$ . In this way, the hyperplane is characterized by  $\theta = \begin{bmatrix} w \\ b \end{bmatrix}$  with  $w \in \mathbb{R}^M$  and  $b \in \mathbb{R}$ . Effectively, the line  $\theta^T \begin{bmatrix} x & 1 \end{bmatrix} = 0$  is the decision boundary. Our goal is to find  $\theta^*$  that maximizes the margin. *Slack* variable  $\xi$  is often added to relax the linear separability constraint and make the algorithm less susceptible to outliers. The optimization problem is formulated as:

$$\begin{aligned} \min_{w, b} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, N \\ & \xi_i \geq 0, \quad i = 1, \dots, N. \end{aligned}$$

**Kernel SVM.** The dual form of SVM optimization problem is given as follows:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \dots, N \\ & \sum_{i=1}^N \alpha_i y_i = 0. \end{aligned}$$

The important point of using the dual form is that the optimization objective can be written entirely in terms of the inner products  $\langle x_i, x_j \rangle$ . Let us assume that there exist a feature mapping function  $\varphi$  which maps the data points in the original vector space to another vector space on which data can be linearly separated. The *trick* here is: instead of explicitly formulating  $\varphi$ , which is a complicated and expensive task, we only have to define the inner product of the mapped features on the target vector space. Effectively, we define kernel  $K : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$  such that  $K(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$ . Applying kernel trick to SVM is simply replacing  $\langle x_i, x_j \rangle$  by  $K(x_i, x_j)$  everywhere.

Intuitively,  $K(x_i, x_j)$  can be viewed as dissimilarity, or distance, between the mapped features  $\varphi(x_i)$  and  $\varphi(x_j)$  on the target feature space. In such a way, we can choose a reasonable kernel given a particular classification or regression problem. Some most common kernel functions used in Computer Vision are Gaussian kernel:  $K(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$ ,  $\chi^2$ -kernel:  $K(x_i, x_j) = \exp(-\alpha \sum_{m=1}^M \frac{(x_i^m - x_j^m)^2}{x_i^m + x_j^m})$  and Histogram Intersection kernel:  $K(x_i, x_j) = \sum_{m=1}^M \min(x_i^m, y_i^m)$

## 3.2 Convolutional Neural Networks

A *deep neural network* (DNN) is a feed-forward artificial neural networks composed of input and output layers, hidden layers and nonlinear activation functions. There are many types of nonlinear activation functions, e.g. Sigmoid, Tanh, Softmax, RELU [Nair and Hinton, 2010], etc.. A *convolutional neural networks* (CNN or ConvNet) is a DNN equipped with additional layers dedicated for processing numerical images, i.e. convolutional and pooling layers [LeCun et al., 1998]. A convolutional

layer is mainly characterized by its number of filters, kernel size, and spatial strides.

Since the success of AlexNet in image classification [Krizhevsky et al., 2012], CNNs have been widely used for many computer vision tasks, including object detection [Girshick et al., 2014, Girshick, 2015, Ren et al., 2015, Vu et al., 2015, Mordan et al., 2017] and segmentation [Long et al., 2015, Chen et al., 2016], human pose and shape recovery [Güler et al., 2018, Varol et al., 2018], image captioning [You et al., 2016], visual question answering [Antol et al., 2015, Ben-Younes et al., 2017], motion estimation [Dosovitskiy et al., 2015] and others [Jain et al., 2017, Li et al., 2017, Lee et al., 2017, Kim et al., 2018, Engilberge et al., 2018]. In the first part of this section, we revisit the base CNN architectures originally designed for image classification tasks. In the second part, we discuss CNN frameworks designed for object detection in images and videos.

### 3.2.1 Architectures

CNN is a feed-forward network consisting of successive layers stacking on top of each other. The first CNN architecture (AlexNet) that has shown success for classification of natural images [Krizhevsky et al., 2012] had five convolutional layers and relatively large filter sizes. Further work has shown advantages of CNNs with more convolutional layers and smaller filters such as VGG architecture [Chatfield et al., 2014]. While the increased network depth has led to the vanishing gradient problem, [He et al., 2016] enabled training of very deep networks by introducing ResNet architectures with skip connections. We overview these three types of CNN architectures below.

#### AlexNet.

The network consists of 8 layers: 5 convolutional layers and 3 fully-connected layers. Figure 3-1 depicts the network architecture. AlexNet takes fixed size  $224 \times 224$  images as inputs. Kernel sizes of the first and the second convolutional layers are  $11 \times 11$  and  $5 \times 5$  respectively. The rest 3 convolutional layers have kernels of size  $3 \times 3$ .

In total, the network has about 60 million parameters and 650,000 neurons.

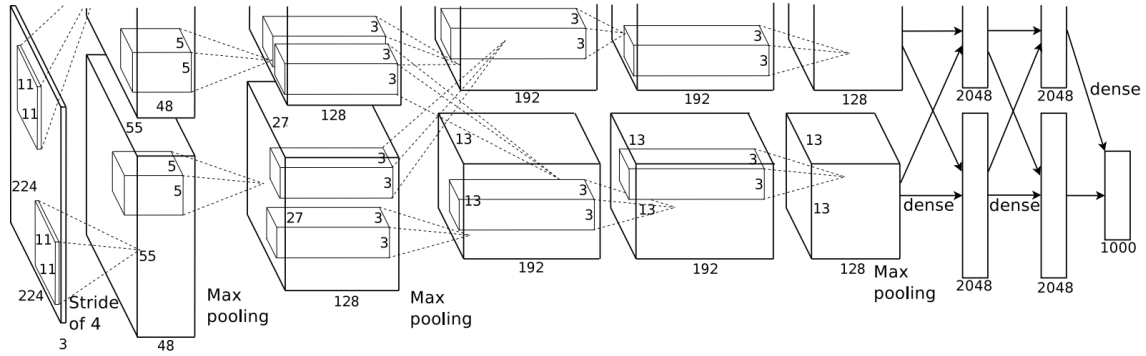


Figure 3-1 – AlexNet architecture for image classification task: 5 successive convolutional layers followed by 3 fully connected layers [Krizhevsky et al., 2012].

## VGG

VGGs are deep convolutional neural networks introduced in [Chatfield et al., 2014]. Although there are several VGG variants introduced in this work, the most common one is VGG-16. VGG-16 consists of 16 layers: 13 convolutional layers and 3 fully connected layers. Unlike AlexNet, all convolutional layers of VGG have a fixed kernel size of  $3 \times 3$ , which is beneficial for reducing the number of parameters. VGG-16 is illustrated in Figure 3-2<sup>1</sup>. The success of VGG models has proved that deeper networks tend to result in improved recognition performance, provided sufficient amount of training data.

## Residual Neural Network

In [He et al., 2016], the authors introduced deep residual neural network (ResNet). While having more layers, ResNet has shown better performance compared to other deep networks such as VGG and GoogleNet thanks to *Batch Normalization* [Ioffe and Szegedy, 2015] and *Skip Connections*. A very deep network is difficult to train due to several reasons including vanishing gradient and internal covariate shift. Skip

1. <http://www.cs.toronto.edu/~frossard/post/vgg16>

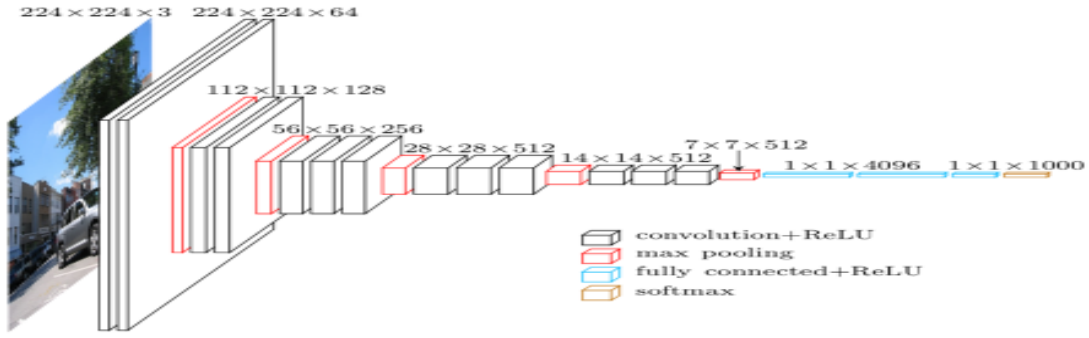


Figure 3-2 – VGG-16 architecture for image classification task: 13 successive convolutional layers followed by 3 fully connected layers [Chatfield et al., 2014].

connections introduced in ResNet have shown to be effective for preserving gradient magnitude and training deeper networks. Batch Normalization reduces internal covariate shift and accelerates deep network training with its stage-wise normalization technique. Figure 3-3 illustrates ResNet with 34 layers alongside with VGG-19.

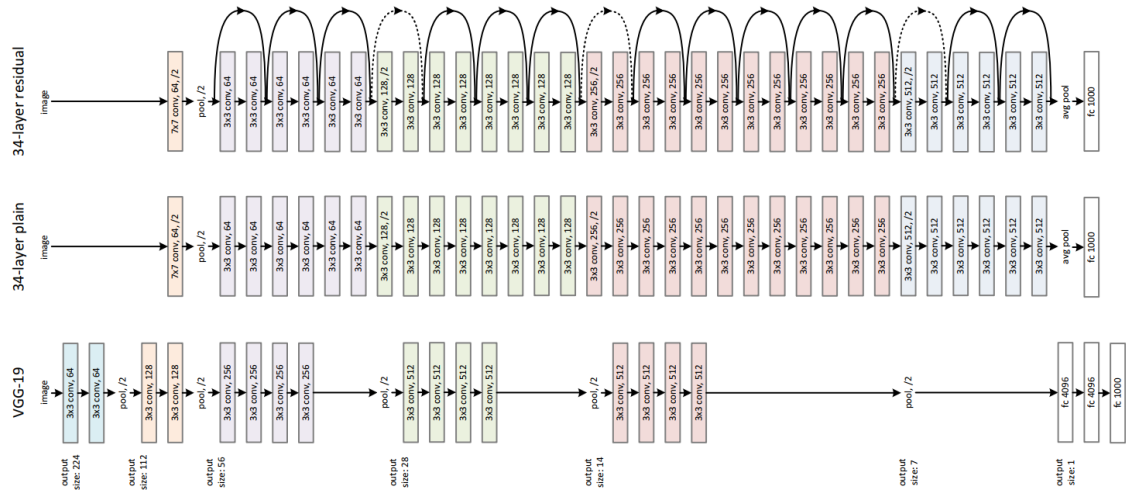


Figure 3-3 – A comparison between VGG-19 and ResNet-34 with respect to the depth [He et al., 2016].

### 3.2.2 Object Detection with CNN

Object detection is the task of simultaneously localizing and recognizing object instances appeared in the input images. To simplify the task, most detection meth-



ods formulate the recognition part as a classification problem, i.e. object classes are predefined beforehand. Such a link allows the adaptation of image classification CNN architectures to the object detection task. A common design principle of most CNN models for object detection is to build upon appearance features produced by pretrained image classification networks. Effectively, only the feature extraction component of the pretrained classification CNNs are retained, e.g. all convolutional layers up to *conv5* in AlexNet. In what follows, we present details of some base CNN models for object detection used in this thesis.

## R-CNN

The R-CNN model is a combination of a CNN and a support vector machine (SVM) operating on object proposals generated by the selective search [Uijlings et al., 2013]. Figure 3-4 illustrates the R-CNN object detection pipeline. Given an input image, we first extract the object proposals using Selective Search. Each image region corresponding to object proposals is then cropped, warped and forwarded through a CNN to produce an appearance feature. The pipeline ends with SVM classifiers, taking the CNN feature as input and producing object scores. R-CNN is a simple yet

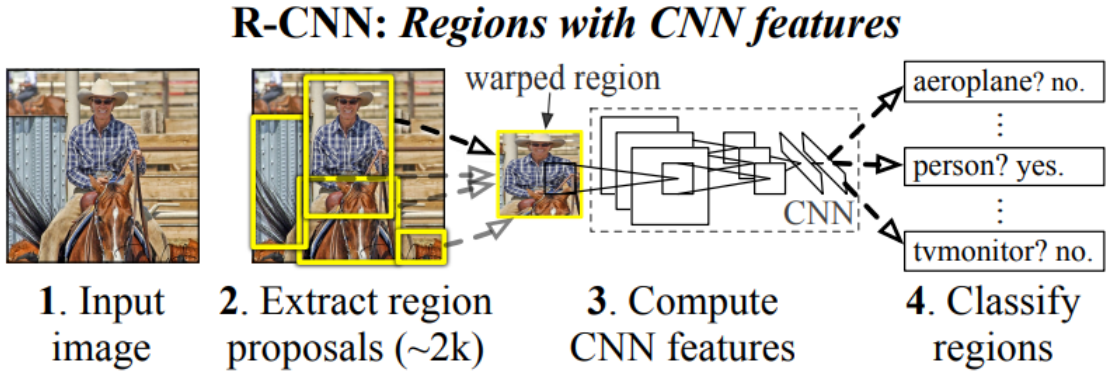


Figure 3-4 – R-CNN object detection pipeline [Girshick et al., 2014].

effective detection model. To achieve the best results, practices such as hard-negative mining and bounding box regressions can be used. In this thesis, the model is used as

a baseline in our work of person detection on individual frames presented in Chapter 5. We modified the original R-CNN by replacing the SVM classifier with a sub-network for classification, which is similarly done in Fast-RCNN and Faster-RCNN later.

## Fast R-CNN

Fast R-CNN [Girshick, 2015] is an upgrade of R-CNN. The two frameworks therefore share similar detection pipeline. Both are based on pre-generated region proposals. Also top convolutional layers of base networks are used for feature extraction. However, instead of forwarding cropped image regions, in Fast R-CNN we extract appearance feature of the whole image by passing it through the stacked convolutional layers. Output features are termed as *feature maps*. Fast R-CNN speeds up R-CNN by a large magnitude with the idea of directly performing *regional-pooling* on feature maps. Regional-pooling is a spatial-aware pooling operation which take box proposals and feature maps as inputs. Given an input image coupled with a feature map, for each box proposal, we proportionally localized the corresponding feature region on the feature map then scale it to a pre-defined size, e.g.  $5 \times 5$ . The scaling step is done by dividing the feature region into equal-sized cells then performing max-pooling on each one. Effectively, outputs of the regional-pooling operation are fixed-size vectors. In this work, the authors introduced the Region-Of-Interest (ROI) pooling layer to this purpose. The ROI-pooling is speed efficient as, for a given input image, it only requires one forward pass to extract the shared feature map for all  $N$  region proposals, while in R-CNN  $N$  forward passes are done. The same argument is also applied for the back-propagation phase on the feature extraction component.

The Fast-RCNN replaces the SVM classifier with a neural network, which not only takes away the high storage-demand but also unifies the whole detection pipeline. Effectively, the model can be trained end-to-end given the precomputed Selective Search proposals. While R-CNN is trained with only one classification loss, Fast-RCNN has two learning branches: region classification and bounding box regression. Moreover, the hard-negative mining step is replaced by a simple heuristic based on overlap ratios of region proposals and object ground-truths. Such modifications have

been proven effective as Fast-RCNN performs better than R-CNN on the detection task. Figure 3-5 depicts the Fast-RCNN object detection pipeline. In this thesis, Fast-RCNN serves as the frame-base baseline in our work of learning video representations for object detection. Also, inspired by the ROI-pooling, we design a nouvel pooling layer operating on spatio-temporal domain. Details are presented in Chapter 6.

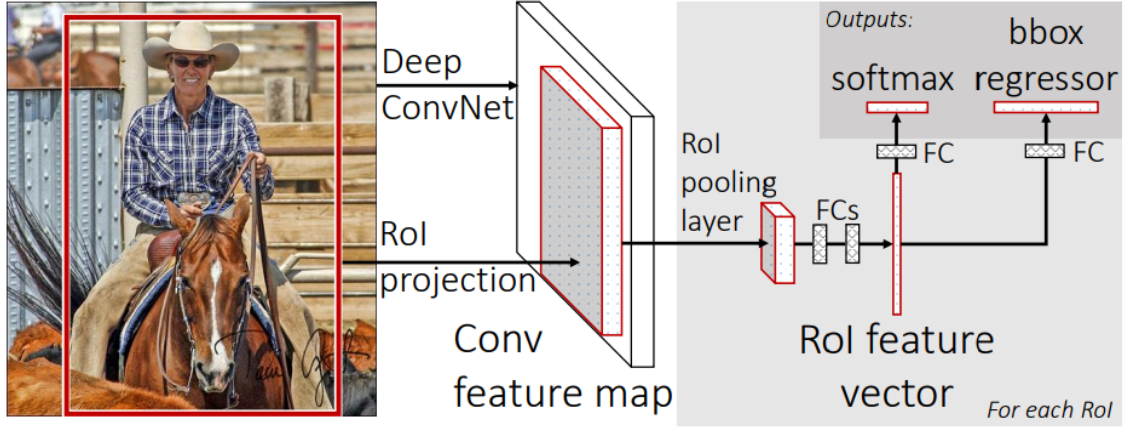


Figure 3-5 – Fast-RCNN object detection pipeline [Girshick, 2015].

## Faster R-CNN

In the follow up work, Faster-RCNN [Ren et al., 2015] improves Fast-RCNN speed and performance with the Region Proposal Networks (RPN), a CNN model which can generate object proposals. Faster R-CNN removes the need of ad-hoc bottom-up object proposals, and increases the speed of object detectors. Instead of only training for the detection task, the Faster-RCNN framework has an additional training phase for proposal generation task. In a nutshell, Faster-RCNN is a combination of two CNN models: Fast-RCNN and RPN. Although the two networks share the feature extraction part, they have different training objectives. The training alternates between the RPN and Fast-RCNN sub-networks.

Figure 3-6 shows the Faster-RCNN object detection pipeline. The RPN branch uses the same feature map used for classification in the Fast-RCNN branch. RPN slides a small convolutional network on the feature map and generate  $K$  object hy-

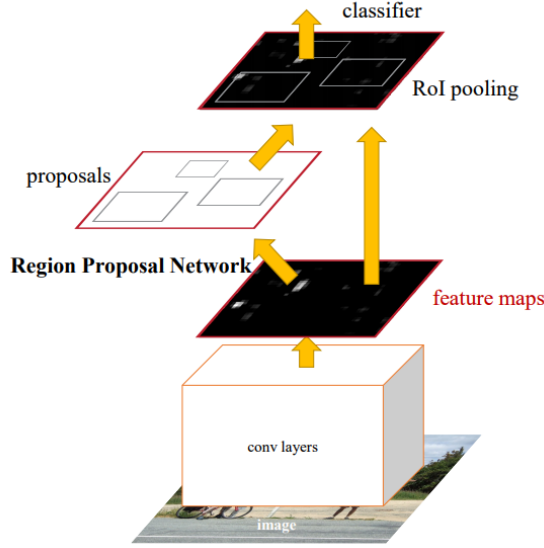


Figure 3-6 – Faster-RCNN object detection pipeline [Ren et al., 2015].

pothesis at every pixel location.  $K$  is the number of pre-defined aspect ratios, e.g.  $1 : 1$ ,  $1 : 2$  and  $2 : 1$ . At test time, only high-confident object proposals are kept. Ren et al. [2015] report that RPN can generate less proposals with higher quality compared to other proposal techniques like Selective Search. Thanks to RPN, Faster-RCNN is both faster and more accurate compared to Fast-RCNN. We take inspiration from the RPN and propose a spatio-temporal proposal generation network. This is part of our contributions presented in Chapter 6.

## R-FCN

R-FCN [Dai et al., 2016] is an efficient detection framework that improves Faster-RCNN in terms of speed. The pipeline of R-FCN still consists of the two alternative phases: generating and classifying object proposals. R-FCN’s architecture takes full advantage of modern fully-convolutional networks like ResNet [He et al., 2016], and resolve Faster-RCNN’s speed bottleneck of fully-connected layers.

To substitute the class-agnostic ROI pooling, R-FCN introduces the Position-Sensitive ROI pooling, a class- and position-specific operation. R-FCN transforms the feature maps of the base network to a position-sensitive score maps, where each

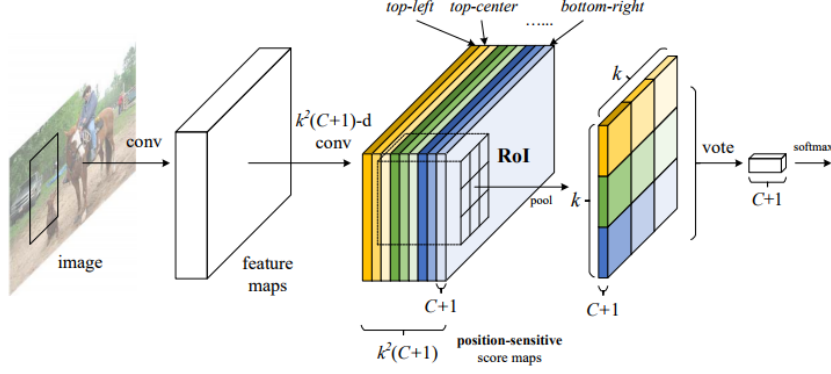


Figure 3-7 – Position-sensitive score maps and Position-sensitive RoI Pooling [Dai et al., 2016].

channel corresponds to a specific object class and a specific pooling position. Position-Sensitive ROI pooling is performed on top of the output position-sensitive score maps. Figure 3-7 demonstrates the pooling process of R-FCN. Such a modification allows a fully-convolutional design for the object classification part of the R-FCN framework, which in turn significantly improves the running speed. In Chapter 7, R-FCN is the frame-base baseline, to which we compare our long-term representation learning framework for object detection in videos. We also use R-FCN as the backbone detection architecture, in this work.

## Contributions Part I



# Chapter 4

## Action Prediction in Images

### 4.1 Introduction

Our environments, such as living rooms, cafés and offices, vary in terms of objects and geometry, but also in *actions* typically performed at these places (e.g., people typically *work* in offices and *cook* or *eat* in kitchens). In this chapter, our goal is to explore such correlation and to develop algorithms able to answer questions such as “What are typical actions for a given scene?”, “Where is a good place to have a picnic?” or “Can I cycle along this path?”. Automatic answers to such questions could be useful for several purposes. First, action prediction could provide scene-specific priors when recognizing human actions. For example, relaxing is common on beaches but not on motorways; cooking is common in kitchens but not in meeting rooms. Second, deviations from an expected set of actions could be used to identify abnormal activities. Moreover, automatic action prediction for geo-localized images could support the search of places suited for particular purposes.

Computer vision has a rich body of work on recognizing human actions [Laptev et al., 2008, Niebles et al., 2010, Sadanand and Corso, 2012, Wang et al., 2011, Marszalek et al., 2009] and scenes [Lazebnik et al., 2006, Oliva and Torralba, 2001, Quattoni and Torralba, 2010, Vogel and Schiele, 2004]. Most of this work addresses the problems of action and scene recognition separately. Recently, several methods have shown advantages of recognizing actions or tracking people in the context of their



environments [Marszalek et al., 2009, Kitani et al., 2012]. Similarly, the interplay between human poses and objects has been studied in [Grabner et al., 2011, Gupta et al., 2011, Delaitre et al., 2012]. While previous work has looked at functional properties for a few selected classes of scenes and objects, here we aim to exploit correlation between scenes and actions at a *large scale* of hundreds of scene categories. Using the discovered correlations, we demonstrate prediction of human actions for test images of outdoor scenes such as, for example, found on Google maps.

To reach our goal, we construct a new SUN Action dataset and collect manual annotations of human actions for 7940 images of 397 scene categories from the SUN dataset [Jianxiong et al., 2010]. Analysis of this data reveals strong action-scene correlation for the majority of scene categories. Notably, we show that an image’s scene category can be determined from corresponding textual descriptions of characteristic actions for that image. We present the SUN Action dataset in Section 4.2 and then study the action-scene correlation in Section 4.3.

Using the discovered relations between scenes and actions, we next address the task of automatic action prediction for images of static scenes. We consider 38 action classes and assign action labels to 397 scene categories. Using such scene-based action annotation we learn visual classifiers for each action category and predict actions for images of static scenes. In Section 4.4, we demonstrate successful action prediction for images of static scenes.

In Section 4.5 we propose and investigate the new task of geo-localized action prediction. Our motivation comes from the large amount of publically-available geo-tagged images (e.g., on Flickr, Instagram and Google Maps) which number constantly increases with the popularity of social networks and the availability of connected camera devices. Application of automatic action prediction on such images will enable the search for places based on their *function*, including specific actions such as swimming, having picnic, hiking and many others. In our experiments reported in Section 4.5, we use geo-localized images collected from [panoramio.com](http://panoramio.com) and demonstrate examples of successful map-based action prediction for the region of France. Section 4.6 concludes the chapter.

## 4.2 SUN Action dataset

**Dataset annotation.** To analyze correlations over a wide range of scene categories and a rich set of actions, we gather the novel *SUN Action* dataset (short for “Scene UNderstanding - Action”) with manual annotations of typical human actions for images of static scenes. We use scene images from the SUN dataset [Xiao et al., 2010]. For each of the 397 well-sampled scene categories we collect free-form annotations of typical actions for the twenty “most typical” images in that category [Ehinger et al., 2011], for a total of 7940 images. Annotations were crowdsourced using Amazon Mechanical Turk (AMT)<sup>1</sup>. AMT workers were shown images of scenes and were asked to list between one and three words or short phrases for each scene describing an action that one would typically perform in a given scene. Scene category labels were not provided. All together we collected 137,558 responses: each image received 17.3 responses on average, and each category received an average of 346.4 responses.

Example images and corresponding responses from the SUN Action dataset are shown in Figure 4-1. We have observed a varying diversity of responses for different scene categories. The top row of Figure 4-1 shows a few examples of scene classes with low entropy of response histograms (high annotator agreement, low response diversity). Such scenes often correspond to places that have been designed for specific purposes (tennis court) or where the natural environment limits the set of possible actions (wave). In contrast, scene classes with high entropy of responses (Figure 4-1, bottom) are places that afford many actions (e.g., a television studio, where many actions need to take place over the course of filming) or unfamiliar places (an anechoic chamber).

The majority of images in the SUN Action dataset contain no people. We found this property to be important for collecting unbiased annotations of typical actions. For a few images containing people we have observed a bias in action annotations towards actions depicted in the image. An example of such a bias is shown in Figure 4-2 illustrating two crosswalk scenes, one without people and one with a cycling person.

---

1. AMT workers gave consent (set by the MIT IRB) for each HIT they chose to perform.

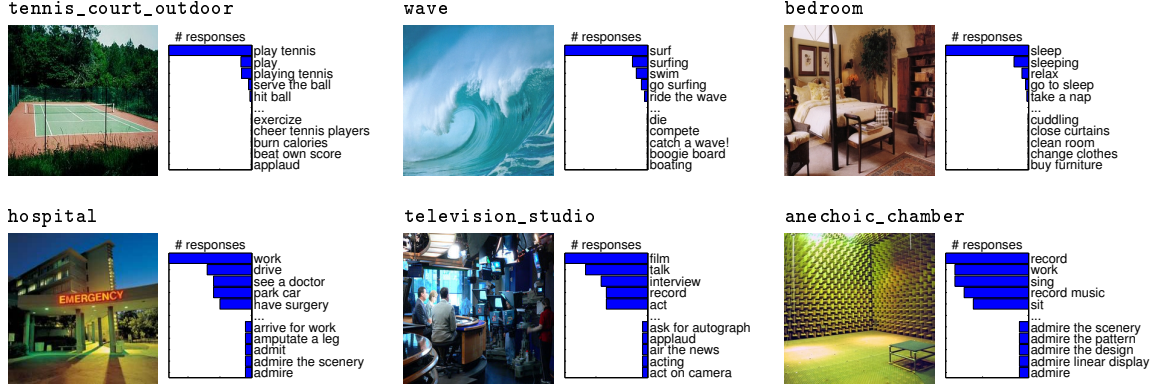


Figure 4-1 – SUN Action scene categories with corresponding histograms of action responses. Top row: Scene categories with low entropy of response histograms. Bottom row: scene categories with higher entropy of response histograms. Low-entropy categories are often places designed for specific purposes (tennis court) or where the environment limits possible actions (wave). By comparison, high-entropy categories are places that afford many actions (television studio) or are unfamiliar (anechoic chamber)



Figure 4-2 – Histograms of words in action responses for two images of the scene class **crosswalk**. The presence of a cyclist in the image on the right biases responses to contain the action “bike”, which is not present in other crosswalk images.

In the scene containing the cyclist, the predominant response was “bike”, unlike other images in the crosswalk category.

**Processing of action responses.** Action responses were gathered in free-form natural language and require preprocessing for our further analysis. Many of responses contain nearly identical information but differ in grammatical structure, such as “read the book while on the flight” and “read a book”. Our first pass of preprocessing converts responses into simplified action annotations by extracting verbs or verb-noun patterns from each response. This strategy reduces the response space while preserving the meaning. For example, responses like “*read the book while on the flight*” or “*avoid eye contact with neighbours*” are trimmed to “*read book*” and “*avoid eye contact*” respectively. We use the Stanford NLP toolbox [Toutanova et al., 2003]

for part-of-speech tagging, stemming, and removal of stop words, and extract either verbs or verb-noun patterns from each response. Responses containing no verbs are removed. The words extracted in this stage of preprocessing are used as input to predict scene categories in Section 4.3.

For the action prediction task in Section 4.4 we manually group semantically similar action responses into action classes. To define action classes, we automatically extract 100 most frequent verb patterns, i.e. single verbs, verb+noun, etc., from action responses. Patterns with similar meaning are then manually merged yielding action labels, for example, “walk on grass” and “walk on sand” are merged into “walk”. We note that the automatic parsing of natural language into action categories is an open problem beyond our work. In particular, we separate scenes into 197 outdoor and 203 indoor categories and define corresponding 38 outdoor and 23 indoor action classes as listed in Figure 4-6.

Given the average of 17.3 action responses per image in our database and a potentially large number of typical actions for a scene, our per-image annotation is not exhaustive. To address this problem, we assume that instances of the same scene category share the same functional properties. We found this assumption to be valid in most cases in our database. We therefore assign the same action labels to all instances of a given scene category using the following *label propagation* strategy. A scene category  $C$  is labeled by an action  $A$  if images of  $C$  are labeled with  $A$  at least 20 times. Following this procedure, for each action label  $A$  we obtain a set of *positive* scene categories. The *negative* scene categories for  $A$  are those containing no  $A$  labels for any of their images. Results of our preprocessing together with the original action responses are available from [<http://www.di.ens.fr/willow/research/actionsfromscenes>].

### 4.3 Analysis of scene-action correlation

Are different scene categories correlated with distinctive sets of actions? Scene categories are often defined by what a person would typically do there: for example, in an office one would typically *work*, whereas in a kitchen one would typically *cook*.

Indeed, most man-made scenes around us have been created to facilitate certain actions.

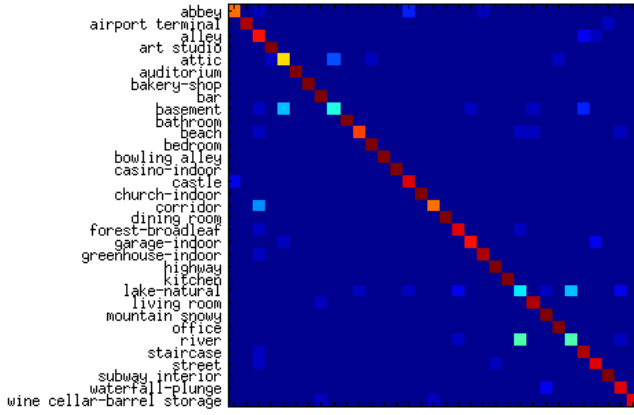
This section verifies and quantifies relations between actions and scenes. We demonstrate successful recognition of a large number of scene categories from associated actions descriptions. We further investigate the structure of action-scene correlations with a hierarchical clustering analysis.

### 4.3.1 Predicting scenes from actions

To verify the hypothesized correlation between scene categories and actions, we conduct two classification experiments using action annotations in the SUN Action dataset. We take inspiration from the field of text classification. In the SUN Action dataset, each image is associated with a collection of natural-language action descriptions. Classifying images based on a collection of associated responses is reminiscent of classifying documents based on their contents. However, there are two notable differences in our approach. First, the number of responses available per image (17.3 responses on average) is significantly lower compared to the number of words in a typical text document. Secondly, we wish to probe category membership using only a small collection of responses per image, to simulate asking a handful of people to provide a most typical action for the image and then performing classification based on the consensus of that set of responses.

#### **Classification methods.**

We classify images using two simple bag-of-words techniques – Nearest Centroid and Naive Bayes. We divide the images in each class into 10 folds for cross-validation. Within the training set, the responses for each image are split into individual words. These word counts are combined and normalized across all images within a given class, to generate a word distribution histogram for each scene category. Within the test set, responses for each image are randomly grouped into chunks of 7 responses for that image, to simulate asking a handful of people at a time to provide a most typical action for each image. Responses within each chunk are then split into individual



(a)

Method	33-cat	397-cat
Chance level	3.00	0.25
Nearest Centroid	85.80	40.31
ML Naive Bayes	<b>91.97</b>	<b>55.86</b>

(b)

Figure 4-3 – Results of action-based scene classification. (a): Confusion matrix for the 33-category subset using Maximum Likelihood Naive Bayes estimation. The high values along the diagonal indicate excellent classification performance. A few pairs of categories e.g., (basement,attic) and (river,lake) are confused due to similarity in their characteristic actions. (b): Average accuracy (%) of scene classification for the 33-category subset and for all 397 scene categories.

words to form bag-of-words queries.

In nearest centroid classification, the bag-of-words queries are normalized to form histograms, which are compared with category histograms according to histogram intersection distance. The scene category centroid with the smallest distance from the query is selected as the class label.

In Maximum Likelihood Naive Bayes classification, the category histograms are interpreted as empirical likelihood estimates: the likelihood  $\Pr(w|c)$  of observing word  $w$  in association with an image of class  $c$  is assumed to be the number of observations of  $w$  within the class  $c$  responses in the training set divided by the total number of words in all class  $c$  responses combined. The word observation likelihoods are assumed to be conditionally independent (the “naive Bayes assumption”), enabling us to compute the class-conditional likelihood of each bag-of-word query as the product of each constituent word’s empirical class likelihood:  $\Pr(w_1, w_2, \dots, w_n|c) = \Pr(w_1|c) \times \Pr(w_2|c) \times \dots \times \Pr(w_n|c)$ . The empirical likelihood estimate makes no explicit provision for estimating the likelihood of unobserved word-class pairs. To address this issue, we compute the minimum class-conditional likelihood over all words

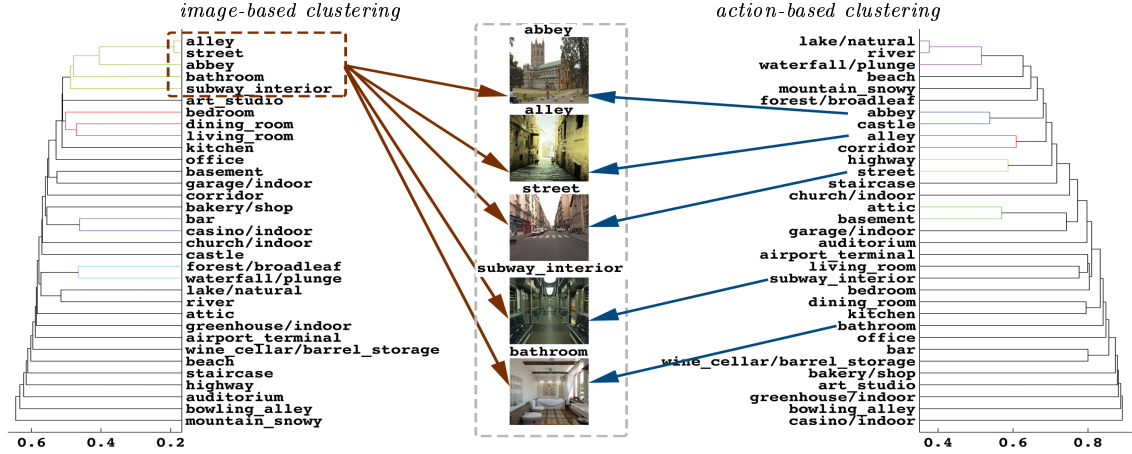


Figure 4-4 – Results of hierarchical clustering of 33 scene categories based on the similarity of image descriptors (left) and action similarity (right). Image-based similarity groups similar-*looking* scenes despite their large difference in semantics such as “alley” and “bathroom”. In contrast, action-based similarity results in more semantically meaningful clusters. For example, “mountain, snowy” is placed in a category of its own according to the visual similarity, whereas it is grouped together with other outdoor places on the basis of action similarity.

and classes in the dataset,  $\min_{w,c}(\Pr(w|c))$ , and use this probability to stand in as the class-conditional likelihood for unobserved words. We assume a uniform prior over scene categories, enabling maximum likelihood estimation: that is to say, bag-of-words queries are classified according to which class provides the largest class-conditional likelihood.

## Results.

Figure 4-3 illustrates results of scene classification. As visualizing results across 397 individual classes is difficult, we select a 33-category subset of well-recognized and semantically important scene categories. To select the 33-category subset, we have asked four of our collaborators to nominate 20-40 most important scene types. Out of the 80 scene types with most annotated images, 35 received at least two nominations and were slated for inclusion. “Cathedral” was removed for not being different enough from “church”, and “abbey” and “coast” was removed for containing only aerial shots, leaving a final slate of 33 scene categories.

The confusion matrix for a Naive Bayes method in Figure 4-3(a) shows a strong

diagonal indicating excellent classification performance. While most classes have almost perfect classification accuracy, a few classes are confused by the classifier due to the sharing of common actions. For example, scene categories “basement” and “attic” are both often annotated by actions “store” and “clean”, while scene categories “river” and “lake” are frequently labeled with “swim” and “fishing”.

Quantitative classification results of the two methods for the 33-category subset and all 397 scene classes are shown in Figure 4-3(b). Notably, both methods perform considerably better than chance while Naive Bayes provides better performance than Nearest Centroid. The fact that such simple classification methods yield very good performance indicates a strong correlation between scene categories and human actions: different scene categories have distinct patterns of associated actions. This confirms our initial hypothesis of a very strong relation between scene categories and their functional properties.

### 4.3.2 Action-based scene clustering

We seek to further investigate the structure of correlations between scene categories and actions: Which scene categories are more similar in terms of their function? We use hierarchical clustering and group scene descriptors at multiple scales. At the finest scale, only the most similar scene types cluster together, whereas at coarse scales, clusters are larger and encompass more dissimilar scene types. Dendrogram visualizations in Figure 4-4 show the progression of clustering patterns from fine to coarse: categories are represented as “leaves”; branchings closer to the leaves of the tree connect classes that cluster together under fine-grained clusterings; and branchings closer to the trunk of the tree encompass broader clusterings. The height of each linkage in the dendrogram indicates the distance between the subclusters it connects.

The two dendrograms in Figure 4-4 illustrate image-based and action-based scene clustering. In the first case, distances between scenes were obtained as Euclidean distances of corresponding image descriptors (see Section 4.4). Clustering based on human action annotations was obtained using  $\chi^2$ -distances between scene representations in terms of bag-of-words histograms (see Section 4.3.1). We observe that



image-based similarity in Figure 4-4(left) captures substantially different information about scene classes as compared to action similarity Figure 4-4(right). For example, “alley”, “bathroom” and “subway interior” are grouped together according to visual similarity due to their similar geometry and texture, but are separated according to action similarity since alleys, bathrooms and subways have different function. Another example is that visual similarity places “mountain, snowy” is a category of its own because no other class commonly depicts open white peaks, whereas action similarity places mountains together with other outdoor places that are associated with hiking, taking photos, and related actions.

## 4.4 Visual Action Prediction

People can easily determine appropriate actions to perform in a given place. Are machines able to do the same thing? In Section 4.3 we have addressed the related problem of predicting scene categories from a set of associated actions. Here we turn to the problem of predicting typical actions for an image of a scene. We approach the problem of visual action prediction using standard image classification techniques in terms of local features and binary classifiers. To train image classifiers we use action labels derived from action annotations as described in Section 4.2. We predict actions separately for indoor and outdoor scenes.

We test two different schemes for action prediction. Under the first scheme (**S1**), we train action classifiers directly from images using action labels only. Under the second scheme (**S2**), we first classify images into scene categories as an intermediate step and then assign action scores based on the obtained scores of scene classifiers. We assume that any particular test image belongs to one scene category only, therefore a score for an action  $a$  in a given image is defined as a max scene score over scene categories  $S$  associated with  $a$ .

### 4.4.1 Implementation Details

#### Image representation.

Our image classification pipeline follows standard approaches and consists of densely extracted local image features, a learned visual vocabulary and a feature encoding step. For local image features we use HOG2x2 [Xiao et al., 2010], SIFT [Lowe, 2004] and CSIFT [Abdel-Hakim and Farag, 2006, van de Sande et al., 2011a] descriptors. Descriptor dimension is reduced by PCA. For the encoding phase we consider two popular encoding techniques: histogram encoding (BoW) and Fisher Vector encoding (FV). To exploit spatial information, we apply Spatial Pyramid framework [Lazebnik et al., 2006], using grids of size 1x1, 2x2, 3x1. Each grid cell is represented either by BoW or FV vectors. The resulting vectors are normalized and concatenated to create the final representation. In the rest of this section we use the format  $\langle descriptor \rangle\_ \langle encoding\ technique \rangle$  to denote image representation techniques, e.g. CSIFT\_FV as Fisher Vector encoding for CSIFT descriptors.

#### Classification.

For the classification, we train SVM classifiers using LIBSVM toolbox [Chang and Lin, 2011]. Linear kernels are used for image representation by Fisher Vector. For the histogram representation (SIFT\_BoW /CSIFT\_BoW), we use  $\chi^2$  kernel [Zhang et al., 2007]. With HOG\_BoW, we exploit Histogram Intersection kernel. Training by SVM, we can boost up the performance by simply using a linear combination of kernels. In our experiments, we aggregate kernels with equal weights.

### 4.4.2 Experimental results

For training and testing the classifiers, we randomly divide the dataset into two equal parts. Our training and testing splits are balanced in the number of images per scene category. Our results for action prediction are summarized in Figures 4-5-4-7 and Table 4.1.

We use mean Average Precision (mAP) as the performance measure. To get mAP,

we first compute the area under precision-recall curve, or Average Precision (AP), for each class. Then mAP is determined as the mean of average precisions across all classes. We obtain best prediction mAP of 60.99% for outdoor actions and 52.09% for indoor actions using combination of HOG\_BOW, HOG\_FV and CSIFT\_FV kernels. Our result is significantly higher than the mAP at chance level, i.e. 6.32% for outdoor action classes and 4.24% for indoor action classes. Figures 4-5-4-7 are produced with our best kernel combination.

In Figure 4-6, we show classification results with 38 outdoor and 23 indoor action classes sorted by AP. For better visualization of prediction results for some action classes, we show example images in Figure 4-7. The last two columns depict some hard positive and hard negative samples for each class. For outdoor scenes, action classes such as “hike”, “pray”, having rather typical color/structure, are easier to classify than other “can-do-almost-everywhere” action classes like “learn”. While people can often differentiate universities from other buildings based e.g., on the text and other cues, the task is still difficult for current vision systems, especially, for those exploiting global image representations. We notice that indoor actions are more structure-dependent than outdoor actions. In our experiment APs of indoor action classes are generally lower than APs of outdoor action classes. We also observe different levels of difficulty among indoor actions, e.g., detecting bowling lanes is easier than detecting sink-like structures. We found building action classifiers challenging, because positive samples are possibly images from very different scene categories, thus covering much larger range of visual texture and structure.

We also aggregate predicted action scores for test images and try to estimate the score contribution. Figure 4-5 shows some test images along with manual action annotations and automatic action predictions. For this visualization, we map SVM scores of test images to probabilities using Platt’s sigmoid [Platt, 1999], with parameters estimated during the training phase. Even though the results are not perfect, we still observe a good match between distributions of annotated and predicted actions. Our predictors successfully give reasonable responses like “swim”, “take a walk” and “relax” to a beach image, or “take picture”, “pray” and “sightsee” to a temple

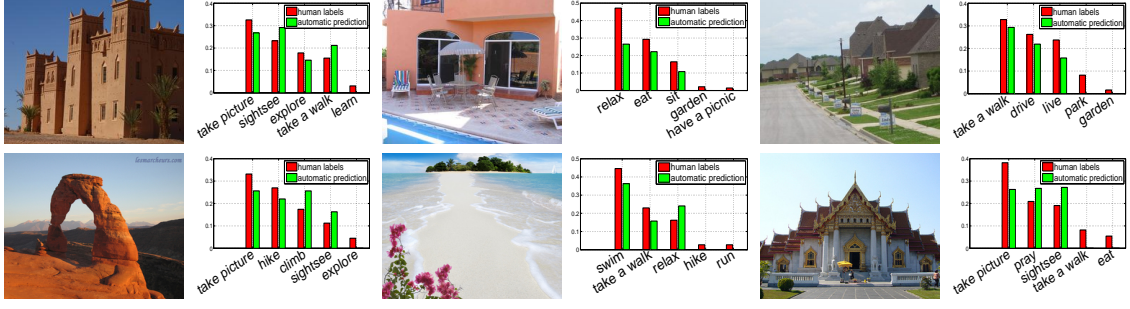


Figure 4-5 – Automatic visual action prediction for test images in SUN Action dataset.

image. Other qualitative results of action prediction by our method are available at [<http://www.di.ens.fr/willow/research/actionsfromscenes>].

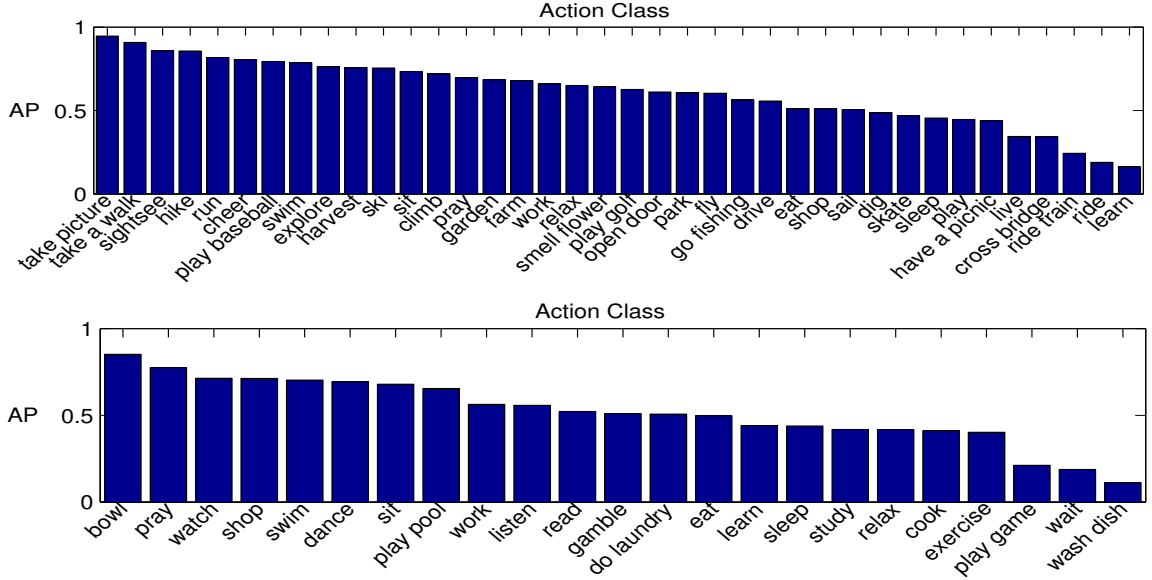


Figure 4-6 – Results of action prediction for all 38 outdoor actions (top) and 23 indoor actions (bottom) sorted in the decreasing order of Average Precision (AP).

For more quantitative analysis, we now consider Table 4.1. The table shows action prediction mAPs of two proposed training schemes combined with different image representation techniques. By comparing results of the two schemes, as shown in two columns **S1** and **S2**, we can conclude that learning action classifiers directly achieve better prediction performance than aggregating multiple scene classifiers. This improvement can be attributed to sharing similar functional properties across different scene classes. In terms of image representation techniques, Fisher Vector encoding

Action	Precision Recall	Correct Predictions	Most Confident False Positives	Least Confident False Negatives
take picture				
hike				
pray				
learn				
bowl				
watch				
sleep				
wash dish				

Figure 4-7 – Selected SUN Action classification results - both outdoor (cyan) and indoor (orange) - with our best kernel combination.

yields better performance compared to Histogram encoding. These results are consistent with recent works on scene classification [Perronnin and Dance, 2007]. Significant performance difference between SIFT and CSIFT proves that color information is useful for the task. Also, linear combination of multiple kernels does improve the performance. Among our three tested kernel combinations, using HOG\_BoW, HOG\_FV and CSIFT\_FV yields the best result. In conclusion, we have shown high accuracy for a new task of action prediction evaluated on a large number of action and scene classes.

## 4.5 Image-based Geo-Mapping of Actions

One possible application of scene-based action prediction is to search for places affording specific action. For example, a user may ask “Where can I camp in the Mont Blanc valley?” or “Where can I sunbathe in Tuscany?”. Such queries are currently not supported by map services such as Google Maps or Bing Maps. To address this problem, we introduce Image-based Geo-Mapping of Action (IGMA), an application

Method	S1	S2	S1-S2	S1	S2	S1-S2
SIFT_BoW	40.92	40.68	0.24	31.75	28.71	3.04
SIFT_FV	41.15	34.51	6.64	31.04	27.13	3.91
CSIFT_BoW	47.78	44.43	3.35	32.53	27.90	4.63
CSIFT_FV	49.52	41.65	7.87	36.29	29.70	6.59
HOG_BoW	47.03	45.93	1.10	37.91	35.50	2.41
HOG_FV	52.66	47.75	4.91	42.78	43.89	-1.11
HOG_BoW+ SIFT_FV+ CSIFT_FV	56.60	50.06	6.54	46.11	40.04	6.07
HOG_BoW+ HOG_FV+ CSIFT_FV	<b>60.99</b>	54.25	6.74	<b>52.09</b>	45.98	6.11
SIFT_FV+ HOG_FV+ CSIFT_FV	56.48	49.61	6.87	45.76	41.41	4.35

Table 4.1 – Scene-based outdoor (cyan) and indoor (orange) action prediction results with different approaches. Note that mAP at chance level is 6.32% (outdoor) and 4.24% (indoor). **S1** and **S2** columns respectively show classification mAP (%) of the two aforementioned training schemes. Column (**S1-S2**) shows the different mAP between two schemes. We observe consistently better performance of scheme **S1**: directly training binary action classifiers over scheme **S2**: aggregating scene classifiers.

for geo-localizing actions on a map and answering map queries of the type “Where can I do X?”. Results are derived from geo-localized scene images publicly available on the Internet. IGMA is the first attempt to automatically answer geo-localized action queries. Our strategy of predicting actions at a broad spatial scale using geo-localized images enables us to go beyond manual location-action labels: for example, one can “have a picnic” not only at a designated picnic area, but also in a grassy countryside field.

**Collecting the Panoramio dataset.** We use Panoramio image sharing service [Google, 2007] to collect a dataset with geo-localized images. Like the SUN Action dataset, the images in Panoramio contain few to no people. The Panoramio service provides a REST API for selecting images: given a range of longitude and latitude values, Panoramio returns a JSON file of image properties including image URL and geographical position. For our experiment, we collected Panoramio images of France, with longitude from  $-5^\circ$  to  $8^\circ$  and latitude from  $41^\circ$  to  $51^\circ$ . In total, our dataset contains over 38,000 distinct geo-tagged images.

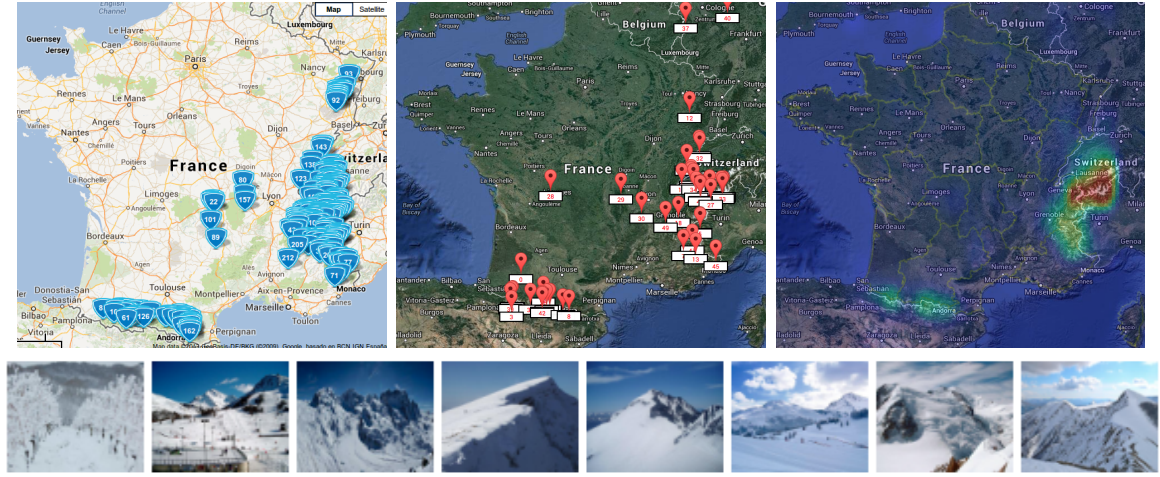


Figure 4-8 – “Where can I ski in France?” - (Top left) Official skiing stations in France [ski, 2013]. (Top middle) Suggested places for skiing by IGMA. (Top right) Dense map of action “ski” generated by IGMA. (Bottom) Panoramio images of suggested places for skiing.

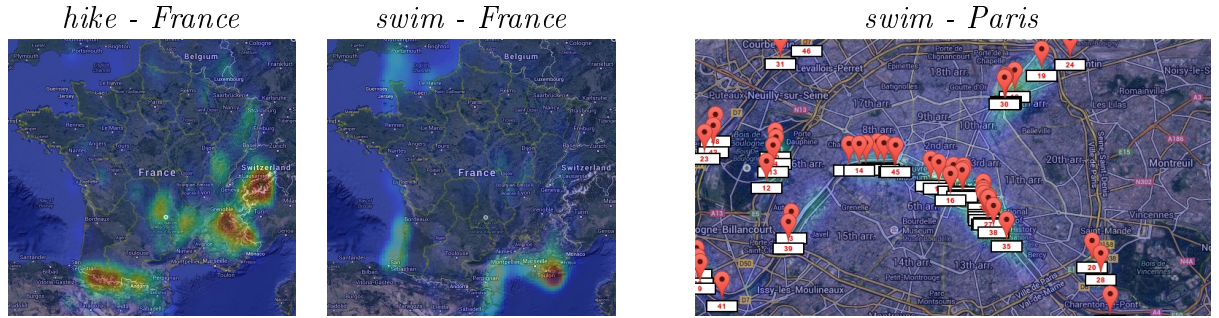


Figure 4-9 – Geo-localized prediction of actions. (left): Predictions for actions “hike” and “swim” on the map of France. (right): Predictions for the action “swim” in Paris.

**Dense map of actions.** Our goal is to construct dense maps visualizing places where people would likely perform certain actions. We construct these maps by applying scene-based action classifiers computed in Section 4.4.2 to the collected Panoramio dataset using the following procedure.

For a given action, we compute the top-scored Panoramio images. We generate a dense map from this list of scores and geo-locations by modeling the map using a Gaussian Mixture Model (GMM) with mixture components centered at the image locations and their weights set to corresponding action scores. The standard deviation  $\sigma$  for each component is set to a fixed value.

This initial dense map estimate is adjusted to compensate for non-uniform sam-



pling of Panoramio images. Different population densities of the examined regions may introduce biases to the action density estimation. Therefore, we estimate the sampling density of Panoramio images. Using the same GMM model above, but with a uniform weight across all mixture model components rather than an action-score-based weight. The initial action map estimated from the highest scored images is normalized by the estimated sampling density of Panoramio images to correct the sampling density bias. We then get the final estimation of action density.

Figure 4-8 illustrates IGMA’s suggestions for the question “Where can I ski in France?”. We compare the estimated dense map produced by IGMA for the action “ski” with the the map of official skiing stations in France, acquired from [ski, 2013]. Qualitatively, our predictions have a high degree of correspondence the locations of skiing areas. Similarly, in Figure 4-9(left) we illustrate predictions for “hike” and “swim” in France. These results visually correspond to the sea-coast and mountain areas of France, confirming good geo-localization of actions.

Figure 4-9(right) illustrates an interesting result of predicting the “swim” action in Paris. This result suggests an area for further investigation: the recommended locations for swimming in Paris fall mainly along the river Seine, where swimming is very uncommon. While it is true that scene categories often have strong correlation with associated actions, not all scenes within a scene category share the same action affordances in practice. One possible approach to this issue might be to subdivide scene categories according to more fine-grained functional affordances, e.g. separately identifying rivers where people can and cannot swim.

## 4.6 Conclusion

In this chapter we have addressed a new problem of action prediction for a wide range of scene images. We have collected a new SUN Action dataset with manual annotations of typical actions for scene images, and discovered strong action-scene correlation for the majority of scene classes. Motivated by this discovery, we have next learned to predict typical actions for a large set of scenes. Using standard state-of-the-



art image classification techniques we have shown high accuracy of action prediction for a large variety of scenes, which is an encouraging result for a new problem. To further demonstrate the potential advantages of our work, we have shown promising results for a new application Geo-Mapping of Actions (IGMA) enabling automatic answers to queries “Where can I do X?”.

## Contributions Part II



## Chapter 5

# Person detection with context-aware CNNs

The second part of this thesis presents our works on person and object detection in videos. This chapter explores two contextual cues for improving person detection in videos. We build on the recent R-CNN model for object detection [Girshick et al., 2014] and extend it to contextual reasoning. We particularly focus on person detection and aim to locate human heads on images coming from video data. The choice of heads is motivated by frequent occlusions of other body parts. When visible, however, other body parts and the rest of the scene constrain locations of heads in the image. Moreover, interactions between people put constraints on the relative positions and appearance of heads. We aim to leverage such constraints for detection by introducing the two models below. Our approach shows a consistent improvement over the R-CNN object detector baseline, which is termed as *Local* model in this chapter.

First, we propose a *Global* CNN model which we train to predict coarse locations and scales of objects given the full low-resolution image on the input. In contrast to the base *Local* model limited to object appearance only, the Global model uses all pixels of the image for prediction. Interestingly, we find this simple model to provide rather accurate localization of heads across positions and scales of the image. Our Global model is related to the more recent line of work on proposal generation [Liu et al., 2016, Redmon and Farhadi, 2017, 2018]. Second, we introduce a *Pairwise* CNN

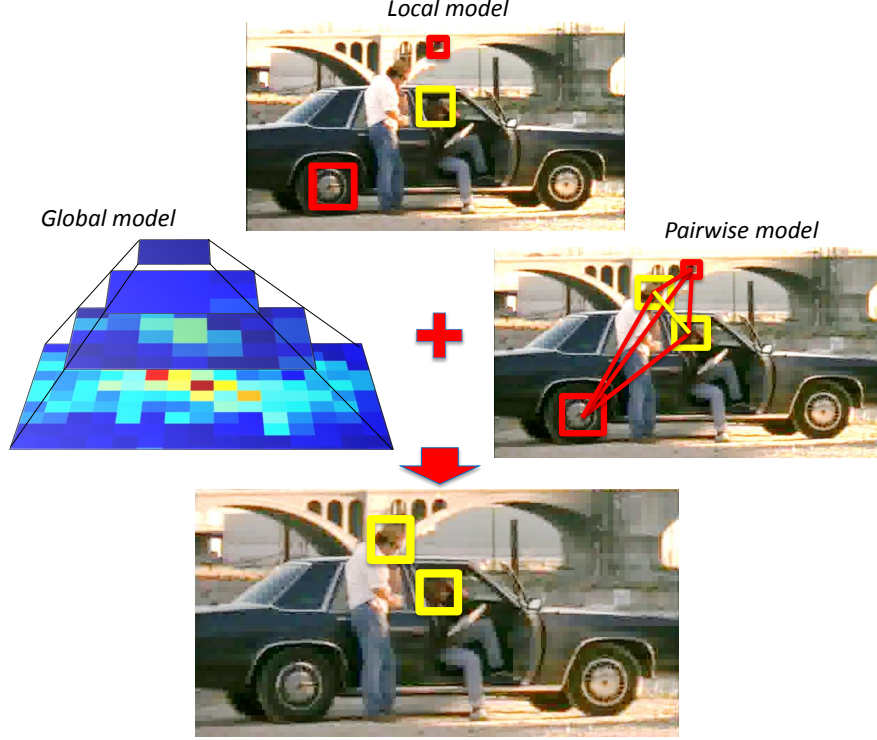


Figure 5-1 – Results of head detection for a sample movie frame. The output of our method (bottom) is obtained from the combination of Local, Global and Pairwise CNN models. Bounding boxes illustrate detections: yellow – correct, red – false. Links between detections correspond to the pairwise potentials of our model: yellow – attractive, red – repulsive.

model that explicitly models relations among pairs of objects. Motivated by [Desai et al., 2011], we build a joint score function for multiple object hypotheses in the image. This score function considers the relative positions, scales and appearance of heads. All parameters of the score function depend on the image data and are learned by optimizing a structured-output loss function. Our final joint model combines Local, Global and Pairwise CNN models (see Figure 5-1).

The rest of this chapter is organized as follows. In Section 5.1, we describes our contextual CNN models. Implementation details are presented in Section 5.2. Section 5.3 introduces benchmark datasets followed by the presentation of experimental results in Section 5.4. Section 5.5 concludes the chapter.

## 5.1 Context-aware CNN model

This section presents main components of our contextual CNN model. In Section 5.1.1, we describe our Local model building on R-CNN [Girshick et al., 2014]. In Section 5.1.2, we introduce the Global CNN model trained to score object proposals using the context of the full image. Section 5.1.3 describes our extension of CNNs with a structured-output loss function aimed to model pairwise relations between objects.

### 5.1.1 Local model

Our Local model follows R-CNN [Girshick et al., 2014] and uses selective search proposals [van de Sande et al., 2011b] to restrict the set of object hypotheses. We extend the bounding box of each proposal with a small margin to capture local image context around objects. The image patch corresponding to each proposal is then resized to fit the input layer of the CNN. As we are interested in head detection, we select bounding boxes with square-like aspect ratios  $\mathcal{R} \in [2/3, 3/2]$  and refer to them as candidates.

The R-CNN model is based on the AlexNet architecture [Krizhevsky et al., 2012] pre-trained on the ImageNet dataset [Deng et al., 2009]. We have considered several alternatives including VGG-S [Chatfield et al., 2014], VGG-verydeep-16 [Simonyan and Zisserman, 2015] and [Oquab et al., 2014]. In our experiments VGG-S slightly outperformed AlexNet but was significantly slower in both training and testing. VGG-verydeep-16 showed better performance but was much slower. The network of [Oquab et al., 2014] had better accuracy and similar speed compared to AlexNet (see Section 5.4.3 for details). For experiments in this work we use the pre-trained network of [Oquab et al., 2014] extended by one fully-connected layer (with 2048 nodes) initialized randomly and followed by ReLu and Dropout.

To train the network, we optimize parameters by minimizing the sum of independent log-losses using stochastic gradient descent (SGD) with momentum. Differently from R-CNN which deploys the second pass of training using SVM, we use the out-

puts of CNN to score candidates. We found this training procedure to work better for our problem compared to the standard R-CNN training.

### 5.1.2 Global model

Our Global model uses image-level information to reason about locations of objects in the image. The Global model is a CNN that takes the whole image as input and outputs a score for each cell of a multi-scale heat map. The input image is isotropically rescaled and zero-padded to fit the standard CNN input of  $224 \times 224$  pixels. The output of the network is defined as a multi-scale grid of scores, corresponding to object hypotheses with coarsely discretized locations and scales in the image (see Figure 5-2). Object hypotheses form a grid of  $C = 284$  square cells of four sizes (28x28, 56x56, 112x112 and 224x224 pixels) and the stride corresponding to the 50% of cell size. Except the output layer, the architecture of the Global CNN is identical to our Local model described in Section 5.1.1.

The Global CNN is trained with SGD, minimizing the sum of  $C$  log-loss functions, one per each grid cell  $c \in \{1 \cdots C\}$ ,

$$\ell(\mathbf{f}_c(\mathbf{x}), y_c) = \sum_{y \in \{0,1\}} \log(1 + \exp((-1)^{y_c+y+1} f_{c,y}(\mathbf{x}))) , \quad (5.1)$$

where  $\mathbf{f}_c(\mathbf{x}) \in \mathbb{R}^2$  is the output of the network for grid cell  $c$  of input image  $\mathbf{x}$ ;  $y_c \in \{0,1\}$  is the label indicating the class of the grid cell  $c$ : *background* or *head*. We set the label of a grid cell to *head* if the Intersection-over-Union (IoU) overlap-ratio between the cell and any ground-truth bounding box in the image  $\mathbf{x}$  is larger than 0.3, otherwise the label is set to *background*.

Due to the coarse resolution of grid cells, our Global model does not provide accurate localization. We therefore use the Global model to rescore the candidates of Local and Pairwise models. For this purpose, we match each candidate with the corresponding grid cell and compute affine combination of their scores. Each candidate is matched to a grid cell with the maximum IoU overlap-ratio. The parameters of affine score combination are optimized by cross validation on the validation set.

### 5.1.3 Pairwise model

In this section we describe our Pairwise model that aims to jointly reason about multiple object candidates. Following [Desai et al., 2011] we formulate the model as a joint score function where variables correspond to object candidates. In the prior work [Desai et al., 2011, Felzenszwalb et al., 2010a, Hoai and Zisserman, 2014] unary potentials of the score function are defined by the response of the local object detector at corresponding locations, whereas higher-order potentials model spatial relations between candidates. Our Pairwise model enriches the model of [Desai et al., 2011] by making all potentials of the score function (5.2) dependent on the image data and, in contrast to [Chen and Yuille, 2014], allows to perform the joint training of all parameters. We describe details of our model in Section 5.1.3.

We train parameters of our model by minimizing the structured surrogate loss using stochastic gradient descent algorithm. The details of our training procedure are presented in Section 5.1.3.

#### Model formulation

**Score function.** Consider a set of  $\mathcal{V}$  candidate bounding boxes (nodes) extracted from an image. Let each bounding box have a binary variable  $y_i$ ,  $i \in \mathcal{V}$  assigned to it. We associate label 1 with the object class and label 0 with the background class. We assume that the ground-truth labels  $\hat{y}_i$  are available for all candidates in training images.

For each pair of nodes we choose an order based on the coordinates of corresponding bounding boxes: the left box is defined to be the first, the right one – the second. Let  $\mathcal{E}$  denote the set of oriented pairs of candidates (set of edges). We cluster all edges based on relative locations and scales of bounding boxes<sup>1</sup> and denote the cluster index of edge  $(i, j) \in \mathcal{E}$  by  $k_{ij} \in \{1, \dots, K\}$ . The clustering step helps to group

---

1. To cluster edges we apply k-means algorithm with  $K = 20$  to a subset of oriented edges in training images. Edges in this subset connect object candidates with positive labels as well as any other candidates with high scores of the pre-trained Local model. For the clustering we use relative location features (horizontal and vertical displacements, ratio of sizes) converted to the log scale and normalized to have zero mean and unit standard deviation.



pairs of heads with similar spatial arrangement and relative scale.

Inspired by [Desai et al., 2011], we construct a joint score function  $S(\mathbf{y}; \mathbf{w})$  that ties together the labels of candidates in the same image:

$$S(\mathbf{y}; \mathbf{w}) = \sum_{i \in \mathcal{V}} \theta_i^U(y_i; \mathbf{w}) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}^P(y_i, y_j, k_{ij}; \mathbf{w}), \quad (5.2)$$

where  $\mathbf{w}$  denotes trainable parameters,  $\theta_i^U$  and  $\theta_{ij}^P$  are unary and pairwise potentials depending on  $\mathbf{w}$ , and  $\mathbf{y} = (y_i)_{i \in \mathcal{V}}$  is a vector of all binary variables.

Note, that different values of potentials in (5.2) can lead to exactly the same score function  $S$ . We rewrite Eq. (5.2) in the more compact form (the set of all representable functions of binary variables stays the same):

$$S(\mathbf{y}; \mathbf{w}) = \sum_{i \in \mathcal{V}} y_i \theta_i^U(\mathbf{w}) + \sum_{(i,j) \in \mathcal{E}} y_i y_j \theta_{ij,k_{ij}}^P(\mathbf{w}) \quad (5.3)$$

where unary potentials  $\theta_i^U$  and pairwise potentials  $\theta_{ij,k_{ij}}^P$  are represented by real values.

**Connecting the score function and the image.** Now we connect the image with potentials of the score function (5.3) using several feed-forward neural networks. First, from the Local model described in Section 5.1.1 we create a feature extractor (FE), i.e. a function  $\varphi^E$  that constructs feature vector  $\mathbf{f}_i$  for the image data  $\mathbf{x}_i$  of candidate  $i$ :  $\mathbf{f}_i = \varphi^E(\mathbf{x}_i, \mathbf{w}^E)$ . Here  $\mathbf{w}^E$  is a vector of trainable parameters of FE.

To connect features  $\mathbf{f}_i$  with potentials in (5.3) we construct two additional feed-forward networks: the unary network (UN) and the pairwise network (PN). The unary network  $\varphi^U$  maps the feature vector  $\mathbf{f}_i$  of a candidate  $i$  to the value of the corresponding unary potential, i.e.  $\theta_i^U = \varphi^U(\mathbf{f}_i, \mathbf{w}^U)$ . The pairwise network  $\varphi^P$  maps the concatenated feature vectors of its two candidates to a vector  $\theta_{ij}^P$  where the  $k$ -th component  $\theta_{ij,k}^P$  corresponds to the one of  $K$  cluster indices, i.e.  $\theta_{ij,k_{ij}}^P = \varphi_{k_{ij}}^P(\mathbf{f}_i, \mathbf{f}_j, \mathbf{w}^P)$ . Vectors  $\mathbf{w}^U$  and  $\mathbf{w}^P$  are the trainable parameters of the UN and PN, correspondingly.

In our experiments we found the following architectures to work best. The FE was of the same structure as our Local model (based on the network of [Oquab et al.,

2014]) leading to 2048 features. In both UN and PN we use just one fully-connected layer. Addition of more hidden layers did not improve results.

**Precision-recall evaluation.** Object detection methods are typically evaluated in terms of precision-recall (PR) and average precision (AP) values. To construct the precision-recall curve given the joint score (5.3), we follow the approach of [Desai et al., 2011]. For each candidate bounding box  $i$ , we compute an individual score  $s_i(\mathbf{w})$  defined as the difference of the max-marginals of the joint score

$$s_i(\mathbf{w}) = \max_{\mathbf{y}: y_i=1} S(\mathbf{y}; \mathbf{w}) - \max_{\mathbf{y}: y_i=0} S(\mathbf{y}; \mathbf{w}). \quad (5.4)$$

The individual scores are used in the standard precision-recall evaluation pipeline [Everingham et al., 2010].

When the number of candidates is small, i.e.  $|\mathcal{V}| \leq 20$ , both maximization problems of (5.4) can be solved exactly using exhaustive search. When the number of candidates becomes larger, the exhaustive search becomes too slow. In this case one can use the cascade of QPBO [Kolmogorov and Rother, 2007] and TRW-S [Kolmogorov, 2006] methods to approximate  $s_i$ . Specifically, QPBO allows to quickly determine the optimal label for some candidates. On our dataset QPBO works surprisingly well, i.e., in many cases it is able to label all nodes. If some nodes are unlabeled by QPBO, one can apply the exhaustive search when the number of unlabelled nodes is at most 20 and TRW-S otherwise.

We have tried using 16 and 32 candidates per image. The exact inference is tractable only in the first case. In this work, we use 16 candidates per image as the large number of candidates did not improve performance on our validation set.

## Training the model

We train parameters of our model by minimizing a structured surrogate loss using the stochastic gradient descent algorithm<sup>2</sup>. The algorithm for parameter update

---

2. As common in the deep learning literature we ignore the non-differentiability issues and assume that in practice we can always compute the gradient.

consists of the following four steps:

1. Select the set of candidates by applying the non-maximum suppression [Felzenszwalb et al., 2010a] on top of the scores produced by the Local model.
2. Perform the forward pass through the model to compute potentials of the joint score function.
3. Perform the inference to compute the structured loss and its gradient (see below).
4. Back-propagate the gradient through the model.

We explain details of the algorithm below.

**Structured surrogate loss.** A structured loss is a function that maps the current values of parameters, image data  $\mathbf{x} = (\mathbf{x}_i)_{i \in \mathcal{V}}$  and the ground-truth labeling  $\hat{\mathbf{y}} = (\hat{y}_i)_{i \in \mathcal{V}}$  to a real number. A popular choice for the surrogate loss for structured-prediction tasks is the structured SVM (SSVM) objective [Taskar et al., 2003, Tsochantaridis et al., 2005]:

$$\ell_{\text{SVM}}(\mathbf{w}, \hat{\mathbf{y}}, \mathbf{x}) = \max_{\mathbf{y}} \left( S(\mathbf{y}; \mathbf{w}, \mathbf{x}) + h(\mathbf{y}, \hat{\mathbf{y}}) \right) - S(\hat{\mathbf{y}}; \mathbf{w}, \mathbf{x}) \quad (5.5)$$

where  $h(\mathbf{y}, \hat{\mathbf{y}}) \geq 0$  measures the agreement between the two labelings. Possible choices for  $h$  include the Hamming loss, the Hamming loss with penalties normalized by the frequency of classes, or higher-order losses making use of assumption that each ground-truth object is assigned to exactly one object candidate [Osokin and Kohli, 2014]. Notice, that in (5.5) the joint score  $S$  depends on parameters  $\mathbf{w}$  and image data  $\mathbf{x}$  implicitly through potentials  $\theta^U$  and  $\theta^P$ .

However, in our experiments we have observed that the SSVM loss is less suited for the detection task, i.e. optimizing the objective (5.5) does not lead to good results in terms of precision-recall measure. To tackle this problem, we propose a new surrogate loss which directly imposes penalties on the wrong values of individual scores (5.4)

extracted from the joint score  $S$ . Specifically, this loss can be written as

$$\ell(\mathbf{w}, \hat{\mathbf{y}}, \mathbf{x}) = \sum_{i: \hat{y}_i=1} v(s_i(\mathbf{w}, \mathbf{x})) + \sum_{i: \hat{y}_i=0} v(-s_i(\mathbf{w}, \mathbf{x})) \quad (5.6)$$

where  $v$  can be any non-increasing function bounded from below. We use  $v(t) = \log(1 + \exp(-t))$  which brings us closer to the training of conventional detector with a soft-max loss.

**Gradient of the structured loss.** To optimize the structured loss w.r.t. the model parameters  $\mathbf{w}$ , we need to compute the gradient of the objective w.r.t. model parameters. We can always achieve this goal using the back-propagation method under two assumptions: 1) the gradient can be back-propagated through the modules of the model, i.e. all the partial derivatives of  $\varphi^E$ ,  $\varphi^U$ ,  $\varphi^P$  w.r.t. the input and the parameters can be computed; 2) the scores of the candidates (5.4) can be computed exactly.

To start the back-propagation procedure, we compute the gradient of structured loss w.r.t. potentials  $\theta_i^U$ ,  $\theta_{ij,k}^P$  of the joint score function  $S$ . [Jaderberg et al., 2015a] have in details explained how to do this for the SSVM loss (5.5). Here we explain how to differentiate the loss (5.6). First, the gradient of the loss (5.6) w.r.t. the scores can be expressed as

$$\frac{d\ell}{ds_i} = (-1)^{\hat{y}_i+1} v'(s_i(-1)^{\hat{y}_i+1}), \quad v'(t) = \frac{-\exp(-t)}{1 + \exp(-t)}.$$

The gradient of the score (when existent) w.r.t. potentials can be computed exactly if we can compute all max-marginals exactly:

$$\frac{ds_i}{d\theta_p^U} = y_p^{i,1} - y_p^{i,0}, \quad \frac{ds_i}{d\theta_{pq,k}^P} = (y_p^{i,1} y_q^{i,1} - y_p^{i,0} y_q^{i,0}) [k_{ij} = k]$$

where  $y_q^{i,t}$  is the  $q$ -th component of  $\mathbf{y}^{i,t} = \arg \max_{\mathbf{y}: y_i=t} S(\mathbf{y}; \mathbf{w})$  for  $t \in \{0, 1\}$ . Here,  $[\cdot]$  is the Iverson bracket notation. Combining the two derivatives via the chain rule we

get

$$\frac{d\ell}{d\theta_p^U} = \sum_{i \in \mathcal{V}} \frac{d\ell}{ds_i} \frac{ds_i}{d\theta_p^U}, \quad \frac{d\ell}{d\theta_{pq,k}^P} = \sum_{i \in \mathcal{V}} \frac{d\ell}{ds_i} \frac{ds_i}{d\theta_{pq,k}^P}.$$

**Back-propagation of the gradient.** The next step of the back-propagation procedure is to compute the derivatives of the loss w.r.t. parameters of the UN and PN

$$\frac{d\ell}{d\mathbf{w}^U} = \sum_{i \in \mathcal{V}} \frac{d\ell}{d\theta_i^U} \frac{d\theta_i^U}{d\mathbf{w}^U}, \quad \frac{d\ell}{d\mathbf{w}^P} = \sum_{(i,j) \in \mathcal{E}} \sum_{k=1}^K \frac{d\ell}{d\theta_{ij,k}^P} \frac{d\theta_{ij,k}^P}{d\mathbf{w}^P} \quad (5.7)$$

and w.r.t. the output of the feature extractor

$$\begin{aligned} \frac{d\ell}{d\mathbf{f}_i} &= \frac{d\ell}{d\theta_i^U} \frac{d\theta_i^U}{d\mathbf{f}_i} + \sum_{j:(i,j) \in \mathcal{E}} \frac{d\ell}{d\theta_{ij,k_{ij}}^P} \frac{d\theta_{ij,k_{ij}}^P}{d\mathbf{f}_i} \\ &\quad + \sum_{j:(j,i) \in \mathcal{E}} \frac{d\ell}{d\theta_{ji,k_{ji}}^P} \frac{d\theta_{ji,k_{ji}}^P}{d\mathbf{f}_i}. \end{aligned} \quad (5.8)$$

Notice that all the derivatives of potentials w.r.t. parameters and features can be computed by propagating the gradient through networks  $\varphi^U$  and  $\varphi^P$ . Finally, propagation of the gradient (5.8) through  $\varphi^E$  gives us the direction of the update for parameters  $\mathbf{w}^E$  of the FE.

## 5.2 Implementation details

In this section we provide implementation details for our models, our approach to merging the models, and some technical information.

### 5.2.1 Local model

To train the Local model, we assign each candidate region to the positive (head) or negative (background) class. For a given image, we make this assignment based on the intersection-over-union (IoU) overlap ratio  $o$  of the candidate bounding box with the best matching ground-truth bounding box. Specifically, candidates with  $o > 0.6$  are labeled as positives and candidates with  $o < 0.5$  are labeled as negatives.

The remaining candidates are considered ambiguous and are not used at the training. Following [Girshick et al., 2014] we exploit the context padding. Each candidate is resized to  $188 \times 188$  square patch which is extended with 18 pixels on each side filled from the original image. The input images of our CNN are of size  $224 \times 224$ . For each image, we form a training batch by sampling 64 proposals such that the balance between classes is roughly maintained.

We initialize parameters of the network using the ImageNet pre-trained network of [Oquab et al., 2014]. We optimize the parameters of the network by minimizing the sum of independent log-losses with a stochastic gradient descent (SGD) algorithm with momentum 0.9 and weight decay 0.0005. We initialize the learning rate at 0.01, and decrease it several times by a factor of 10 after the validation error reaches saturation.

### 5.2.2 Global model

The Global model takes the whole image (isotropically rescaled and zero-padded to size  $224 \times 224$ ) as the input and provides a vector of 284 numbers as the output. Each element of the output vector is associated with a cell of our multi-scale grid. For each cell we construct a target objective: 1 is the corresponding image patch has at least 0.3 IoU ratio with at least one ground-truth object bounding box. To train the Global model we optimize the sum of independent log-losses with an SGD algorithm. We initialize the model with the ImageNet pre-trained network [Oquab et al., 2014]. The learning rate of SGD is set to 0.0001, momentum – to 0.9, weight decay – to 0.0005.

### 5.2.3 Pairwise model

The number of candidates from one image that our Pairwise model can process is quite limited due to the complexity of the inference procedure. To select the “good” candidates out of the thousands produced by the selective search [Uijlings et al., 2013] we use the non-maximum suppression (with threshold 0.3) on top of the scores

provided by the Local model. We find that 16 candidates per image produced this way provide good balance between quality and speed.

To construct clusters of candidate pair (edges) incorporating the layout information we use the three features representing the vertical and horizontal displacements and the ratio of the candidate sizes. To be precise, if the position of each candidate is defined by a tuple  $(x_i, y_i, w_i, h_i)$  we define the size of the candidate as  $s_i = (w_i + h_i)/2$ , its horizontal position as  $x_i^c = x_i + w_i/2$  and its vertical position as  $y_i^c = y_i + h_i/2$ . For the two candidates sorted such that  $x_i \leq x_j$  we compute the features as follows:  $f_{ij}^1 = \log(s_i/s_2)$ ,  $f_{ij}^2 = \varphi((x_j^c - x_i^c)/s_i)$ ,  $f_{ij}^3 = \varphi((y_j^c - y_i^c)/s_i)$ , where  $\varphi(x) = \text{sign}(x) \log(|x| + 1)$ . All the three features are normalized to have zero mean and unit standard deviation on the training set. We find that increasing number of clusters beyond 20 does not improve the performance.

To train the Pairwise model we assign each selected candidate a target binary label based on the maximum IoU ratio with the ground-truth bounding boxes (threshold 0.5). We form a training batch from 64 candidates coming from 4 different images. The FE part of the model is initialized from the Local model. The weights of the UN and PN were initialized randomly using zero-mean Gaussians with standard deviation 0.01. The structured surrogate objective is optimized with and SGD with momentum 0.9, weight decay 0.000005, and learning rate 0.00001. We decreased the learning rate by a factor of 10 after 4 passes over the training data.

#### 5.2.4 Combining models

**Local and Pairwise models.** We now describe the process of computing the scores of the joint model. First, we compute the scores of the Local model for all candidates and perform the non-maximum suppression [Felzenszwalb et al., 2010a] using NMS threshold 0.3. The 16 top-scoring detections produced by NMS are then used as input for the Pairwise model. This number of candidates is sufficient on scenes with a few people, but can cause the drop of recall for crowded scenes (especially for some scenes of Casablanca dataset). To compensate for this drop, we combine scores produced by the Local and Pairwise models  $s_l, s_p$  respectively. For candidates with both scores

existing, we use the affine combination  $s_{lp} = \alpha s_l + (1 - \alpha)s_p + \beta$ . For candidates with the score of the Pairwise model non-existent, we use the score of the Local model  $s_{lp} = s_l$ . Parameters  $\alpha \in [0, 1]$  and  $\beta \in [-10, 10]$  are selected by maximizing AP on the validation set using grid search.

**Local, Pairwise and Global models.** To combine scores  $s_{lp}$  with the Global model, we associate each candidate with the output cell of the Global model having maximum IoU overlap-ratio. The score of the joint model  $s^*$  is computed as an affine combination of the detection score  $s_{lp}$  and the grid cell score  $s_g$ , i.e.  $s^* = \gamma s_{lp} + (1 - \gamma)s_g$  where  $\gamma \in [0, 1]$  is obtained by maximizing AP on the validation set.

## 5.3 Datasets

In this section we present our new head detection dataset, HollywoodHeads (HH), and discuss two other datasets we use for evaluation: TVHI [Patron-Perez et al., 2012, Hoai and Zisserman, 2014] and Casablanca [Ren, 2008].

### 5.3.1 HollywoodHeads dataset

HollywoodHeads dataset contains 369,846 human heads annotated in 224,740 video frames from 21 Hollywood movies<sup>3</sup>. The movies vary in genres and represent different time epochs. To create annotation, we have manually annotated tracks of human heads in action-rich movie clips. For each head track, head bounding boxes, i.e., the smallest axis-parallel rectangles including all visible pixels of the head, were manually annotated on several key frames. The bounding boxes on remaining frames were linearly interpolated and manually verified to be correct. In total, we have collected 2,380 clips with 3,872 human tracks, spanning over 3.5 hours of video. The dataset is divided into the training, validation and test subsets which

---

3. List of movies used in HollywoodHeads dataset. Training set: *American beauty*, *As Good As It Gets*, *Big Fish*, *Big Lebowski*, *Bringing out the dead*, *Capote*, *Clerks*, *Crash*, *Dead Poets Society*, *Double Indemnity*, *Erin Brockovich*, *Fantastic 4*, *Fargo*, *Fear And Loathing In Las Vegas*, *Fight Club*. Validation set: *Five Easy Pieces*, *Forrest Gump*, *Gang Related*. Test set: *Gandhi*, *Charade*, *I Am Sam*.



have no overlap in terms of movies<sup>3</sup>. Given the redundancy of consequent video frames, we have temporally subsampled videos in the validation and test subsets. In summary, the training set of HollywoodHeads contains 216,719 frames from 15 movies, the validation set contains 6,719 frames from 3 movies and the test set contains 1,302 frames from another set of 3 movies. Human heads with poor visibility (e.g., strong occlusions, low lighting conditions) were marked by the “difficult” flag and were excluded from the evaluation. The HollywoodHeads dataset is available from <http://www.di.ens.fr/willow/research/headdetection>.

### 5.3.2 TVHI dataset

The extended TV Human Interaction (TVHI) dataset [Patron-Perez et al., 2012, Hoai and Zisserman, 2014] consists of 1,313 frames of TV show episodes annotated with bounding boxes of human upper bodies. Frames are split into the two sets: 599 for training and 714 for testing. To evaluate our method using upper-body annotation, we have regressed head bounding boxes to upper-body bounding boxes [Mathias et al., 2014]. The parameters of regression were tuned on the TVHI training subset for each tested method.

### 5.3.3 Casablanca dataset

The Casablanca dataset [Ren, 2008] contains 1,466 frames from the movie “Casablanca”. The frames are annotated with head bounding boxes, however, the annotation of frontal heads is typically reduced to face bounding boxes and, therefore differs in the scale and aspect ratio from the HollywoodHeads annotation. Given some mistakes in the original annotation in [Ren, 2008], we have added missing bounding boxes for the heads of all people in the foreground. Similar to the TVHI dataset, we have applied bounding-box regression to compensate dataset-specific annotation biases.

## 5.4 Experiments

This section presents our experimental results. First, we demonstrate the effect of different combinations of proposed models (Section 5.4.1) and provide the comparison with the state-of-the-art (Section 5.4.2). Section 5.4.3 compares different architectures of the Local model. We then justify the need of our new large dataset for training (Section 5.4.4) and show improvements in computational complexity that can be achieved with the Global model (Section 5.4.5).

To evaluate the detection performance, we use the standard Average Precision (AP) measure based on the Precision-Recall (PR) curve [Everingham et al., 2010]. Detections having high overlap ratio with the ground truth ( $\text{IoU} > 0.5$ ) are considered as true positives. Multiple detections assigned to the same ground truth are penalized and declared as false positives. Matches to “difficult” head annotations are ignored in the evaluation, i.e. such detections are considered neither as true positives nor as false positives.

### 5.4.1 Results of context-aware models

We compare performance of the following four models: the Local model (Sec 5.1.1), the combination of the Local and Global models (Section 5.1.2), the combination of the Local and Pairwise models (Section 5.1.3) and the combination of all the three proposed models. The performance of head detection is evaluated on HollywoodHeads, Casablanca and TVHI datasets. Qualitative results of the Global and Pairwise models are illustrated in Figures 5-2 and 5-4 respectively. Table 5.1 presents quantitative results for all models. We observe that the Global and Pairwise models consistently improve the performance of the baseline Local model. The combination of all three models demonstrates the best performance on all three datasets.

### 5.4.2 Comparison with the state-of-the-art methods

We compare our approach against several baselines: the CNN-based object detector [Girshick et al., 2014] (R-CNN), DPM-based face detector [Mathias et al., 2014]

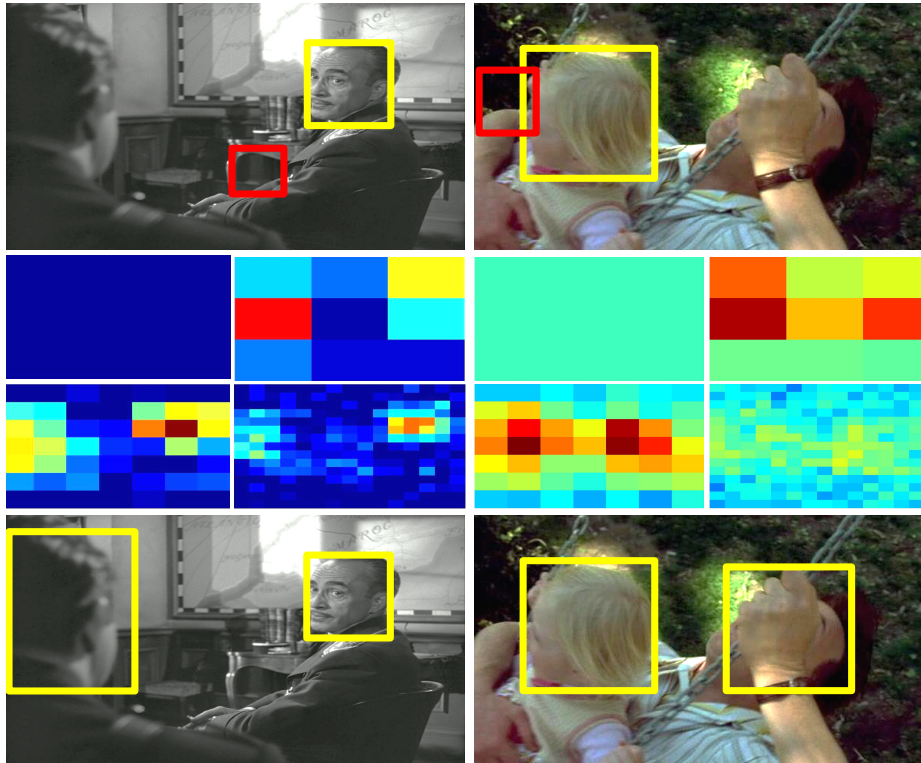


Figure 5-2 – Qualitative results for the Global model. The top row shows detections produced by the Local model. The middle row illustrates the multi-scale score map produced by our Global model. Red color correspond to high score values for the “head” class, blue color – to low score values. The bottom row demonstrates detections by the combination of the Local and Global models.

Test set	Local	Local Global	Local Pairwise	Local Pairwise Global
Casablanca	71.8	72.1	72.5	<b>72.7</b>
HH	71.8	72.5	71.9	<b>72.7</b>
TVHI	87.8	89.5	89.2	<b>89.8</b>

Table 5.1 – Performance (% AP) of different context-aware models on three datasets: Casablanca, HollywoodHeads (HH) and TVHI.

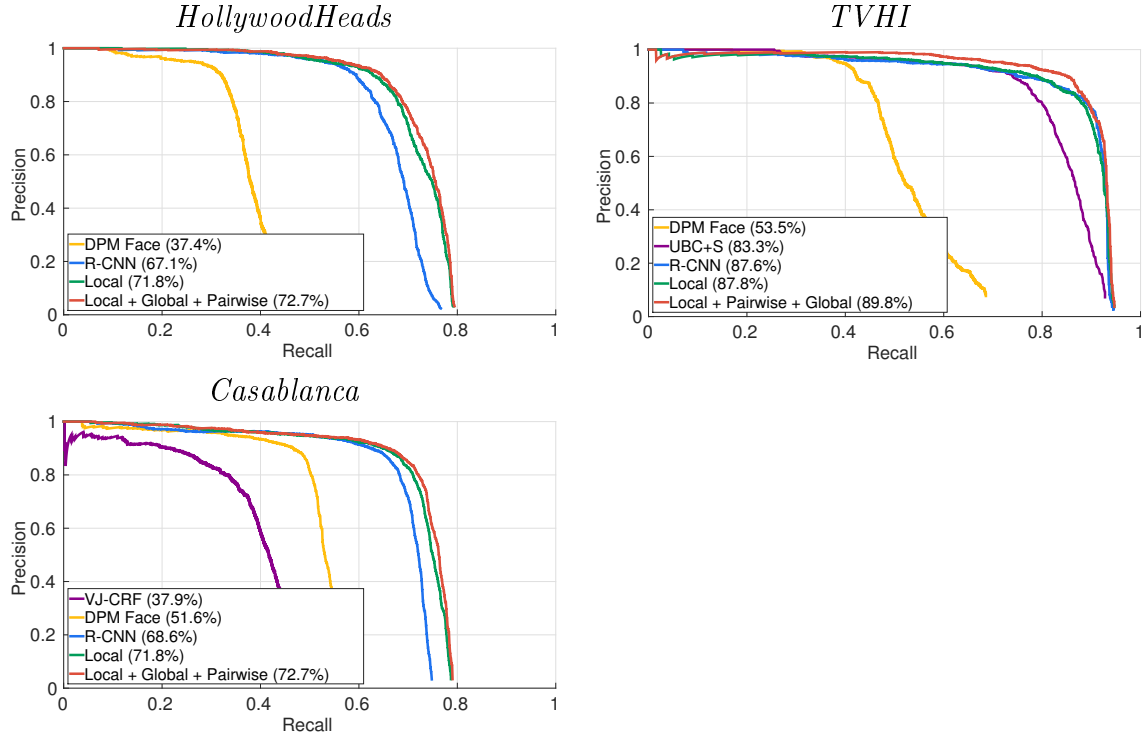


Figure 5-3 – Results of our method compared to the state-of-the-art on the three datasets.

(DPM Face) as well as other methods reporting results on TVHI [Hoai and Zisserman, 2014] (UBC+S) and Casablanca [Ren, 2008] (VJ-CRF). We have trained R-CNN<sup>4</sup> object detector on human heads using the training subset of HollywoodHeads dataset. The CNN model was first fine-tuned on all region proposals used to train our Local model. Given memory limitations, the SVM phase of R-CNN training was done on a subset of training images. For the DPM-based face detector we have used the vanilla DPM model provided by [Mathias et al., 2014]. Results of other methods were taken from original papers.

4. <https://github.com/rbgirshick/rcnn>

Results of all compared methods are presented in Figure 5-3. Our joint model outperforms other methods on all three datasets. Consistently with other recent evaluations, we observe the advantage of CNN-based methods compared to other baselines. As expected, methods trained to detect faces achieve lower recall on the head detection task given the large variation of view points in natural images. Our method significantly outperforms R-CNN on two out of three datasets and performs slightly better than R-CNN on the TVHI dataset.

Note that our evaluation on the Casablanca dataset differs from [Ren, 2008] due to the improved annotation and the use of VOC evaluation procedure. Animated results of our method are available from the project web-page <http://www.di.ens.fr/willow/research/headdetection>.

### 5.4.3 Architectures of the Local model.

In this section we compare the performance and speed of different architectures of the Local model. We consider AlexNet architecture [Krizhevsky et al., 2012], VGG-S [Chatfield et al., 2014], VGG-verydeep-16 [Simonyan and Zisserman, 2015] provided with the MatConvNet framework [Vedaldi and Lenc, 2015]<sup>5</sup> and the model of [Oquab et al., 2014]. All models were pre-trained on the ImageNet dataset [Deng et al., 2009] and fine-tuned on the training set of the HollywoodHeads dataset as the Local model. In Table 5.2 we report values of AP produced by different models together with the train/test speed. We measure the speed as the number of image patches processed per second. For each model we choose the size of the training batch such that the training speed is maximal. In all cases it happens to be the maximum batch size that fits into the GPU memory. Experiments of this section were run on NVIDIA TITAN X with 12G RAM.

---

5. <http://www.vlfeat.org/matconvnet/pretrained/>

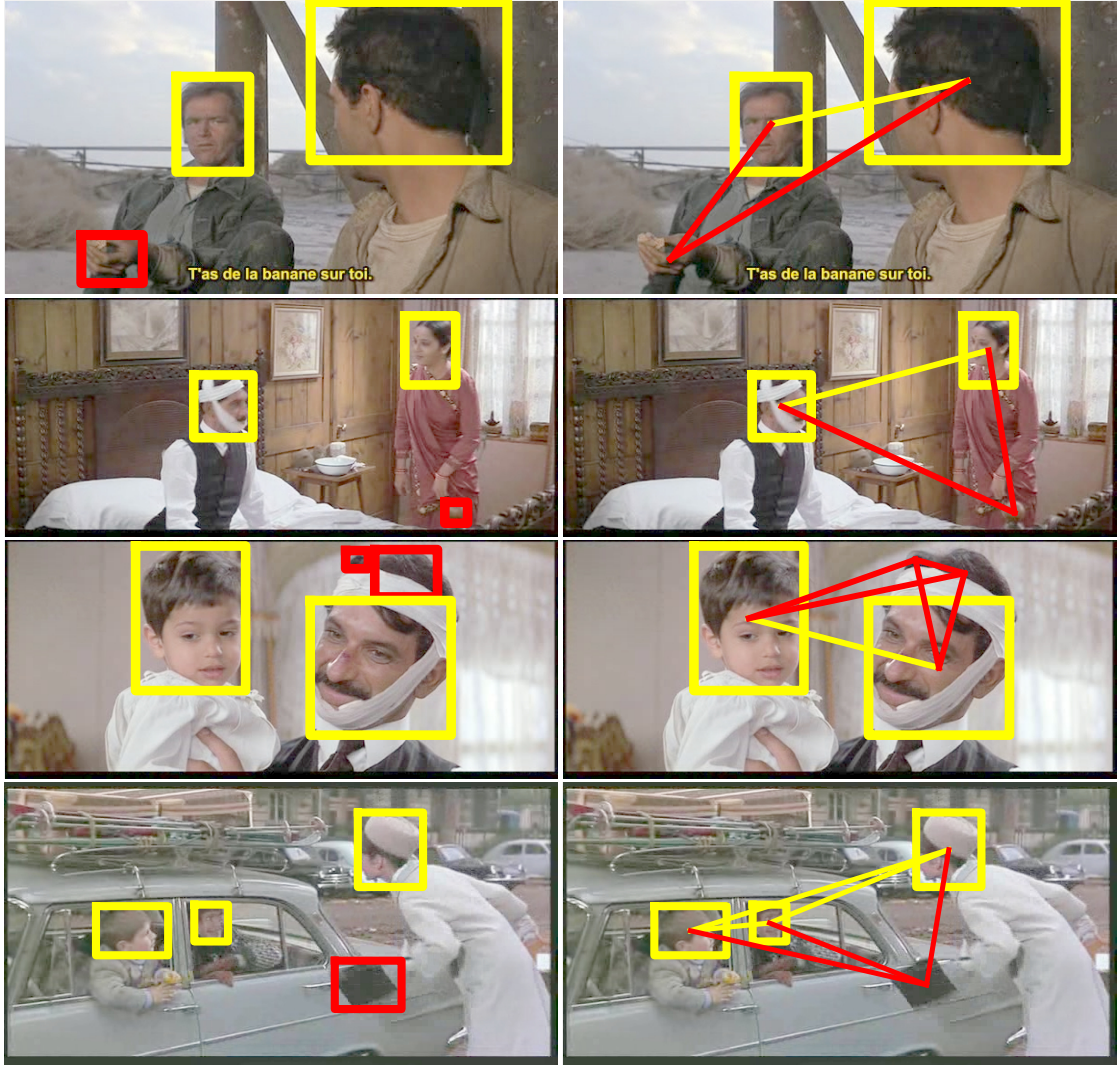


Figure 5-4 – Qualitative results for the Pairwise model. For each video frame we show results of the Local model (left) and the Pairwise model (right). For both methods we choose the score thresholds such that the precision equals the recall on the validation set. The plotted bounding boxes show the detections with the scores above the selected thresholds. Yellow boxes correspond to correct detections, red – to false positives. For the Pairwise model we show the strength of links between the candidates detected by at least one method. Links above a strength threshold (attractive) are plotted yellow and others – red (repulsive).

#### 5.4.4 Size of the training set

In this experiment we analyze the amount of training data required to train our models. Our full training set is constructed from 15 movies. We also examine the use of smaller training sets corresponding to the first 8 movies and the first 4 movies

	AlexNet	Oquab	VGG-S	verydeep-16
AP	76.3	76.7	77.2	<b>78.5</b>
Train speed	<b>445</b>	284	147	30
Test speed	<b>1490</b>	980	510	74

Table 5.2 – Performance (% AP) of Local models of different architectures on the HollywoodHeads validation set. Bottom lines report training and testing speed, measured by the number of image patches processed per second.

Test set	4 movies	8 movies	15 movies
Casablanca	51.2	62.5	<b>72.7</b>
HollywoodHeads	63.3	67.7	<b>72.7</b>
TVHI	88.6	88.8	<b>89.8</b>

Table 5.3 – Performance of models trained on the training sets of different sizes. We report % AP for each test set.

of the full training set respectively. We use each training set to train parameters of our full model and evaluate it on three datasets. Corresponding results are reported in Table 5.3. We observe that the amount of the training data and, maybe more importantly, its diversity significantly helps the performance.

#### 5.4.5 Complexity reduction with the Global model

Here we show that the Global model can suppress false candidates and reduce the computational complexity of R-CNN and our Local model at test time. We achieve this by transferring scores of the Global model detection proposals. We then filter out low-score candidates and thus reduce the number of candidates that have to pass through Local CNN. We evaluate the performance of detectors with different percentage of candidates remaining after the filtering. Table 5.4 presents results of this experiment. We observe that the detection performance remains high despite aggressive filtering by the scores of the Global model.

% remained	100	30	20	10	6	4	2
R-CNN	67.1	65.0	63.9	59.0	53.7	48.9	41.3
Local	71.8	68.3	66.8	60.2	53.4	48.8	41.9

Table 5.4 – Performance of the R-CNN method and of our Local model (% AP) on the test set of HollywoodHeads with different percentage of candidates remaining after filtering using the Global model.

## 5.5 Conclusion

In this chapter we have addressed the task of detecting people in still images. We proposed two context-aware CNN-based models. To train and evaluate our method, we have collected the new large-scale HollywoodHeads dataset consisting of movie frames and human head annotations. The combination of our context-aware models and the CNN-based local detector achieves state-of-the-art results on our dataset and the two existing human detection datasets, TVHI and Casablanca.

We believe that our context-aware models can be extended to tackle general object classes. In particular, the Microsoft COCO dataset [Lin et al., 2014] contains many small object classes with implied spatial constraints. Another possible direction for future work is to take into account motion information to extend our methods to perform long-term tracking.





## Chapter 6

# Tube-CNN: Modeling temporal evolution of appearance for object detection in video

In the previous chapter, we have shown how the detection performance can benefit from contextual cues in static images, such as the global image context and the pairwise context capturing relations between objects. In this chapter we explore temporal cues and construct spatio-temporal models for object detection. Indeed, static images may sometimes lack information to identify the presence of an object. Video, on the other hand, provides additional cues which can help to disambiguate the detection problem. For example, sequences with objects undergoing dynamic occlusions and rotations are structured and contain space-time patterns with object-specific properties, see Figure 6-1. We argue that dynamic changes of appearance provide discriminative information that can boost object detection in difficult situations.

Object tracking in video can compensate some failures of object detectors by imposing temporal smoothness. Tracking, however, often assumes the temporal continuity of appearance and can be misleading when such assumptions break due to strong occlusions or fast motions. In contrast to tracking, our goal in this paper is to learn discriminative models for the *temporal evolution of object appearance* and to use such models for object detection.

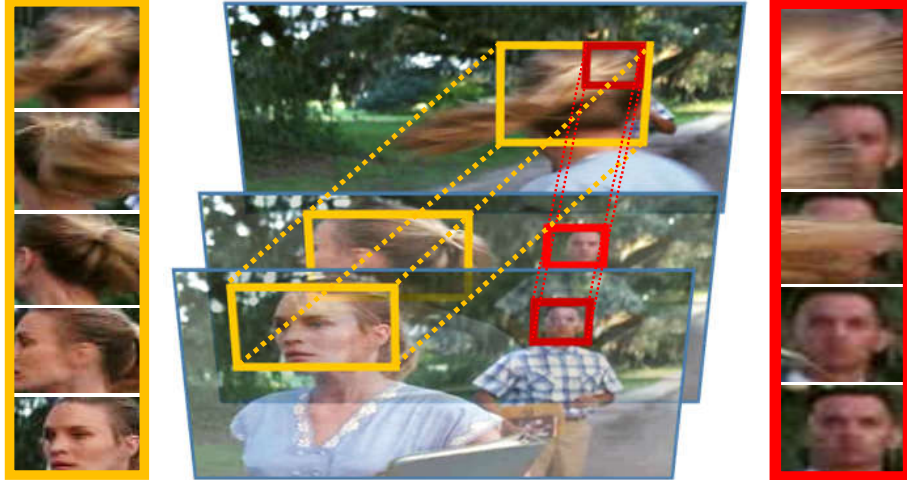


Figure 6-1 – Space-time object tubes used in our work to capture evolution of object appearance over short sequences of video frames.

To model the temporal evolution of appearance, we introduce space-time *tubes* corresponding to sequences of object bounding boxes in consecutive frames. We propose two CNN architectures for generating and classifying tubes, respectively. Our tube proposal network (TPN) first generates a large number of spatio-temporal tube proposals maximizing the objectness score and guaranteeing high recall for the ground truth object tubes. The Tube-CNN implements a tube-level object detector in the video using tube proposals as input. Examples of tubes are shown in Figure 6-1.

People are among the most frequent and difficult objects in images and video. Changes in clothing, poses and hair-style as well as frequent occlusions and dynamic variations of appearance make person detection particularly challenging. In this work, we pay close attention to this object class. To span the variety of dynamic scenes and activities, we consider people in Hollywood movies and address detection of person heads as in Chapter 5. To train our models we use the recent HollywoodHeads dataset presented in Section 5.3 and extend it with the annotation of head tracks in 3,8 hours of video from 21 different movies. We test our method on two datasets and report significant improvements compared to the state of the art. We also evaluate and demonstrate advantages of our method in difficult scenes with strong dynamic occlusions.

The following presentation in this chapter is organized as follows. Section 6.1 de-

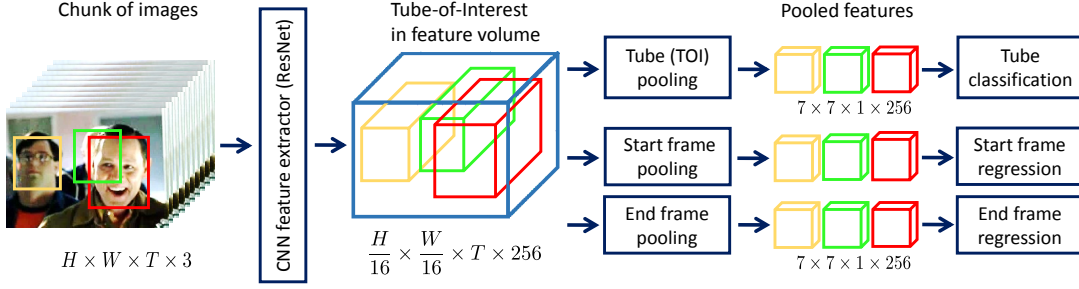


Figure 6-2 – Architecture of Tube-CNN for object detection. The input is a chunk of images and a set of tube proposals. The model starts with the CNN feature extractor to get a spatio-temporal feature volume. The following network splits into the three branches: tube classification (starts with Tube-of-Interest, TOI, pooling), regression on the start and end frames, which begins with the frame-level Region-of-Interest pooling, ROI.

scribes in detail our Tube-CNN model. In Section 6.2 we present our two approaches to generate tube proposals and describe our Tube Proposal Network (TPN). Section 6.3 explains our experimental setups and demonstrates our results. Section 6.4 concludes the chapter.

## 6.1 Tube-CNN for object detection

We now describe the Tube-CNN model for detecting objects in video. Our model operates on short chunks of consecutive video frames. Instead of rectangular image regions, our elementary units are spatio-temporal tubes spanning several frames. In this section, we assume that a set of candidate tubes (*tube proposals*) is provided and describe our Tube-CNN that classifies and refines them.

The number of all possible tubes is huge and does not allow to consider them exhaustively. In this work, we consider only linear tubes, i.e., tubes corresponding to uniform linear motion in the video. Such an approximation is reasonable only locally for small chunks of consecutive frames. In what follows, we always represent a linear tube by the two rectangles on its first and last frames. We study the effect of different tube lengths in Section 6.3.5.

### 6.1.1 Architecture

Tube-CNN is a convolutional neural network operating on chunks of consecutive frames and the corresponding sets of tube proposals. For each proposal, the model outputs the class scores and the refined proposal position. Tube-CNN is an end-to-end model consisting of three main blocks: CNN feature extraction, tube classification and tube regression. The overall network architecture is shown in Figure 6-2.

**CNN feature extractor.** The first block of the network extracts a feature map independently for every frame in the input chunk of frames. Extracted features are stacked along the temporal dimension to form a spatio-temporal feature volume. To achieve good performance, we reuse the idea of sharing computations from first CNN layers among all proposals [He et al., 2014, Girshick, 2015]. For the architecture on this block, we can reuse most of single frame CNNs (we try CaffeNet<sup>1</sup> and ResNet [He et al., 2016]).

**Tube classification.** The classification starts with Tube-of-Interest (TOI) pooling on the feature volume. Within the TOI-pooling layer, we first map the coordinates of tube proposals to subvolumes of features. Next, we max-pool within each frame to obtain a fixed-size feature map. The last stage consists in the temporal aggregation of the maps coming from all the frames of a chunk. We have tried max-pooling, average pooling, and 1-dimensional convolution. Our conclusion is that temporal max-pooling works best (without significant difference although) and we use it in what follows. Note, that if temporal aggregation is done by max or average pooling, the number of parameters in Tube-CNN does not depend on the chunk length  $T$ , and the learned model can be readily applied to chunks of arbitrary length.

After the TOI-pooling, we obtain a feature map of a fixed size for each tube proposal. This map is fed into the final part of the classification network. For example, in the case of ResNet-101 pipeline [He et al., 2016], it is the 4-th block. The tube classification branch ends with the cross-entropy loss.

---

1. [https://github.com/BVLC/caffe/tree/master/models/bvlc\\_reference\\_caffenet](https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet)

**Tube regression.** Tube regression adjusts spatial positions of tube proposals to better localize the object. Tube regression consists of two networks for bounding box regression [Girshick, 2015] at the two ends of the tube. Both branches of the tube regression start with the ROI pooling [Girshick, 2015] on the corresponding frame (the start and end frames of the chunk). Tube regression branches of the network end with the smooth L1 loss [Girshick, 2015].

### 6.1.2 Supervision

To train the Tube-CNN model, one needs a dataset with annotated object tracks. Details of the datasets used for our training are given in Section 6.3.1.

At the training time, Tube-CNN model takes a chunk of  $T$  consecutive frames and a set of tube proposals as inputs. Given the chunk, we find a set of ground-truth tracks passing through it. Each ground-truth track is approximated within a chunk by a linear ground-truth tube. We use these tubes to assign labels to tube proposals in this chunk.

Label assignment for a tube proposal is based on the tube overlap between the tube proposal and the ground-truth tubes. We define the *tube overlap* between two tubes as minimum of the spatial Intersection-over-Union (IoU) overlaps at their ends. Each sampled tube proposal of a training chunk is assigned with a label: positive (class label of an object) and negative (background). Tube proposals having tube overlap  $\theta_{GT} \geq 0.5$  with the best matched ground-truth are marked as positives. Tube proposals with  $0.1 \leq \theta_{GT} < 0.5$  are marked as negatives. Additional details on the training setup are given in Section 6.3.2.

## 6.2 Generating tube proposals

The Tube-CNN model uses tube proposals as input. Tube proposals are defined as the sequences of region proposals on consecutive video frames that hypothesize spatio-temporal positions of objects. To reduce the number of potential proposals, we consider only the ones corresponding to uniform linear motion.

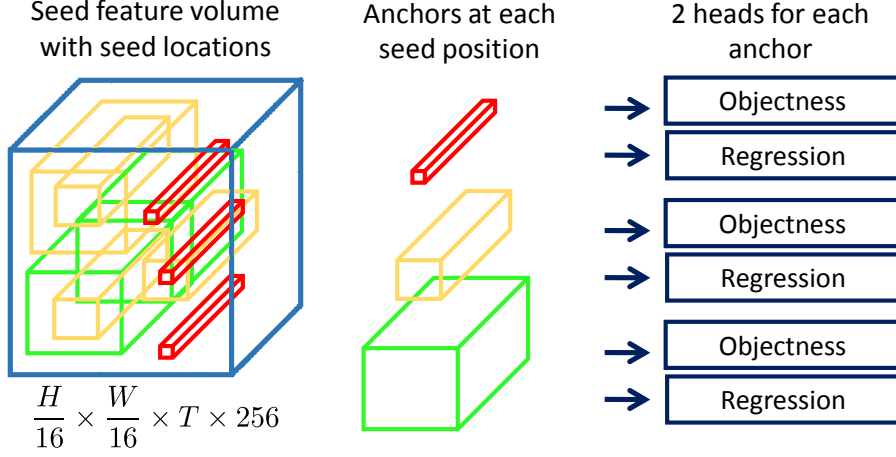


Figure 6-3 – Architecture of Tube Proposal Network (TPN). Tube anchors are plotted in red, yellow and green.

In this section, we introduce a CNN model that generates tube proposals given an input chunk of frames: Tube Proposal Network. We also propose a tracking-based baseline to build tube proposals on top of single-frame region proposals. Evaluation of different tube proposals is reported in Section 6.3.3.

### 6.2.1 Tube proposal network

We now present our Tube Proposal Network (TPN). This is a CNN model that can generate a small set of high quality class-agnostic tube proposals used for object detection.

**Architecture.** Tube Proposal Network is a fully convolutional network [Long et al., 2015], taking a chunk of consecutive frames as input and producing a set of tube proposals. Each tube proposal has an objectness score and a refined region location. Figure 6-3 illustrates the TPN model.

Similar to the Tube-CNN model, TPN passes input chunks of consecutive frames through several CNN layers. The output feature maps are stacked along the temporal dimension and passed through a  $3 \times 3 \times 3$  volumetric convolutional layer (conv3D) [Tran et al., 2015] to form a feature volume.

We refer to all spatial positions of the feature volume as *seed locations* and the corresponding features as *seed features*. Each seed feature vector is associated with a set of  $K$  reference tubes, which we call *tube anchors*. Tube anchors sharing the same

center axis in the feature volume correspond to the same seed location as well as the same seed feature vector.

Each seed feature vector is then passed through an *anchor network* to produce two outputs per each of the  $K$  tube anchors: the objectness score and parameters for the tube regression. In our setting, the anchor network consists of two fully-connected layers. Finally, we obtain the tube proposals by collecting the regressed locations of all the tube anchors. Scores of the tube proposals directly come from the objectness scores of the anchors.

To prune tube proposals, we apply Non Maximum Suppression (NMS) based on the objectness score. We modify the standard NMS to the case of tubes by replacing the spatial IoU overlap ratio with the tube overlap ratio.

**Training.** TPN and Tube-CNN have many common properties, thus, their training procedures are similar. Compared to Tube-CNN operating with tube proposals, TPN is trained on mini-batches of tube anchors.

Each seed feature vector is connected to  $K$  tube anchors. Tube anchors have the same center axis with the receptive field of the seed features. In the related work [Ren et al., 2015], anchors for region proposals correspond to several scales and aspect ratios. While adopting this setup to tubes, we have tried to add anchors with motion between the start and end regions, but it has not improved the recall while significantly increasing the number of anchors, thus complexity at the testing stage. We therefore use tube anchors, with spatial location fixed, but with varying scales and aspect ratios. Figure 6-4 shows examples of initial tube anchors and the corresponding final TPN proposals. We show that TPN can regress an anchor without motion to a tube proposal with motion, because the receptive field of the anchor is typically much larger compared to the size of the anchor, thus, TPN can get enough information about the spatio-temporal neighborhood.

To train TPN, we label each tube anchor as either positive (object) or negative (background). The label assignment procedure is similar to one of the Tube-CNN training procedure. The only difference is that the supervision of TPN is class-agnostic. Positive anchors have tube overlap ratio  $\theta_{GT} \geq 0.5$  with a ground-truth





Figure 6-4 – Examples of TPN proposal. Examples of TPN proposals. Tube anchors and final proposal are plotted by green and magenta respectively.

tube of any object class. Anchors having  $\theta_{GT} \leq 0.3$  with all ground-truth tubes are labeled as negatives.

### 6.2.2 Tube proposals by tracking box proposals

We introduce a tracking-based method to produce tube proposals from region proposals (which we refer to as *box proposals*) of single frames. Given a chunk of  $T$  consecutive frames, we generate box proposals of the start frame by the Selective Search method [Uijlings et al., 2013]. For each of those box proposals, we hypothesize its corresponding positions on the end frame by using a tracker. Specifically, we use the Kanade-Lucas-Tomasi (KLT) tracker [Shi et al., 1994] to obtain point tracks between the start and the end frames of the chunk. Each box proposal is associated with a set of interior point tracks, corresponding to a set of movement directions. Those directions are then clustered into  $N_b$  directional bins. We apply RANSAC on  $N_h$  most populated bins to construct  $N_h$  candidate temporal paths in the interval of  $T$  frames<sup>2</sup>. Tube proposals are then constructed by linearly moving the box along all hypothesized temporal paths.

The above process could be built upon the output of any box proposal method. To have a rough comparison with [Kang et al., 2016a], we generate tube proposals

---

2. We use  $N_b = 16$  and  $N_h = 4$  in our experiments

from high-scoring detections produced by our Fast R-CNN baseline<sup>3</sup>.

## 6.3 Experiments

In this section, we present and analyze experimental results of our method. We first introduce video datasets for object detection in Section 6.3.1. We then clarify our experimental setup in Section 6.3.2. Section 6.3.3 evaluates the quality of tube proposals. Section 6.3.4 presents our main results for object detection and compares them to the state of the art. In Section 6.3.5 we present an ablation study analyzing parameters and design choices of our method.

### 6.3.1 Datasets

**HollywoodHeads dataset.** HollywoodHeads<sup>4</sup> is a recent large-scale dataset for person head detection with video clips from 21 movies. The original dataset provides head annotation for all the frames. To train tube models in this work, we extend the annotation by combining detections to form tracks. Our extended HollywoodHeads dataset contains 331 746 video frames with complete annotation of head tracks, split into 216 719, 67 181 and 47 846 frames for training, validation and test respectively. For evaluation and comparison to the previous work, we keep the test set of [Vu et al., 2015] with 1 302 annotated frames<sup>5</sup>.

Occlusions are among the most challenging factors for object detection. To study the detection performance under occlusions, we select a subset of video clips, named HollywoodHeads-Hard, with partially visible heads. HollywoodHeads-Hard is automatically composed from clips with multiple annotated heads, where ground-truth bounding boxes have significant overlap on the horizontal axis. In total, we obtain 266 difficult clips and select one frame for testing from each of them.

**Casablanca dataset.** In addition to HollywoodHeads, we evaluate head detection on the Casablanca dataset. This dataset was introduced in [Ren, 2008] and extended

---

3. This baseline is described in details in Section 6.3.4

4. <http://www.di.ens.fr/willow/research/headetection/>

5. We have removed 10 test frames from the original test set, because they belonged to short video clips of length less than 10 frames.

in Section 5.3.3 with the annotation of missing ground-truth heads.

**ImageNet VID dataset.** We also evaluate our method on the multi-class object detection benchmark. ImageNet VID [Russakovsky et al., 2015] is a large-scale dataset for object detection providing complete annotation of 30 object classes in 1.3 million video frames. The dataset contains 5 354 short videos split into 3 862 videos for training, 555 videos for validation and 937 videos for testing. The 30 object classes of the ImageNet VID dataset are a subset of 200 classes used in ImageNet DET challenge. As the ground-truth for the test set is not published, we evaluate and compare our results on the validation set with 176 126 frames.

**Youtube-Objects dataset.** Youtube-Objects (YTO) is a video dataset for object localization providing annotation for 10 object classes on a sparse set of key frames [Prest et al., 2012]. The 10 object classes in YTO form a subset of 30 object classes in ImageNet VID. We use YTO for additional evaluation of the models pretrained on ImageNet VID.

### 6.3.2 Training details

**Base networks.** In our experiments, Tube-CNN is initialized either from CaffeNet [Girshick, 2015] or from Resnet-101 [He et al., 2016] models pre-trained on ImageNet [Deng et al., 2009]. The TPN model is based on the CaffeNet architecture.

*CaffeNet.* Tube-CNN initialized from CaffeNet has a CNN feature extractor block composed of the first 5 convolutional layers. The last fully-connected (FC) layers are used for tube classification and tube regression. The TOI pooling used for tube classification, and the ROI pooling used for tube regression are applied to the feature volume created by stacking *conv5* feature maps. As described in Section 6.1.1, our TOI pooling consists of: spatial max-pooling and temporal max-pooling. Similarly to [Girshick, 2015], the spatial max-pooling produces fixed-size  $6 \times 6$  output features. On the other hand, temporal max-pooling collapses all values along temporal dimension into one max value. Therefore, for every input tube proposal, TOI pooling layer outputs a 256-channels feature map with a fixed spatio-temporal extent of  $6 \times 6 \times 1$ .

Each feature map is flattened into a 9126-dimensional feature vector, which is then passed through FC layers to compute the classification score. We use the same spatial hyper-parameters of  $6 \times 6$  for ROI pooling. Given an input tube proposal, the ROI pooling layer outputs two 9126-dimensional feature vectors corresponding to the two ends of the tube proposal. We pass the two feature vectors through FC layers to produce regression parameters for the beginning and the end of the tube.

*Resnet-101.* The TOI pooling and the ROI pooling layers are inserted between the third and the fourth blocks of the network, i.e. *conv\_4x* and *conv\_5x* of Table 1 in [He et al., 2016]. In detail, 91 convolutional layers before and including *conv\_4x* belong to CNN feature extractor block of Tube-CNN. All layers after and including *conv\_5x* construct tube classification and tube regression blocks. The spatial extent of both TOI and ROI pooling layers has a fixed-size of  $7 \times 7$ . The temporal max-pooling is done similarly as for CaffeNet. Without being flattened, pooled feature maps are passed into *conv\_5x* followed by an average pooling layer. In our setting, output size of *conv\_5x* is different from the one of the original network. We adapt to this change by adjusting kernel size of the average pooling layer to  $4 \times 4$ . The last FC layers of tube classification and tube regression blocks take the average pooling output to produce final object score and position.

**Training parameters.** We optimize Tube-CNN and TPN with the stochastic gradient descent (SGD) algorithm with momentum 0.9 and weight decay 0.0005 on mini-batches [Krizhevsky et al., 2012]. The classification and regression are trained with the cross-entropy loss and smooth L1 loss [Girshick, 2015], respectively. We fix tube length  $T = 10$  for all tube models. This design choice is analyzed in Section 6.3.5.

**Hard negative mining.** Hard negative mining enables training on difficult samples and has been proven effective for object detection with HOG and DPM models [Dalal and Triggs, 2005, Felzenszwalb et al., 2010a] and more recently with CNN methods [Arandjelović et al., 2016, Shrivastava et al., 2016]. In our experiments, we find that a few iterations of hard negative mining improves detection performance

for all of our models. We here describe the procedure in an abstract manner. The term *proposal* stands for box proposal in case of Box-CNN, and tube proposal in case of Tube-CNN. The overlap ratio between proposal and ground-truth is defined separately for each model. An iteration of hard negative mining consists of three steps: (1) a forward pass on a subset of input proposals to find hard negatives; (2) composition of several training batches with hard negatives and random positives; (3) training the network on the constructed batches. We define hard negatives as high-scoring false positives with no overlap to any ground-truth. The hard negative training batch contains 25% positives and 50% hard negatives at most. The rest of the batch is composed of negatives having overlap ratio to ground-truth in the range  $[0.1, 0.5)$ . Section 6.3.5 analyzes the effect of hard negative mining on the detection performance.

### 6.3.3 Evaluation of tube proposals

In this section, we evaluate the quality of tube proposals produced by different methods on HollywoodHeads and ImageNet VID datasets.

**Tube proposal baselines.** Tube proposals have been previously proposed for the task of action detection in video. We here evaluate and compare two state-of-the-art methods for action proposals [Gemert et al., 2015, Oneata et al., 2014]. For the 3D-Proposal method [Oneata et al., 2014], we extract 10,000 tube proposals for every shots of 10 frames. For APT [Gemert et al., 2015], we use longer shots due to the length constraint imposed by the method.

**Evaluation of recall.** Object proposals should maximize the coverage of ground-truth object bounding boxes. We evaluate the quality of tube proposals by examining their recall on both frame level and video level. We refer to the two measures as *box-recall* and *tube-recall*, respectively. Tube-recall is the percentage of ground-truth tubes having tube overlap-ratio more than 0.5 with at least one tube proposal (tube overlap-ratio is defined in Section 6.1.2). To compute box-recall, we first split tube proposals

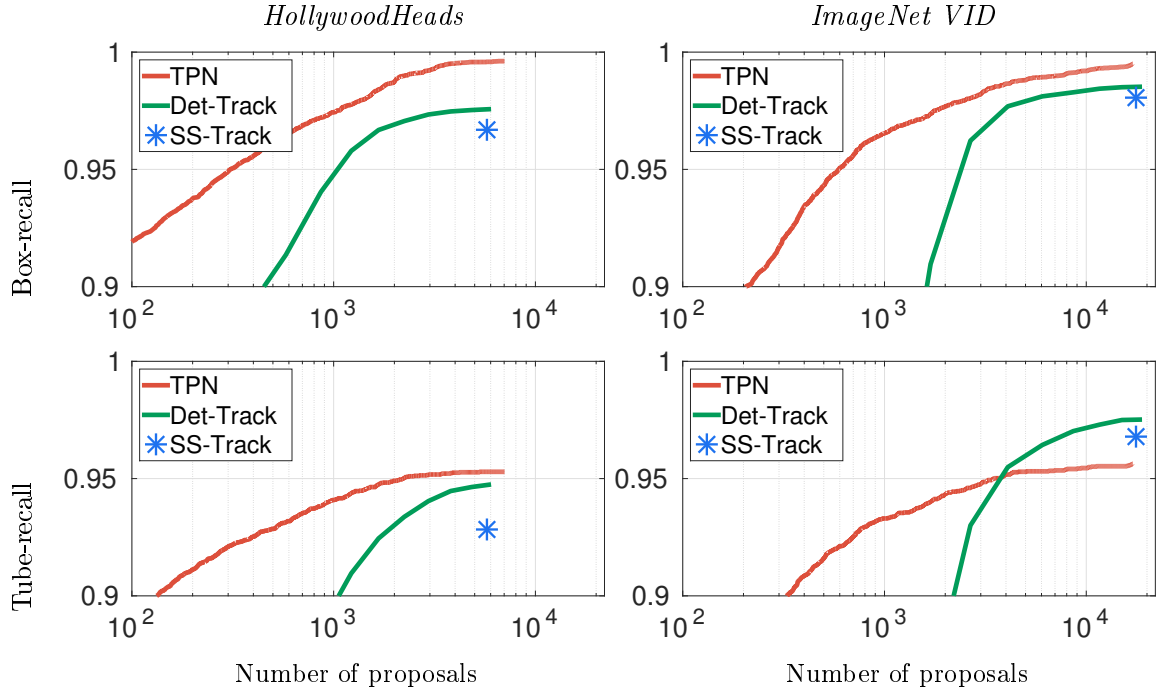


Figure 6-5 – Box-recall and tube-recall *vs.* average number of tube proposals per chunk on HollywoodHeads and ImageNET VID datasets.

into per-frame box proposals. Box-recall is then computed as the percentage of still images ground-truth bounding boxes having IoU overlap-ratio more than 0.5 with at least one box proposal.

We evaluate the recall on HollywoodHeads and ImageNet VID datasets in Table 6.1. For each dataset we select a validation set of about 3000 frame chunks. We first observe that action proposals do not achieve satisfactory level of recall for object detection. Due the low performance, we omit action proposals from further evaluation. We next evaluate the two track-based tube proposals introduced in Section 6.2.2. The proposals based on Selective Search and single-frame object detections are denoted by SS-Track and Det-Track respectively. Both methods obtain high recall at the cost of the large number of tube proposals.

For TPN we experiment with different numbers of top-scoring tube proposals after tube-NMS. For HollywoodHeads TPN outperforms all other methods in both Tube recall and Box recall. For ImageNet VID TPN achieves slightly worse recall compared to SS-Track and Det-Track. This might be due to the large variation of

Method	Avg.# TP	Tube Rec.	Box Rec.
3D Proposals [Oneata et al., 2014]	10K	55.3	61.3
APT Proposals [Gemert et al., 2015]	4.2K	48.7	62.1
SS-Track (ours)	6.7K	92.8	96.7
Det-Track (ours)	6K	94.8	97.6
TPN (ours)	100	91.1	91.9
	500	93.2	96.2
	1.6K	94.8	98.3
	5.2K	<b>95.3</b>	<b>99.6</b>

(a) HollywoodHeads

Method	Avg.# TP	Tube Rec.	Box Rec.
SS-Track (ours)	17K	96.8	98.1
Det-Track (ours)	17K	<b>97.5</b>	98.5
TPN (ours)	100	81.3	83.2
	1K	93.3	96.5
	5K	95.3	98.8
	10K	95.5	<b>99.2</b>

(b) ImageNet VID

Table 6.1 – Evaluation of tube proposals in terms of tube-recall and box-recall on HollywoodHeads and VID datasets. Avg.#TP stands for the average number of tube proposals per chunk of consecutive frames.

object aspect ratio in ImageNet VID. Figure 6-5 illustrates recall depending on the number of tube proposals. In all cases TPN achieves a significant reduction in the number of proposals by the cost of a minor drop in recall. This property can be used to significantly reduce computational complexity of detection at test time.

### 6.3.4 Detection results

In this section we report results for object detection. Following the PASCAL VOC detection challenge [Everingham et al., 2010], we use Average Precision (AP) measure to evaluate object detection on the level of individual frames. Given results of a tube object detector, we decompose tubes into per-frame object detections. On each frame we aggregate detections from multiple tubes using the standard NMS procedure. Resulting detections are evaluated against ground truth object bounding boxes on test frames.

Model	HH	Casa	VID
Context-aware CNN [Vu et al., 2015]	72.7	72.7	-
Kang [Kang et al., 2017]	-	-	68.4
Box-CNN	82.4	81.9	68.7
Box-CNN + track	84.7	82.1	68.9
Tube-CNN	<b>86.8</b>	<b>84.0</b>	<b>72.7</b>

Table 6.2 – Detection performance mAP (%) on the HollywoodHeads, Casablanca and ImageNet VID datasets.

**Baselines.** We compare our method to the single-frame Fast R-CNN object detectors [Girshick, 2015] which we re-train on our data. We refer to such detectors as Box-CNN to contrast with the tube-level models. Box-CNNs are trained on single frames of training videos and frame-level annotations.

Single-frame object detectors often produce noisy detections along object tracks [Kang et al., 2016b]. A common practice to improve object detection in video is to link detections on consecutive frames with tracking-by-detection methods [Everingham et al., 2006, Pirsivash et al., 2011]. We here follow the procedure in [Everingham et al., 2006] and link object detections across frames based on the consensus of KLT point tracks. This procedure enables to overcome some failures of object detectors by bridging the gaps in object tracks along time and by discarding short tracks. We apply tracking-by-detection to the Box-CNN method and denote this strong baseline as "Box-CNN + track".

**Head detection result.** In the first and second columns of Table 6.2 we compare our method to the the results presented in Chapter 5, as well as to Box-CNN and Box-CNN + track baselines on HollywoodHeads and Casablanca datasets. The Tube-CNN significantly outperforms the frame-based approach introduced in Section 5.1 by more than 10% AP on both datasets. As will be detailed in Section 6.3.5 this improvement originates both from the Tube-CNN object detectors and from the more powerful ResNet-101 base network. Our best Tube-CNN model is trained on SS-Track tube proposals. Tube-CNN achieves consistent improvement over other baseline methods using comparable base CNN architectures. Figure 6-6 illustrates

































Model									
Kang [Kang et al., 2017]	<b>84.6</b>	<b>78.1</b>	72.0	67.2	68.0	80.1	54.7	<b>61.2</b>	61.6
Box-CNN	79.1	73.7	79	66.3	68	80.3	56.8	59.1	70.8
Tube-CNN	81.1	77	<b>79.9</b>	<b>72.5</b>	<b>72.5</b>	<b>84.2</b>	<b>56.2</b>	60.3	<b>77.8</b>
									
Kang [Kang et al., 2017]	<b>78.9</b>	71.6	83.2	78.1	91.5	66.8	21.6	74.4	36.6
Box-CNN	75.6	71.7	82.5	76.9	84.1	62.9	25.1	78.3	44
Tube-CNN	77.4	<b>73.5</b>	<b>83.9</b>	<b>82</b>	<b>94.9</b>	<b>71.6</b>	<b>40.5</b>	<b>78.8</b>	<b>50</b>
									
Kang [Kang et al., 2017]	76.3	51.4	<b>70.6</b>	<b>64.2</b>	61.2	42.3	84.8	78.1	77.2
Box-CNN	76.9	58.1	53.6	56.8	72.6	52.2	88.5	78.9	77.6
Tube-CNN	<b>83.7</b>	<b>55.5</b>	56.7	58.5	<b>75.6</b>	<b>62.0</b>	<b>90.4</b>	<b>81.4</b>	<b>80.3</b>
				mAP					
Kang [Kang et al., 2017]	61.5	<b>66.9</b>	<b>88.5</b>	68.4					
Box-CNN	64	60	87.3	68.7					
Tube-CNN	<b>70.7</b>	65.6	87.4	<b>72.7</b>					

Table 6.3 – Object detection performance AP (%) on ImageNet VID validation set.











Method											Avg.
Kang [Kang et al., 2016b]	94.1	69.7	88.2	79.3	76.6	18.6	89.6	89.0	87.3	75.3	76.8
Galteri [Galteri et al., 2017]	87.8	94.8	81.7	95.1	84.3	97.5	78.0	61.0	94.8	76.8	85.2
Kang [Kang et al., 2017]	91.2	99.4	<b>93.1</b>	94.8	94.3	99.3	90.2	87.8	89.7	84.2	92.4
Tube-CNN	<b>98.2</b>	<b>100</b>	92.4	<b>97.5</b>	<b>96.5</b>	<b>99.3</b>	<b>92.1</b>	<b>96.1</b>	<b>95.8</b>	<b>89.9</b>	<b>95.8</b>

Table 6.4 – Localization performance CorLoc (%) on the YTO dataset.

results for the HollywoodHeads-Hard test set with occluded heads. Compared to the full test set, the performance on HollywoodHeads-Hard is significantly lower for all tested methods. Compared to the Box-CNN, Tube-CNN shows improvement of almost 10% AP, confirming the advantage of our method for particularly difficult scenes. Qualitative results of Tube-CNN for difficult examples of occluded heads are illustrated in Figure 6-7 and in supplementary materials.

**Detection results on ImageNet VID and YTO.** We next evaluate Tube-CNN for the more general case of multi-class object detection. Table 6.3 presents per-class

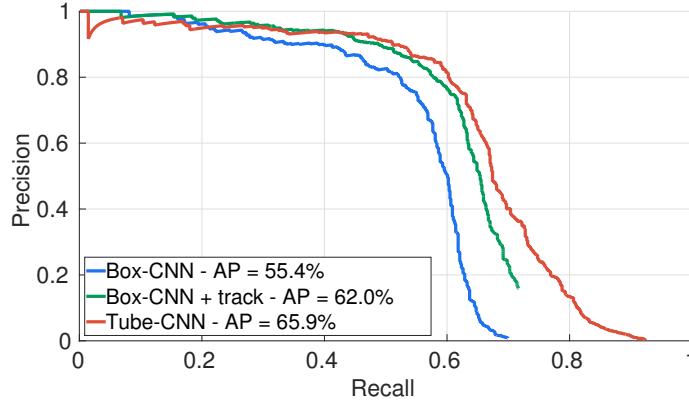


Figure 6-6 – Detection results on HollywoodHeads-Hard.

results for the validation set of the ImageNet VID benchmark. Similar to experiments on head detection in HollywoodHeads dataset, we observe the consistent improvement of the proposed Tube-CNN over Box-CNN for 28 out of 30 object categories. We, hence, conclude that our method generalizes well to a large number of object classes. The VID column of Table 6.2 compares results of Tube-CNN to baselines on the ImageNet VID validation set. As in the case of head detection, our method improves all tested baselines. In Figure 6-7 we visualize some qualitative results comparing Box-CNN and Tube-CNN.

On YTO we evaluate results using the standard Correct Localization (CorLoc) measure used for this dataset in [Prest et al., 2012, Kang et al., 2016b, 2017, Galteri et al., 2017]. Given an object class, CorLoc is defined as the percentage of positive images correctly localized according to the PASCAL criterion. Table 6.4 compares CorLoc results of the Tube-CNN method with the best results reported in the literature. Our method provides best performance on YTO even if it was not trained nor fine-tuned on this dataset.

### 6.3.5 Ablation study

In this section, we analyze design choices and parameters of our method. We also investigate and demonstrate advantages of TPN in terms of computational complexity.

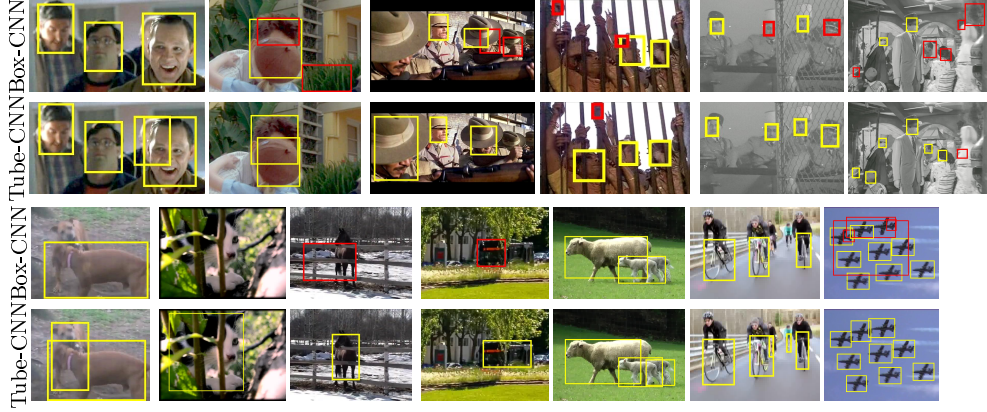


Figure 6-7 – Qualitative results for HollywoodHeads, Casablanca and ImageNet VID datasets. On each frame we illustrate results of the Box-CNN model and the Tube-CNN model. For both methods we choose thresholds such that precision equals recall on the corresponding dataset. All detections with scores above threshold are plotted. Yellow boxes: correct detections, red boxes: false detections.

**Base architectures and hard negative mining.** Box-CNN and Tube-CNN methods depend on the training schemes, base network architectures as well as the type and the amount of used tube proposals. Here we analyze the detection performance by varying each of these factors. Table 6.5 reports performance of our models on HollywoodHeads and ImageNet VID datasets under different settings.

With respect to the training scheme, we achieve consistent improvement with hard negative mining (HN) for all models. Regarding the choice of the base network, ResNet architecture (RN) provides significant improvements compared to the more shallow CaffeNet (CN) network. Our strongest models use Resnet-101 base network. For the ImageNet VID dataset we only report results of models based on Resnet-101.

We train and test Tube-CNN models using tube proposals generated by our TPN model and by tracking box proposals (i.e. SS-Track). In Table 6.5, we denote these two Tube-CNN models as "Tube-CNN + SS-Track" and "Tube-CNN + TPN" respectively. In both cases, Tube-CNN models outperform Box-CNN and other baselines. The main advantage of TPN is in test time, which grows linearly with the number of proposals. Using a smaller set of proposals, Tube-CNN + TPN is able to achieve similar (or even better) results compared to Tube-CNN + SS-Track. This together

Model	Num. Prop	CN	CN+HN	RN	RN+HN
Box-CNN	~2K	71.2	76.4	75.5	82.4
Box-CNN + track	~2K	73.1	76.9	78.0	84.7
Tube-CNN + SS-Track	~6K	<b>76.9</b>	78.3	<b>83.3</b>	<b>86.8</b>
Tube-CNN + TPN	100	74.9	78.6	81.7	83.9
	300	75.8	78.7	82.3	85.0
	2K	73.7	<b>79.1</b>	82.7	86.2

(a) HollywoodHeads

Model	Num. Prop	RN	RN+HN
Box-CNN	~2K	66.1	68.7
Box-CNN + track	~2K	67.2	68.9
Tube-CNN + SS-Track	~17K	-	<b>72.7</b>
Tube-CNN + TPN	300	69.5	70.6
	1K	70	70.9
	2K	69.8	70.9

(b) ImageNet VID

Table 6.5 – Detection performance mAP (%) on HollywoodHeads and ImageNet VID dataset. CN and RN and HN stands for CaffeNet, Resnet and hard negative mining respectively.

with the analysis of recall in Section 6.3.3 confirms the advantage the proposed TPN scheme.

Tube-CNN performs slightly better on SS-Track proposals compared to TPN. We believe that Tube-CNN + TPN detection results could be further improved if using deeper networks for the TPN model.

**Effect of the tube length.** We restrict our method to tubes with linear motion to enable a tractable approach for generating tube proposals. We believe linear tubes might be sufficient to represent short time intervals, which is the main target in this work. In Table 6.6 we report results comparing performance of Tube-CNN on the HollywoodHeads dataset for different values of tube length  $T$  (all models are based on CaffeNet without hard negative mining). We observe the best performance obtained for  $T = 10$ .

Tube length	T=1	T=5	T=10	T=15	T=20
AP (%)	71.2	76.6	<b>76.9</b>	73.4	70.1

Table 6.6 – Tube-CNN performance for different values of T.

Box-CNN	fc7 max	fc7 avg	Tube-CNN
76.4	77.0	77.3	<b>79.1</b>

Table 6.7 – Aggregation of Box-CNN and Tube-CNN.

**Advantage of feature-level aggregation.** In this experiment, we show the advantages of early feature aggregation with TOI pooling and learning features for tube classification. We compare our Tube-CNN framework with baselines of temporal aggregation of single-frame image features. In particular, we use our best CaffeNet-based single frame Box-CNN detector to extract FC7 features on all the frames of training tubes and train a linear SVM classifier based on the temporally aggregated features. SVM parameters for each setup are chosen with 5-fold cross-validation. In Table 6.7 we report results for max and average aggregation baselines. The improvement of Tube-CNN over baselines confirms the advantage of jointly learning features and their temporal aggregation as proposed in this work.

**Running time.** Table 6.8 compares running times of object detectors (in frames per second) for Tube-CNN on SS-Track and TPN proposals. To obtain results for SS-Track, we accumulate the time required for per-frame Selective Search proposals, KLT point tracking and Tube-CNN evaluation. Results for TPN are obtained by summing times for the evaluation of the TPN network, applying tube-NMS and evaluating Tube-CNN. When using ResNet-based Tube-CNN object detectors, our method with 100 TPN is about 28 times faster compared to Tube-CNN + SS-Track. The improvements of running time come from the efficient generation of TPN proposals and from the reduced number of proposals required for the Tube-CNN evaluation.

Method	SS-Track	RPN-Track	TPN			
Num.TP	~6K	~1.2K	100	300	500	1K
CN	0.3	2.3	<b>4.3</b>	3.7	3.0	2.3
RN	0.06	0.7	<b>1.7</b>	1.1	1.0	0.7

Table 6.8 – Running times (frames per second) for Tube-CNN object detectors using SS-Track proposals and different numbers of TPN proposals.

## 6.4 Conclusion

We have addressed the task of object detection in video. To this end, we proposed two CNN models for generating and classifying object tube proposals respectively. Our tube classification model achieves state-of-the-art results on all four tested datasets for object detection in video. The unified framework of the two models is both accurate and computationally efficient at the running time. Our method particularly improves object detection in difficult situations with dynamic occlusions. We therefore believe it could be highly valuable when used as input for object tracking and other challenging scenes.



## Chapter 7

# Long-Term Representation Learning for Efficient Object Detection in Videos

In this chapter, we continue addressing the problem of object detection in videos. Different to the approaches presented in Chapter 5 and Chapter 6, this work explores the feature learning aspect in videos. We aim at learning good video representation which can both improve detection performance and increase the running speed.

Motion is an important cue enabling the human visual system to perceive its environments [Cutting, 1986, Gibson, 1979]. By encapsulating motion information, video provides a rich medium for computer vision to understand and analyze the visual world. While the advent of convolutional neural networks (CNNs) has led to rapid improvements in learning spatial features, a persistent challenge remains to learn efficient representations that derive significant benefits from long-term temporal information in videos.

In this work we aim to learn video representations that incorporate multi-scale information on longer time horizons and design practical frameworks that achieve accuracy, efficiency and predictive power. While our frameworks are applicable to diverse problems, we demonstrate its application to the problem of object detection in videos. Motivated by the Gestalt principle of common fate [Ellis, 1938, Wertheimer,



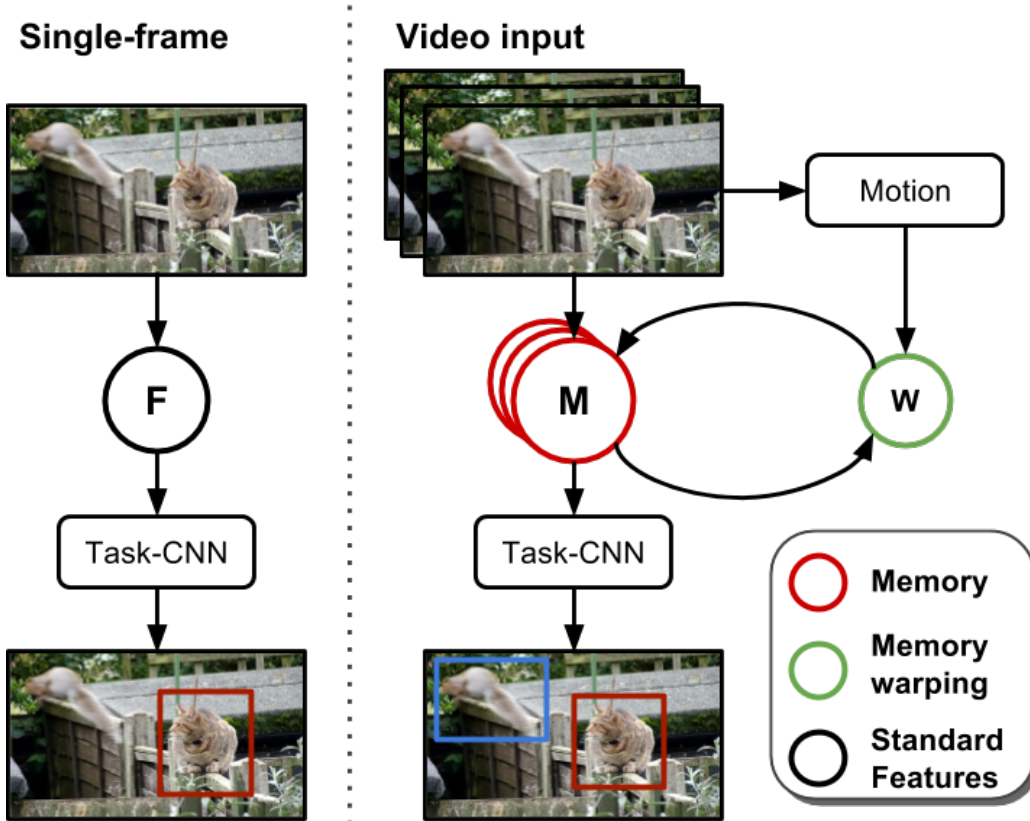


Figure 7-1 – A schematic comparison between a typical per-frame method (left) and the proposed video representation learning approach (right) for object detection in videos. We propose a multi-scale memory that efficiently aggregates image evidence over longer time horizons and also accounts for camera and object motion by feature warping, which enables learning better representations that lead to higher accuracy.

1938], we hypothesize that the temporal coherence of appearance can be used to learn more powerful video representations.

In recent years, object detection in videos has attracted significant interest with benchmarks such as ImageNet VID [Russakovsky et al., 2015] or Youtube-8M [Abu-El-Haija et al., 2016]. A popular approach has been to detect bounding boxes independently for each frame using CNN-based methods [Lin et al., 2017a,b, Liu et al., 2016, Ren et al., 2015], followed by temporal reasoning in terms of tracking [Kang et al., 2016b], re-scoring detections [Feichtenhofer et al., 2017b] and performing non-maximum suppression [Han et al., 2016]. While such methods improve over per-frame baselines, we here wish to investigate alternative and more powerful strategies to aggregate temporal information for object detection. A few recent works temporally aggregate features to improve representation power Feichtenhofer et al. [2017b], Zhu

et al. [2017b], but use a fixed set of nearby frames and do not maintain causality or efficiency. In contrast, we propose a video representation that composes information across time in an *online* fashion (see Figure 7-1), which is not only faster, but also enables *predictive* applications.

We apply our video representation to object detection in Section 7.2.1 and evaluate it on the ImageNet VID Russakovsky et al. [2015] data set in Section 7.3.1. Our proposed architectures improve over per-frame baselines by up to 2.2% in mean average precision (mAP).<sup>1</sup> We achieve state-of-the-art results close to flow-guided feature aggregation (FGFA) [Zhu et al., 2017b] *without having access to future frames* and with considerably lower runtime, while outperforming a causal variant of FGFA.

A key benefit of the online nature of our video representation is that it imparts predictive abilities, which enables novel applications. First, in Section 7.2.2, we enhance the accuracy of an online real-time detector, by leveraging a stronger but less efficient detector in another thread. While the strong detector lags due to higher latency, our memory warping enables *propagating and aligning* its representation with the real-time detector, boosting the accuracy of the latter by more than 10% mAP, with no impact on speed or online operation (see Section 7.3.3). This is non-trivial, since parallelizing standard detectors in an online setup is not straightforward. Next, our predictive warping of video representations enables *anticipating* features in future frames, which allows solving visual tasks without actually observing future images. Sections 7.2.3 and 7.3.2 demonstrate this for the novel application of anticipating objects in future frames.

Finally, we note that our contributions are architecture-independent. The speed, accuracy and predictive benefits of our online representation are available for any detection method on video inputs.

---

1. To put this accuracy gain in context for object detection, the gains from hard example mining [Wang et al., 2017] or hard positive generation [Shrivastava et al., 2016] are around 2% mAP.

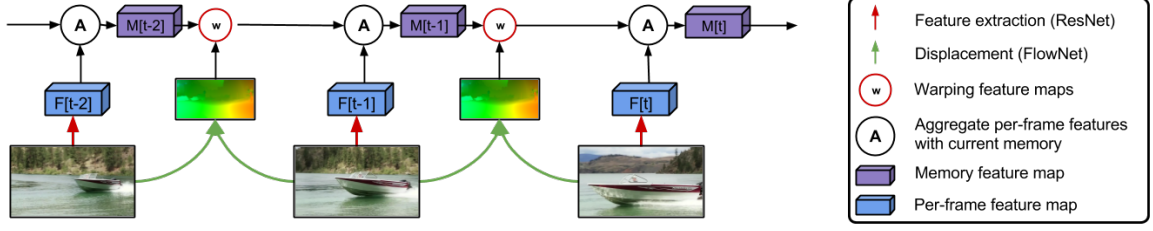


Figure 7-2 – A detailed illustration of the proposed **MemNet** running for three frames. At every time step, features from the current frame (blue) are aggregated with the memory of the previous frame (purple), either by simple averaging or with a learned adaptive weighting. Then, the memory feature map is warped via bilinear sampling based on a learned displacement field. The detection output in every frame is computed from the current memory.

## 7.1 Feature propagation via memory networks

The goal of this work is to improve the feature representation for objects in videos, by leveraging temporal information and motion. Exploiting past frames can also help predictions in the current frame when occlusions or motion blur distorts image evidence, *c.f.* [Zhu et al., 2017b]. We propose to continuously aggregate and update features over time to provide a stable and powerful representation of the scene captured by the video, which is illustrated in Figure 7-2. In Sections 7.1.1 and 7.1.2, we describe our approach to learn long-term video representations where feature memory is accumulated and transformed over time to account for observer and scene motions. Section 7.1.3 discusses the differences between our models and other memory-base architectures such as FGFA [Zhu et al., 2017b] and convolutional RNNs [Pinheiro and Collobert, 2014, Liang and Hu, 2015].

### 7.1.1 Aggregating features over time

Given a single image  $\mathcal{I} \in \mathbb{R}^{h_{\mathcal{I}} \times w_{\mathcal{I}} \times 3}$ , a convolutional neural network (CNN) with parameters  $\Theta_{\mathcal{F}}$  first extracts a feature map  $\mathcal{F} \in \mathbb{R}^{h_{\mathcal{F}} \times w_{\mathcal{F}} \times d_{\mathcal{F}}}$ , where  $d_{\mathcal{F}}$  is the number of feature maps and we typically have  $h_{\mathcal{F}} = \frac{1}{16}h_{\mathcal{I}}$  and  $w_{\mathcal{F}} = \frac{1}{16}w_{\mathcal{I}}$ . In the following, we show how these single image feature representations are effectively aggregated over time. While we use a single feature map per image for ease of presentation, note that we can make use of multiple feature maps at different resolutions to handle scale variations, which was shown to be useful in [Lin et al., 2017a, Yang et al., 2016].

**Tracking features over time:** In every frame  $t$ , we hold a feature map  $\mathcal{M}_t \in \mathbb{R}^{h_{\mathcal{F}} \times w_{\mathcal{F}} \times d_{\mathcal{F}}}$  that acts as a memory on the feature representation of the video. Since the scene is dynamic and the camera is moving, the same objects will appear at different locations of the image plane in frames  $t-1$  and  $t$ . In order for the memory of the past frame  $\mathcal{M}_{t-1}$  to benefit detection in the current frame  $t$ ,  $\mathcal{M}_{t-1}$  needs to be transformed according to the scene dynamics. Similar to [Zhu et al., 2017b], we use bilinear sampling to implement this transformation,

$$\hat{\mathcal{M}}_t = \varphi(\mathcal{M}_{t-1}, \mathcal{D}_{(t,t-1)}) , \quad (7.1)$$

where  $\varphi(\cdot)$  is the bilinear sampling function and  $\mathcal{D}_{(t,t-1)} \in \mathbb{R}^{h_{\mathcal{F}} \times w_{\mathcal{F}} \times 2}$  is a displacement (or flow) field between frames  $t$  and  $t-1$ , which is estimated by a CNN with parameters  $\Theta_{\mathcal{D}}$ . This CNN is a pre-trained FlowNet [Dosovitskiy et al., 2015], which takes images  $\mathcal{I}_t$  and  $\mathcal{I}_{t-1}$  as input and predicts the displacement, but we fine-tune the parameters  $\Theta_{\mathcal{D}}$  for the task at hand. Note that for fast computation of the displacement field, we feed FlowNet with half-resolution images and up-scale the displacement field. Also note that in the absence of ground truth data for the displacement field, this CNN predicts displacements suitable for the task at hand, which is demonstrated in Section 7.3.1.

**Updating with image evidence:** After having transformed the memory to the current frame  $t$ , *i.e.*,  $\hat{\mathcal{M}}_t$ , we need to aggregate the newly available image evidence  $\mathcal{F}_t$  extracted by the feature CNN into the memory,

$$\mathcal{M}_t = \psi(\hat{\mathcal{M}}_t, \mathcal{F}_t) , \quad (7.2)$$

which defines one step of the proposed *MemNet*. We implement (and experimentally evaluate) two variants of the aggregation function  $\psi(\cdot)$ . The first is a parameter-free combination that leads to exponential decay of memory over time,

$$\psi(\hat{\mathcal{M}}, \mathcal{F}) := \frac{1}{2}(\mathcal{M} + \mathcal{F}) , \quad (7.3)$$

and the second is a weighted combination of memory and image features,

$$\psi(\hat{\mathcal{M}}, \mathcal{F}) := \alpha^{\mathcal{M}} \cdot \mathcal{M} + \alpha^{\mathcal{F}} \cdot \mathcal{F}, \quad (7.4)$$

with  $\alpha^{\mathcal{M}}, \alpha^{\mathcal{F}} \in \mathbb{R}^{h_{\mathcal{F}} \times w_{\mathcal{F}} \times 1}$  and  $\alpha^{\mathcal{M}} + \alpha^{\mathcal{F}} = \mathbf{1}$ . The weights are computed by a separate CNN, named *weight-CNN*, with parameters  $\xi_{\mathcal{M}}$  and  $\xi_{\mathcal{F}}$  operating on  $\mathcal{M}$  and  $\mathcal{F}$ , respectively, and the constraint  $\alpha^{\mathcal{M}} + \alpha^{\mathcal{F}} = \mathbf{1}$  is always satisfied by passing the concatenated output of the CNNs through a per-pixel softmax function. The parameters of *weight-CNN* are automatically learned together with the rest of the network without any additional supervision. In the first frame  $t = 1$ , we simply assign the memory  $\mathcal{M}_1$  to be the feature representation of the image  $\mathcal{F}_1$ .

**Training MemNet:** Training the proposed video representation requires a supervisory signal from a task module acting on top of the memory features  $\mathcal{M}$ . In general,  $\mathcal{M}$  can be used for different tasks, for example to predict future frames [Mathieu et al., 2016]. In this work we explore object detection in videos, where the supervisory signal comes from a combination of object localization and classification loss functions, see Section 7.2.1.

All parts of the video representation can be trained end-to-end. Since the bilinear sampling and the grid generation of the warping module are both differentiable [Jaderberg et al., 2015b], we can back-propagate the training signal over time to previous frames, to the image feature extractor, as well as to the FlowNet generating the displacement fields.

While the network architecture in theory allows gradients to flow over the memory warping module to learn a good feature propagation, it also opens a shortcut for minimizing the loss because image evidence is available at every frame. While for some tasks past information is truly essential for prediction in the present, for several tasks the image of the current frame already provides most evidence for a good prediction (or at least a signal to minimize the loss). To encourage the network to learn a good feature propagation module, we randomly drop image evidence with probability 0.8

at frame  $t$ , which we found to improve results by a few percentage points.

### 7.1.2 Extending the temporal scale

The basic MemNet operates on just a single temporal scale, which has limited capability to leverage information at a larger temporal horizon. While, in theory, information from the whole video sequence is contained in the feature representation of the current frame  $t$ , this portion can be vanishingly small, particularly for the aggregation function relying on the exponential decay.

We thus propose to use a clock-work structure similar to Koutník et al. [2014], Shelhamer et al. [2016] that operates on multiple temporal scales, which we denote *ClockNet* and illustrate in Figure 7-3. Formally, instead of having a single memory feature map, we have  $K$  memories  $\mathcal{M}_t^k$  at frame  $t$  with  $k \in \{1, \dots, K\}$ , each of them operating at different rates. In our implementation, we update memory  $\mathcal{M}^k$  every  $2^{k-1}$  frames with new image evidence, although other schedules are also possible. Note that when  $K = 1$ , the basic *MemNet* is obtained.

In order to exchange information across the different time scales  $k$ , we aggregate all memory maps at a single frame  $t$  by simply averaging them, *i.e.*,  $\mathcal{M}_t = \frac{1}{K} \sum_{k=1}^K \mathcal{M}_t^k$ . As with the feature map aggregation in the basic *MemNet*, different strategies for combining feature maps are possible. We chose the simpler parameter-free averaging, as a more complex learning-based weighting scheme did not show any performance gains. The aggregated memory  $\mathcal{M}_t$  can then be used as input to any task-specific modules.

### 7.1.3 Discussion

Our proposed video representations have a simple and intuitive structure, can be trained end-to-end and fulfill the basic requirements for a fast and causal system that can be applied to videos in any real-world application. In contrast to FGFA Zhu et al. [2017b], the proposed model does not look at future frames and is also not limited to a specific temporal horizon in the past, rather can carry information from the whole

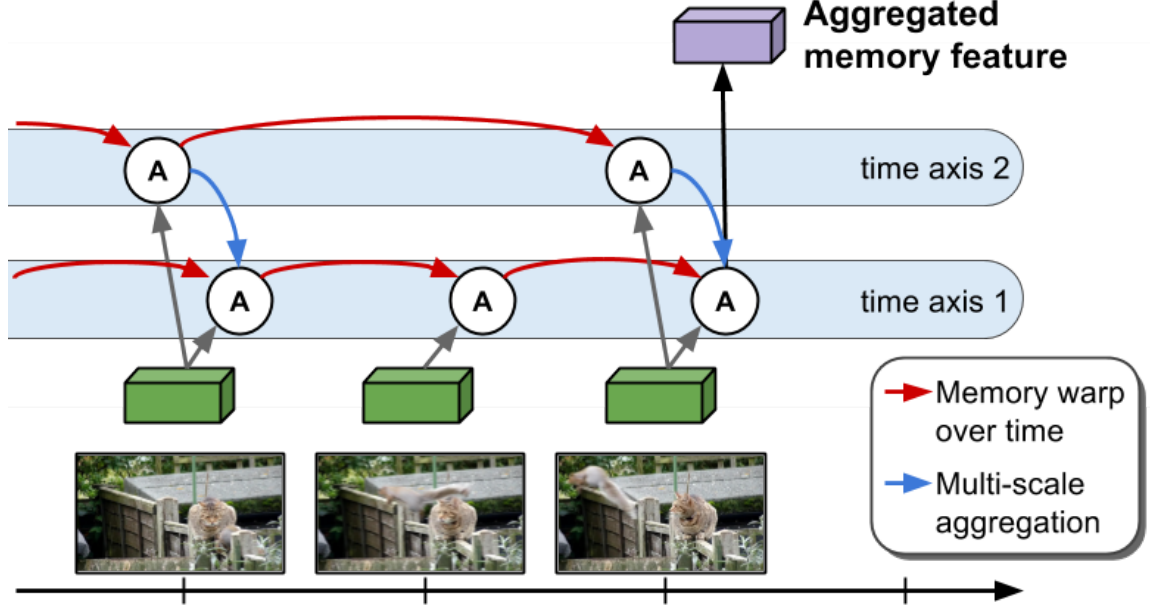


Figure 7-3 – Our **ClockNet** extends the MemNet by adding multiple time axes with increasing time scales to aggregate more information from further back in time. Each additional time axis  $k > 1$  skips  $2^{k-1} - 1$  frames. We only illustrate two time scales to avoid clutter.

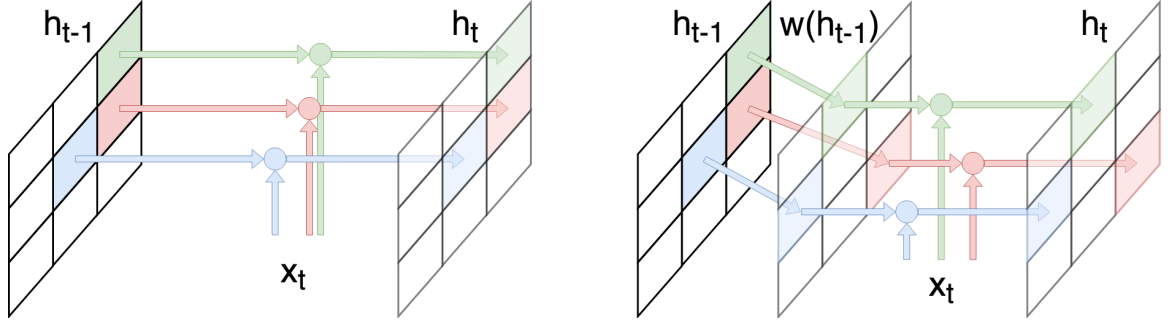


Figure 7-4 – Relation between standard convolutional recurrent neural networks (left) and the proposed video representation (right).

(past) sequence in its memory. An even longer temporal horizon is utilized by the ClockNet architecture.

There also exists a relation to convolutional recurrent neural networks (cRNN) [Pinheiro and Collobert, 2014, Liang and Hu, 2015], however, with one crucial difference. While cRNNs keep their hidden memory fixed across spatial dimensions ( $h_t = \text{RNN}(h_{t-1}, x_t)$ ), our model enables the memory to be spatially aligned with observer and scene motion in the actual video content ( $h_t = \text{RNN}(\text{warp}(h_{t-1}, \mathcal{D}_{t,t-1}), x_t)$ ), see Figure 7-4. While our aggregation function  $\psi(\cdot)$  for new input and previous hidden states is simple, we did not observe any improvements for our particular applications

with more complex architectures like LSTM [Hochreiter and Schmidhuber, 1997] or GRU [Cho et al., 2014].

## 7.2 Detection, Propagation and Anticipation in Videos

To demonstrate the benefits of propagating features over time with the proposed *MemNet* and *ClockNet*, we show its impact for three practical applications.

### 7.2.1 Object detection in videos

While our representation  $\mathcal{M}$  is generic and can be used for various tasks, we here focus on object detection in video. Modern object detectors such as Faster-RCNN [Ren et al., 2015], R-FCN [Dai et al., 2016], SSD [Liu et al., 2016] or RetinaNet [Lin et al., 2017b] share a similar high-level structure and rely on a convolutional neural network to extract features  $\mathcal{F}$  from a single image. The detection-specific modules applied on top of  $\mathcal{F}$  define the differences between the detectors, *e.g.*, proposal-based [Ren et al., 2015, Dai et al., 2016] or proposal-free [Liu et al., 2016, Lin et al., 2017b], making  $\mathcal{F}$  an interface between one generic module and detection-specific modules. Our proposed *MemNet* and *ClockNet* operate on  $\mathcal{F}$  and compute a novel feature representation  $\mathcal{M}$ , making our video representation applicable to all of these detectors. In this paper, we choose R-FCN [Dai et al., 2016] because it has shown a good trade-off between the accuracy and speed and given its publicly available implementation.

Given a representation  $\mathcal{M}_t$  of a video sequence at frame  $t$ , the object detector first computes object proposals with a region proposal network (RPN) as proposed in [Ren et al., 2015]. Object proposals define potential locations of objects of interest (independent of the actual category) and reduce the search space for the final classification stage. Each proposal is then classified into one of the  $C$  categories and the corresponding proposal location is further refined. In contrast to Faster-RCNN [Ren et al., 2015], the per-proposal computation costs in R-FCN are minimal by using position-sensitive ROI pooling. This special type of ROI pooling is applied on the



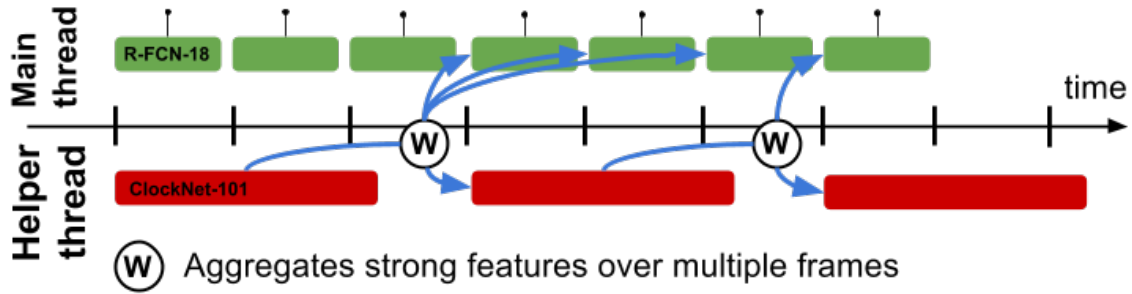


Figure 7-5 – In a multi-GPU setup, a fast but weak object detector (R-FCN-18, green blocks) leverages the features of a strong but slow object detector (ClockNet-101, red blocks), see Section 7.2.2. The width of blocks represent computation time. At a frame  $t$ , strong features from ClockNet-101 are only available from  $t - \Delta$ , but efficiently warped into frame  $t$  with our propagation module. The warped features boost the representational power of R-FCN-18 significantly, without increasing latency of the real-time system.

output of the region classification network (RCN).

### 7.2.2 Real-time detection by propagating strong features

Assuming an input stream capturing images at 20 frames-per-second (FPS), we ideally want an object detector that can process one image in less than 50 ms to avoid latency in the output. One easy option to speed-up a modern object detector is to use a more light-weight feature extraction CNN, *e.g.*, by using ResNet-18 instead of ResNet-101. Note that this is a viable option for any detection framework, *e.g.*, Faster-RCNN [Ren et al., 2015], R-FCN [Dai et al., 2016], YOLO [Redmon et al., 2016, Redmon and Farhadi, 2017] or SSD [Liu et al., 2016]. However, accuracy will decrease. Here, we explore another option to speed-up a modern object detector. Instead of using a single model, we demonstrate how to exploit two models with complementary properties running simultaneously (but asynchronously) on two threads (two GPUs) to achieve both speed and accuracy, using our feature propagation.

We run a fast detector, R-FCN-18 (*i.e.*, R-FCN with ResNet-18 [He et al., 2016]) in one thread and a slower but also stronger detector, ClockNet-101, in the other thread. R-FCN-18 runs at the required frame rate and can provide output for every frame, however at a lower quality than ClockNet-101 could do if no time requirements existed. The main problem with the strong object detector is that it will always have

some delay (or latency)  $\Delta$  to produce an output. If  $\Delta$  is too large for a practical system, the strong detector is not usable. It is important to note that achieving a speed-up with two GPUs is not trivial in a real-time setting. For the offline case it is easy to distribute computation of different images on multiple GPUs. However, this is not an option for streaming data. In Section 7.3.3, we still empirically compare with two alternative baselines that also leverage two GPUs.

With our design, on the other hand, we can still leverage the strong features by making up for the delay via feature propagation. We compute the displacement field between frames at times  $t + \Delta$  and  $t$  and warp the strong features  $\mathcal{M}_t^{101}$  into the current frame  $t + \Delta$ , where the fast object detector has already computed features  $\mathcal{F}_{t+\Delta}^{18}$ , see Figure 7-5. We boost the representational power of R-FCN-18 by combining the feature maps. Again, we take the average of both features (the dimensionality is the same), but more advanced aggregation schemes are possible. We experimentally evaluate this application in Section 7.3.3.

### 7.2.3 Anticipating features

In the previous application of Section 7.2.2, we deploy feature propagation over several frames, but the displacement fields are still computed from image evidence, similar to [Zhu et al., 2017a]. For a true visual anticipation, however, future images are not available.

We propose to extrapolate the displacement fields into future frames and use them to propagate the feature (or memory) maps. For demonstration, we use a simple extrapolation technique. Given the two previous displacement fields  $\mathcal{D}_{t-1,t-2}$  and  $\mathcal{D}_{t,t-1}$ , we compute the difference of aligned displacement vectors (with bilinear sampling), which gives us the acceleration of pixels. We then apply a constant acceleration motion model to each displacement vector and extrapolate for one or multiple frames. Obviously, this extrapolation technique has limitations but it is sufficient for our demonstrations of feature anticipation. We analyze the quality of the anticipated features in Section 7.3.2 by measuring the object detection quality in future frames.

## 7.3 Experiments

Our experimental evaluation focuses on the performance of our feature propagation and aggregation methods for object detection in videos. In Sections 7.3.1, 7.3.2 and 7.3.3, we evaluate the performance of the proposed *MemNet* and *ClockNet* on the three applications introduced in Section 7.2, respectively.

**Dataset:** All our experiments are conducted on the ImageNet VID data set [Rusakovsky et al., 2015], which is most suitable for object detection in videos and has been used to evaluate recent approaches for this task [Feichtenhofer et al., 2017b, Kang et al., 2016b, Zhu et al., 2017a,b]. ImageNet VID is a large scale data set consisting of 5344 video clips, captured at frames rates between 25 and 30 FPS and divided into training, validation and testing sets with 3862, 555 and 937 clips respectively. Each clip is fully annotated with bounding box tracks of 30 different object classes.

**Implementation details:** We use the ResNet-101 architecture [He et al., 2016] as the basic feature extractor in all experiments with the exception of some ablation studies. In particular, we use *à trous* convolutions as in [Chen et al., 2016, Zhu et al., 2017b] to increase the feature resolution. Similar to [Zhu et al., 2017b] the extracted features are passed through a  $3 \times 3$  convolutional layer and a non-linear activation (ReLU [Nair and Hinton, 2010]) before we provide them as inputs to *MemNet* and *ClockNet*. To estimate displacement fields we use FlowNet [Dosovitskiy et al., 2015]. All the parameters are jointly fine-tuned end-to-end. In general, we closely follow the experimental setup of [Zhu et al., 2017b] using their publicly available MXNet [Chen et al., 2015] implementation. All models are trained for 2 epochs with an initial learning rate of 0.001, which is decreased by a factor of 10 after  $\frac{4}{3}$  epochs. We train our models on the same mix of ImageNet DET and ImageNet VID training sets as in [Zhu et al., 2017b]. All experiments, including runtime measurement, are done with NVIDIA TITAN Xp GPUs. For training, we use a setup with 4 parallel GPUs.

### 7.3.1 Object detection in videos

Object detection in videos aims at localizing objects in every video frame, *i.e.*, estimating bounding boxes around objects associated with a confidence score.

**Evaluation metrics:** We measure detection performance as mean average precision (mAP) over all object classes, where we additionally differentiate between fast, medium and slowly moving objects using the subsets of videos introduced in [Zhu et al., 2017b]. We also measure the average runtime per frame in milliseconds (ms) for each model (using the same framework and GPU setup).

**Baselines:** We compare the proposed *MemNet* and *ClockNet* models with several baselines. The first one is the per-frame object detector itself (R-FCN [Dai et al., 2016, Zhu et al., 2017b]) that does not exploit temporal information. The second baseline is FGFA [Zhu et al., 2017b], which, for every frame, aggregates features from nearby frames both in the past and the future. Obviously, this makes FGFA a non-causal system not applicable to real-time tasks. Note that these two baselines represent two extremes of using temporal information, with R-FCN not exploiting the video at all and FGFA looking not only into the past but also into future frames. For a more fair comparison, we thus created a causal variant of FGFA (Cau. FGFA) that aggregates information only from nearby features in the past but not from future frames. While FGFA can only operate in the off-line setting where future frames are accessible, causal FGFA is an on-line detector, making it the most comparable baseline to our proposed models.

**Main results:** Table 7.1 summarizes our quantitative results and Figures 7-7 gives qualitative examples. We can first see that all models leveraging temporal data improve over the per-frame baseline R-FCN [Dai et al., 2016]. Looking into both past and future frames, as FGFA [Zhu et al., 2017b] does, gives the best overall results, but comes at a considerable runtime cost and, more importantly, is a non-causal system. Among all causal systems leveraging data only from the past (eight

Method	mAP	mAP (fast)	mAP (med)	mAP (slow)	ms
R-FCN [Dai et al., 2016]	73.4	51.4	71.6	82.4	108
FGFA [Zhu et al., 2017b]	76.2	56.0	75.2	83.8	286
FGFA (half)	66.7	42.2	66.1	77.5	152
Cau. FGFA	75.2	53.9	<b>74.1</b>	83.8	204
Cau. FGFA*	66.0	40.0	65.3	79.3	181
MemNet-3	74.3	51.9	72.5	83.6	122
MemNet-6	75.1	53.8	73.5	83.3	122
MemNet-6-wgts	75.3	51.8	73.6	83.8	124
MemNet-6-strd-4	74.4	51.7	72.4	<b>84.2</b>	122
MemNet-6-strd-8	74.2	51.6	72.6	82.7	122
ClockNet	<b>75.6</b>	<b>55.4</b>	73.7	83.4	169

Table 7.1 – Detection performance and runtime on ImageNet-VID validation of different methods. We also report results on the three validation subsets of fast, medium and slowly moving objects, denoted as mAP(fast), mAP(med) and mAP(slow).

bottom models in Table 7.1), the proposed *ClockNet* gives the best results overall and is particularly strong for fast moving objects. Its mAP value is only 0.6 percentage points behind FGFA without having access to future frames.

Looking at the running times, we see that the proposed *MemNet* is clearly the fastest, except for the still-frame baseline, and *ClockNet* already ranks second. Both proposed models are faster than causal FGFA and consequently, FGFA [Zhu et al., 2017b]. Built upon the detection framework of R-FCN, all those models have the same computational complexity for feature extraction, proposals generation and classification. Therefore, the causes of speed difference are the numbers of flow computations  $N_F$  and feature warps  $N_W$ . For FGFA and Causal FGFA,  $N_F$  and  $N_W$  equal the number of frames within aggregation range, i.e. 20 and 10, respectively. The *ClockNet* reported in Table 7.1 requires  $N_F = 3$  and  $N_W = 3$ , corresponding to its 3 temporal scales, and *MemNet* has a speed advantage because it only needs  $N_F = 1$  and  $N_W = 1$  to process an incoming frame.

We also note that the computation time of Causal FGFA can be reduced by aggregating displacement fields in an online manner, see Cau. FGFA\* in Table 7.1, thus reducing  $N_F$  to 1 as in MemNet. While the runtime is reduced, the error accu-

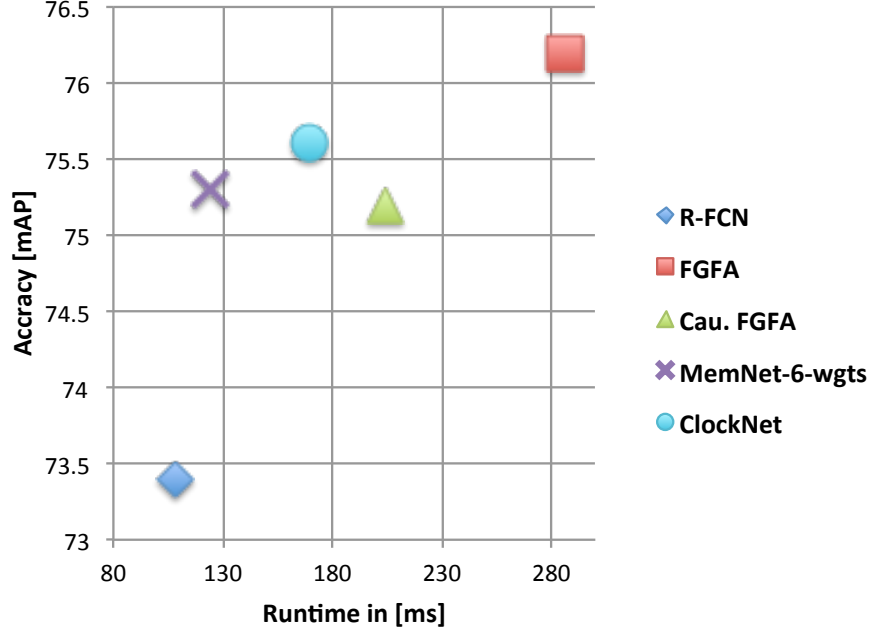


Figure 7-6 – Accuracy and runtime trade-off of various methods.

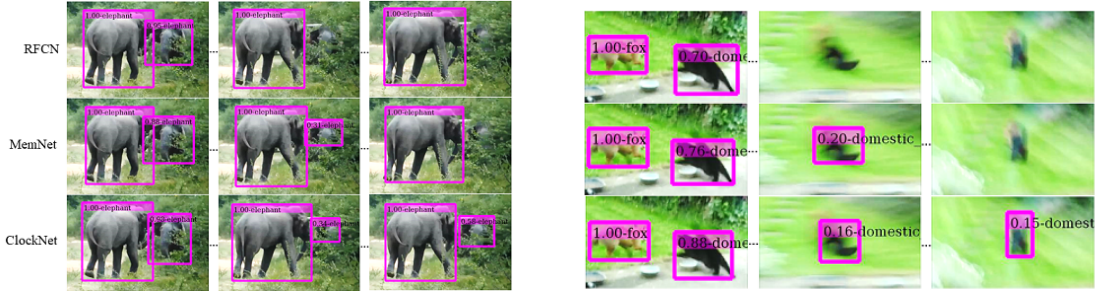


Figure 7-7 – Qualitative examples of our memory models and the R-FCN baseline.

mulation in online aggregation of displacement fields leads to a significant accuracy drop. To highlight the reduction in runtime from MemNet and ClockNet, we show another trivial way to speed-up FGFA by halving the image resolution, FGFA (half). However, this variant also leads to a large performance drop.

In summary, Table 7.1 and Figure 7-6 demonstrates the advantages of our memory propagation mechanism and our temporal multi-scale architecture ClockNet in terms of accuracy and runtime. Moreover, our memory-based architectures have the additional benefit (over FGFA) of being able to propagate features into future frames, which we analyze in Section 7.3.2 and demonstrate in a practical application in Section 7.3.3.

**Effect of different sequence lengths and temporal scales:** We investigate the importance of the length of the sequences used for training *MemNet*. In the testing phase there is no limitation on the sequence length, but GPU memory is a hard constraint during training as gradients need to flow back through the whole sequence. We can observe in Table 7.1 that a longer temporal window for training *MemNet* (*MemNet-6* vs. *MemNet-3*) is indeed beneficial. In addition, we analyze the importance of multiple temporal scales in *ClockNet*, which operates on three time axes  $k = [1, 3, 4]$ , corresponding to temporal strides of 1, 4 and 8, respectively. To emphasize the benefit of feature aggregation across multiple temporal scales, we compare *ClockNet* to *MemNet* trained with longer strides (*MemNet-6-strd-4* and *MemNet-6-strd-8*), given it access to larger temporal horizon during training. We can see from Table 7.1 that both baselines perform worse than *MemNet-6* (with temporal stride of 1), which illustrates the importance of the information provided by actual neighboring frames. Therefore, the success of *ClockNet* provides a signal for the benefits of a temporal multi-scale architecture, where different temporal scales are complementary to each other. More experiments on various details of our methods can be found in the supplemental material.

**Different aggregation schemes and aggregation levels:** We further want to investigate the impact of several aspects of the proposed model and conduct an ablation study on (i) different feature aggregation schemes and (ii) memories at different feature representation levels. For all these experiments, we use a ResNet-50 model and train only on half resolution images.

Table 7.2 demonstrates results of this ablation study. We first compare our weighting approach with the adaptive weight [Zhu et al., 2017b] and the recently proposed AdaScan [Kar et al., 2016]. Interestingly, we found that simply averaging features from the current frame and the warped memory gives on par results with more complex weighting schemes, which is also consistent with the results reported in Table 7.1. Next, we investigate the impact of employing the memory module at different representation levels of the underlying feature extraction network, i.e., ResNet-50. We

	Avg.	Adap. [Zhu et al., 2017b]	Ada. [Kar et al., 2016]	
mAP (%)	66.4	66.4	66.2	
(a)				
	Conv3	Conv4	Conv4+Conv5	Conv5
mAP (%)	66	66	66.2	66.4
(b)				

Table 7.2 – Detection performance of MemNet in ablation studies on ImageNet-VID validation set. (a) shows the impact of different aggregation schemes and (b) shows different levels of the feature representation where the memory module is employed.

compared the memory module at "conv3", "conv4", "conv5", and a combination of the latter two. Combining the combination of feature maps outperforms the lowerlevel feature maps but using the highest level representation, i.e., "conv5", still gives the overall best results.

**Fine-tuning FlowNet:** Finally, we want to understand the effect of fine-tuning FlowNet during training the video representation for the detection task. Similar to [Zhu et al., 2017b], we observe a performance drop in mAP if FlowNet is not fine-tuned. Given that displacement fields after fine-tuning are apparently better for the detection task, we visualize the difference for some examples in Figure 7-8. One difference that we can observe is that the object tends to move as a whole and ignores the motion of individual parts.

### 7.3.2 Propagating and anticipating features

In the previous experiment, the appearance features of the current frame are always available and the main purpose was to analyze the influence of additional information from the past. In this experiment, we want to give more insights into the quality of the feature propagation and feature anticipation, which has several applications as discussed in Section 7.2.





Figure 7-8 – We illustrate the impact of jointly fine-tuning the FlowNet with the object detector. For each example consists of 4 images: the 2 input images (“Image0, Image1”) and the displacement field before and after fine-tuning. The figure shows 2 columns and 4 rows of examples.

**Propagation:** In this experiment, we provide image information up to frame  $t - \delta$  and then propagate features up to frame  $t$ . Note that displacement fields are still available but only for warping the memory; no image evidence in form of appearance features is available to the detector after frame  $t - \delta$ . The propagated features are then used to compute the detections. We compare our model with a baseline that takes the

Methods	mAP			FPS		
$\delta$	0	4	8	0	4	8
<i>Box - Propagation</i>						
MemNet	75.1	64.6	55.4	8.2	14.7	16.9
ClockNet	75.6	64.9	56.2	5.9	12.5	15.3
<i>Feature - Propagation</i>						
MemNet	75.1	68.9	56.1	8.2	14	16
ClockNet	75.6	70.9	62.3	5.9	6.6	8.3
<i>Feature - Anticipation</i>						
MemNet	-	68.8	55.9	-	5.1	8.7
ClockNet	-	67.0	57.3	-	2.5	3.4

Table 7.3 – Detection performance (runtime fps and accuracy mAP) on ImageNet-VID validation set of MemNet and ClockNet with feature propagation and feature anticipation. We also compare our results to the box-propagating baselines.

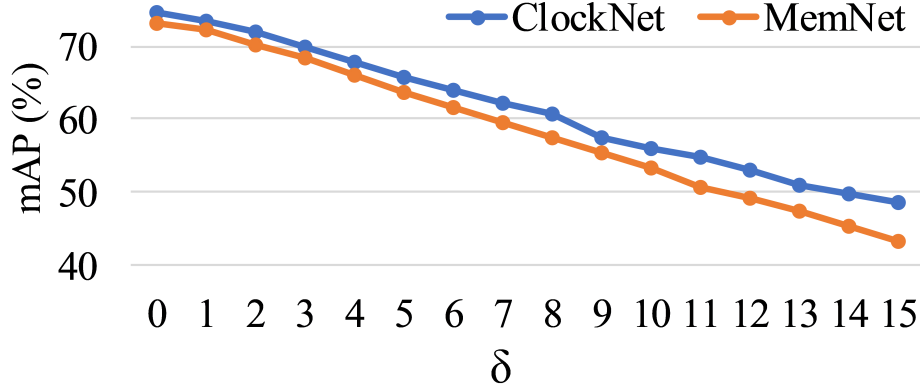


Figure 7-9 – Mean AP of MemNet and ClockNet with respect to different propagation lengths  $\delta$ .

detected bounding boxes at frame  $t - \delta$  and propagates them with the computed displacement fields to frame  $t$ . The mean displacement inside a bounding box serves as the translational vector. Table 7.3 shows the impact of skipping the image features for different amounts of frames for MemNet, ClockNet and the box-propagating baselines. In general, all models gracefully degrade performance with larger  $\delta$ , but at the same time reduce running time. It is evident that feature propagation outperforms propagation on the bounding box level, particularly for ClockNet. From Figure 7-9, we can see that the performance gap between MemNet and ClockNet increases when  $\delta$  gets larger, demonstrating the impact of multiple temporal scales and the extend

time horizon of ClockNet.

**Anticipation:** We next evaluate feature anticipation as discussed in Section 7.2.3, which differs to the previous experiment since no image information is available at all for frames after  $t - \delta$ . This requires us to extrapolate the displacement field as described in Section 7.2.3. Specifically, in our setting feature propagation and feature anticipation are only different at the step of estimating optical flows.

Comparing the anticipation results with the propagation model in Table 7.3, we can see that our flow extrapolation strategy works well for MemNet which has short temporal stride, although the runtime speed drops due to our current non-optimized implementation of flow extrapolation. Reversely, on ClockNet memories, feature anticipation performs worse than feature propagation. One explanation is that quality of the extrapolated flows is heavily degraded with the long temporal strides of ClockNet.

### 7.3.3 Fast detection by propagating strong features

In this section we analyze the application described in Section 7.2.2 and Figure 7-5. A fast but weak object detector is running in the main thread providing detection output at every frame, while leveraging features from a strong but slow feature extractor. In order to use the strong features, they have to be propagated to the current frame, *i.e.*, aligned over time, to compensate for the delay  $\Delta$ .

We use R-FCN based on the ResNet-18 architecture as the fast main-thread detector, which runs at 14.3 FPS. The helper thread runs *ClockNet* based on ResNet-101 (same as in Table 7.1), which is able to propagate features over time as demonstrated in Section 7.3.2. In practice, ClockNet-101 runs 2.4 times slower than R-FCN-18. To compensate for additional overhead (*e.g.*, feature propagation), we update ClockNet-101 with image evidence once every 4 frames (“ClockNet-101-FeatProp”). For training the fast detector, *i.e.*, R-FCN-18, we only leverage fixed features coming from the second thread but do not fine-tune ClockNet-101.

Method	#G	mAP	FPS
R-FCN-18	1	60.2	14.3
ClockNet-101	1	75.6	5.9
ClockNet-101-FeatProp	1	70.9	6.6
R-FCN-18 (split)	2	21.7	28.6
ClockNet-101 (split)	2	28.9	11.8
ClockNet-101-FeatProp (split)	2	23.3	13.2
R-FCN-18+ClockNet-101-BoxProp	2	66.9	14.3
R-FCN-18+ClockNet-101-FeatProp	2	71.9	14.3

Table 7.4 – Accuracy and runtime of our two-threaded detection setup. The number of GPUs utilized is denoted as #G.

**Quantitative results:** Table 7.4 shows the results of this experiment. As expected, R-FCN-18 is the fastest model but also gives the worst accuracy. ClockNet-101 is the model shown in Section 7.3.1 and can be considered an upper bound in terms of accuracy but is very slow compared to R-FCN-18. ClockNet-101-FeatProp is the model running in the helper thread which receives image evidence every 4 frames and uses propagation to make predictions in other frames. The performance drop compared to the upper bound is not much, as also seen in Section 7.3.2, but the runtime is still high. However, when following the design proposed in Section 7.2.2, *i.e.*, feeding R-FCN-18 with propagated strong features from ClockNet-101-FeatProp, we observe a significant 10% mAP boost compared to R-FCN-18 which is getting close to the ClockNet-101 upper bound, while maintaining the low runtime of R-FCN-18.

**Comparison to two-thread baselines:** Recall that achieving speed-up with two GPUs in a real-time (data-streaming) setting is non-trivial as parallelizing frames over multiple GPUs is not possible. We still evaluate alternative baselines that leverage two GPUs. First, a trivial two times speed-up can be achieved by splitting the image into two halves, running the detector individually and merging the detections before NMS, denoted “(split)” in Table 7.4. While this gives the expected speed-up, the performance drop is significant, which can be explained by the fact that objects in Imagenet-VID are mostly centered, thus effectively truncating them. The second

baseline is more evolved and similar to our proposed design. However, instead of propagating features, (delayed) detections from the strong model are propagated to align with the faster detector (“BoxProp”), as in Section 7.3.2. As in the previous experiment, we observe that feature-level propagation is superior to propagating bounding boxes.

## 7.4 Conclusions

Our work demonstrates how to learn a long-term video representation that can effectively leverage past information. We rely on the concept of a memory that is being updated regularly with image evidence and is being warped over time for a proper spatio-temporal alignment of features. The proposed ClockNet is a temporal multi-scale architecture that can store information at multiple temporal scales. Our experimental evaluation illustrates benefits of introduced components and demonstrates applications in terms of: feature anticipation and a fast multi-threaded object detector.

In future work it will be interesting to investigate the benefits of our learned video representations to other tasks such as action recognition or semantic segmentation in videos. We also plan to further explore the utility of feature anticipation and postulate that learning representations in causal settings is beneficial for visual anticipation.

# Chapter 8

## Conclusion and Future Work

This work has addressed problems of action prediction in still images as well as object detection in videos. Towards action prediction, we have collected a new dataset with manual annotations of typical actions for 397 scene categories. The strong correlations between actions and scenes are justified by statistical methods. Based on such correlations, we have proposed a new task of predicting actions in still images and introduced a potential application of affordance geo-localization. In the second part of the thesis we have addressed object detection in videos from different angles. First, on the frame-level, we have leveraged contextual cues to improve detection performance with *Global* and *Pairwise* models. Second, on the clip-level, we have designed CNN models that can learn spatio-temporal representations to capture the temporal evolution of object appearance. Last, we have designed an efficient and effective memory-base framework that can store and transfer long-term appearance information. Our approaches show significant improvements over the baselines.

Despite recent progress computer vision is facing important challenges. Some of these challenges are described below.

**Self-supervised Feature Learning.** The power of learning visual features currently comes with the high cost of annotating very large datasets. In recent years much work has been focused on creating large and high quality fully-annotated datasets with the goal of learning more robust visual representations [Cordts et al., 2016, Neuhold

et al., 2017]. Although such a brute-force approach will advance computer vision in the short-term, its sustainability in the long run is questionable. Motivated by this issue, another direction of recent work addresses self-supervised feature learning from a massive amount of unannotated data using adhoc tasks [Doersch et al., 2015, Pathak et al., 2017, Larsson et al., 2017, Doersch and Zisserman, 2017]. Despite some progress, the performance of such methods is still not on a par with supervised results by fully-supervised approaches.

Images are not random collections of pixels. We strongly believe that the latent structure of images and video should be more closely explored and used as a regularizer for feature learning. Other strong cues for self-supervised feature learning could include motion and sound, as these modalities are widely used by people to understand the world [Ellis, 1938]. This motivates self-supervised multi-modal learning algorithms that can exploit the underlying structure not only of images alone but in combination with other modalities.

**Domain Adaptation.** There is a growing interest in using synthetic data to overcome the difficulty of collecting annotations. Although the rendering engines are producing more and more realistic images, the underlying distributions of synthetic images and the real ones still differ. That raises a problem of adapting or transferring features and knowledges learned from synthetic data to real data. A direct use of such features often leads to poor performance on real datasets. An ambitious solution for domain adaptation is to learn a semantic embedding playing as a common protocol that can translate the semantic world, either real or synthetic. Such a system could then be deployed to solve practical problems such as reliable recognition of traffic scenes in various weather conditions.

**Reliable Predictive Model.** From a very young age, people possesses a natural gift of inferring others' intentions. Such an ability is important for social adaptation in an interconnected society. We believe that intelligent systems should be able to not only recognize but also anticipate human intentions and act accordingly to. This is

an ambitious goal which needs advancement in multiple areas including recognition, end-to-end deep reinforcement learning and uncertainty deep modelling.





# Bibliography

Map of ski stations in france. [www.skiinfo.fr/france/carte.html](http://www.skiinfo.fr/france/carte.html), 2013. 52, 53, 149

A. E. Abdel-Hakim and A. A. Farag. Csift: A sift descriptor with color invariant characteristics. In *Proc. CVPR*, 2006. 24, 47

Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 102

Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proc. CVPR*, 2016. 15

Yali Amit, Donald Geman, and Xiaodong Fan. A coarse-to-fine strategy for multiclass shape detection. *IEEE TPAMI*, 2004. 5

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proc. ICCV*, 2015. 27

Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. *Proc. CVPR*, 2016. 89

Sean M Arietta, Alexei A Efros, Ravi Ramamoorthi, and Maneesh Agrawala. City

- forensics: Using visual elements to predict non-visual city attributes. *IEEE TVCG*, 2014. 15
- Stylianos Asteriadis, Paraskevi Tzouveli, Kostas Karpouzis, and Stefanos Kollias. Estimation of behavioral user state based on eye gaze and head pose—application in an e-learning environment. *Multimedia Tools and Applications*, 2009. 4
- Sandra Avila, Nicolas Thome, Matthieu Cord, Eduardo Valle, and Arnaldo De A Araújo. Pooling in image representation: The visual codeword point of view. *Computer Vision and Image Understanding*, 2013. 24
- Hedi Ben-Younes, Rémi Cadene, Matthieu Cord, and Nicolas Thome. Mutan: Multimodal tucker fusion for visual question answering. In *Proc. ICCV*, 2017. 27
- Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *IEEE TPAMI*, 2001. 13
- Piotr Bojanowski, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Finding actors and actions in movies. In *Proc. ICCV*, 2013. 20
- Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proc. COLT*, 1992. 25
- Junxia Cai and A Goshtasby. Detecting human faces in color images. *Image and Vision Computing*, 1999. 17
- John Canny. A computational approach to edge detection. In *Readings in Computer Vision*. 1987. 17
- C. C. Chang and C. J. Lin. Libsvm: A library for support vector machines. *ACM TIST*, 2011. 47
- K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. BMVC*, 2014. 27, 28, 29, 59, 74, 147

- Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE TPAMI*, 2016. 27, 112
- Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, 2015. 112
- X. Chen and A. L. Yuille. Articulated pose estimation with image-dependent preference on pairwise relations. In *Proc. NIPS*. 2014. 61
- Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *Proc. CVPR*, 2014. 5
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proc. EMNLP*, 2014. 109
- Eunji Chong, Katha Chanda, Zhefan Ye, Audrey Southerland, Nataniel Ruiz, Rebecca M Jones, Agata Rozga, and James M Rehg. Detecting gaze towards eyes in natural social interactions and its use in child assessment. *Proc. ACM IMWUT*, 2017. 4
- Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. *IEEE TPAMI*, 2001. 17
- Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. CVPR*, 2016. 123
- Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 1995. 25

- Ian Craw, H Ellis, and J Rowland Lishman. Automatic extraction of face-features. *Pattern recognition letters*, 1987. 17
- Ian Craw, David Tock, and Alan Bennett. Finding face features. In *Proc. ECCV*, 1992. 17
- James E. Cutting. *Perception with an Eye for Motion*. 1986. 101
- Niels da Vitoria Lobo and Young Ho Kwon. Face detection using templates, 1998. 17
- Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object detection via region-based fully convolutional networks. In *Proc. NIPS*, 2016. 19, 33, 34, 109, 110, 113, 114, 148
- Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *Proc. ICCV*, 2017. 20
- Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, 2005. 4, 5, 16, 24, 89
- V. Delaitre, D.F. Fouhey, I. Laptev, J. Sivic, A. Gupta, and A.A. Efros. Scene semantics from long-term observation of people. In *Proc. ECCV*. 2012. 15, 38
- Vincent Delaitre, Ivan Laptev, and Josef Sivic. Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *Proc. BMVC*, 2010. 4, 15
- Vincent Delaitre, Josef Sivic, and Ivan Laptev. Learning person-object interactions for action recognition in still images. In *Proc. NIPS*, 2011. 4
- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 59, 74, 88
- C. Desai, D. Ramanan, and C. C. Fowlkes. Discriminative models for multi-class object layout. *IJCV*, 2011. 58, 61, 62, 63

- Sidney D’Mello, Andrew Olney, Claire Williams, and Patrick Hays. Gaze tutor: A gaze-reactive intelligent tutoring system. *International Journal of human-computer studies*, 2012. 4
- Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proc. ICCV*, 2017. 124
- Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proc. ICCV*, 2015. 124
- Piotr Dollár, Serge J Belongie, and Pietro Perona. The fastest pedestrian detector in the west. In *Proc. BMVC*, 2010. 5
- Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. In *Proc. CVPR*, 2015. 21
- Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow With Convolutional Networks. In *Proc. ICCV*, 2015. 22, 27, 105, 112
- Charles Dubout and François Fleuret. Exact acceleration of linear object detectors. In *Proc. ECCV*, 2012. 18
- Alexei A Efros, Alexander C Berg, Greg Mori, Jitendra Malik, et al. Recognizing action at a distance. In *Proc. ICCV*, 2003. 13
- K. A. Ehinger, J. Xiao, A. Torralba, and A. Oliva. Estimating scene typicality from human ratings and image features. 2011. 39
- W. Ellis. *A Source Book of Gestalt Psychology*. 1938. 6, 8, 101, 124

- Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. Finding beans in burgers: Deep semantic-visual embedding with localization. In *Proc. CVPR*, 2018. 27
- M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 2010. 63, 71, 92
- Mark Everingham, Josef Sivic, and Andrew Zisserman. Hello! my name is... buffy”—automatic naming of characters in tv video. In *Proc. BMVC*, 2006. 20, 93
- Li Fei-Fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *Proc. CVPR*, 2005. 24
- Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *Proc. CVPR*, 2016. 21, 22
- Christoph Feichtenhofer, Axel Pinz, and Richard P. Wildes. Spatiotemporal Multiplier Networks for Video Action Recognition. In *Proc. CVPR*, 2017a. 21, 22
- Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to Track and Track to Detect. In *Proc. ICCV*, 2017b. 22, 102, 112
- P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE TPAMI*, 2010a. 5, 18, 61, 64, 68, 89
- Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *IJCV*, 2005. 18
- Pedro F Felzenszwalb, Ross B Girshick, and David McAllester. Cascade object detection with deformable part models. In *Proc. CVPR*, 2010b. 18
- R Feraund, Olivier J Bernier, J-E Viallet, and Michel Collobert. A fast and accurate face detector based on neural networks. *IEEE TPAMI*, 2001. 17

- Vittorio Ferrari, Manuel Marin-Jimenez, and Andrew Zisserman. Pose search: retrieving people using their pose. In *Proc. CVPR*, 2009. 4
- Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 1973. 16, 18
- Francois Fleuret and Donald Geman. Coarse-to-fine face detection. *IJCV*. 5
- Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Spatio-temporal closed-loop object detection. *IEEE TIP*, 2017. 94, 95
- JC Gemert, Mihir Jain, Ella Gati, Cees GM Snoek, et al. *APT: Action localization Proposals from dense Trajectories*. 2015. 90, 92
- J.J. Gibson. *The Ecological Approach to Visual Perception*. 1979. 101
- Ross Girshick. Fast r-cnn. In *Proc. ICCV*, 2015. 4, 17, 19, 27, 31, 32, 82, 83, 88, 89, 93, 147
- Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, 2014. 5, 10, 17, 18, 27, 30, 57, 59, 67, 71, 147
- Google. Panoramio service. <http://www.panoramio.com>, 2007. 51
- Venu Govindaraju. Locating human faces in photographs. *IJCV*, 1996. 16
- H. Grabner, J. Gall, and L. Van Gool. What makes a chair a chair? In *Proc. CVPR*, 2011. 15, 38
- Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. *Proc. CVPR*, 2018. 27
- A. Gupta, S. Satkin, A.A. Efros, and M. Hebert. From 3d scene geometry to human workspace. In *Proc. CVPR*, 2011. 15, 38



- Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. *arXiv preprint arXiv:1803.10892*, 2018. 16
- Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S Huang. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*, 2016. 22, 102
- Yugo Hayashi. Gaze feedback and pedagogical suggestions in collaborative learning. In *Proc. ITS*, 2018. 4
- K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proc. ECCV*, 2014. 19, 82
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proc. CVPR*, 2016. 19, 27, 28, 29, 33, 82, 88, 89, 110, 112, 147
- M. Hoai and A. Zisserman. Talking heads: Detecting humans and recognizing their interactions. In *Proc. CVPR*, 2014. 61, 69, 70, 73
- Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997. 21, 109
- <http://www.di.ens.fr/willow/research/actionsfromscenes>. 41, 49
- Nazli Ikizler, R Gokberk Cinbis, Selen Pehlivan, and Pinar Duygulu. Recognizing actions from still images. In *Proc. ICPR*, 2008. 4, 14
- Nazli Ikizler-Cinbis, R Gokberk Cinbis, and Stan Sclaroff. Learning actions from the web. In *Proc. ICCV*, 2009. 4
- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015. 28
- M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Deep structured output learning for unconstrained text recognition. In *Proc. ICLR*, 2015a. 65

- Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial Transformer Networks. In *Proc. NIPS*, 2015b. 106
- Himalaya Jain, Joaquin Zepeda, Patrick Pérez, and Rémi Gribonval. Subic: A supervised, structured binary code for image search. In *Proc. ICCV*, 2017. 27
- Patrick Jermann, Dejana Mullins, Marc-Antoine Nüssli, and Pierre Dillenbourg. Collaborative gaze footprints: Correlates of interaction quality. In *Proc. CSCL*, 2011. 4
- X. Jianxiong, J. Hays, K.A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*, 2010. 38
- Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *arXiv preprint arXiv:1604.02532*, 2016a. 10, 86
- Kai Kang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Object detection from video tubelets with convolutional neural networks. In *Proc. CVPR*, 2016b. 20, 22, 93, 94, 95, 102, 112
- Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. Object detection in videos with tubelet proposal networks. *Proc. CVPR*, 2017. 21, 93, 94, 95
- Amlan Kar, Nishant Rai, Karan Sikka, and Gaurav Sharma. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. 2016. 116, 117
- Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proc. CVPR*, 2014. 14

- Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 14
- A. Khosla, B. An, J.J. Lim, and A. Torralba. Looking beyond the visible scene. In *Proc. CVPR*, 2014. 15
- Hyeonwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Nießner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *Proc. SIGGRAPH*, 2018. 27
- Michael Kirby and Lawrence Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE TPAMI*, 1990. 17
- K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *Proc. ECCV*. 2012. 4, 15, 38
- Iasonas Kokkinos. Rapid deformable object detection using dual-tree branch-and-bound. In *Proc. NIPS*, 2011. 5, 18
- V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE TPAMI*, 2006. 63
- V. Kolmogorov and C. Rother. Minimizing non-submodular functions with graph cuts – a review. *IEEE TPAMI*, 2007. 63
- Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE TPAMI*, 2016. 15
- Jan Koutník, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. A Clockwork RNN. In *Proc. ICML*, 2014. 107
- Philipp Krähenbühl and Vladlen Koltun. Geodesic object proposals. In *Proc. ECCV*, 2014. 5

- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, 2012. 18, 19, 27, 28, 59, 74, 89, 147
- Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *Proc. ICCV*, 2011. 14
- Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature Space Optimization for Semantic Video Segmentation. In *Proc. CVPR*, 2016. 22
- Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Proc. CVPR*, 2008. 5
- Andreas Lanitis, Christopher J Taylor, and Timothy F Cootes. Automatic face identification system using flexible appearance models. *Image and vision computing*, 1995. 17
- Ivan Laptev. On space-time interest points. *IJCV*, 2005. 13
- Ivan Laptev. *Modeling and visual recognition of human actions and interactions*. Habilitation à diriger des recherches en informatique, École normale supérieure, Paris, France, July 2013. 3
- Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Proc. CVPR*, 2008. 14, 37
- Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *Proc. CVPR*, 2017. 124
- S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006. 37, 47
- Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese cnn for robust target association. In *Proc. CVPRW*, 2016. 21

- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Intelligent signal processing*, 1998. 26
- Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proc. CVPR*, 2017. 27
- Chi Li, M Zeeshan Zia, Quoc-Huy Tran, Xiang Yu, Gregory D Hager, and Manmohan Chandraker. Deep supervision with shape concepts for occlusion-aware 3d object parsing. *Proc. CVPR*, 2017. 27
- Ming Liang and Xiaolin Hu. Recurrent Convolutional Neural Network for Object Recognition. In *Proc. CVPR*, 2015. 104, 108
- Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proc. ECCV*. 2014. 77
- Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *Proc. CVPR*, 2017a. 102, 104
- Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *Proc. ICCV*, 2017b. 102, 109
- Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In *Proc. ECCV*, 2016. 20, 57, 102, 109, 110
- Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 2015. 27, 84
- D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 24, 47

- Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. Predicting Deeper into the Future of Semantic Segmentation. In *Proc. ICCV*, 2017. 22
- M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *Proc. CVPR*, 2009. 37, 38
- Markus Mathias, Rodrigo Benenson, Marco Pedersoli, and Luc Van Gool. Face detection without bells and whistles. In *Proc. ECCV*, 2014. 70, 71, 73
- Michael Mathieu, Camille Couprie, and Yann Lecun. Deep Multi Scale Video Prediction Beyond Mean Square Error. In *Proc. ICLR*, 2016. 106
- Stephen J McKenna, Shaogang Gong, and Yogesh Raja. Modelling facial colour and identity with gaussian mixtures. *Pattern recognition*, 1998. 17
- Taylor Mordan, Nicolas Thome, Matthieu Cord, and Gilles Henaff. Deformable part-based fully convolutional network for object detection. *Proc. BMVC*, 2017. 27
- Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. ICML*, 2010. 26, 112
- Ara V Nefian and Monson H Hayes. Face detection and recognition using hidden markov models. In *Proc. ICIP*, 1998. 17
- Gerhard Neuhold, Tobias Ollmann, S Rota Buló, and Peter Kotschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proc. ICCV*, 2017. 123
- J.C. Niebles, C.-W. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *Proc. ECCV*. 2010. 37
- A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001. 37
- Nuria Oliver, Alex Pentland, and François Bérard. Lafter: a real-time face and lips tracker with facial expression recognition. *Pattern recognition*, 2000. 17

- Dan Oneata, Jerome Revaud, Jakob Verbeek, and Cordelia Schmid. Spatio-temporal object detection proposals. In *Proc. ECCV*, 2014. 90, 92
- M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proc. CVPR*, 2014. 59, 62, 67, 74
- Vicente Ordonez, Girish Kulkarni, and Tamara L Berg. Im2text: Describing images using 1 million captioned photographs. In *Proc. NIPS*, 2011. 20
- A. Osokin and P. Kohli. Perceptually inspired layout-aware losses for image segmentation. In *Proc. ECCV*, 2014. 64
- Edgar Osuna, Robert Freund, and Federico Girosit. Training support vector machines: an application to face detection. In *Proc. CVPR*, 1997. 17
- Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *IJCV*, 2000. 16
- Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Proc. ICCV*, 1998. 17
- Seymour A Papert. The summer vision project. 1966. 3
- Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *Proc. CVPR*, 2017. 124
- A. Patron-Perez, M. Marszalek, I. Reid, and A. Zisserman. Structured learning of human interactions in tv shows. *IEEE TPAMI*, 2012. 69, 70
- Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *Proc. CVPR*, 2012. 15
- Marco Pedersoli, Andrea Vedaldi, and Jordi Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *Proc. CVPR*, 2011. 18

- F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proc. CVPR*, 2007. 24, 50
- F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *Proc. ECCV*, 2010. 25
- Pedro O. Pinheiro and Ronan Collobert. Recurrent Convolutional Neural Networks for Scene Labeling, 2014. 104, 108
- Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Proc. CVPR*, 2011. 20, 21, 93
- John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 1999. 48
- Alessandro Prest, Christian Leistner, Javier Civera, Cordelia Schmid, and Vittorio Ferrari. Learning object class detectors from weakly annotated video. In *Proc. CVPR*, 2012. 10, 88, 95
- A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Proc. CVPR*, 2010. 37
- Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. In *Proc. CVPR*, 2017. 20, 57, 110
- Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 20, 57
- Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. CVPR*, 2016. 20, 110
- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. NIPS*, 2015. 4, 17, 19, 27, 32, 33, 85, 102, 109, 110, 147
- X. Ren. Finding people in archive films through tracking. In *Proc. CVPR*, 2008. 69, 70, 73, 74, 87



- Ma Mercedes T Rodrigo and Ryan Sjd Baker. Coarse-grained detection of student frustration in an introductory programming course. In *Proc. ICER*, 2009. 4
- Karl Rohr. Towards model-based recognition of human movements in image sequences. *Proc. CVGIP*, 1994. 13
- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 22, 88, 102, 103, 112
- S. Sadanand and J.J Corso. Action bank: A high-level representation of activity in video. In *Proc. CVPR*, 2012. 37
- Toshiyuki Sakai, Makoto Nagao, and Shinya Fujibayashi. Line extraction and pattern detection in a photograph. *Pattern recognition*, 1969. 17
- Ferdinando Samaria and Steve Young. Hmm-based architecture for face identification. *Image and vision computing*, 1994. 17
- J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 2013. 24
- Henry Schneiderman and Takeo Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *Proc. CVPR*, 1998. 17
- Henry Schneiderman and Takeo Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proc. CVPR*, 2000. 17
- Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. In *Proc. ICPR*, 2004. 13
- Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork Convnets for Video Semantic Segmentation. In *Proc. ECCVW*, 2016. 22, 107
- Jianbo Shi et al. Good features to track. In *Proc. CVPR*, 1994. 86

- Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. 2016. 89, 103
- K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*, 2015. 59, 74
- Karen Simonyan and Andrew Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Proc. NIPS*, 2014. 14, 21, 22
- Saad Ahmed Sirohey. Human face segmentation and identification. Technical report, 1998. 16
- Karin Sobottka and Ioannis Pitas. Extraction of facial regions and features using color and shape information. In *Proc. ICPR*, 1996. 17
- Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 14
- Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised Learning of Video Representations using LSTMs. In *Proc. ICML*, 2015. 21
- Ömer Sümer, Patricia Goldberg, Kathleen Stürmer, Tina Seidel, Peter Gerjets, Ulrich Trautwein, and Enkelejda Kasneci. Teacher’s perception in the classroom. *CoRR*, 2018. 4
- Siyu Tang, Bjoern Andres, Miykhaylo Andriluka, and Bernt Schiele. Subgraph decomposition for multi-target tracking. In *Proc. CVPR*, 2015. 20, 21
- B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Proc. NIPS*, 2003. 64
- K. Toutanova, D. Klein, C.D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. NAACL HLT*, 2003. 40
- D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proc. ICCV*, 2015. 14, 21, 84

- I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 2005. 64
- Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 1991. 17
- Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 2013. 5, 19, 30, 67, 86
- K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Empowering visual categorization with the gpu. *IEEE Transactions on Multimedia*, 2011a. 47
- K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as search for object recognition. In *Proc. ICCV*, 2011b. 59
- Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE TPAMI*, 2016. 14
- Gül Varol, Duygu Ceylan, Bryan Russell, Jimei Yang, Ersin Yumer, Ivan Laptev, and Cordelia Schmid. Bodynet: Volumetric inference of 3d human body shapes. *Proc. ECCV*, 2018. 27
- A. Vedaldi and K. Lenc. MatConvNet – convolutional neural networks for MATLAB. In *Proc. ACMMM*, 2015. 74
- Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, 2001. 4, 16, 18
- J. Vogel and B. Schiele. Natural scene retrieval based on a semantic modeling step. In *Proc. CIVR*. 2004. 37
- Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating Visual Representations From Unlabeled Video. In *Proc. CVPR*, 2016. 22
- Tuan-Hung Vu, Catherine Olsson, Ivan Laptev, Aude Oliva, and Josef Sivic. Predicting actions from static scenes. In *Proc. ECCV*, 2014. 4, 6, 9, 147

- Tuan-Hung Vu, Anton Osokin, and Ivan Laptev. Context-aware CNNs for person head detection. In *Proc. ICCV*, 2015. 4, 7, 10, 27, 87, 93, 147
- J. Walker, A. Gupta, and M. Hebert. Patch to the future: Unsupervised visual prediction. In *Proc. CVPR*, 2014. 4, 15
- Heng Wang, Alexander Kläser, Cordelia Schmid, and Liu Cheng-Lin. Action Recognition by Dense Trajectories. In *Proc. CVPR*, 2011. 14, 37
- X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proc. ICCV*, 2015. 21
- X. Wang, A. Shrivastava, and A. Gupta. A-Fast-RCNN: Hard positive generation via adversary for object detection. In *Proc. CVPR*, 2017. 103
- Max Wertheimer. Laws of organization in perceptual forms. *Psychologische Forschung*, 1938. 8, 101
- Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*, 2010. 6, 39, 47, 147
- Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit All the Layers: Fast and Accurate CNN Object Detector With Scale Dependent Pooling and Cascaded Rejection Classifiers. In *Proc. CVPR*, 2016. 104
- Jie Yang and Alex Waibel. A real-time face tracker. In *Proc. WACV*, 1996. 17
- Weilong Yang, Yang Wang, and Greg Mori. Recognizing human actions from still images with latent poses. In *Proc. CVPR*, 2010. 4
- Bangpeng Yao and Li Fei-Fei. Grouplet: A structured image representation for recognizing human and object interactions. In *Proc. CVPR*, 2010a. 4, 15
- Bangpeng Yao and Li Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *Proc. CVPR*, 2010b. 15

- Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing Videos by Exploiting Temporal Structure. In *Proc. ICCV*, 2015. 21
- Tae-Woong Yoo and Il-Seok Oh. A fast algorithm for tracking human faces based on chromatic histograms. *Pattern Recognition Letters*, 1999. 17
- YoungJoon Yoo, Kimin Yun, Sangdoo Yun, JongHee Hong, Hawook Jeong, and Jin Young Choi. Visual path prediction in complex scenes with crowded moving objects. In *Proc. CVPR*, 2016. 16
- Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *Proc. CVPR*, 2016. 27
- J. Yuen and A. Torralba. A data-driven approach for event prediction. In *Proc. ECCV*, 2010. 15
- Alan L Yuille, Peter W Hallinan, and David S Cohen. Feature extraction from faces using deformable templates. *IJCV*, 1992. 17
- J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 2007. 47
- Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep Feature Flow for Video Recognition. In *Proc. CVPR*, 2017a. 22, 111, 112
- Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-Guided Feature Aggregation for Video Object Detection. *Proc. ICCV*, 2017b. 4, 22, 102, 103, 104, 105, 107, 112, 113, 114, 116, 117
- C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *Proc. ECCV*, 2014. 5

# List of Figures

1-1	Illustration of scene-action correlation [Vu et al., 2014]: Images of scene classes <code>sandbar</code> and <code>temple_east_asia</code> from the SUN dataset [Xiao et al., 2010] together with probabilities for the five most likely actions, predicted manually by people (red) and by an automatic method (green).	6
1-2	Illustration of contextual models [Vu et al., 2015]. . . . .	7
1-3	Space-time object tubes used in our work to capture evolution of object appearance over short sequences of video frames. . . . .	7
1-4	A schematic comparison between a typical per-frame method (left) and the memory-based video representation learning approach (right) for object detection in videos. . . . .	8
3-1	AlexNet architecture for image classification task: 5 successive convolutional layers followed by 3 fully connected layers [Krizhevsky et al., 2012]. . . . .	28
3-2	VGG-16 architecture for image classification task: 13 successive convolutional layers followed by 3 fully connected layers [Chatfield et al., 2014]. . . . .	29
3-3	A comparison between VGG-19 and ResNet-34 with respect to the depth [He et al., 2016]. . . . .	29
3-4	R-CNN object detection pipeline [Girshick et al., 2014]. . . . .	30
3-5	Fast-RCNN object detection pipeline [Girshick, 2015]. . . . .	32
3-6	Faster-RCNN object detection pipeline [Ren et al., 2015]. . . . .	33

3-7	Position-sensitive score maps and Position-sensitive RoI Pooling [Dai et al., 2016]. . . . .	34
4-1	SUN Action scene categories with corresponding histograms of action responses. Top row: Scene categories with low entropy of response histograms. Bottom row: scene categories with higher entropy of response histograms. Low-entropy categories are often places designed for specific purposes (tennis court) or where the environment limits possible actions (wave). By comparison, high-entropy categories are places that afford many actions (television studio) or are unfamiliar (anechoic chamber) . . . . .	40
4-2	Histograms of words in action responses for two images of the scene class <b>crosswalk</b> . The presence of a cyclist in the image on the right biases responses to contain the action “bike”, which is not present in other crosswalk images. . . . .	40
4-3	Results of action-based scene classification. (a): Confusion matrix for the 33-category subset using Maximum Likelihood Naive Bayes estimation. The high values along the diagonal indicate excellent classification performance. A few pairs of categories e.g., (basement,attic) and (river,lake) are confused due to similarity in their characteristic actions. (b): Average accuracy (%) of scene classification for the 33-category subset and for all 397 scene categories. . . . .	43
4-4	Results of hierarchical clustering of 33 scene categories based on the similarity of image descriptors (left) and action similarity (right). Image-based similarity groups similar- <i>looking</i> scenes despite their large difference in semantics such as “alley” and “bathroom”. In contrast, action-based similarity results in more semantically meaningful clusters. For example, “mountain, snowy” is placed in a category of its own according to the visual similarity, whereas it is grouped together with other outdoor places on the basis of action similarity. . . . .	44
4-5	Automatic visual action prediction for test images in SUN Action dataset. . . . .	49

4-6	Results of action prediction for all 38 outdoor actions (top) and 23 indoor actions (bottom) sorted in the decreasing order of Average Precision (AP).	49
4-7	Selected SUN Action classification results - both outdoor (cyan) and indoor (orange) - with our best kernel combination. . . . .	50
4-8	“Where can I ski in France?” - (Top left) Official skiing stations in France [ski, 2013]. (Top middle) Suggested places for skiing by IGMA. (Top right) Dense map of action “ski” generated by IGMA. (Bottom) Panoramic images of suggested places for skiing. . . . .	52
4-9	Geo-localized prediction of actions. (left): Predictions for actions “hike” and “swim” on the map of France. (right): Predictions for the action “swim” in Paris. . . . .	52
5-1	Results of head detection for a sample movie frame. The output of our method (bottom) is obtained from the combination of Local, Global and Pairwise CNN models. Bounding boxes illustrate detections: yellow – correct, red – false. Links between detections correspond to the pairwise potentials of our model: yellow – attractive, red – repulsive. . . . .	58
5-2	Qualitative results for the Global model. The top row shows detections produced by the Local model. The middle row illustrates the multi-scale score map produced by our Global model. Red color correspond to high score values for the “head” class, blue color – to low score values. The bottom row demonstrates detections by the combination of the Local and Global models. . . . .	72
5-3	Results of our method compared to the state-of-the-art on the three datasets. . . . .	73



5-4	Qualitative results for the Pairwise model. For each video frame we show results of the Local model (left) and the Pairwise model (right). For both methods we choose the score thresholds such that the precision equals the recall on the validation set. The plotted bounding boxes show the detections with the scores above the selected thresholds. Yellow boxes correspond to correct detections, red – to false positives. For the Pairwise model we show the strength of links between the candidates detected by at least one method. Links above a strength threshold (attractive) are plotted yellow and others – red (repulsive).	75
6-1	Space-time object tubes used in our work to capture evolution of object appearance over short sequences of video frames.	80
6-2	Architecture of Tube-CNN for object detection. The input is a chunk of images and a set of tube proposals. The model starts with the CNN feature extractor to get a spatio-temporal feature volume. The following network splits into the three branches: tube classification (starts with Tube-of-Interest, TOI, pooling), regression on the start and end frames, which begins with the frame-level Region-of-Interest pooling, ROI.	81
6-3	Architecture of Tube Proposal Network (TPN). Tube anchors are plotted in red, yellow and green.	84
6-4	Examples of TPN proposal. Examples of TPN proposals. Tube anchors and final proposal are plotted by green and magenta respectively.	86
6-5	Box-recall and tube-recall <i>vs.</i> average number of tube proposals per chunk on HollywoodHeads and ImageNET VID datasets.	91
6-6	Detection results on HollywoodHeads-Hard.	95

6-7	Qualitative results for HollywoodHeads, Casablanca and ImageNet VID datasets. On each frame we illustrate results of the Box-CNN model and the Tube-CNN model. For both methods we choose thresholds such that precision equals recall on the corresponding dataset. All detections with scores above threshold are plotted. Yellow boxes: correct detections, red boxes: false detections. . . . .	96
7-1	A schematic comparison between a typical per-frame method (left) and the proposed video representation learning approach (right) for object detection in videos. We propose a multi-scale memory that efficiently aggregates image evidence over longer time horizons and also accounts for camera and object motion by feature warping, which enables learning better representations that lead to higher accuracy. . . . .	102
7-2	A detailed illustration of the proposed <b>MemNet</b> running for three frames. At every time step, features from the current frame (blue) are aggregated with the memory of the previous frame (purple), either by simple averaging or with a learned adaptive weighting. Then, the memory feature map is warped via bilinear sampling based on a learned displacement field. The detection output in every frame is computed from the current memory. . .	104
7-3	Our <b>ClockNet</b> extends the MemNet by adding multiple time axes with increasing time scales to aggregate more information from further back in time. Each additional time axis $k > 1$ skips $2^{k-1} - 1$ frames. We only illustrate two time scales to avoid clutter. . . . .	108
7-4	Relation between standard convolutional recurrent neural networks (left) and the proposed video representation (right). . . . .	108

7-5	In a multi-GPU setup, a fast but weak object detector (R-FCN-18, green blocks) leverages the features of a strong but slow object detector (ClockNet-101, red blocks), see Section 7.2.2. The width of blocks represent computation time. At a frame $t$ , strong features from ClockNet-101 are only available from $t - \Delta$ , but efficiently warped into frame $t$ with our propagation module. The warped features boost the representational power of R-FCN-18 significantly, without increasing latency of the real-time system. . . . .	110
7-6	Accuracy and runtime trade-off of various methods. . . . .	115
7-7	Qualitative examples of our memory models and the RFCN baseline. . . .	115
7-8	We illustrate the impact of jointly fine-tuning the FlowNet with the object detector. For each example consists of 4 images: the 2 input images (“Image0, Image1”) and the displacement field before and after fine-tuning. The figure shows 2 columns and 4 rows of examples. . . .	118
7-9	Mean AP of MemNet and ClockNet with respect to different propagation lengths $\delta$ . . . . .	119

# List of Tables

4.1	Scene-based outdoor (cyan) and indoor (orange) action prediction results with different approaches. Note that mAP at chance level is 6.32% (outdoor) and 4.24% (indoor). <b>S1</b> and <b>S2</b> columns respectively show classification mAP (%) of the two aforementioned training schemes. Column ( <b>S1-S2</b> ) shows the different mAP between two schemes. We observe consistently better performance of scheme <b>S1</b> : directly training binary action classifiers over scheme <b>S2</b> : aggregating scene classifiers.	51
5.1	Performance (% AP) of different context-aware models on three datasets: Casablanca, HollywoodHeads (HH) and TVHL. . . . .	73
5.2	Performance (% AP) of Local models of different architectures on the HollywoodHeads validation set. Bottom lines report training and testing speed, measured by the number of image patches processed per second. . . . .	76
5.3	Performance of models trained on the training sets of different sizes. We report % AP for each test set. . . . .	76
5.4	Performance of the R-CNN method and of our Local model (% AP) on the test set of HollywoodHeads with different percentage of candidates remaining after filtering using the Global model. . . . .	77
6.1	Evaluation of tube proposals in terms of tube-recall and box-recall on HollywoodHeads and VID datasets. Avg.#TP stands for the average number of tube proposals per chunk of consecutive frames. . . . .	92

6.2	Detection performance mAP (%) on the HollywoodHeads, Casablanca and ImageNet VID datasets. . . . .	93
6.3	Object detection performance AP (%) on ImageNet VID validation set. . . . .	94
6.4	Localization performance CorLoc (%) on the YTO dataset. . . . .	94
6.5	Detection performance mAP (%) on HollywoodHeads and ImageNet VID dataset. CN and RN and HN stands for CaffeNet, Resnet and hard negative mining respectively. . . . .	97
6.6	Tube-CNN performance for different values of T. . . . .	98
6.7	Aggregation of Box-CNN and Tube-CNN. . . . .	98
6.8	Running times (frames per second) for Tube-CNN object detectors using SS-Track proposals and different numbers of TPN proposals. . . . .	99
7.1	Detection performance and runtime on ImageNet-VID validation of different methods. We also report results on the three validation subsets of fast, medium and slowly moving objects, denoted as mAP(fast), mAP(med) and mAP(slow). . . . .	114
7.2	Detection performance of MemNet in ablation studies on ImageNet-VID validation set. (a) shows the impact of different aggregation schemes and (b) shows different levels of the feature representation where the memory module is employed. . . . .	117
7.3	Detection performance (runtime fps and accuracy mAP) on ImageNet-VID validation set of MemNet and ClockNet with feature propagation and feature anticipation. We also compare our results to the box-propagating baselines. . . . .	119
7.4	Accuracy and runtime of our two-threaded detection setup. The number of GPUs utilized is denoted as #G. . . . .	121



## Abstract

In this thesis, we address person detection and action prediction in visual data. We develop models that learn representations for visual data and the structure in the output space while making use of contextual cues and temporal consistency. We also propose a predictive model to anticipate person's attention in given static scenes.

In the first part of the thesis, we explore the strong association between scene categories and actions. Based on that understanding, we formulate a new task of predicting human actions in static scenes. To train and evaluate the proposed model, we collect a new dataset of scene-action associations, named SUN Action dataset. The success of this task enables potential applications such as affordance geo-localization.

The second part of the thesis is focused on person and generic object detection in videos. First, we construct contextual models to enhance person detection in individual frames. We train and evaluate our method on our new HollywoodHeads dataset with annotated human heads in movies. Our models consistently improve detection performance over baseline detectors. Second, we introduce a novel convolutional neural network architecture operating on short clips of frames to leverage temporal consistency and to learn spatio-temporal representations. By empirical experiments, we demonstrate the benefit of our spatio-temporal representations for object detection in videos. Last, we learn video representations that incorporate multiscale information on coarse time scales and design practical frameworks that achieve accuracy, efficiency and predictive power. Compared to per-frame features, our video representations show best detection improvement on frames degraded by fast motions.

## Keywords

Scene understanding, action prediction, interaction analysis, spatio-temporal visual representation, object detection in videos, deep convolutional neural networks, representation learning.